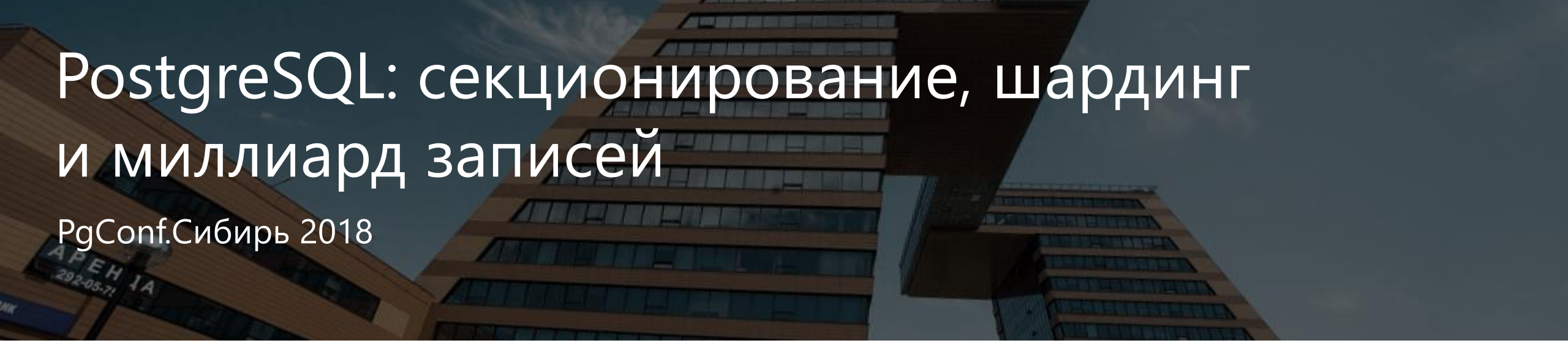


PostgreSQL: секционирование, шардинг и миллиард записей

PgConf.Сибирь 2018

Обо мне

Фадеев Алексей Сергеевич

C# разработчик

Компания SibEDGE

Сфера деятельности

Backend-разработка

Работа с СУБД, оптимизация

Используем: PostgreSQL, Microsoft SQL Server


see beyond
THE TOP
KOMTAR PENANG
PG
PENANG STATE GALL

Прототип проекта

От нескольких систем мониторинга периодически собираются значения параметров и пишутся в БД.

Количество параметров: 500000

Секционирование в PostgreSQL 10

```
CREATE TABLE test (id int not null, val real)
PARTITION BY RANGE (id);
```

Нельзя сделать секционированной уже существующую таблицу

Изменение границ секций: сложно

Проблемы с индексами

Блокирующие операции

Возможности pg_pathman

HASH и RANGE секционирование

Оптимизации планирования и исполнения

PartitionFilter вместо триггеров (быстрый INSERT)

Автоматическое создание секций при вставке

Пользовательские колбеки на создание секций

Поддержка FDW-партиций

Неблокирующее секционирование

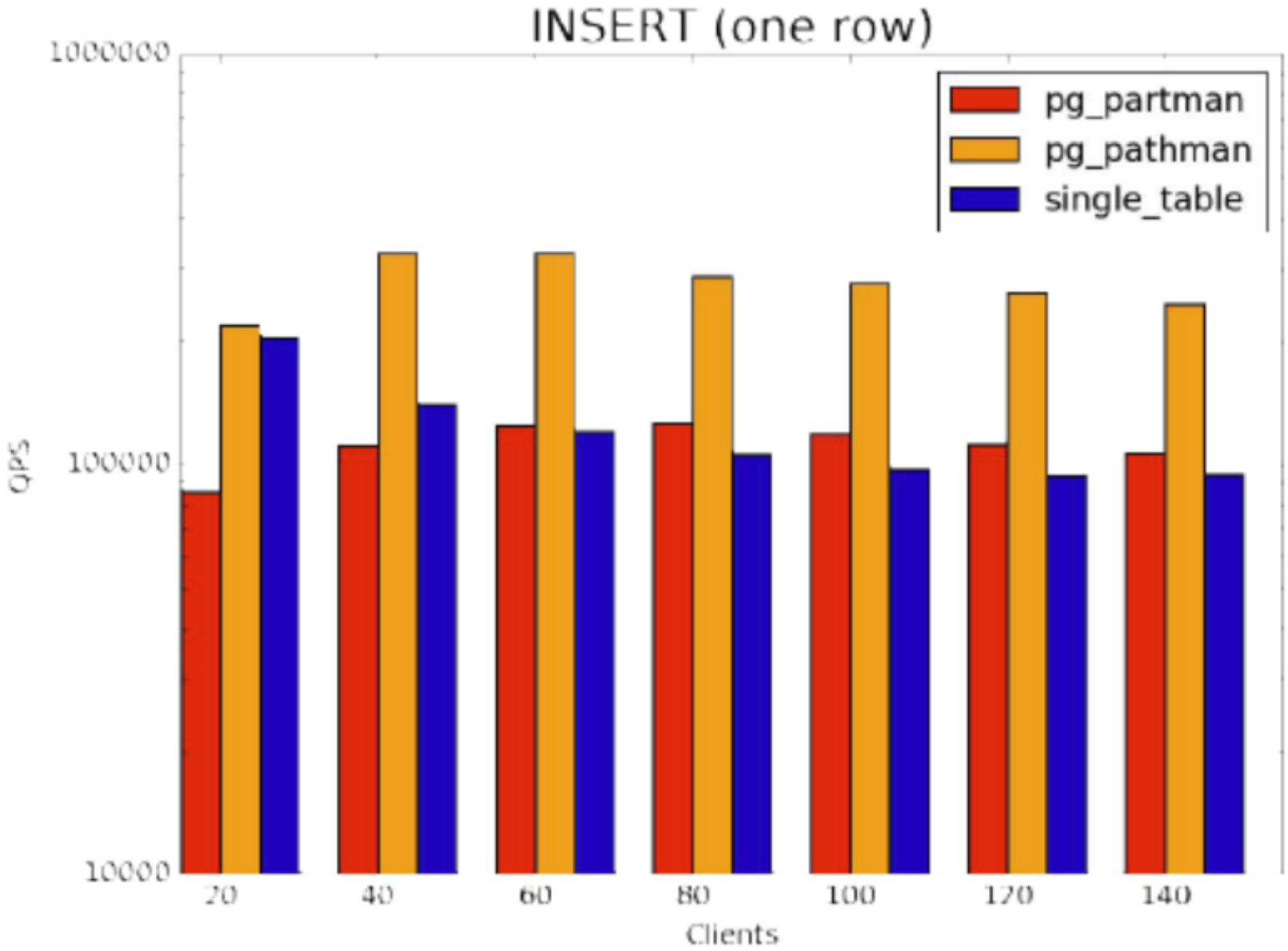
Удобный API

Добавим секционирование по дате-времени

```
SELECT create_range_partitions('parameter_value',
    'datetime_event', '2018-04-01'::timestamp, '1 day'::interval, 10)
```

```
SELECT * FROM pathman_partition_list
```

parent regclass	partition regclass	parttype integer	expr text	range_min text	range_max text
parameter_value	parameter_value_1	2	datetime_event	2018-04-01 00:00:00	2018-04-02 00:00:00
parameter_value	parameter_value_2	2	datetime_event	2018-04-02 00:00:00	2018-04-03 00:00:00
parameter_value	parameter_value_3	2	datetime_event	2018-04-03 00:00:00	2018-04-04 00:00:00
parameter_value	parameter_value_4	2	datetime_event	2018-04-04 00:00:00	2018-04-05 00:00:00
parameter_value	parameter_value_5	2	datetime_event	2018-04-05 00:00:00	2018-04-06 00:00:00
parameter_value	parameter_value_6	2	datetime_event	2018-04-06 00:00:00	2018-04-07 00:00:00
parameter_value	parameter_value_7	2	datetime_event	2018-04-07 00:00:00	2018-04-08 00:00:00
parameter_value	parameter_value_8	2	datetime_event	2018-04-08 00:00:00	2018-04-09 00:00:00
parameter_value	parameter_value_9	2	datetime_event	2018-04-09 00:00:00	2018-04-10 00:00:00
parameter_value	parameter_value_10	2	datetime_event	2018-04-10 00:00:00	2018-04-11 00:00:00


INSERT (one row)
1000000
100000
10000
QPS
pg_partman
pg_pathman
single_table
20
40
60
80
100
120
140
Clients

Вставка данных

Multi-insert, по 20000 записей за 1 запрос, всего 1 млн.

	Time
Single table	38 с
1 partition	39 с
30 partitions	37 с
30 partitions ordered	34 с

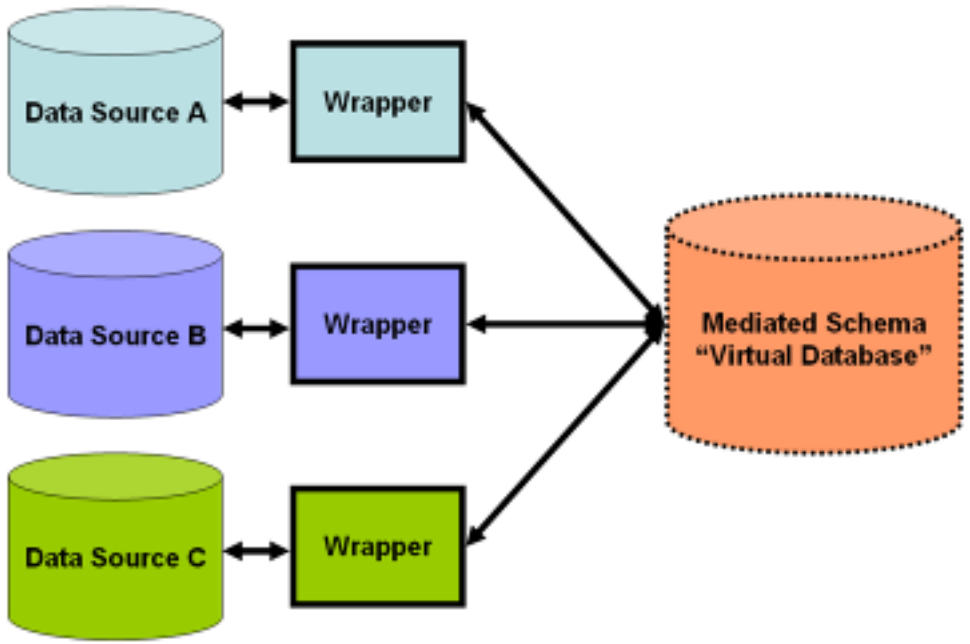
Прототип проекта

От нескольких систем мониторинга периодически собираются значения параметров и пишутся в БД.

Количество систем мониторинга: 5 (пронумерованы, поле **system_id**)

Количество параметров: 500000

Foreign Data Wrapper



With PostgreSQL

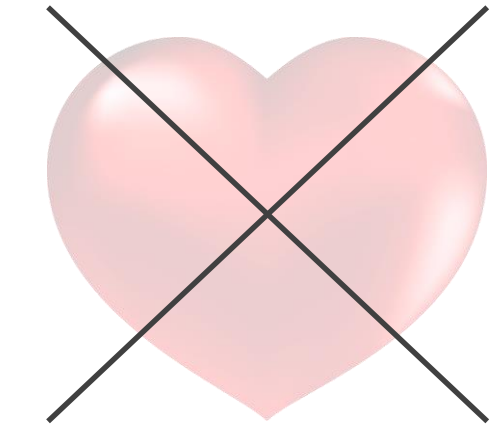
pg_pathman + FDW

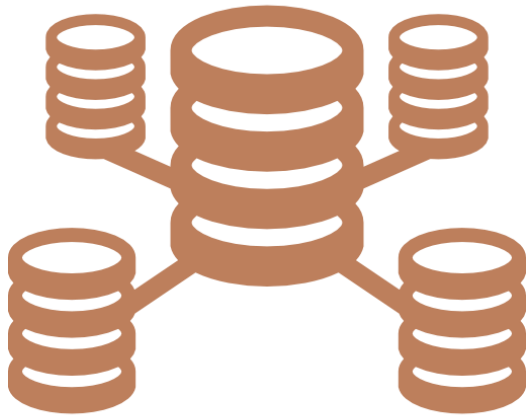
pg_pathman + FDW

= 

pg_pathman + FDW

=



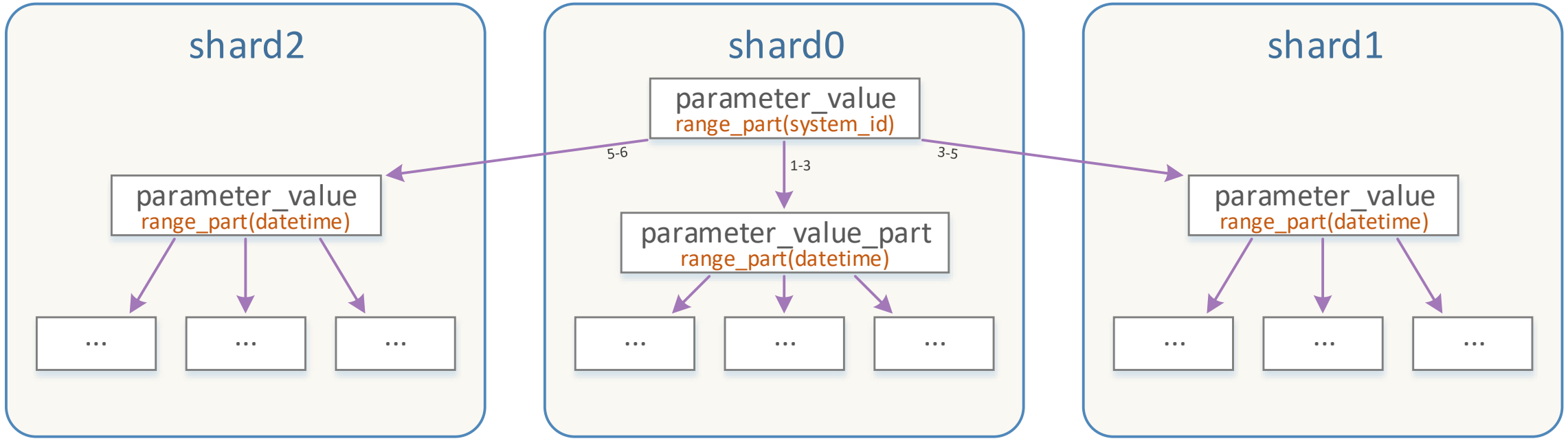


```
CREATE SERVER shard1
        FOREIGN DATA WRAPPER postgres_fdw
        OPTIONS (host '192.168.33.86', port '5432', dbname 'cluster');


CREATE FOREIGN TABLE parameter_value_shard1 (
  system_id integer NOT NULL,
  parameter_id integer NOT NULL,
  datetime_event timestamp without time zone NOT NULL,
  datetime_store timestamp without time zone NOT NULL,
  value real NOT NULL)
SERVER shard1
OPTIONS (schema_name 'public', table_name 'parameter_value');
```

```
SELECT create_range_partitions('parameter_value', 'system_id', 1, 2, 1);

SELECT attach_range_partition('parameter_value', 'parameter_value_part', 1, 3);
SELECT attach_range_partition('parameter_value', 'parameter_value_shard1', 3, 5);
SELECT attach_range_partition('parameter_value', 'parameter_value_shard2', 5, 6);
```



Каскадное секционирование

Напрямую pg_pathman не поддерживает

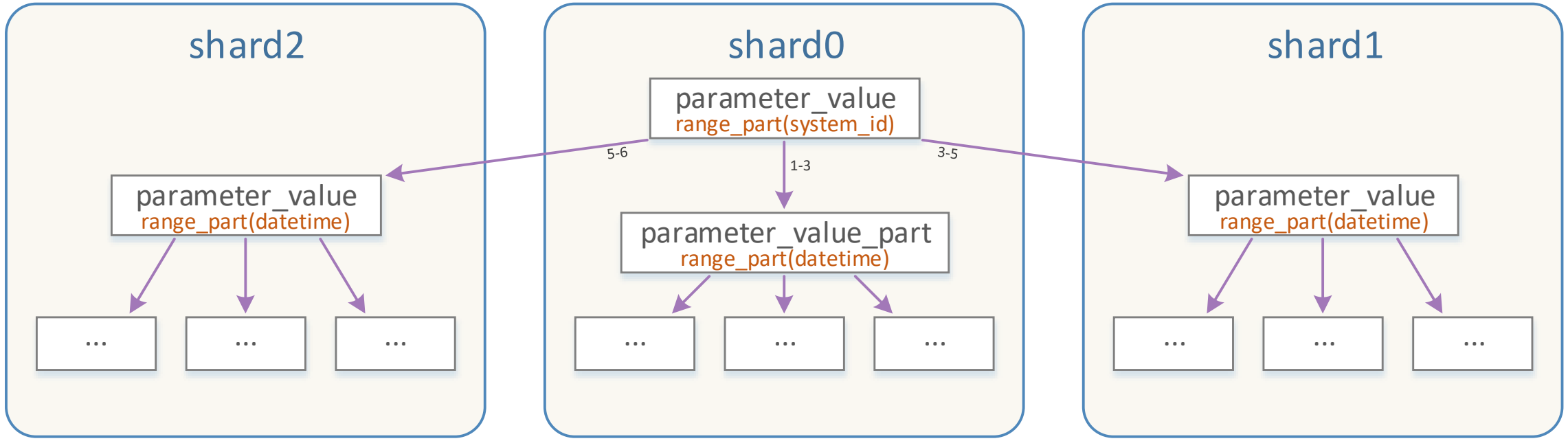
Каскадное секционирование

Напрямую pg_pathman не поддерживает

Хак: создаём FDW для текущей БД

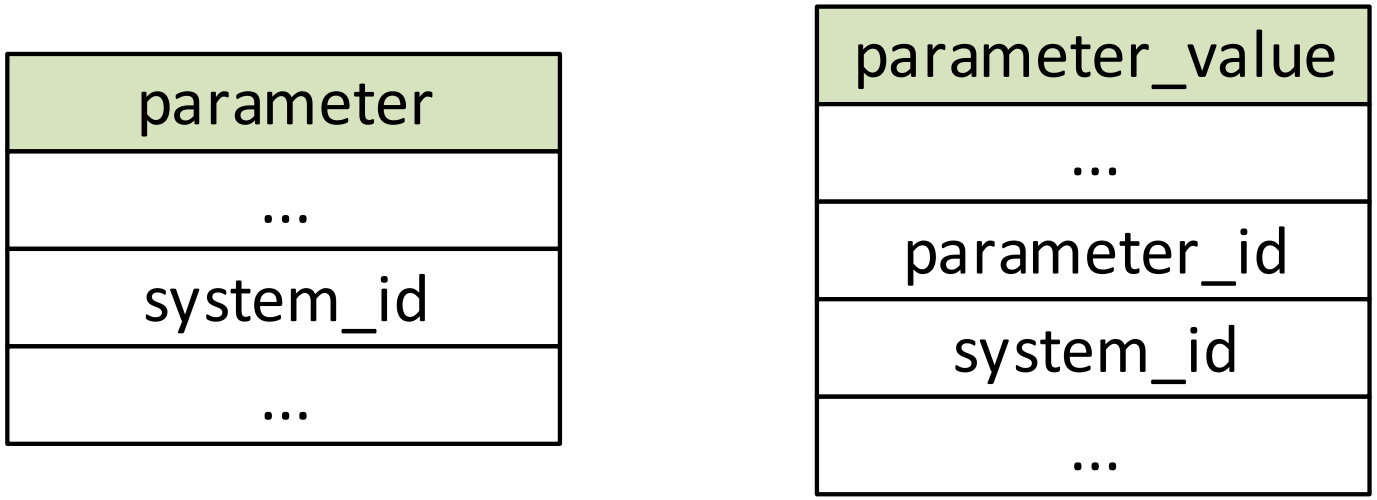
```
SELECT create_range_partitions('parameter_value', 'system_id', 1, 2, 1);

SELECT attach_range_partition('parameter_value', 'parameter_value_shard0', 1, 3);
SELECT attach_range_partition('parameter_value', 'parameter_value_shard1', 3, 5);
SELECT attach_range_partition('parameter_value', 'parameter_value_shard2', 5, 6);
```



Денормализация

Добавляем **поле system_id** в таблицу значений параметров

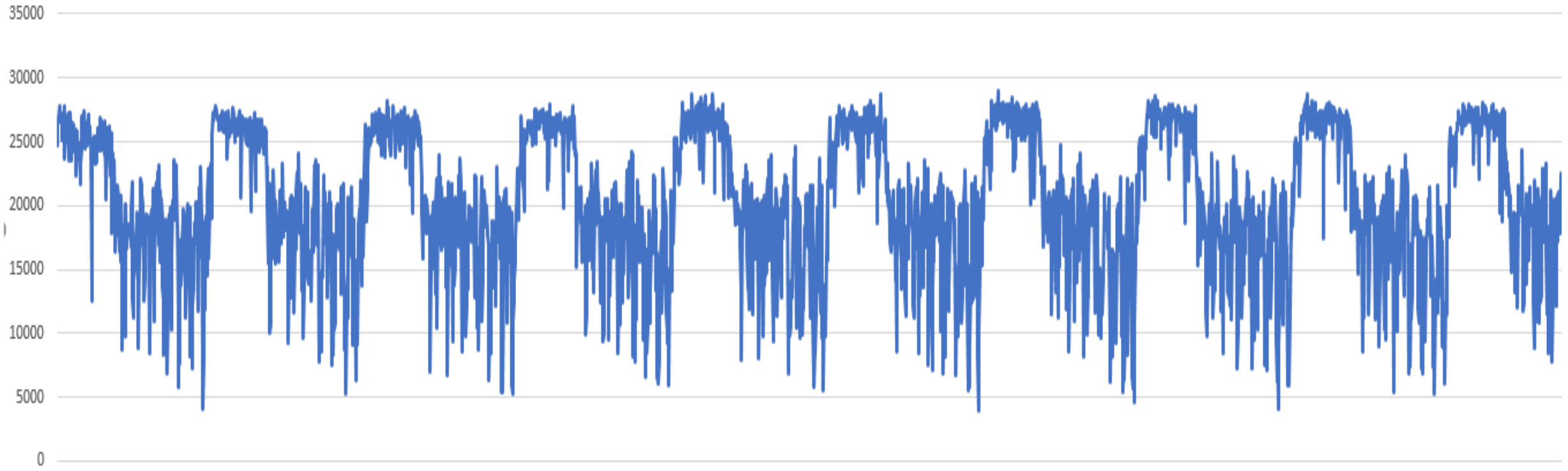


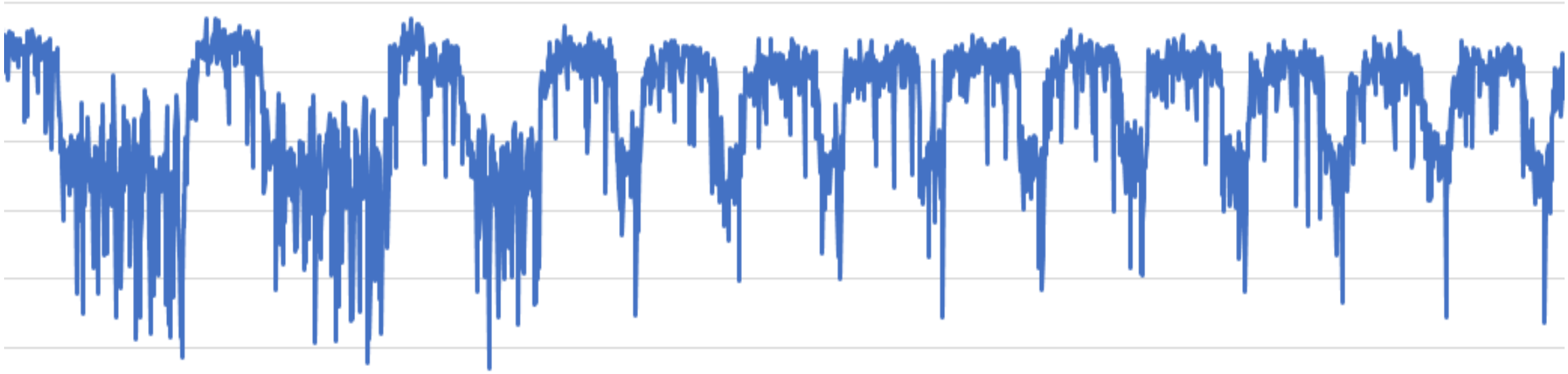
Вставка данных

Multi-insert, 20000 записей за 1 запрос

Главная БД, INSERT INTO parameter_value	14-15с
Главная БД, INSERT INTO parameter_value_shard1	14-15с
Прямое подключение к БД shard1, INSERT INTO parameter_value	1.8-2.2с

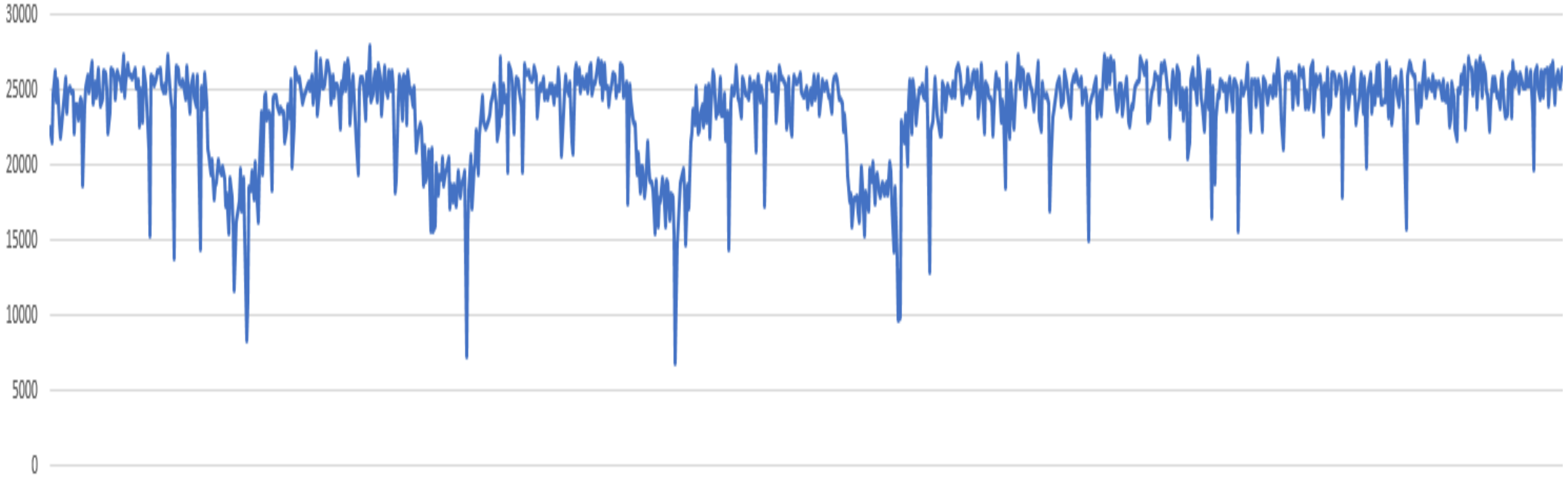
Каждую итерацию генерируется 150-155 тыс. записей,
в пропорции примерно 15/13/9 (shard0/shard1/shard2)





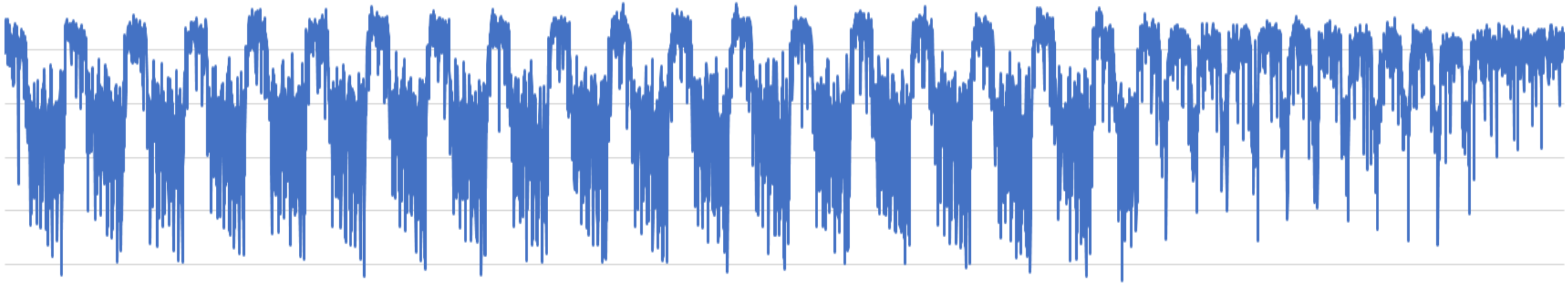
Секция: ~18.14 млн

Секция: ~9.08 млн



Секция: ~9.08 млн

Секция: ~6.42 млн



Всего записей: 1 231.4 млн

Конфигурация:
shard0 – физическая машина, Windows 10 x64, SSD
shard1 – виртуальная машина, Windows 10 x64
shard2 – физическая машина, Debian 7, HDD

На одной ноде

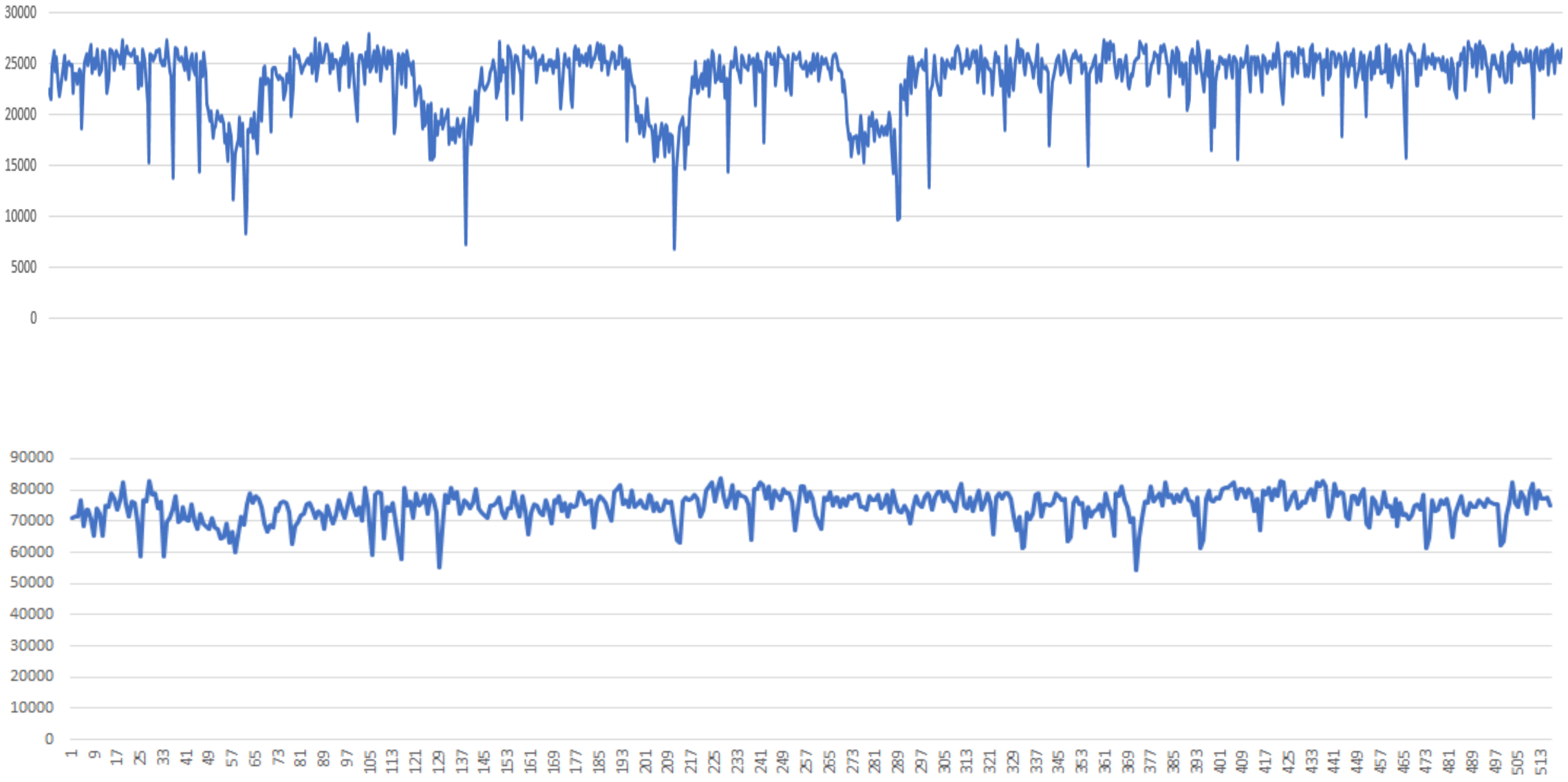
Скорость записи не падает

Высокая скорость SELECT-запросов

Преимущества шардинга

Сложные запросы: выполняются параллельно на узлах кластера

Параллельная запись


30000
25000
20000
15000
10000
5000
0
90000
80000
70000
60000
50000
40000
30000
20000
10000
0
1
9
17
25
33
41
49
57
65
73
81
89
97
105
113
121
129
137
145
153
161
169
177
185
193
201
209
217
225
233
241
249
257
265
273
281
289
297
305
313
321
329
337
345
353
361
369
377
385
393
401
409
417
425
433
441
449
457
465
473
481
489
497
505
513

Преимущества кластера

Параллельная вставка в ноды кластера: кратное увеличение скорости записи

Если будет очень много запросов – меньше нагрузка

Меньше секций на каждой ноде

Меньше места на диске на каждой ноды

Можно вынести архивные данные на отдельный узел

SELECT

На всех таблицах созданы индексы по полям
parameter_id и **datetime_event**

```
SELECT * FROM parameter_value
  WHERE parameter_id=86
    AND system_id=3
    AND datetime_event BETWEEN '2018-04-15 03:00:00'
                       AND '2018-04-20 03:00:00'
```

Время выполнения: 30-40 мс

```
EXPLAIN ANALYZE SELECT COUNT(*) FROM parameter_value
  WHERE system_id=3 AND parameter_id<1000
```

```
Aggregate  (cost=161.33..161.34 rows=1 width=8) (actual time=3704.307..3704.307 rows=1 loops=1)
 ->  Append  (cost=100.00..161.32 rows=6 width=0) (actual time=12.251..3672.906 rows=504581 loops=1)
       ->  Foreign Scan on parameter_value_shard1  (cost=100.00..161.32 rows=6 width=0) (actual
             time=12.250..3646.413 rows=504581 loops=1)
Planning time: 0.217 ms
Execution time: 3705.927 ms
```

```
EXPLAIN ANALYZE SELECT COUNT(*) FROM parameter_value_shard1
  WHERE system_id=3 AND parameter_id<1000
```

```
Foreign Scan  (cost=100.02..161.24 rows=1 width=8) (actual time=430.953..430.954 rows=1 loops=1)
  Relations: Aggregate on (public.parameter_value_shard1)
Planning time: 19.221 ms
Execution time: 501.327 ms
```

Postgres можно использовать:

Big data

Горизонтальное масштабирование

Postgres или NoSQL?

Контакты

FadeevAS@sibedge.com

tomsklab@yandex.ru

https://vk.com/fadeev

https://www.facebook.com/alexey.fadeev.3745

ВСЁ!