9.2

Создаем стенд

Текущий стенд состоит из 2-ух узлов: `vs01.sXXXXXX`, `vs02.sXXXXXX` (XXXXXX - ваш номер студента). Работа стенда 6 часов. 2 попытки запуска стенда.

1. Над текстом нажмите кнопку "Создать стенд". Каждый стенд создан под определенную тему курса. Сейчас вы находитесь в теме - **9. Мониторинг в кейсах** , этот стенд не подойдет под практические занятия из других пунктов курса.

Запуск обычно идёт до 10 минут, в редких случаях - до 30 минут.

2. После создания стенда вам нужно зайти по SSH на **sbox.slurm.io** с логином и паролем из личного кабинета (https://edu.slurm.io/):

```
ssh s000001@sbox.slurm.io
```

s000001 нужно заменить на ваш номер студента

Далее вам нужно перейти на сам стенд (sbox является просто jump-хостом и не понадобится кроме как для входа):

```
ssh vs01.s000001
```

s000001 снова нужно заменить на ваш личный номер студента

На хосте vs01 у вас есть полные права (включая беспарольный sudo):

```
sudo -i
```

Будьте осторожны и не сломайте стенд до прохождения всех заданий :)

Практика: Параметры конфигурации PostgreSQL

Так же можете пройтись по шагам текущей практики
- https://gitlab.slurm.io/postgres/slurm_course/-/blob/main/practice/lecture6/1.statistics.md, дублированной ниже по тексту.

Статистика

1. Давайте попробуем сломать статистику и посмотрим, как это отобразится в системе мониторинга

2. Для начала давайте запустим нагрузку на наш сервер

```
student$ sudo -u postgres -i
postgres$ cd ~
postgres$ ./tank.sh DB="demo" CLUSTER_HOST="localhost" CLUSTER_PORT="5432"
TIME="36000" CNT="10"
```

3. Этот скрипт запускает файл `./tank.sql`. При желании можно его открыть и посмотреть

4. Посмотрим таблицу `bookings.flights` и выполним запрос

В ДРУГОЙ СЕССИИ !!!

```
postgres$ psql demo

demo=> \d+ bookings.flights
                                                                                Table
"bookings.flights"
       Column        |           Type           | Collation | Nullable |
Default                        | Storage  | Stats target |         Description
---------------------+--------------------------+-----------+----------+-------------
-----------------------------------+----------+--------------+-----------------
-----------
 flight_id           | integer                  |           | not null |
nextval('bookings.flights_flight_id_seq'::regclass) | plain    |              |
Идентификатор рейса
 flight_no           | character(6)             |           | not null |
| extended |              | Номер рейса
 scheduled_departure | timestamp with time zone |           | not null |
| plain    |              | Время вылета по расписанию
 scheduled_arrival   | timestamp with time zone |           | not null |
| plain    |              | Время прилёта по расписанию
 departure_airport   | character(3)             |           | not null |
| extended |              | Аэропорт отправления
 arrival_airport     | character(3)             |           | not null |
| extended |              | Аэропорт прибытия
 status              | character varying(20)    |           | not null |
| extended |              | Статус рейса
 aircraft_code       | character(3)             |           | not null |
| extended |              | Код самолета, IATA
 actual_departure    | timestamp with time zone |           |          |
| plain    |              | Фактическое время вылета
 actual_arrival      | timestamp with time zone |           |          |
| plain    |              | Фактическое время прилёта
Indexes:
    "flights_pkey" PRIMARY KEY, btree (flight_id)
    "flights_flight_no_scheduled_departure_key" UNIQUE CONSTRAINT, btree (flight_no,
scheduled_departure)
    "ix__flights__scheduled_departure" btree (scheduled_departure)
Check constraints:
    "flights_check" CHECK (scheduled_arrival > scheduled_departure)
    "flights_check1" CHECK (actual_arrival IS NULL OR actual_departure IS NOT NULL
AND actual_arrival IS NOT NULL AND actual_arrival > actual_departure)
    "flights_status_check" CHECK (status::text = ANY (ARRAY['On Time'::character
varying::text, 'Delayed'::character varying::text, 'Departed'::character
varying::text, 'Arrived'::character varying::text, 'Scheduled'::character
varying::text, 'Cancelled'::character varying::text]))
Foreign-key constraints:
```

```
    "flights_aircraft_code_fkey" FOREIGN KEY (aircraft_code) REFERENCES
bookings.aircrafts(aircraft_code)
    "flights_arrival_airport_fkey" FOREIGN KEY (arrival_airport) REFERENCES
bookings.airports(airport_code)
    "flights_departure_airport_fkey" FOREIGN KEY (departure_airport) REFERENCES
bookings.airports(airport_code)
Referenced by:
    TABLE "bookings.ticket_flights" CONSTRAINT "ticket_flights_flight_id_fkey"
FOREIGN KEY (flight_id) REFERENCES bookings.flights(flight_id)
Access method: heap

5432=> explain (analyze, buffers) select count(*), a.model  from bookings.tickets t
join bookings.ticket_flights tf on t.ticket_no = tf.ticket_no join bookings.flights f
on tf.flight_id = f.flight_id join bookings.aircrafts a on a.aircraft_code =
f.aircraft_code where f.scheduled_departure between '2016-10-01' and '2016-11-01'
group by a.model;


QUERY PLAN
------------------------------------------------------------------------------------
------------------------------------------------------------------------------------
-------------------------------------------------
 Finalize GroupAggregate  (cost=66357.77..66360.05 rows=9 width=24) (actual
time=2073.009..2088.056 rows=8 loops=1)
   Group Key: a.model
   Buffers: shared hit=13208 read=22242, temp read=6401 written=6524
   ->  Gather Merge  (cost=66357.77..66359.87 rows=18 width=24) (actual
time=2072.999..2088.041 rows=24 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         Buffers: shared hit=13208 read=22242, temp read=6401 written=6524
         ->  Sort  (cost=65357.75..65357.77 rows=9 width=24) (actual
time=2051.453..2051.461 rows=8 loops=3)
               Sort Key: a.model
               Sort Method: quicksort  Memory: 25kB
               Worker 0:  Sort Method: quicksort  Memory: 25kB
               Worker 1:  Sort Method: quicksort  Memory: 25kB
               Buffers: shared hit=13208 read=22242, temp read=6401 written=6524
               ->  Partial HashAggregate  (cost=65357.52..65357.61 rows=9 width=24)
(actual time=2051.404..2051.411 rows=8 loops=3)
                     Group Key: a.model
                     Buffers: shared hit=13192 read=22242, temp read=6401
written=6524
                     ->  Hash Join  (cost=39194.40..64098.95 rows=251713 width=16)
(actual time=1322.816..1873.465 rows=222358 loops=3)
                           Hash Cond: (f.aircraft_code = a.aircraft_code)
                           Buffers: shared hit=13192 read=22242, temp read=6401
written=6524
                           ->  Parallel Hash Join  (cost=39193.19..63125.86
rows=251713 width=4) (actual time=1322.738..1693.474 rows=222358 loops=3)
                                 Hash Cond: (t.ticket_no = tf.ticket_no)
                                 Buffers: shared hit=13154 read=22242, temp read=6401
written=6524
                                 ->  Parallel Seq Scan on tickets t
(cost=0.00..17406.40 rows=346340 width=14) (actual time=0.025..177.330 rows=277072
loops=3)
                                       Buffers: shared hit=6135 read=7808
                                 ->  Parallel Hash  (cost=34571.78..34571.78
rows=251713 width=18) (actual time=968.819..968.822 rows=222358 loops=3)
                                       Buckets: 65536  Batches: 16  Memory Usage:
2848kB
```

```
                                          Buffers: shared hit=7019 read=14434, temp
written=3000
                                          ->  Parallel Hash Join
(cost=2292.46..34571.78 rows=251713 width=18) (actual time=17.122..782.220
rows=222358 loops=3)
                                                Hash Cond: (tf.flight_id = f.flight_id)
                                                Buffers: shared hit=7019 read=14434
                                                ->  Parallel Seq Scan on ticket_flights
tf  (cost=0.00..29691.55 rows=985755 width=18) (actual time=0.028..330.941
rows=788604 loops=3)
                                                      Buffers: shared hit=5409
read=14425
                                                ->  Parallel Hash
(cost=2168.86..2168.86 rows=9888 width=8) (actual time=9.911..9.912 rows=5601
loops=3)
                                                      Buckets: 32768  Batches: 1  Memory
Usage: 928kB
                                                      Buffers: shared hit=1579 read=9
                                                      ->  Parallel Seq Scan on flights f
(cost=0.00..2168.86 rows=9888 width=8) (actual time=0.068..5.729 rows=5601 loops=3)
                                                            Filter:
((scheduled_departure >= '2016-10-01 00:00:00+00'::timestamp with time zone) AND
(scheduled_departure <= '2016-11-01 00:00:00+00'::timestamp with time zone))
                                                            Rows Removed by Filter:
16343
                                                            Buffers: shared hit=1579
read=9
                               ->  Hash  (cost=1.09..1.09 rows=9 width=20) (actual
time=0.025..0.026 rows=9 loops=3)
                                     Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                     Buffers: shared hit=3
                                     ->  Seq Scan on aircrafts a  (cost=0.00..1.09 rows=9
width=20) (actual time=0.015..0.017 rows=9 loops=3)
                                           Buffers: shared hit=3
 Planning Time: 1.813 ms
 Execution Time: 2088.432 ms
(46 rows)
```

5. Давайте поговорим про time, cost и buffers. Что вы про это знаете?

6. Проверяем нашу статистику и автовакуум

```
demo=> \x
demo=> select * from pg_settings where name = 'autovacuum';
-[ RECORD 1 ]---+---------------------------------
name            | autovacuum
setting         | on
unit            |
category        | Autovacuum
short_desc      | Starts the autovacuum subprocess.
extra_desc      |
context         | sighup
vartype         | bool
source          | default
min_val         |
max_val         |
enumvals        |
boot_val        | on
reset_val       | on
sourcefile      |
sourceline      |
pending_restart | f
```

```
demo=> select * from pg_settings where name = 'default_statistics_target';
-[ RECORD 1 ]---+-----------------------------------------------------------
-----------------------------------
name            | default_statistics_target
setting         | 100
unit            |
category        | Query Tuning / Other Planner Options
short_desc      | Sets the default statistics target.
extra_desc      | This applies to table columns that have not had a column-specific
target set via ALTER TABLE SET STATISTICS.
context         | user
vartype         | integer
source          | default
min_val         | 1
max_val         | 10000
enumvals        |
boot_val        | 100
reset_val       | 100
sourcefile      |
sourceline      |
pending_restart | f
```

7. Выключаем их

```
demo=> alter system set autovacuum=off;
ALTER SYSTEM

demo=> alter system set  default_statistics_target=1;
ALTER SYSTEM
```

8. Перезагружаем конфиги

```
demo=> select pg_reload_conf();
-[ RECORD 1 ]--+--
pg_reload_conf | t
```

9. Проверяем, что выключен автовакуум и сбор статистики поставлен на минимум

```
demo=> select * from pg_settings where name = 'autovacuum';
-[ RECORD 1 ]---+---------------------------------
name            | autovacuum
setting         | off
unit            |
category        | Autovacuum
short_desc      | Starts the autovacuum subprocess.
extra_desc      |
context         | sighup
vartype         | bool
source          | default
min_val         |
max_val         |
enumvals        |
boot_val        | on
reset_val       | on
sourcefile      |
sourceline      |
pending_restart | f

demo=> select * from pg_settings where name = 'default_statistics_target';
-[ RECORD 1 ]---+-----------------------------------------------------------
-----------------------------------
name            | default_statistics_target
```

```
setting         | 1
unit            |
category        | Query Tuning / Other Planner Options
short_desc      | Sets the default statistics target.
extra_desc      | This applies to table columns that have not had a column-specific
target set via ALTER TABLE SET STATISTICS.
context         | user
vartype         | integer
source          | default
min_val         | 1
max_val         | 10000
enumvals        |
boot_val        | 100
reset_val       | 100
sourcefile      |
sourceline      |
pending_restart | f
```

10. Запускаем анализ

```
demo=> \q
postgres$ vacuumdb demo -Zq -j4
```

11. Ломаем дальше

```
postgres$ psql demo
psql (12.6)
Type "help" for help.

demo=> update bookings.flights set flight_no=flight_no;
UPDATE 65829

demo=> update bookings.flights set flight_id=flight_id;
UPDATE 65848
```

12. Выполним наш старый запрос

```
demo=> explain (analyze, buffers) select count(*), a.model  from bookings.tickets t
join bookings.ticket_flights tf on t.ticket_no = tf.ticket_no join bookings.flights f
on tf.flight_id = f.flight_id join bookings.aircrafts a on a.aircraft_code =
f.aircraft_code where f.flight_no = 'PG0405' and f.scheduled_departure between '2016-
10-01' and '2016-11-01' group by a.model;

demo=> update bookings.flights set flight_no=fligh_no;

QUERY PLAN
------------------------------------------------------------------------------
------------------------------------------------------------------------------
-----------------------------------------------
 Finalize GroupAggregate  (cost=49109.95..49112.23 rows=9 width=24) (actual
time=10440.568..10441.031 rows=8 loops=1)
   Group Key: a.model
   Buffers: shared hit=2674468 read=15327
   ->  Gather Merge  (cost=49109.95..49112.05 rows=18 width=24) (actual
time=10440.555..10441.014 rows=24 loops=1)
         Workers Planned: 2
         Workers Launched: 2
         Buffers: shared hit=2674468 read=15327
         ->  Sort  (cost=48109.93..48109.95 rows=9 width=24) (actual
time=10420.790..10420.798 rows=8 loops=3)
               Sort Key: a.model
               Sort Method: quicksort  Memory: 25kB
               Worker 0:  Sort Method: quicksort  Memory: 25kB
```

```
               Worker 1:  Sort Method: quicksort  Memory: 25kB
               Buffers: shared hit=2674468 read=15327
               ->  Partial HashAggregate  (cost=48109.70..48109.79 rows=9 width=24)
(actual time=10420.757..10420.766 rows=8 loops=3)
                     Group Key: a.model
                     Buffers: shared hit=2674452 read=15327
                     ->  Hash Join  (cost=2769.50..47974.17 rows=27105 width=16)
(actual time=29.419..10249.919 rows=222358 loops=3)
                           Hash Cond: (f.aircraft_code = a.aircraft_code)
                           Buffers: shared hit=2674452 read=15327
                           ->  Nested Loop  (cost=2768.30..47868.31 rows=27105
width=4) (actual time=29.359..10060.418 rows=222358 loops=3)
                                 Buffers: shared hit=2674417 read=15327
                                 ->  Parallel Hash Join  (cost=2767.87..35091.22
rows=27105 width=18) (actual time=29.307..958.314 rows=222358 loops=3)
                                       Hash Cond: (tf.flight_id = f.flight_id)
                                       Buffers: shared hit=6210 read=15240
                                       ->  Parallel Seq Scan on ticket_flights tf
(cost=0.00..29726.48 rows=989248 width=18) (actual time=0.022..366.615 rows=788604
loops=3)
                                             Buffers: shared hit=6180 read=13654
                                       ->  Parallel Hash  (cost=2741.54..2741.54
rows=2107 width=8) (actual time=18.552..18.553 rows=5601 loops=3)
                                             Buckets: 32768 (originally 4096)
Batches: 1 (originally 1)  Memory Usage: 1216kB
                                             Buffers: shared hit=2 read=1586
                                             ->  Parallel Seq Scan on flights f
(cost=0.00..2741.54 rows=2107 width=8) (actual time=0.033..9.976 rows=5601 loops=3)
                                                   Filter: ((scheduled_departure >=
'2016-10-01 00:00:00+00'::timestamp with time zone) AND (scheduled_departure <=
'2016-11-01 00:00:00+00'::timestamp with time zone))
                                                   Rows Removed by Filter: 16343
                                                   Buffers: shared hit=2 read=1586
                                 ->  Index Only Scan using tickets_pkey on tickets t
(cost=0.42..0.47 rows=1 width=14) (actual time=0.040..0.040 rows=1 loops=667073)
                                       Index Cond: (ticket_no = tf.ticket_no)
                                       Heap Fetches: 667073
                                       Buffers: shared hit=2668207 read=87
                           ->  Hash  (cost=1.09..1.09 rows=9 width=20) (actual
time=0.024..0.024 rows=9 loops=3)
                                 Buckets: 1024  Batches: 1  Memory Usage: 9kB
                                 Buffers: shared hit=3
                                 ->  Seq Scan on aircrafts a  (cost=0.00..1.09 rows=9
width=20) (actual time=0.013..0.016 rows=9 loops=3)
                                       Buffers: shared hit=3
 Planning Time: 0.784 ms
 Execution Time: 10441.125 ms
(44 rows)
```

13. Что вы можете по нему сказать? Что изменилось в нем? Какие появлись дополнительные процессы?

14. Возвращаем наши параметры обратно

```
demo=> alter system set  default_statistics_target=100;
ALTER SYSTEM

demo=> alter system set autovacuum=on;
ALTER SYSTEM

demo=> select pg_reload_conf();
```

```
 pg_reload_conf
----------------
 t
(1 row)

demo=> \q
postgres$ vacuumdb demo -q -j 4
```

А что у нас в мониторинге? Можем ли мы увидеть проблему там?

Давайте рассмотрим такой параметр из плана как recheck

15. Посмотрим на таблицу `bookings.flights`

```
postgres$ psql demo
psql (12.6)
Type "help" for help.

demo=> \d+ bookings.flights
                                                                    Table
"bookings.flights"
       Column        |           Type           | Collation | Nullable |
Default                       | Storage  | Stats target |         Description
---------------------+--------------------------+-----------+----------+-------------
-----------------------------------+----------+--------------+----------------
-----------
 flight_id           | integer                  |           | not null |
nextval('bookings.flights_flight_id_seq'::regclass) | plain    |              |
Идентификатор рейса
 flight_no           | character(6)             |           | not null |
| extended |              | Номер рейса
 scheduled_departure | timestamp with time zone |           | not null |
| plain    |              | Время вылета по расписанию
 scheduled_arrival   | timestamp with time zone |           | not null |
| plain    |              | Время прилёта по расписанию
 departure_airport   | character(3)             |           | not null |
| extended |              | Аэропорт отправления
 arrival_airport     | character(3)             |           | not null |
| extended |              | Аэропорт прибытия
 status              | character varying(20)    |           | not null |
| extended |              | Статус рейса
 aircraft_code       | character(3)             |           | not null |
| extended |              | Код самолета, IATA
 actual_departure    | timestamp with time zone |           |          |
| plain    |              | Фактическое время вылета
 actual_arrival      | timestamp with time zone |           |          |
| plain    |              | Фактическое время прилёта
Indexes:
    "flights_pkey" PRIMARY KEY, btree (flight_id)
    "flights_flight_no_scheduled_departure_key" UNIQUE CONSTRAINT, btree (flight_no,
scheduled_departure)
    "ix__flights__scheduled_departure" btree (scheduled_departure)
Check constraints:
    "flights_check" CHECK (scheduled_arrival > scheduled_departure)
    "flights_check1" CHECK (actual_arrival IS NULL OR actual_departure IS NOT NULL
AND actual_arrival IS NOT NULL AND actual_arrival > actual_departure)
    "flights_status_check" CHECK (status::text = ANY (ARRAY['On Time'::character
varying::text, 'Delayed'::character varying::text, 'Departed'::character
varying::text, 'Arrived'::character varying::text, 'Scheduled'::character
varying::text, 'Cancelled'::character varying::text]))
Foreign-key constraints:
```

```
    "flights_aircraft_code_fkey" FOREIGN KEY (aircraft_code) REFERENCES
bookings.aircrafts(aircraft_code)
    "flights_arrival_airport_fkey" FOREIGN KEY (arrival_airport) REFERENCES
bookings.airports(airport_code)
    "flights_departure_airport_fkey" FOREIGN KEY (departure_airport) REFERENCES
bookings.airports(airport_code)
Referenced by:
    TABLE "bookings.ticket_flights" CONSTRAINT "ticket_flights_flight_id_fkey"
FOREIGN KEY (flight_id) REFERENCES bookings.flights(flight_id)
Access method: heap
```

16. Создадим индекс в этой таблице по полю `scheduled_departure` и выполним простой запрос к этой таблице

```
demo=> create index concurrently ix__flights__scheduled_departure  on
bookings.flights(scheduled_departure);
CREATE INDEX

demo=> explain(analyze, buffers) select flight_id from bookings.flights f where
f.scheduled_departure between '2016-10-01' and '2016-11-01';

QUERY PLAN
------------------------------------------------------------------------------
------------------------------------------------------------------------------
---------
 Bitmap Heap Scan on flights f  (cost=353.33..2188.69 rows=16491 width=4) (actual
time=1.511..6.087 rows=16803 loops=1)
   Recheck Cond: ((scheduled_departure >= '2016-10-01 00:00:00+00'::timestamp with
time zone) AND (scheduled_departure <= '2016-11-01 00:00:00+00'::timestamp with time
zone))
   Heap Blocks: exact=915
   Buffers: shared hit=963
   ->  Bitmap Index Scan on ix__flights__scheduled_departure  (cost=0.00..349.20
rows=16491 width=0) (actual time=1.391..1.392 rows=16803 loops=1)
         Index Cond: ((scheduled_departure >= '2016-10-01 00:00:00+00'::timestamp
with time zone) AND (scheduled_departure <= '2016-11-01 00:00:00+00'::timestamp with
time zone))
         Buffers: shared hit=48
 Planning Time: 0.103 ms
 Execution Time: 7.179 ms
(9 rows)
```

17. Появился Recheck - перепроверка. Почему она появилась? Что это значит?

18. Давайте перестроим запрос так, чтобы не было Recheck

```
demo=> explain(analyze, buffers) select f.scheduled_departure from bookings.flights f
where f.scheduled_departure between '2016-10-01' and '2016-11-01';

QUERY PLAN
------------------------------------------------------------------------------
------------------------------------------------------------------------------
---
 Index Only Scan using ix__flights__scheduled_departure on flights f
(cost=0.29..514.11 rows=16491 width=8) (actual time=0.029..2.212 rows=16803 loops=1)
   Index Cond: ((scheduled_departure >= '2016-10-01 00:00:00+00'::timestamp with time
zone) AND (scheduled_departure <= '2016-11-01 00:00:00+00'::timestamp with time
zone))
   Heap Fetches: 0
   Buffers: shared hit=49
 Planning Time: 0.128 ms
 Execution Time: 3.168 ms
```

```
(6 rows)

-----------------------------------------------
```

19. Как еще можно избавиться от Recheck?

Параметры конфигурации PostgreSQL: Параметр work_mem | autovacuum_work_mem | maintenance_work_mem

Можете запустить скрипт на стенде `~/practice/lecture6/2.work_mem.sh`, который спикер запускал в видео.

Или пройтись по шагам текущей практики
- https://gitlab.slurm.io/postgres/slurm_course/-/blob/main/practice/lecture6/2.work_mem.md, также представленной ниже по тексту.

1. Посмотрим, какие параметры выставлены сейчас в системе

```
postgres$ psql -d demo

demo=> select * from pg_settings where name like '%work_mem%';

-[ RECORD 1 ]---+-----------------------------------------------------------
-----------------------------------------------
name            | autovacuum_work_mem
setting         | -1
unit            | kB
category        | Resource Usage / Memory
short_desc      | Sets the maximum memory to be used by each autovacuum worker
process.
extra_desc      |
context         | sighup
vartype         | integer
source          | default
min_val         | -1
max_val         | 2147483647
enumvals        |
boot_val        | -1
reset_val       | -1
sourcefile      |
sourceline      |
```

```
pending_restart | f
-[ RECORD 2 ]---+-------------------------------------------------------------
-----------------------------------------------
name            | maintenance_work_mem
setting         | 65536
unit            | kB
category        | Resource Usage / Memory
short_desc      | Sets the maximum memory to be used for maintenance operations.
extra_desc      | This includes operations such as VACUUM and CREATE INDEX.
context         | user
vartype         | integer
source          | default
min_val         | 1024
max_val         | 2147483647
enumvals        |
boot_val        | 65536
reset_val       | 65536
sourcefile      |
sourceline      |
pending_restart | f
-[ RECORD 3 ]---+-------------------------------------------------------------
-----------------------------------------------
name            | work_mem
setting         | 4096
unit            | kB
category        | Resource Usage / Memory
short_desc      | Sets the maximum memory to be used for query workspaces.
extra_desc      | This much memory can be used by each internal sort operation and
hash table before switching to temporary disk files.
context         | user
vartype         | integer
source          | default
min_val         | 64
max_val         | 2147483647
enumvals        |
boot_val        | 4096
reset_val       | 4096
sourcefile      |
sourceline      |
pending_restart | f
```

2. Сильно уменьшим параметр `work_mem`

```
demo=> alter system set work_mem='64';
ALTER SYSTEM

demo=> select * from pg_reload_conf();
-[ RECORD 1 ]--+--
pg_reload_conf | t
```

3. Проверим, какой стал сейчас параметр

```
demo=> select * from pg_settings where name like work_mem;

-[ RECORD 1 ]---+-------------------------------------------------------------
-----------------------------------------------
name            | work_mem
setting         | 64
unit            | kB
category        | Resource Usage / Memory
short_desc      | Sets the maximum memory to be used for query workspaces.
extra_desc      | This much memory can be used by each internal sort operation and
hash table before switching to temporary disk files.
```

```
context           | user
vartype           | integer
source            | configuration file
min_val           | 64
max_val           | 2147483647
enumvals          |
boot_val          | 4096
reset_val         | 64
sourcefile        | /var/lib/pgsql/12/data/postgresql.auto.conf
sourceline        | 5
pending_restart   | f
```

4. Выполним запрос и разберем план его выполнения

```
demo=> explain (analyze,buffers)select count(*) from (select distinct flight_id,
flight_no from bookings.flights) f where flight_id in (select flight_id from
bookings.boarding_passes b where seat_no = 'A2');
                                                                        QUERY PLAN
-----------------------------------------------------------------------------------
-----------------------------------------------------------------
 Aggregate  (cost=41654.16..41654.17 rows=1 width=8) (actual time=671.702..671.758
rows=1 loops=1)
   Buffers: shared hit=11304 read=4907 dirtied=15 written=8, temp read=360
written=606
   ->  Hash Join  (cost=37262.02..39134.38 rows=1007911 width=0) (actual
time=671.697..671.753 rows=0 loops=1)
         Hash Cond: (f.flight_id = b.flight_id)
         Buffers: shared hit=11304 read=4907 dirtied=15 written=8, temp read=360
written=606
         ->  Subquery Scan on f  (cost=10896.02..12048.08 rows=65832 width=4) (actual
time=98.297..98.299 rows=1 loops=1)
               Buffers: shared hit=1591, temp read=360 written=606
               ->  Unique  (cost=10896.02..11389.76 rows=65832 width=11) (actual
time=98.295..98.296 rows=1 loops=1)
                     Buffers: shared hit=1591, temp read=360 written=606
                     ->  Sort  (cost=10896.02..11060.60 rows=65832 width=11) (actual
time=98.293..98.294 rows=1 loops=1)
                           Sort Key: flights.flight_id, flights.flight_no
                           Sort Method: external merge  Disk: 1384kB
                           Buffers: shared hit=1591, temp read=360 written=606
                           ->  Seq Scan on flights  (cost=0.00..2246.32 rows=65832
width=11) (actual time=0.010..46.799 rows=65835 loops=1)
                                 Buffers: shared hit=1588
         ->  Hash  (cost=26241.65..26241.65 rows=7548 width=4) (actual
time=573.384..573.438 rows=0 loops=1)
               Buckets: 2048  Batches: 8  Memory Usage: 16kB
               Buffers: shared hit=9713 read=4907 dirtied=15 written=8
               ->  Gather  (cost=1000.00..26241.65 rows=7548 width=4) (actual
time=573.383..573.436 rows=0 loops=1)
                     Workers Planned: 2
                     Workers Launched: 2
                     Buffers: shared hit=9713 read=4907 dirtied=15 written=8
                     ->  Parallel Seq Scan on boarding_passes b  (cost=0.00..24486.85
rows=3145 width=4) (actual time=549.651..549.651 rows=0 loops=3)
                           Filter: ((seat_no)::text = 'A2'::text)
                           Rows Removed by Filter: 631480
                           Buffers: shared hit=9713 read=4907 dirtied=15 written=8
 Planning Time: 0.557 ms
 Execution Time: 672.123 ms
(28 rows)
```

Что вы можете сказать про него?
А что у нас в мониторинге? Можем ли мы увидеть там нашу проблему?

5. Посмотрим, какое значение параметра default_statistics_target выставлены сейчас в системе

```
demo=> select * from pg_settings where name = 'default_statistics_target';

            name             | setting | unit |                category                |
short_desc                | extra_desc
| context | vartype | source  | min_val | max_val | enumvals | boot_val | reset_val |
sourcefile | sourceline | pending_restart
---------------------------+---------+------+----------------------------------------+-
----------------------------------+-----------------------------------------------
-----------------------------------------------------------+---------+---------+--
-------+---------+---------+---------+---------+----------+-----------+----------
--+-----------------
 default_statistics_target | 100     |      | Query Tuning / Other Planner Options |
Sets the default statistics target. | This applies to table columns that have not had
a column-specific target set via ALTER TABLE SET STATISTICS. | user    | integer |
default | 1       | 10000   |          | 100     | 100       |           |
| f
(1 row)
```

6. Выставим параметр work_mem в 20МБ и сильно уменьшим параметр `default_statistics_target`

```
demo=> alter system set default_statistics_target=1;
ALTER SYSTEM

demo=> alter system set work_mem='20MB';
ALTER SYSTEM

demo=> select * from pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)

demo=> \q
```

7. Сделаем анализ таблиц bookings.flights и bookings.boarding_passes

```
postgres$ vacuumdb -d demo -t bookings.flights --analyze-only --verbose
vacuumdb: vacuuming database "demo"
INFO:  analyzing "bookings.flights"
INFO:  "flights": scanned 300 of 2639 pages, containing 24426 live rows and 139 dead
rows; 300 rows in sample, 214867 estimated total rows

postgres$ vacuumdb -d demo -t bookings.boarding_passes --analyze-only --verbose
vacuumdb: vacuuming database "demo"
INFO:  analyzing "bookings.boarding_passes"
INFO:  "boarding_passes": scanned 300 of 59065 pages, containing 40534 live rows and
266 dead rows; 300 rows in sample, 7980469 estimated total rows
```

8. Выполним запрос и разберем план его выполнения

```
postgres$ psql -d demo
```

```
demo=> explain (analyze,buffers)select count(*) from (select distinct flight_id,
flight_no from bookings.flights) f where flight_id in (select flight_id from
bookings.boarding_passes b where seat_no = 'A2');

------------------------------------------------------------------------------
------------------------------------------------------------------------------
------
 Aggregate  (cost=140375.66..140375.67 rows=1 width=8) (actual
time=17380.464..17380.466 rows=1 loops=1)
   Buffers: shared hit=598761 read=48480 written=3600
   ->  Nested Loop  (cost=5864.65..111352.49 rows=11609269 width=0) (actual
time=17380.461..17380.462 rows=0 loops=1)
         Buffers: shared hit=598761 read=48480 written=3600
         ->  HashAggregate  (cost=5864.22..8013.70 rows=214948 width=11) (actual
time=756.563..909.272 rows=214867 loops=1)
               Group Key: flights.flight_id, flights.flight_no
               Buffers: shared hit=2640
               ->  Seq Scan on flights  (cost=0.00..4789.48 rows=214948 width=11)
(actual time=0.013..284.882 rows=214867 loops=1)
                     Buffers: shared hit=2640
         ->  Index Only Scan using boarding_passes_flight_id_seat_no_key on
boarding_passes b  (cost=0.43..2.50 rows=54 width=4) (actual time=0.071..0.071 rows=0
loops=214867)
               Index Cond: ((flight_id = flights.flight_id) AND (seat_no =
'A2'::text))
               Heap Fetches: 0
               Buffers: shared hit=596121 read=48480 written=3600
 Planning Time: 0.537 ms
 Execution Time: 17383.021 ms
(15 rows)
```

9. Вернем обратно параметр default_statistics_target

```
alter system set default_statistics_target=100;
ALTER SYSTEM

select * from pg_reload_conf();
 pg_reload_conf
----------------
 t
(1 row)

demo=> \q
```

10. Сделаем анализ таблиц bookings.flights и bookings.boarding_passes

```
vacuumdb -d demo -t bookings.flights --analyze-only --verbose
vacuumdb: vacuuming database "demo"
INFO:  analyzing "bookings.flights"
INFO:  "flights": scanned 2640 of 2640 pages, containing 214867 live rows and 1294
dead rows; 30000 rows in sample, 214867 estimated total rows

vacuumdb -d demo -t bookings.boarding_passes --analyze-only --verbose
vacuumdb: vacuuming database "demo"
INFO:  analyzing "bookings.boarding_passes"
INFO:  "boarding_passes": scanned 30000 of 59194 pages, containing 4013733 live rows
and 53968 dead rows; 30000 rows in sample, 7919630 estimated total rows
```

11. Выполним запрос и разберем план его выполнения

```
postgres$ psql -d demo
```

```
explain (analyze,buffers)select count(*) from (select distinct flight_id, flight_no
from bookings.flights) f where flight_id in (select flight_id from
bookings.boarding_passes b where seat_no = 'A2');

                                                                        QUERY PLAN
-----------------------------------------------------------------------------------
-------------------------------------------------------------------
 Aggregate  (cost=113088.10..113088.11 rows=1 width=8) (actual
time=6848.021..6870.766 rows=1 loops=1)
   Buffers: shared hit=3881 read=57973 dirtied=305 written=19
   ->  Hash Join  (cost=108149.80..113011.21 rows=30753 width=0) (actual
time=6848.005..6870.749 rows=0 loops=1)
         Hash Cond: (flights.flight_id = b.flight_id)
         Buffers: shared hit=3881 read=57973 dirtied=305 written=19
         ->  HashAggregate  (cost=5863.01..8011.68 rows=214867 width=11) (actual
time=1693.340..1693.341 rows=1 loops=1)
               Group Key: flights.flight_id, flights.flight_no
               Buffers: shared hit=2640
               ->  Seq Scan on flights  (cost=0.00..4788.67 rows=214867 width=11)
(actual time=0.017..406.716 rows=214867 loops=1)
                     Buffers: shared hit=2640
         ->  Hash  (cost=102196.71..102196.71 rows=7207 width=4) (actual
time=5154.604..5177.345 rows=0 loops=1)
               Buckets: 8192  Batches: 1  Memory Usage: 64kB
               Buffers: shared hit=1241 read=57973 dirtied=305 written=19
               ->  Gather  (cost=1000.00..102196.71 rows=7207 width=4) (actual
time=5154.603..5177.334 rows=0 loops=1)
                     Workers Planned: 2
                     Workers Launched: 2
                     Buffers: shared hit=1241 read=57973 dirtied=305 written=19
                     ->  Parallel Seq Scan on boarding_passes b
(cost=0.00..100476.01 rows=3003 width=4) (actual time=5058.988..5058.988 rows=0
loops=3)
                           Filter: ((seat_no)::text = 'A2'::text)
                           Rows Removed by Filter: 2641937
                           Buffers: shared hit=1241 read=57973 dirtied=305 written=19
 Planning Time: 0.573 ms
 Execution Time: 6873.942 ms
(23 rows)
```

Домашнее задание.
Запаздывание реплики
1. Создать слот репликации на сервере PostgreSQL на порту 5432;

2. Создайте реплику на 12 версии PostgreSQL на порту 5433, чтобы она работала через слот репликации;

3. На реплике выставите параметр `max_standby_streaming_delay = 1h`. Примените его

3. Выполните внутри транзакции запрос с подсчетом количества записей в таблицах `bookings.ticket_flights`, `bookings.tickets`, `bookings.flights` и паузой в 120 секунд;

4. Дождитесь сбора метрик в pgwatch2 и убедитесь, что репликация у вас начала запаздывать.

ЗАМЕЧАНИЕ
Если на графиках **pgwatch2** нет запаздывания, то можно выполнить следующие действия:
1. Нужно увеличить число секунд в параметре `pg_watch2`
2. На лидере выполнить дополнительную команду обвноления данных, например `update bookings.tickets set ticket_no = ticket_no;`

Чек-лист

1. Создан слот репликации на сервере PostgreSQL на порту 5432;
2. Создана реплика на 12 версии PostgreSQL на порту 5433 и она работает через слот репликации;
3. На реплике выставлен параметр `max_standby_streaming_delay = 1h`
4. Выполнен внутри транзакции запрос с подсчетом количества записей в таблицах `bookings.ticket_flights`, `bookings.tickets`, `bookings.flights` и паузой в 120 секунд;
5. Подтверждено состояние отставание реплики.

Решение:

1. Создать слот репликации на сервере PostgreSQL на порту `5432`;

```
student$ sudo -u postgres -i
postgres$ cd ~
postgres$ ./tank.sh DB="demo" CLUSTER_HOST="localhost" CLUSTER_PORT="5432"
TIME="36000" CNT="10"
postgres$ psql
psql (12.6)
Type "help" for help.

postgres=# SELECT pg_create_physical_replication_slot('synch_slot5433');
 pg_create_physical_replication_slot
-------------------------------------
 (synch_slot5433,)
(1 row)
```

2. Создайте реплику на 12 версии PostgreSQL на порту `5433`, чтобы она работала через слот репликации;

```
postgres=# \q
postgres$ pg_basebackup -D /var/lib/pgsql/12/main5433 -R -S synch_slot5433 -v
pg_basebackup: initiating base backup, waiting for checkpoint to complete
pg_basebackup: checkpoint completed
pg_basebackup: write-ahead log start point: 2/13000028 on timeline 1
pg_basebackup: starting background WAL receiver
pg_basebackup: write-ahead log end point: 2/13000138
pg_basebackup: waiting for background process to finish streaming ...
pg_basebackup: syncing data to disk ...
pg_basebackup: base backup completed
```

3. На реплике выставите параметр `max_standby_streaming_delay = 1h`. Примените его Строка с `port=5433` может не корректно встать в конфиг `/var/lib/pgsql/12/main5433/postgresql.conf`. Нужно пройти и исправить это.

```
postgres$ echo "port=5433" >> /var/lib/pgsql/12/main5433/postgresql.conf
postgres$ echo "max_standby_streaming_delay = 1h" >>
~/12/main5433/postgresql.conf
postgres$ pg_ctl -D /var/lib/pgsql/12/main5433 start
waiting for server to start....2021-04-24 14:54:28.832 UTC [23138] LOG:
starting PostgreSQL 12.6 on x86_64-pc-linux-gnu, compiled by gcc (GCC) 4.8.5
20150623 (Red Hat 4.8.5-44), 64-bit
2021-04-24 14:54:28.833 UTC [23138] LOG:  listening on IPv4 address
"0.0.0.0", port 5433
2021-04-24 14:54:28.833 UTC [23138] LOG:  listening on IPv6 address "::",
port 5433
2021-04-24 14:54:28.836 UTC [23138] LOG:  listening on Unix socket
"/var/run/postgresql/.s.PGSQL.5433"
2021-04-24 14:54:28.848 UTC [23138] LOG:  listening on Unix socket
"/tmp/.s.PGSQL.5433"
2021-04-24 14:54:28.866 UTC [23138] LOG:  redirecting log output to logging
collector process
2021-04-24 14:54:28.866 UTC [23138] HINT:  Future log output will appear in
directory "log".
 done
server started
```

4. Выполните внутри транзакции запрос с подсчетом количества записей в таблицах `bookings.ticket_flights`, `bookings.tickets`, `bookings.flights` и паузой в 120 секунд;

```
postgres$ psql -p5433 demo
psql (12.6)
```

```
Type "help" for help.

demo=# do $$
begin
  perform count(*) from bookings.ticket_flights;
  perform count(*) from bookings.tickets;
  perform count(*) from bookings.flights;
  perform pg_sleep(120);
end;
$$;

DO
```

5. Дождитесь сбора метрик в pgwatch2 и убедитесь, что репликация у вас начала запаздывать.

В ДРУГОЙ СЕССИИ !!!

```
demo$ psql
psql (12.6)
Type "help" for help.

demo=# \x
Expanded display is on.
demo=# select * from pg_stat_replication ;
-[ RECORD 1 ]----+------------------------------
pid              | 28087
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-24 15:03:41.419127+00
backend_xmin     |
state            | streaming
sent_lsn         | 2/16FA3478
write_lsn        | 2/16FA3478
flush_lsn        | 2/16FA3478
replay_lsn       | 2/16E559B8
write_lag        | 00:00:00.000148
flush_lag        | 00:00:00.002821
replay_lag       | 00:00:00.937678
sync_priority    | 0
sync_state       | async
reply_time       | 2021-04-24 15:04:26.274246+00


demo=# select * from pg_stat_replication ;
-[ RECORD 1 ]----+------------------------------
pid              | 28087
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-24 15:03:41.419127+00
backend_xmin     |
state            | streaming
sent_lsn         | 2/2F080000
write_lsn        | 2/2F000000
```

```
flush_lsn         | 2/2F000000
replay_lsn        | 2/16E559B8
write_lag         | 00:00:11.237752
flush_lag         | 00:00:11.237752
replay_lag        | 00:01:19.455937
sync_priority     | 0
sync_state        | async
reply_time        | 2021-04-24 15:05:44.78153+00

demo=# select * from pg_stat_replication ;
-[ RECORD 1 ]----+------------------------------
pid              | 28087
usesysid         | 10
usename          | postgres
application_name | walreceiver
client_addr      |
client_hostname  |
client_port      | -1
backend_start    | 2021-04-24 15:03:41.419127+00
backend_xmin     |
state            | streaming
sent_lsn         | 2/305C0000
write_lsn        | 2/30560000
flush_lsn        | 2/30520000
replay_lsn       | 2/16E559B8
write_lag        | 00:00:11.638886
flush_lag        | 00:00:11.64735
replay_lag       | 00:01:21.058311
sync_priority    | 0
sync_state       | async
reply_time       | 2021-04-24 15:05:46.394854+00

demo=#
```

ЗАМЕЧАНИЕ Если на графиках pgwatch2 нет запаздывания, то можно выполнить следующие действия:

1. Нужно увеличить число секунд в параметре `pg_watch2`
2. На лидере выполнить дополнительную команду обвноления данных, например update `bookings.tickets set ticket_no = ticket_no;`