Разгоняем железо и операционную систему на максимальную производительность. Бенчмаркаем на PostgreSQL

Михаил Жилин

Postgres Professional

m.zhilin@postgrespro.ru

Михаил Жилин

@mizhka

Bio

- 15 лет в нагрузочном тестировании
- Руководитель группы производительности Postgres Professional
- Контрибьютор в Open Source проекты

О чём разговор?

- О бенчмарках СУБД, в частности PostgreSQL
- О проблемах бенчмарков
- Научимся решать проблемы оптимизацией железа

О чём НЕ БУДЕТ разговор?

- О бенчмарках OS, железа
- Об обработке результатов и математики
- О методиках и параметрах тестирования

Зачем запустить бенчмарк?

- Какую базу лучше выбрать для X?
- А вдруг ляжет продакшен?
- А точно новая версия лучше/не хуже? И насколько?
- Оценить маркетинг материал
- Поменяли параметр X

Как запустить бенчмарк?

- Найти в интернете
- Создать виртуалку, устанавливать СУБД
- Настроить бенчмарк, сгенерить данные
- Запустить, подождать и…
- Получить профит!

Как подготовиться?

[1] HammerDB TPC-C: Поехали

- Order Entry Benchmark
  - https://www.tpc.org/tpcc/
  - В базе есть склады, районы и т.п.
  - Создаются ордера

[1] HammerDB TPC-C: Поехали

- Оборудование
  - 8 ядер, 16 GiB памяти, 100 GB диск
- Объём базы
  - 10 складов (20GiB)
- Нагрузка
  - 20 потоков
- Запуски на 10 минут

[1] HammerDB TPC-C: Поехали

- 3 запуска

  120,000 - 130,000 ордеров в минуту

[1] HammerDB TPC-C: Поехали

- 3 запуска

  120,000 - 130,000 ордеров в минуту

- Ещё 3 запуска

  80,000 - 90,000 ордеров в минуту

[1] HammerDB TPC-C: Поехали

- 3 запуска

  120,000 - 130,000 ордеров в минуту

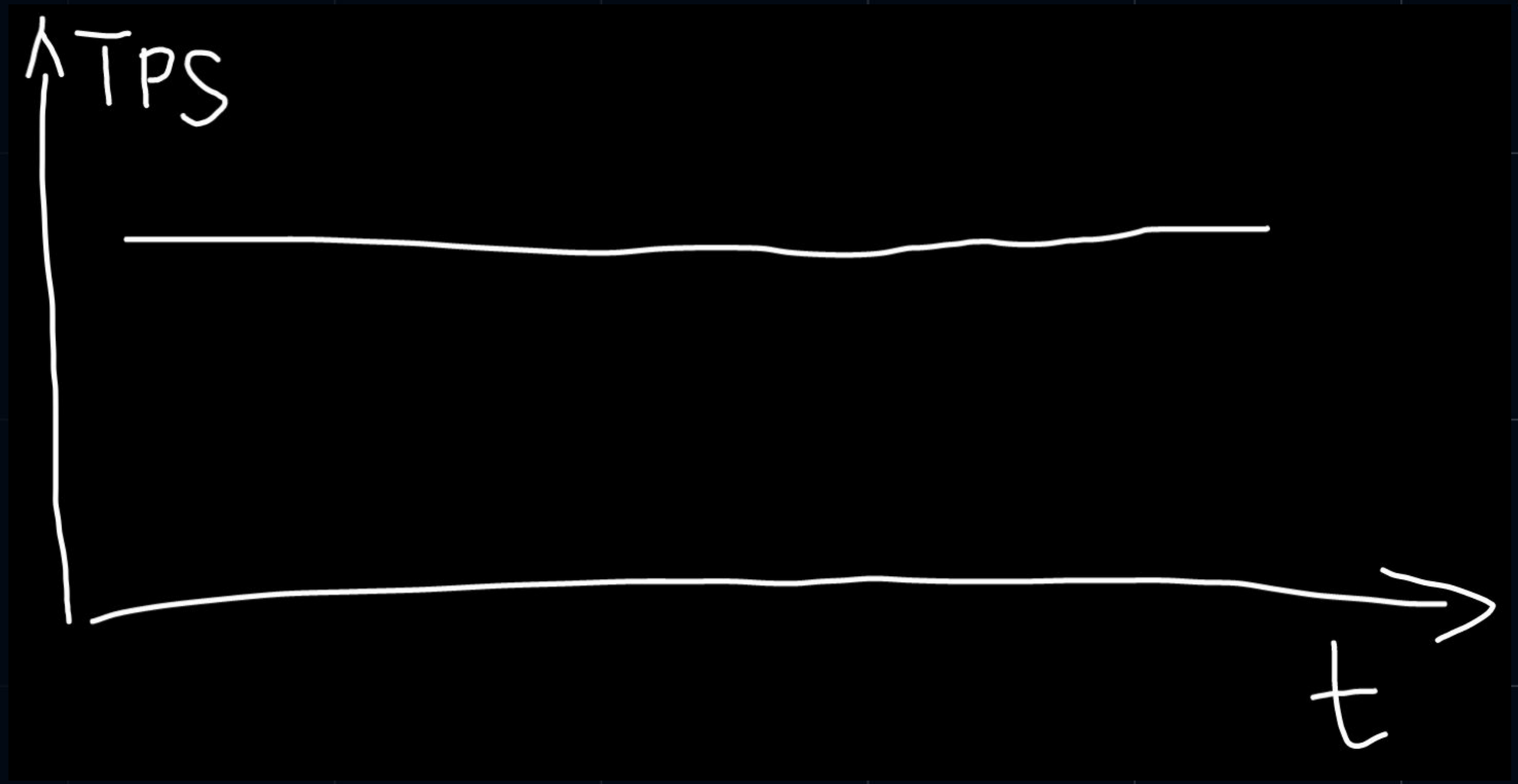
- Ещё 3 запуска

  80,000 - 90,000 ордеров в минуту

- Ещё 3 запуска

  15,000 ордеров в минуту

[1] HammerDB TPC-C: Поехали



[1] HammerDB TPC-C: Приехали

[1] HammerDB TPC-C: Запрос

```
-- 160ms
SELECT COUNT(DISTINCT (s_i_id))
  FROM order_line, stock, district
 WHERE ol_w_id = 10
   AND ol_d_id = 8
   AND d_w_id= 10
   AND d_id = 8
   -- 20 последних ордеров
   AND (ol_o_id < d_next_o_id)
   AND ol_o_id >= (d_next_o_id - 20)
   AND s_w_id = 10
   AND s_i_id = ol_i_id
   AND s_quantity < 100000;
```

[1] HammerDB TPC-C: Запрос

- **Таблица district**
  - 10 строк на 1 склад, фиксированное число
- **Таблица stock**
  - 100,000 строк на 1 склад, фиксированное число
- **Таблица order_line**
  - 10-15 строк на каждый ордер!
  - В начале теста - 0, в конце - 130 миллионов!!!

https://bit.ly/3KSY7mi

[1] HammerDB TPC-C: Запрос

- Запрос проверки уровня stock-ов
- Пункт 2.8.2.2 спецификации TPC-C

  https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf

2.8 The Stock-Level Transaction

The Stock-Level business transaction determines the number of recently sold items that have a stock level below a specified threshold. It represents a heavy read-only database transaction with a low frequency of execution, a relaxed response time requirement, and relaxed consistency requirements.

2.8.1 Input Data Generation

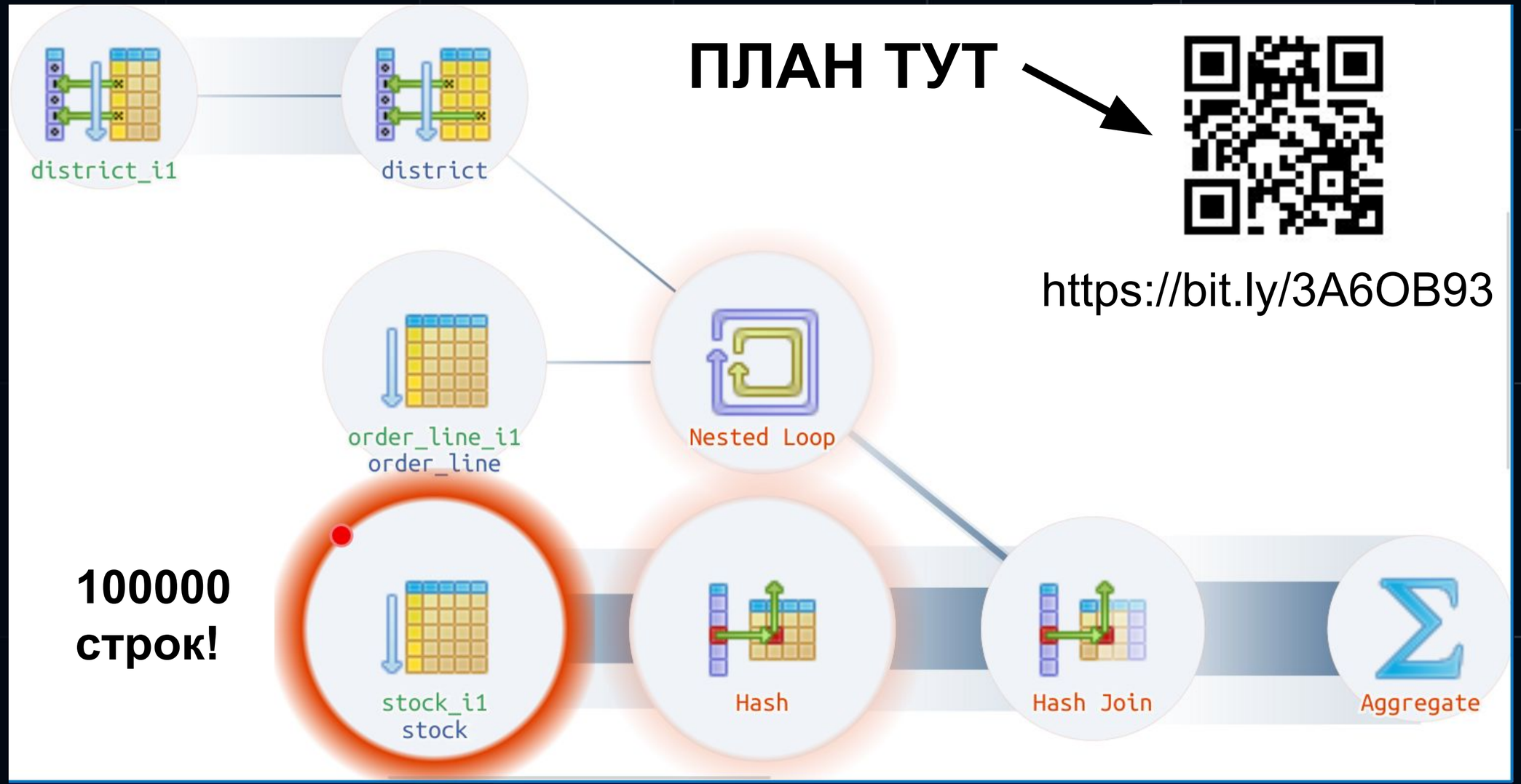
2.8.1.1 Each terminal must use a unique value of (W_ID, D_ID) that is constant over the whole measurement, i.e., D_IDs cannot be re-used within a warehouse.

2.8.1.2 The threshold of minimum quantity in stock (threshold) is selected at random within [10 .. 20].

[1] HammerDB TPC-C: Запрос

Почему тормозит запрос?

[1] HammerDB TPC-C: Запрос



ПЛАН ТУТ

https://bit.ly/3A6OB93

100000 строк!

[1] HammerDB TPC-C: Запрос

```
Aggregate
  ->  Hash Join
        Hash Cond: (order_line.ol_i_id = stock.s_i_id)
        ->  Nested Loop
              ->  Bitmap Heap Scan on district
                    Recheck Cond: ((d_w_id = 10) AND (d_id = 8))
                    ->  Bitmap Index Scan on district_i1
                          Index Cond: ((d_w_id = 10) AND (d_id = 8))
              ->  Index Scan using order_line_i1 on order_line
                    Index Cond: ((ol_w_id = 10) AND (ol_d_id = 8) AND (ol_o_id
< district.d_next_o_id) AND (ol_o_id >= (district.d_next_o_id - 20)))
        ->  Hash
              ->  Index Scan using stock_i1 on stock
                    Index Cond: (s_w_id = 10)
                    Filter: (s_quantity < 100000)
```

[1] HammerDB TPC-C: Запрос

```
    (ol_w_id = 10)
AND (ol_d_id = 8)
-- 20 последних ордеров
AND (ol_o_id < district.d_next_o_id)
AND (ol_o_id >= (district.d_next_o_id - 20))
```

All rows in the ORDER-LINE table with matching OL_W_ID (equals W_ID), OL_D_ID (equals D_ID), and OL_O_ID (lower than D_NEXT_O_ID and greater than or equal to D_NEXT_O_ID minus 20) are selected. They are the items for 20 recent orders of the district.

[1] HammerDB TPC-C: Запрос

```
    (ol_w_id = 10)
AND (ol_d_id = 8)
-- 20 последних ордеров
AND (ol_o_id < district.d_next_o_id)
AND (ol_o_id >= (district.d_next_o_id - 20))
```

[1] HammerDB TPC-C: Запрос

```
    (ol_w_id = 10)
AND (ol_d_id = 8)
-- 20 последних ордеров
AND (ol_o_id < district.d_next_o_id)
AND (ol_o_id >= (district.d_next_o_id - 20))
```

[1] HammerDB TPC-C: Запрос

```
    (ol_w_id = 10)
AND (ol_d_id = 8)
-- 20 последних ордеров
AND (ol_o_id < district.d_next_o_id)
AND (ol_o_id >= (district.d_next_o_id - 20))
```

↓

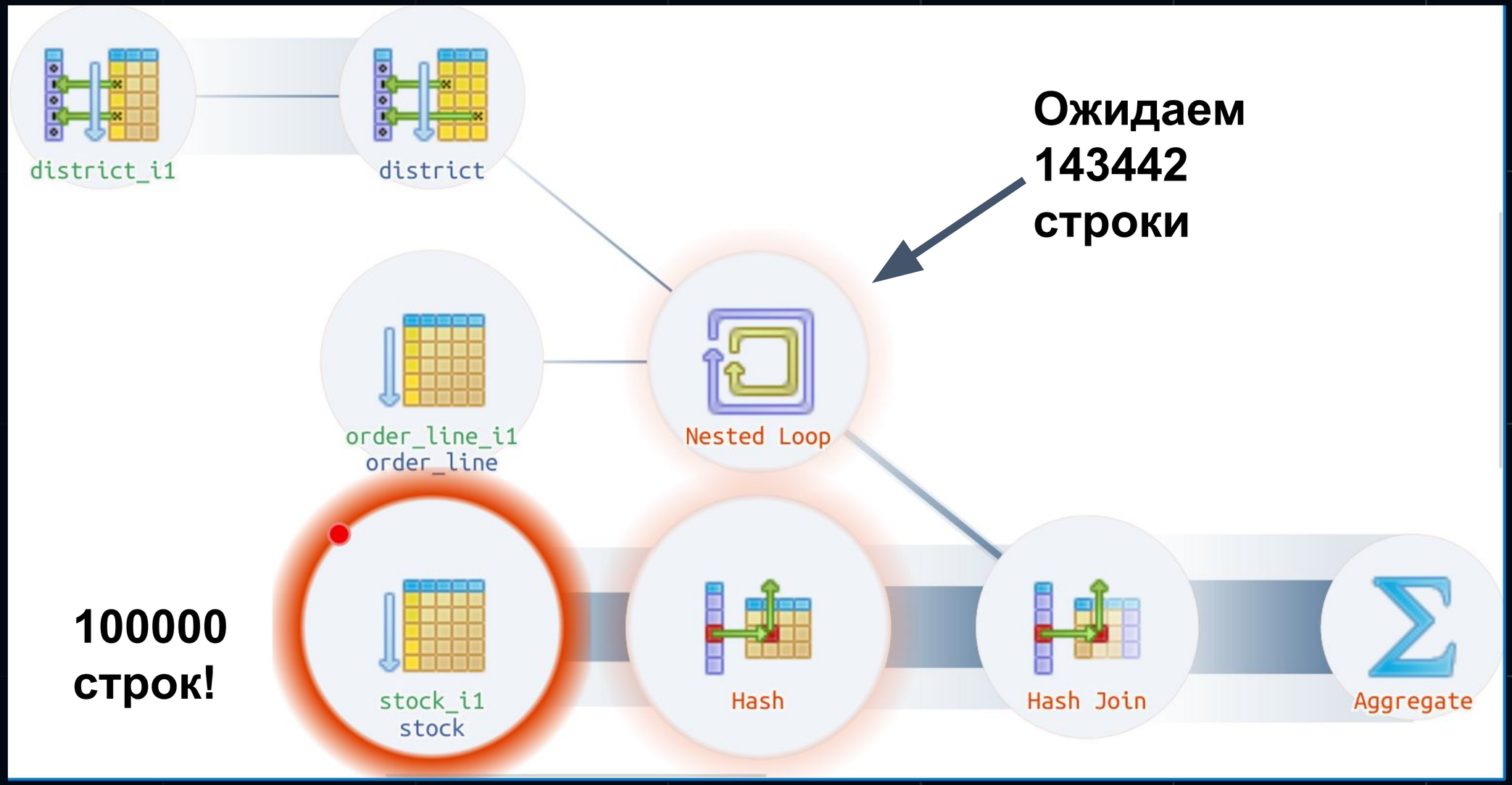
```
ol_o_id >= expression (district.d_next_o_id)
```

[1] HammerDB TPC-C: Запрос

- **Index Scan using order_line_i1 on order_line**
  - Estimated rows=**143,442**
  - Actual rows=**219**

- **Ошибка в оценке числа строк (cardinality)**

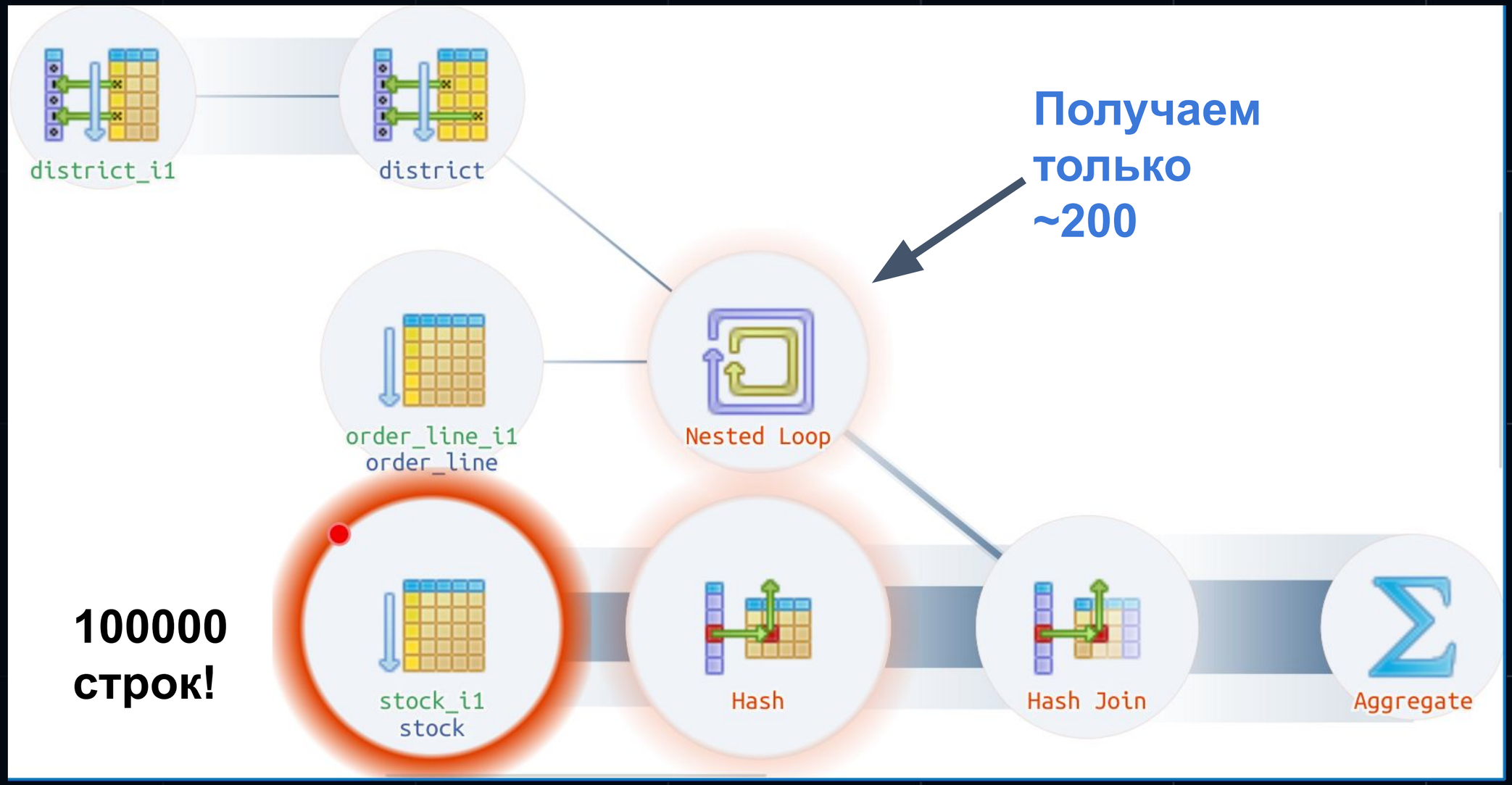
[1] HammerDB TPC-C: Запрос



Ожидаем 143442 строки

100000 строк!

[1] HammerDB TPC-C: Запрос



Получаем только ~200

100000 строк!

[1] HammerDB TPC-C: Запрос

```
SELECT COUNT(DISTINCT (s_i_id))
  FROM order_line, stock, district
 WHERE ol_w_id = 10
   AND ol_d_id = 8
   AND d_w_id= 10
   AND d_id = 8
 -- 20 последних ордеров
   AND (ol_o_id < d_next_o_id)
   AND ol_o_id >= (d_next_o_id - 20)
   AND s_w_id = 10
   AND s_i_id = ol_i_id
   AND s_quantity < 100000; -- 160ms
```

PostgreSQL не умеет оценивать интервалы с неизвестными

[1] HammerDB TPC-C: Запрос

```
-- 2 ms
select COUNT(DISTINCT (s_i_id))
  from district d, stock s, order_line ol
  join lateral (select o.o_id
                 from orders o
               where o.o_w_id = d.d_w_id
                 and o.o_d_id = d.d_id
                 and o.o_id < d.d_next_o_id
                 order by o.o_id desc limit 20) o on (true)
  where d.d_w_id = 10
   and d.d_id = 8
   and ol.ol_w_id = d.d_w_id
   and ol.ol_d_id = d.d_id
   and ol.ol_o_id = o.o_id
   and s_w_id = d.d_w_id
   and s_i_id = ol.ol_i_id
   and s_quantity < 100000;
```

[1] HammerDB TPC-C: Запрос

- **go-tpc**
  - https://github.com/pingcap/go-tpc/blob/master/tpcc/stock_level.go
- 2 запроса вместо 1
  - ```
    SELECT d_next_o_id
      FROM district
     WHERE d_w_id = ? AND d_id = ?
    ```
  - ```
    SELECT …
     WHERE ol_o_id < ?
       AND ol_o_id >= ? - 20 …
    ```

[1] HammerDB TPC-C: Выводы

- Не доверяйте именитым бенчмаркам!
- Надо анализировать запуски
- Идеальный бенчмарк - тот что написан вами под ваше решение

Как подготовиться?

[2] Simple Update

- `update t set c1 = ?, c2 = ? where id = ?`
- Нет ошибок планировщика в отличии от HammerDB TPC-C
- Простая реализация на JMeter-е
- Цель - число запросов в секунду

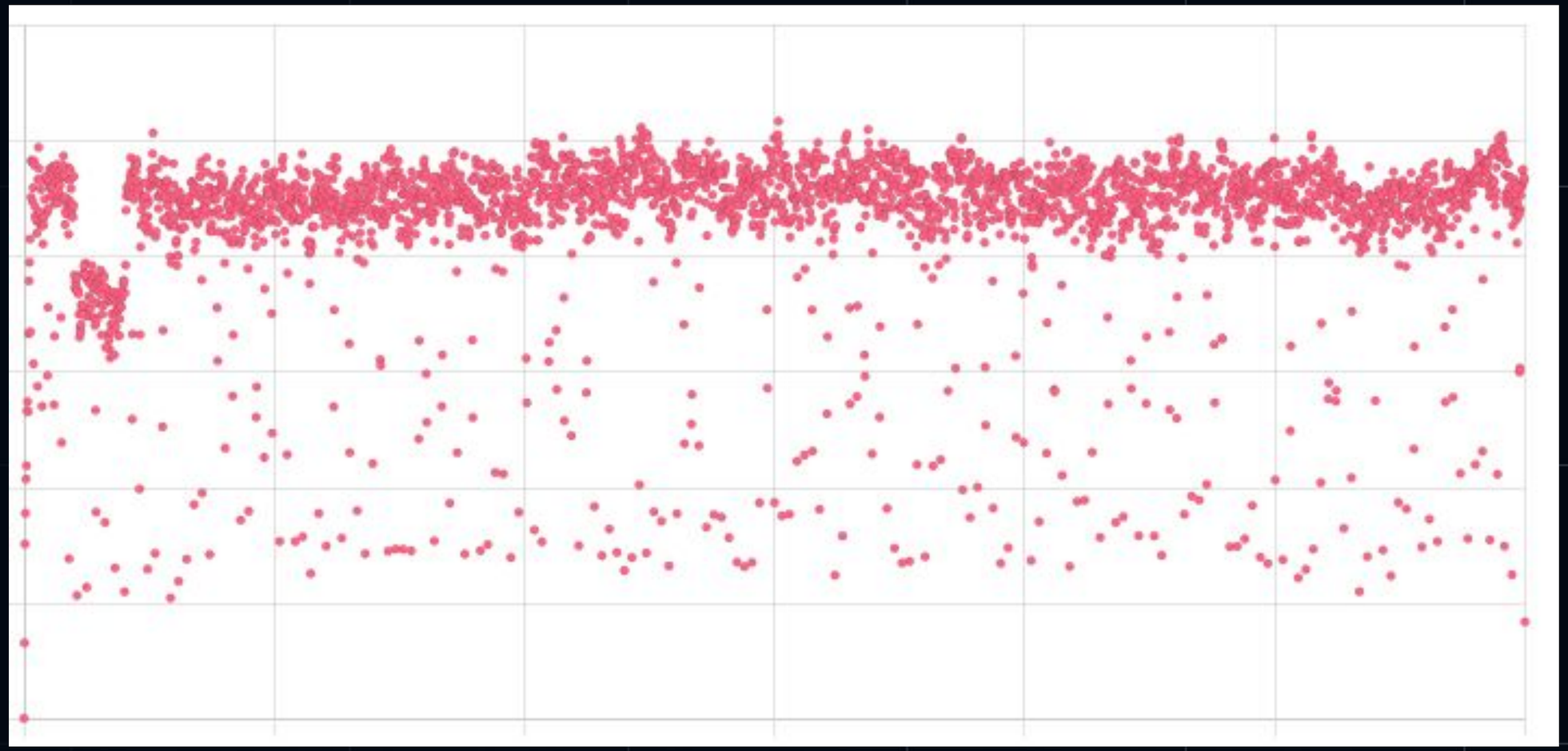
[2] Simple Update

- Оборудование
  - 16 ядер, много памяти и диска
- Объём базы
  - несколько гигабайт
- Нагрузка
  - 10 потоков JMeter-а на 5 минут
- JMeter выдал 43051 запрос в секунду! Круто!

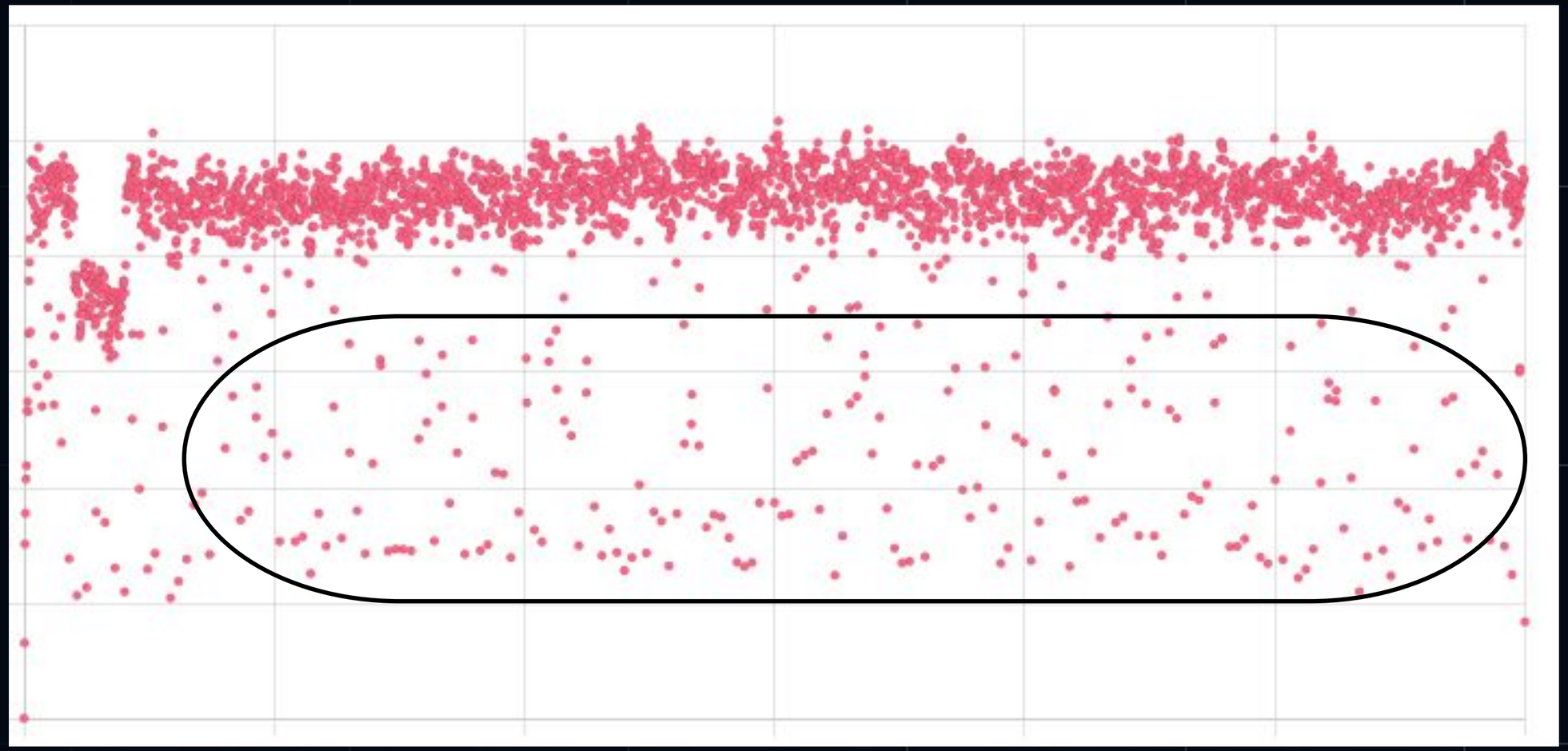
[2] Simple Update

TPS (Time) ?

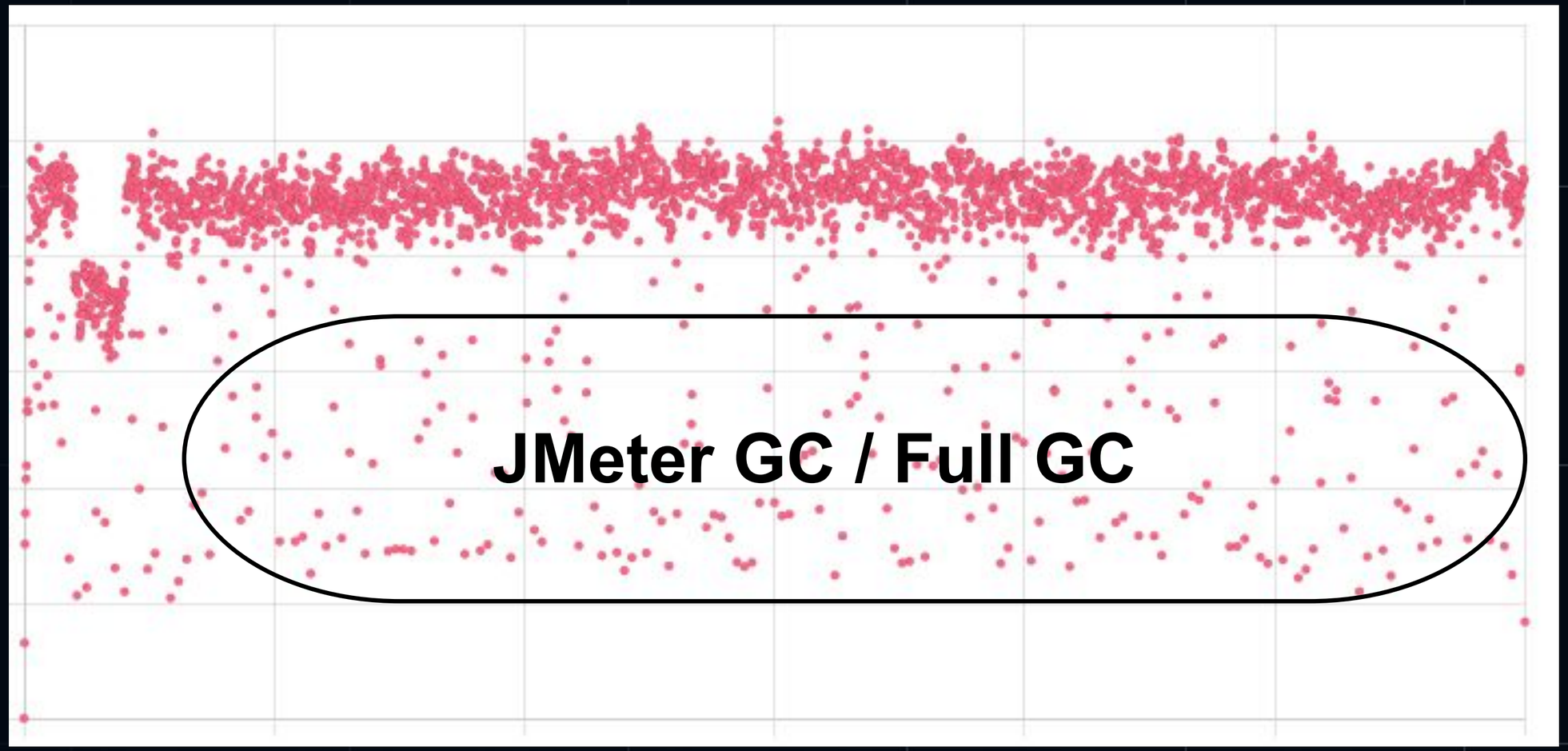
[2] Simple Update



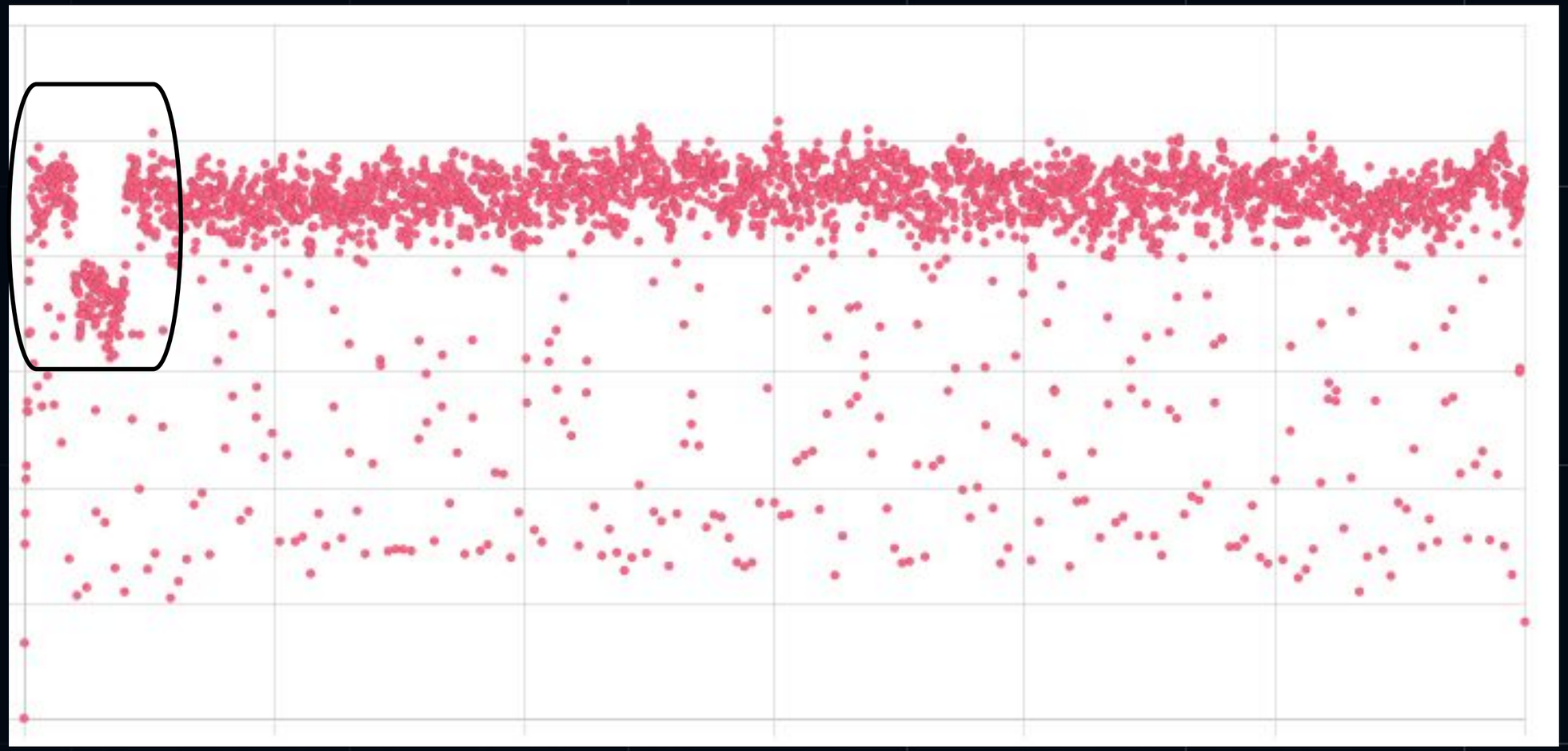
[2] Simple Update



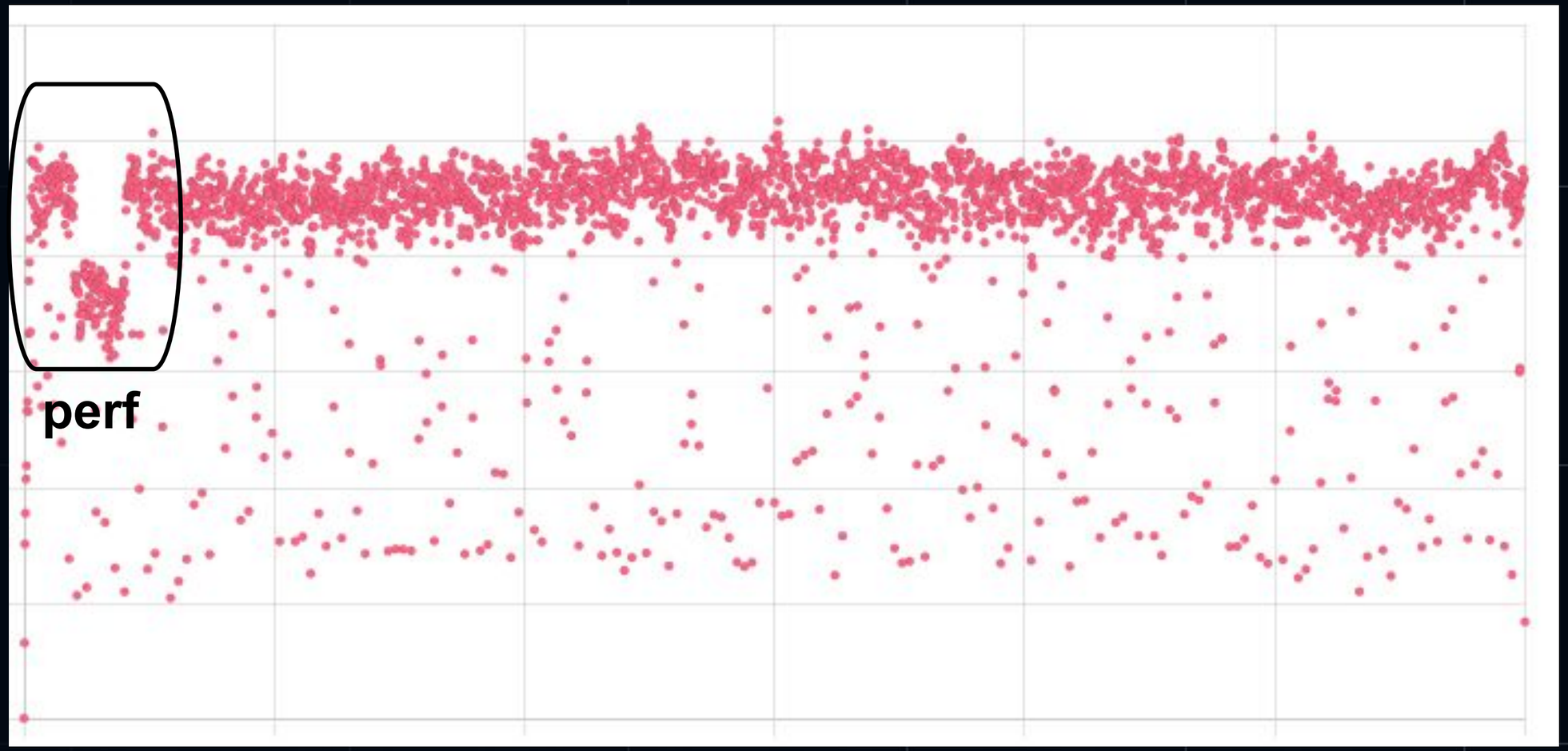
[2] Simple Update



[2] Simple Update



[2] Simple Update



[2] Simple Update

Чему равен TPS ?

[2] Simple Update

- **Андрей Акиньшин**

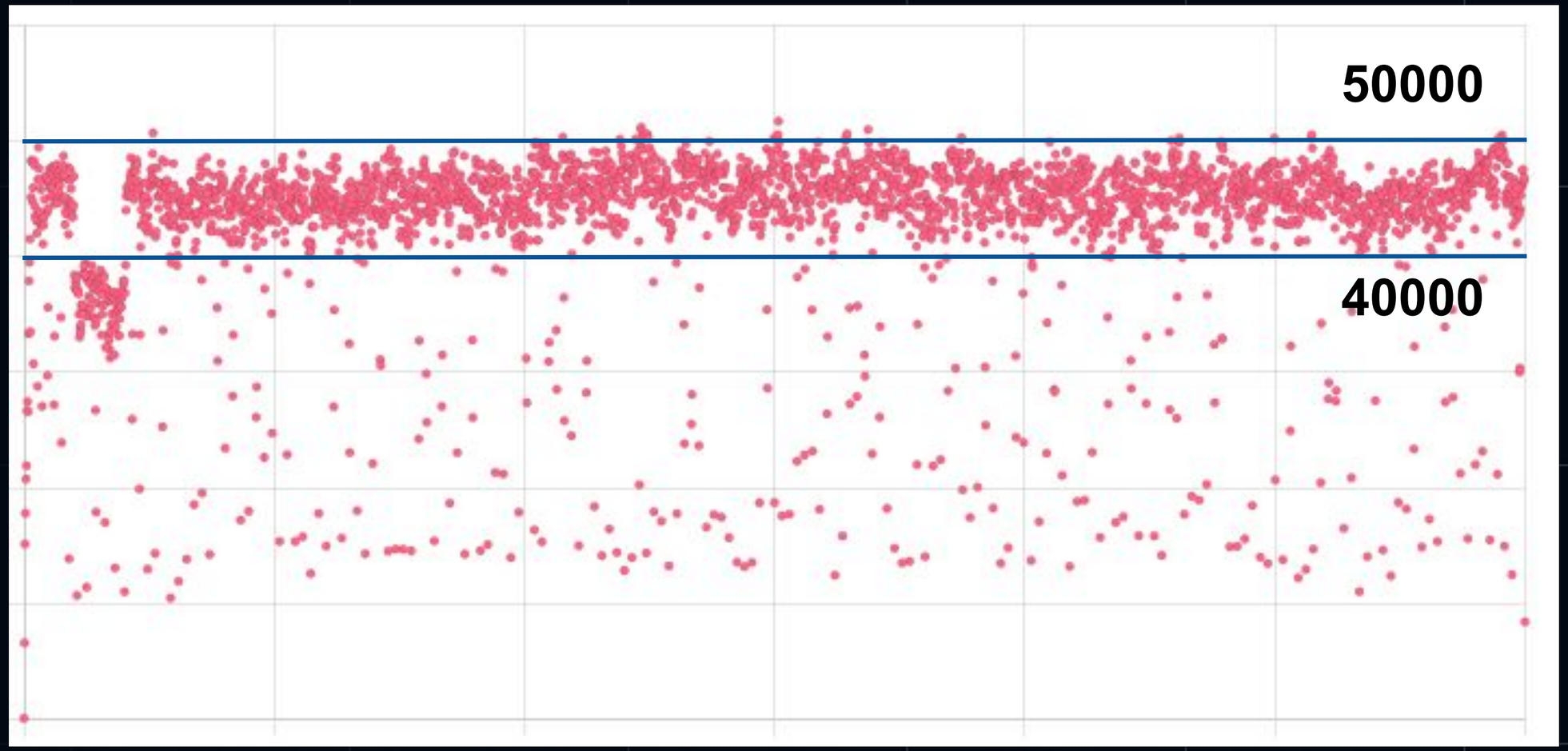
  “Описательная статистика

  перформанс-распределений”

Perfolizer
Performance analysis toolkit

- https://heisenbug.ru/talks/737c5ad2c665484d8504b745ef19a607/
- https://habr.com/ru/companies/jugru/articles/722342/

[2] Simple Update



[2] Simple Update

[2] Simple Update

Что делать ?

[2] Simple Update

Тюнить железо и OS

[2] Simple Update: Железо

- По умолчанию OS+HW настроены под Power Saving
- Настройка Intel платформы под MaxPerformance
  - Спасибо Михаилу Цветкову и Михаилу Синявину

[2] Simple Update: Железо

- BIOS
  - Отключаем энергосбережение
  - Turbo Mode = ON
  - FAN, CPU, etc: MaxPerformance

[2] Simple Update: Железо

- turbostat - аналог vmstat, iostat, mpstat
  - проверка частоты по ядрам
  - %time в C-State (0, 1, 1E)

[2] Simple Update: Железо

- Команда “cpupower frequency-info”
- driver: intel_pstate
  - GRUB_CMDLINE_LINUX_DEFAULT: intel_pstate=enable
- governor: performance
  - /etc/default/cpufrequtils: GOVERNOR="performance"

[2] Simple Update: Железо

- CPU C-State: C0 - running, C6 - sleep/idle
- На постоянку
  - GRUB: processor.max_cstate=1 idle=poll
- На лету
  - echo 1 > /dev/cpu_dma_latency

[2] Simple Update: Железо

- Уменьшаем частоту миграции процессов
  - kernel.sched_migration_cost_ns=50000000
- Выключаем desktop оптимизации
  - kernel.sched_autogroup_enabled=0
- Прочие
  - kernel.sched_min_granularity_ns=100000000
  - kernel.sched_wakeup_granularity_ns=10000000
  - kernel.sched_nr_migrate=2

[2] Simple Update: Железо

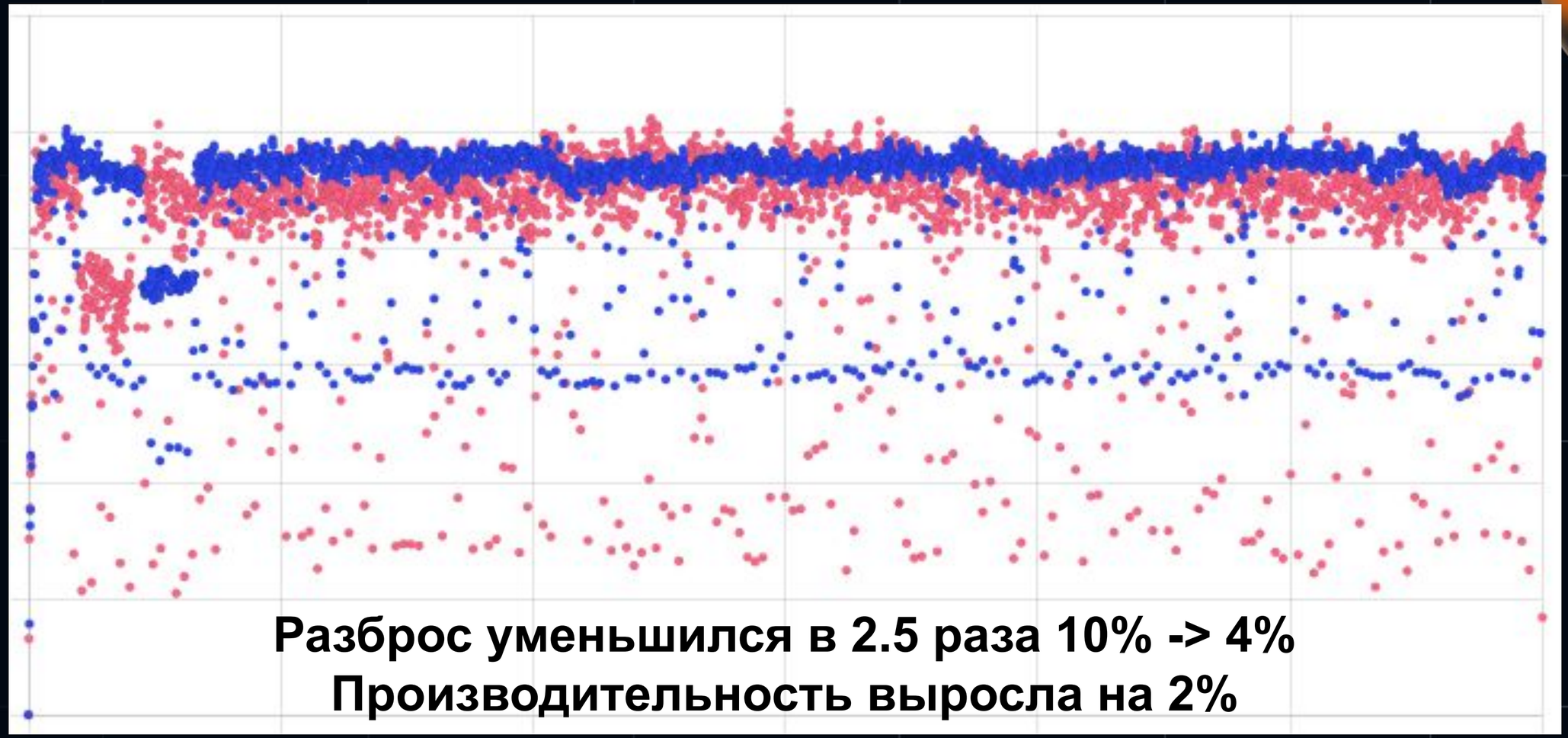
- Linux 5.13+ (sysctl -> sysfs)
  - cd /sys/kernel/debug/sched
  - echo 50000000 > migration_cost_ns
- Не понятно выставить значение на постоянку

[2] Simple Update: Железо

- CPU Pinning: прибивать гвоздями VM к ядрам 1 socket-а
- Стараться избегать sibling ядер
- Poor man:

```
ls -1 /proc/$(pgrep -f kvm-heizenbug)/task | \
xargs -I$ taskset -cp 5-20 $
```

[2] Simple Update: Результат



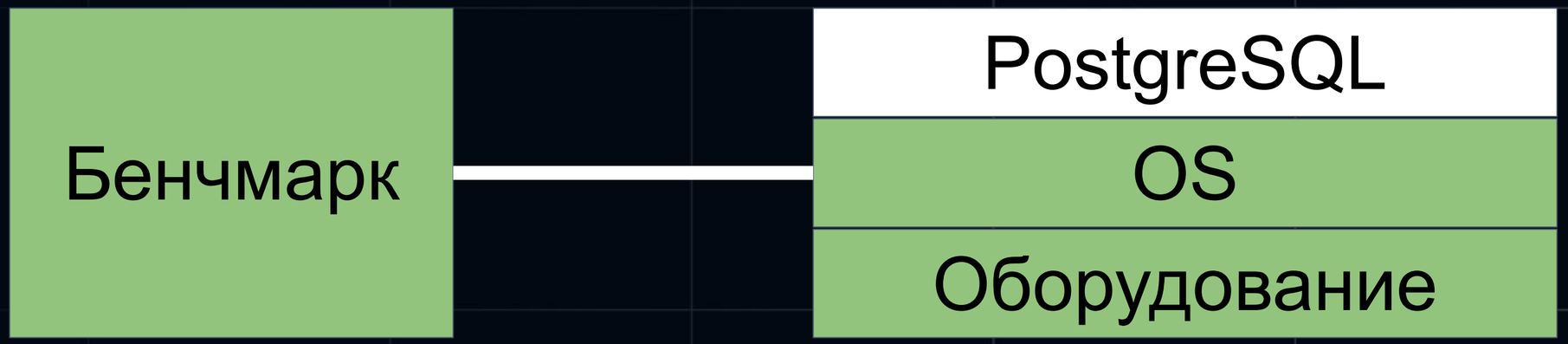
[2] Simple Update: Бонус

- Гостевой диск: HDD vs SSD
  - lsblk -fit
  - echo 0 > /sys/block/vdb/queue/rotational

[2] Simple Update: Выводы

- Нагрузочные инструменты вносят помехи в результаты
- Качественные результаты требуют тюнинга железа и OS
  - C-State, P-State, Frequency
  - Минимизируем миграции процессов
  - Прибиваем процессы к ядрам процессора
- Бонус! Ansible playbook - http://bit.ly/3KZ4Yek

Как подготовиться?



[3] PostgreSQL

- Множество настроек
- Готовые конфигураторы
  - pgtune
  - tuned
- Минимальный набор для бенчмарков

[3] PostgreSQL: Минималка

- Shared buffers
  - ¼ RAM = 1-500 GiB
  - Миллионы 4K страницы
  - Или тысячи 2M страниц
  - Или десятки 1G страниц
- huge_pages = on

  https://www.postgresql.org/docs/15/kernel-resources.html#LINUX-HUGE-PAGES

[3] PostgreSQL: Минималка

- Время запроса не должно страдать из-за dirty pages
- bgwriter
  - bgwriter_delay = 10ms
  - bgwriter_flush_after = 0
  - bgwriter_lru_maxpages = 4000
  - bgwriter_lru_multiplier = 10

[3] PostgreSQL: Минималка

- Чекпоинты притормаживают систему
- Уменьшаем частоту
  - checkpoint_timeout = 30min
  - checkpoint_completion_target = 0.9
  - max_wal_size = 16GB
  - wal_compression = on

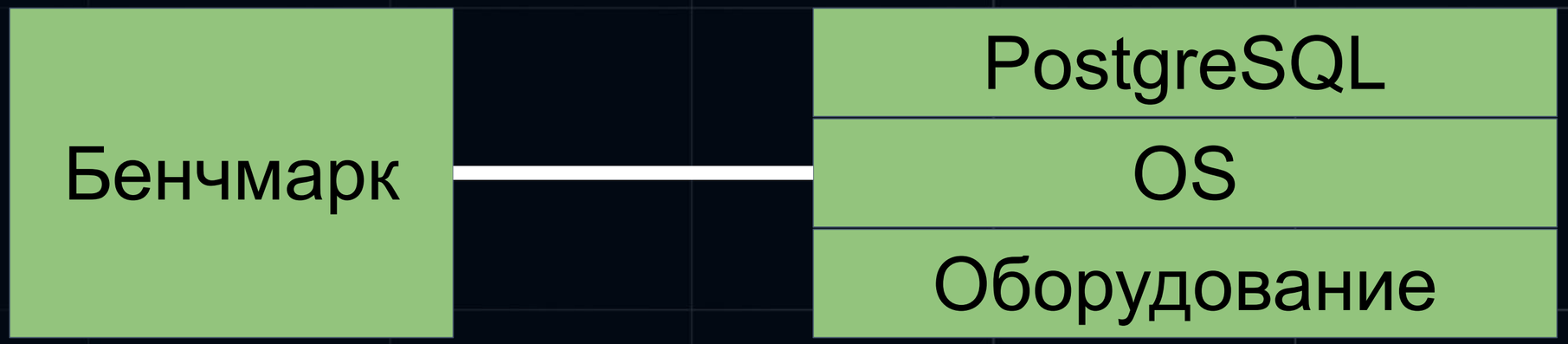
[3] PostgreSQL: Минималка

- Параллелизм мешает для OLTP замеров
- Отключить параллелизм
  - max_parallel_workers_per_gather = 0

[3] PostgreSQL: Бонус

- LWLock WALWrite
- Цель замера - запросы или commit-ы?
- Ускоряем коммит
  - synchronous_commit = off (только для тестов)
  - commit_delay = 0 (особенно если max_connection велико)

Как подготовиться?



К запуску готовы!

В заключение [1]

- Не доверяйте именитым бенчмаркам!
- Анализируйте тесты на аномалии и помехи
- Делайте тюнинг железа, OS на производительность
- Минимизируйте настройки СУБД
- Сохраняйте сырые результаты
- Развивайте автоматизацию запусков

В заключение [2]

“Нельзя теоретизировать, прежде чем появятся факты. Неизбежно начинаешь подстраивать факты под свою теорию, а не строить теорию на основе фактов.”

“It is a capital mistake to theorize before one has data. Insensibly one begins to twist facts to suit theories, instead of theories to suit facts.”

Arthur Conan Doyle

СЛАЙДЫ ТУТ

Михаил Жилин

Telegram:
@mizhka
E-mail:
m.zhilin@postgrespro.ru