Как подготовиться к бенчмарку СУБД?

Михаил Жилин

Postgres Professional

mizhka@gmail.com

HEISENBUG

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 и их решениях
- О как мы теперь запускаем бенчмарки?

О чём НЕ БУДЕТ разговор?

- О бенчмарках OS, приложения
- О тюнинге PostgreSQL
- О обработке результатов и математики

Зачем запустить бенчмарк?

- Какую базу лучше выбрать для X?
- А вдруг ляжет продакшен?
- А точно новая версия лучше? 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!

[1] HammerDB TPC-C: Поехали

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

[1] HammerDB TPC-C: Поехали

- Оборудование
  - 8 ядер, 16 GiB памяти, 100 GB диск
- Объём базы
  - 10 складов (20GiB)
- Нагрузка
  - 20 потоков
- Запуск на 5, 10, 15 минут

[1] HammerDB TPC-C: Поехали

- 3 запуска на 5 минут:

  120,000 - 130,000 ордеров в секунду

[1] HammerDB TPC-C: Поехали

- 3 запуска на 5 минут:

  120,000 - 130,000 ордеров в секунду

- Ещё 3 запуска на 10 минут:

  80,000 - 90,000 ордеров в секунду

[1] HammerDB TPC-C: Поехали

- 3 запуска на 5 минут:

  120,000 - 130,000 ордеров в секунду

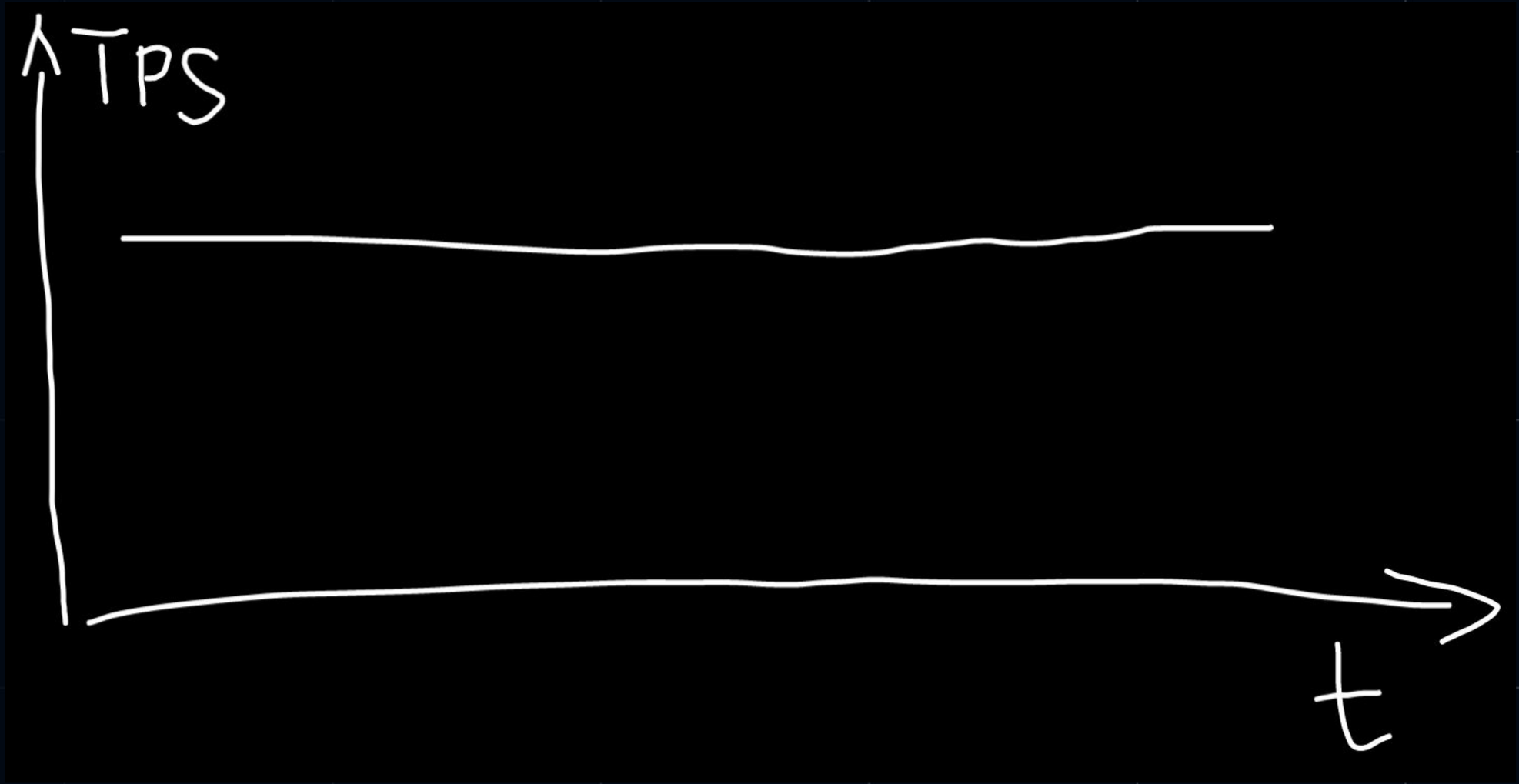
- Ещё 3 запуска на 10 минут:

  80,000 - 90,000 ордеров в секунду

- Ещё 3 запуска на 15 минут:

  15,000 ордеров в секунду

[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

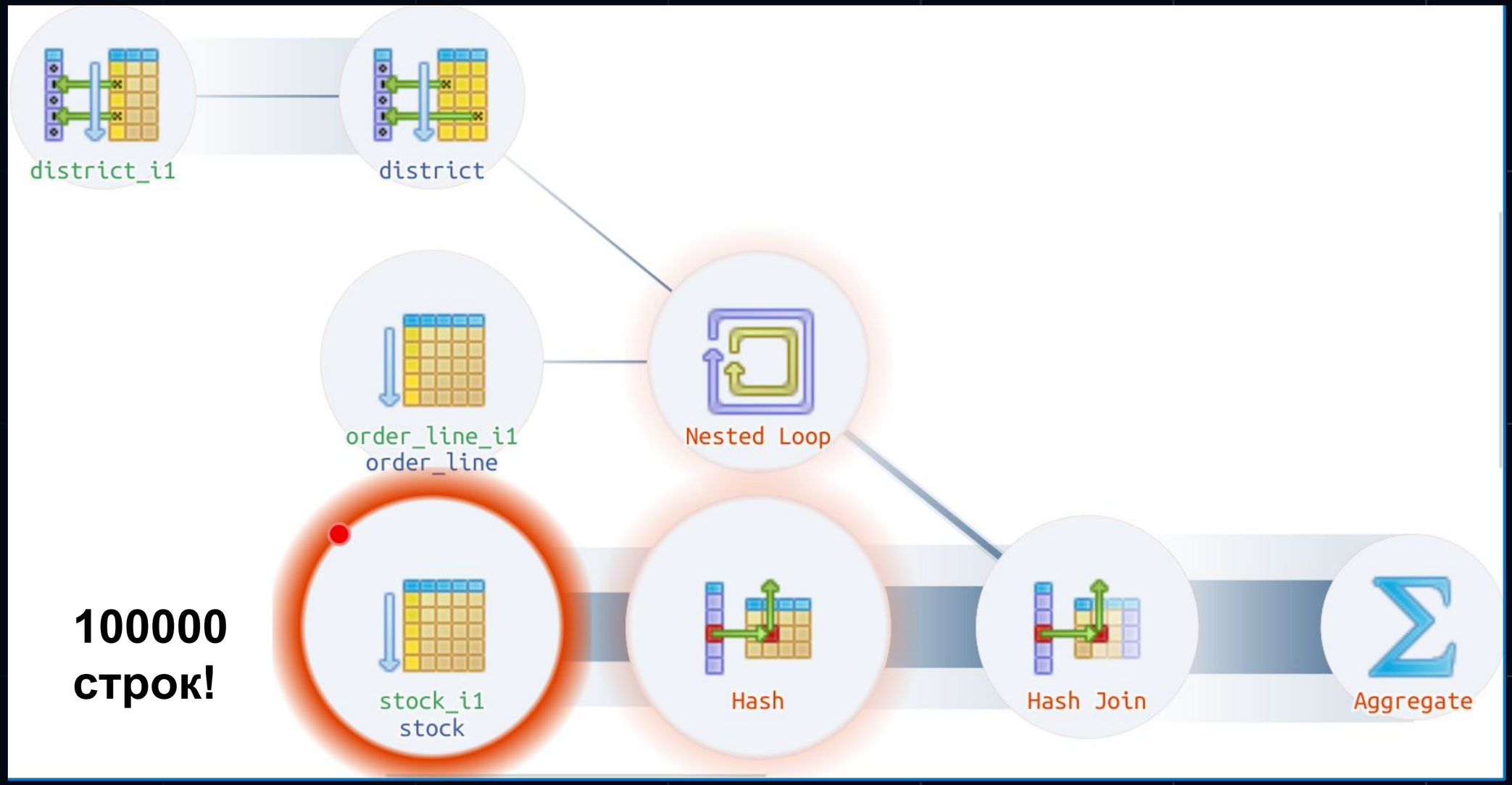
- **Таблица order_line**
  - 10-15 строк на каждый ордер!
  - В начале теста - 0, в конце - 130 миллионов
- **Таблица district**
  - 10 строк на 1 склад, фиксированное число
- **Таблица stock**
  - 100,000 строк на 1 склад, фиксированное число

- https://bit.ly/3KSY7mi

[1] HammerDB TPC-C: Запрос

- Запрос проверки уровня stock-ов
  - TPC-C глава 2.8
  - https://www.tpc.org/tpc_documents_current_versions/pdf/tpc-c_v5.11.0.pdf
  - Запрос из 2.8.2.2

[1] HammerDB TPC-C: Запрос



100000 строк!

[1] HammerDB TPC-C: Запрос

```
      (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))
```

[1] HammerDB TPC-C: Запрос

(**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)**)

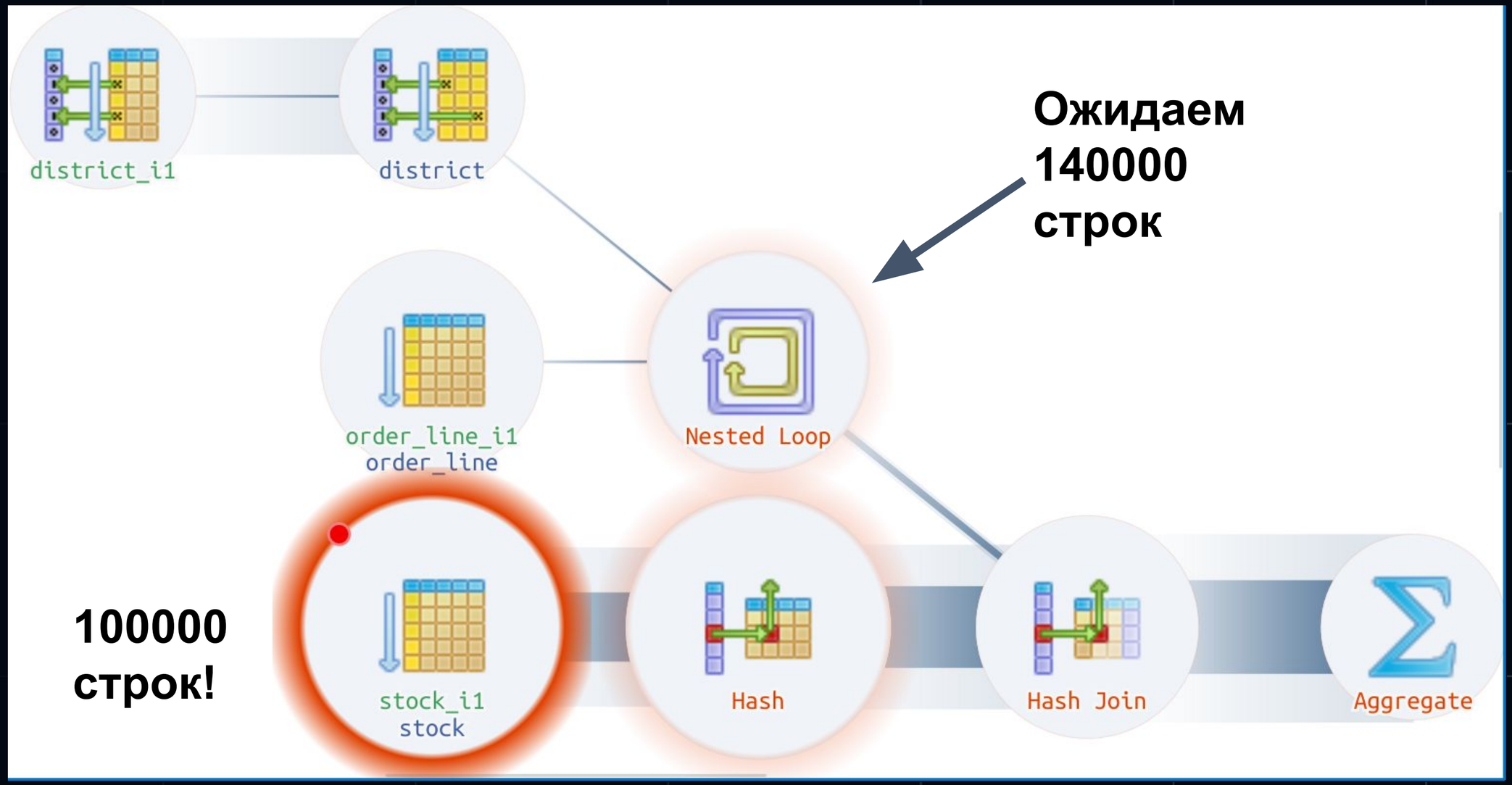
ol_o_id >= expression (district.d_next_o_id)

[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)**
- Rows by (ol_w_id + ol_d_id) =**1,300,000**

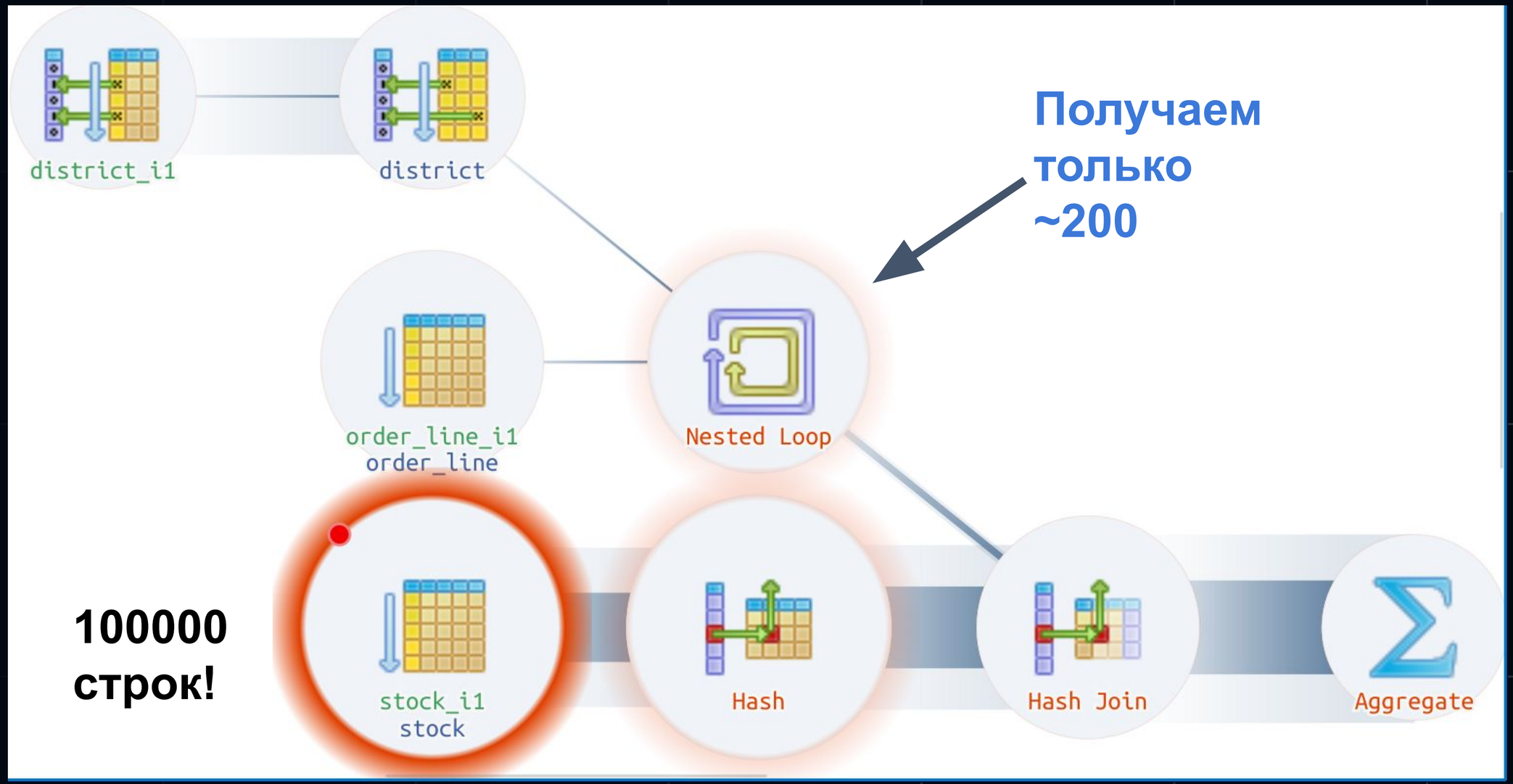
[1] HammerDB TPC-C: Запрос



Ожидаем 140000 строк

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 = ?
  - ol_o_id < ? AND ol_o_id >= ? - 20

[1] HammerDB TPC-C: Выводы

- **Проверяйте, наблюдайте, исследуйте!**
- Производительность может меняться во времени
  - Transaction log, HDR, heat map
  - Любой движок БД может ошибиться
- Каждый кейс требует анализа

[2] Simple Update

- **update t set c1 = ?, c2 = ? where id = ?**
- Без ошибок планировщика
- Просто реализовать самому

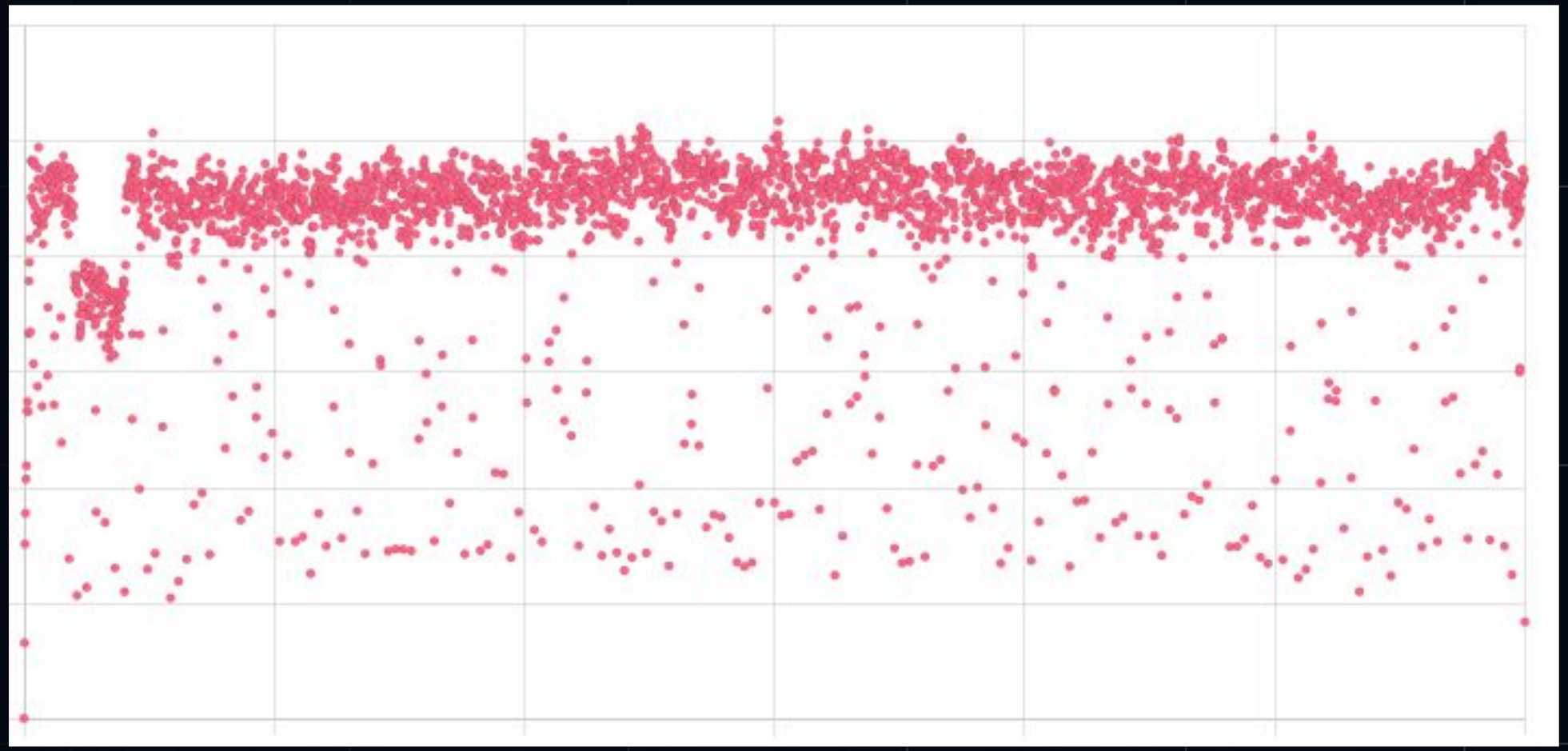
[2] Simple Update

- 16 ядер
- Все данные помещаются в кэш
- Замер
  - 10 потоков нагрузки (JMeter)
  - на 5 минут
- JMeter выдал **43051** запрос в секунду! **Круто!**

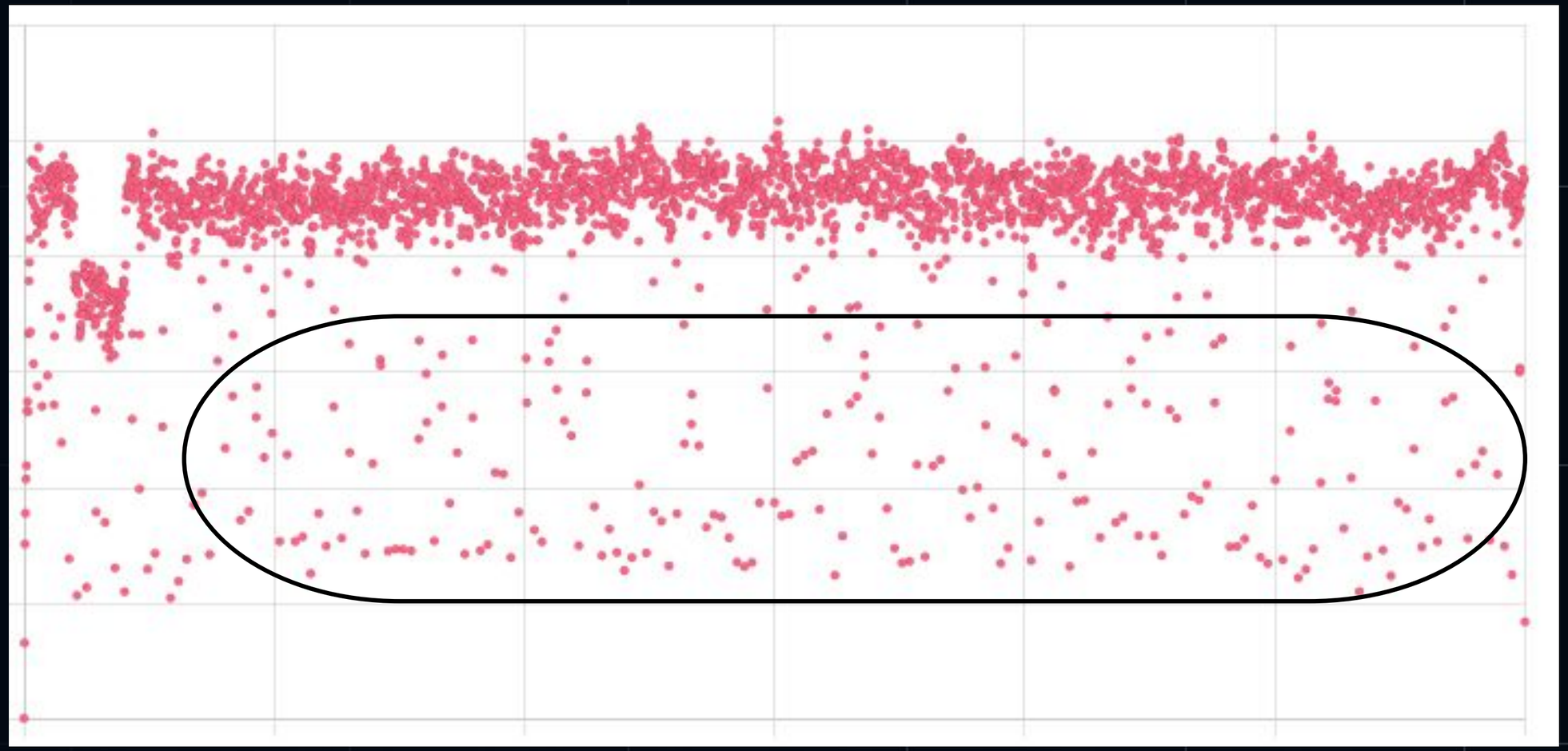
[2] Simple Update

TPS (Time) ?

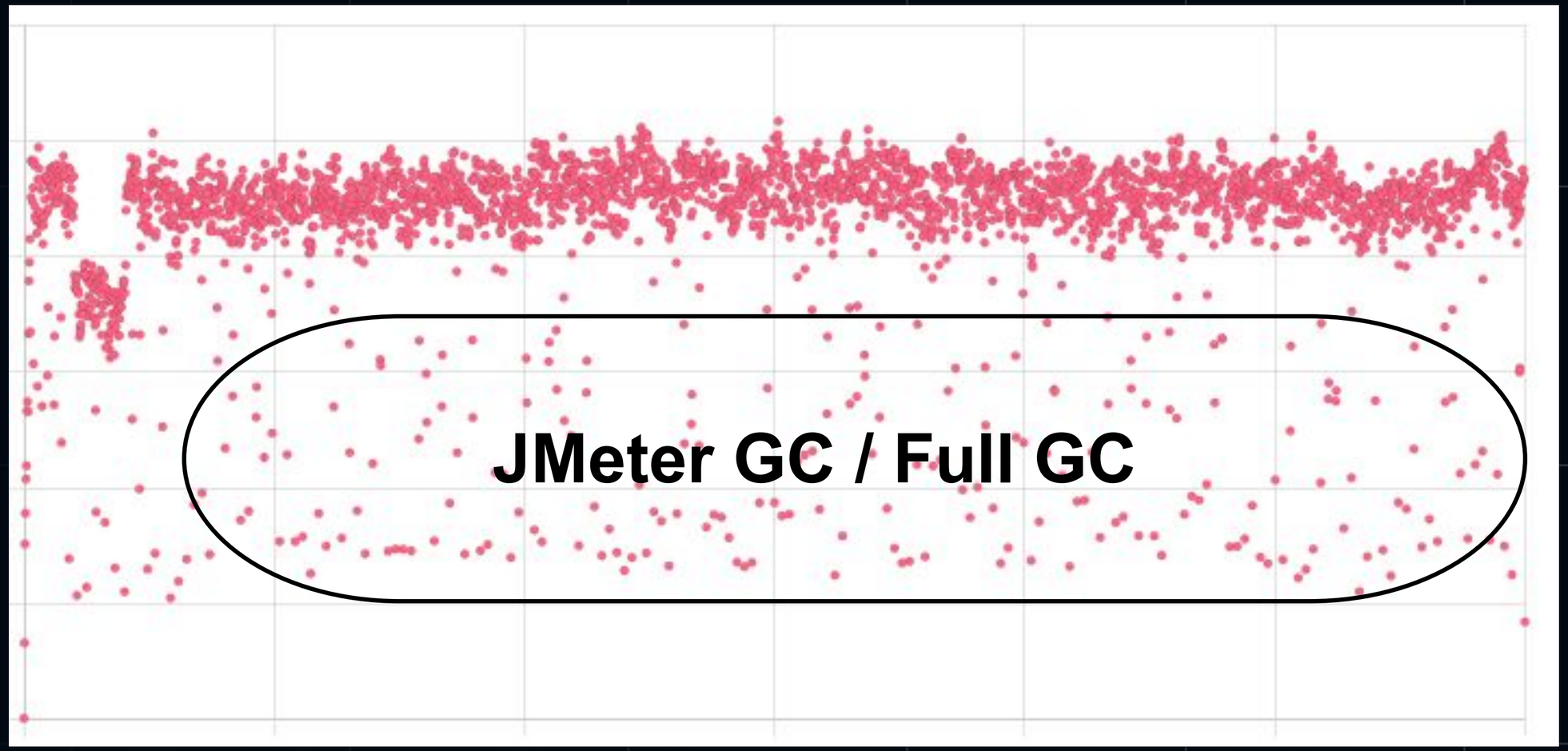
[2] Simple Update



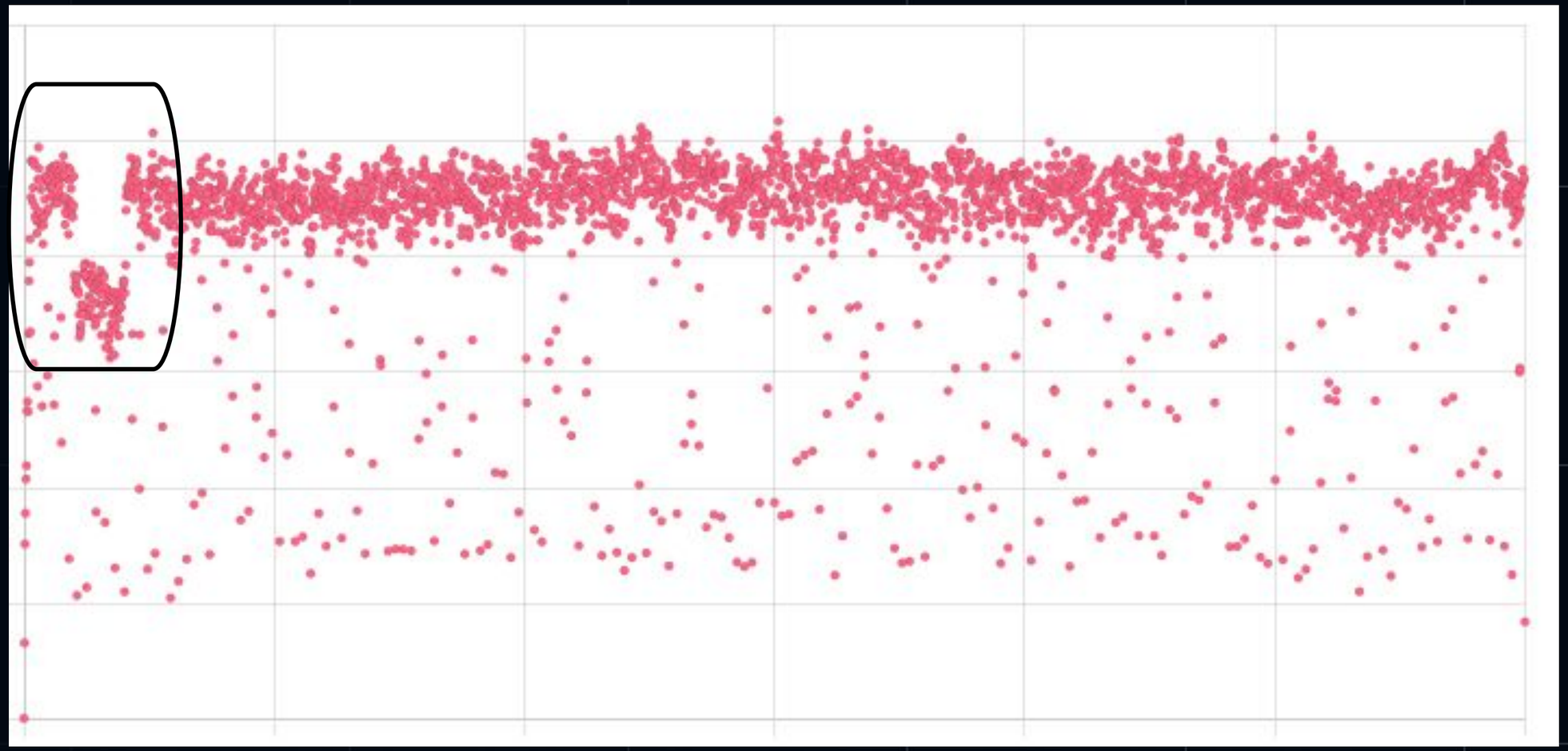
[2] Simple Update



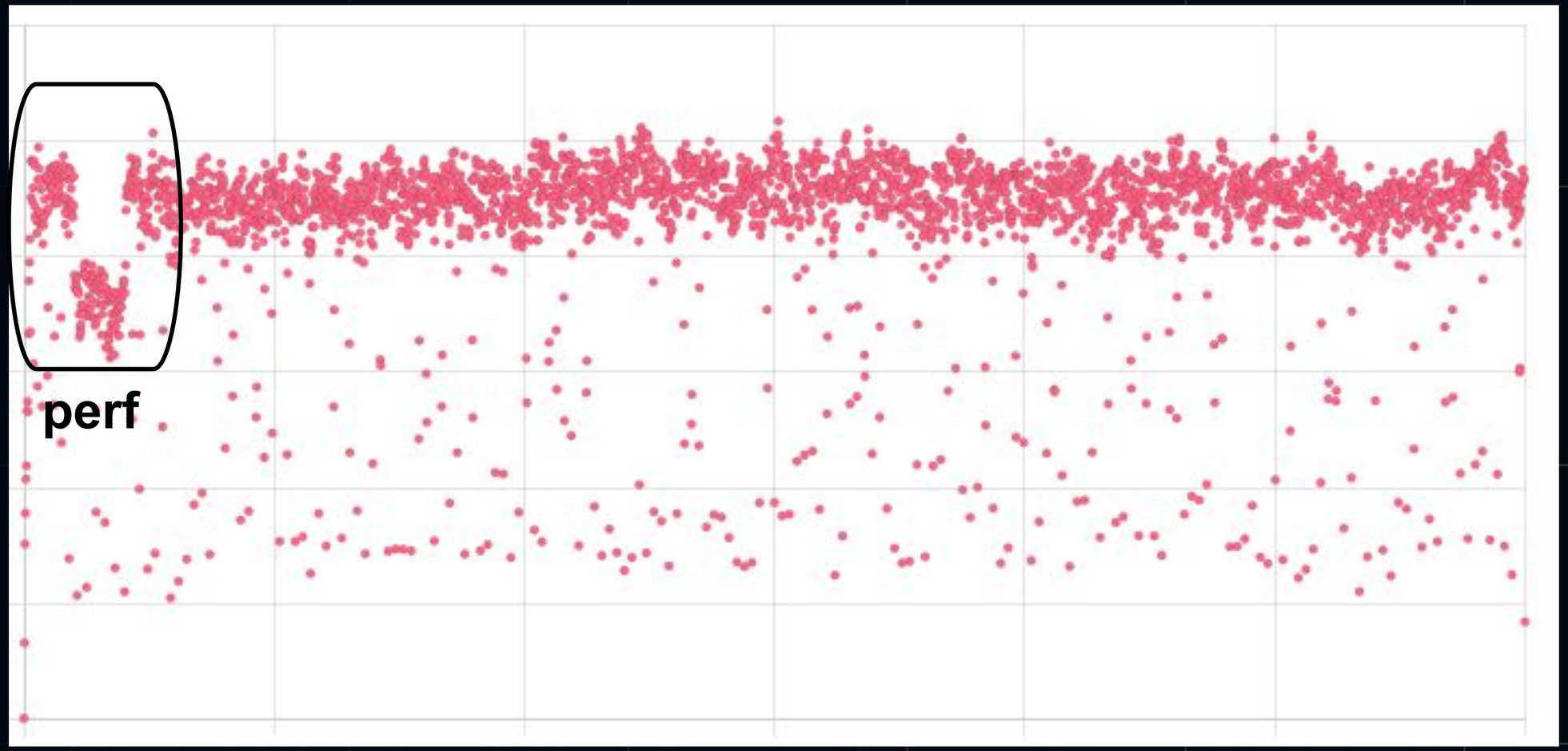
[2] Simple Update



[2] Simple Update



[2] Simple Update



[2] Simple Update

Average 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: HyperVisor

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

[2] Simple Update: HyperVisor

- BIOS
  - Отключаем энергосбережение
  - Turbo = ON
  - MaxPerformance (FAN, CPU, etc)

[2] Simple Update: HyperVisor

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

[2] Simple Update: HyperVisor

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

[2] Simple Update: HyperVisor

- CPU C-State
  - power mode: C0 - running, C6 - sleep/idle
  - GRUB: processor.max_cstate=1 idle=poll
  - Setting: echo 1 > /dev/cpu_dma_latency

[2] Simple Update: HyperVisor

- OS Scheduler (sysctl)
  - **kernel.sched_migration_cost_ns=50000000**
  - **kernel.sched_autogroup_enabled=0**
  - kernel.sched_min_granularity_ns=100000000
  - kernel.sched_wakeup_granularity_ns=10000000
  - kernel.sched_nr_migrate=2

[2] Simple Update: HyperVisor

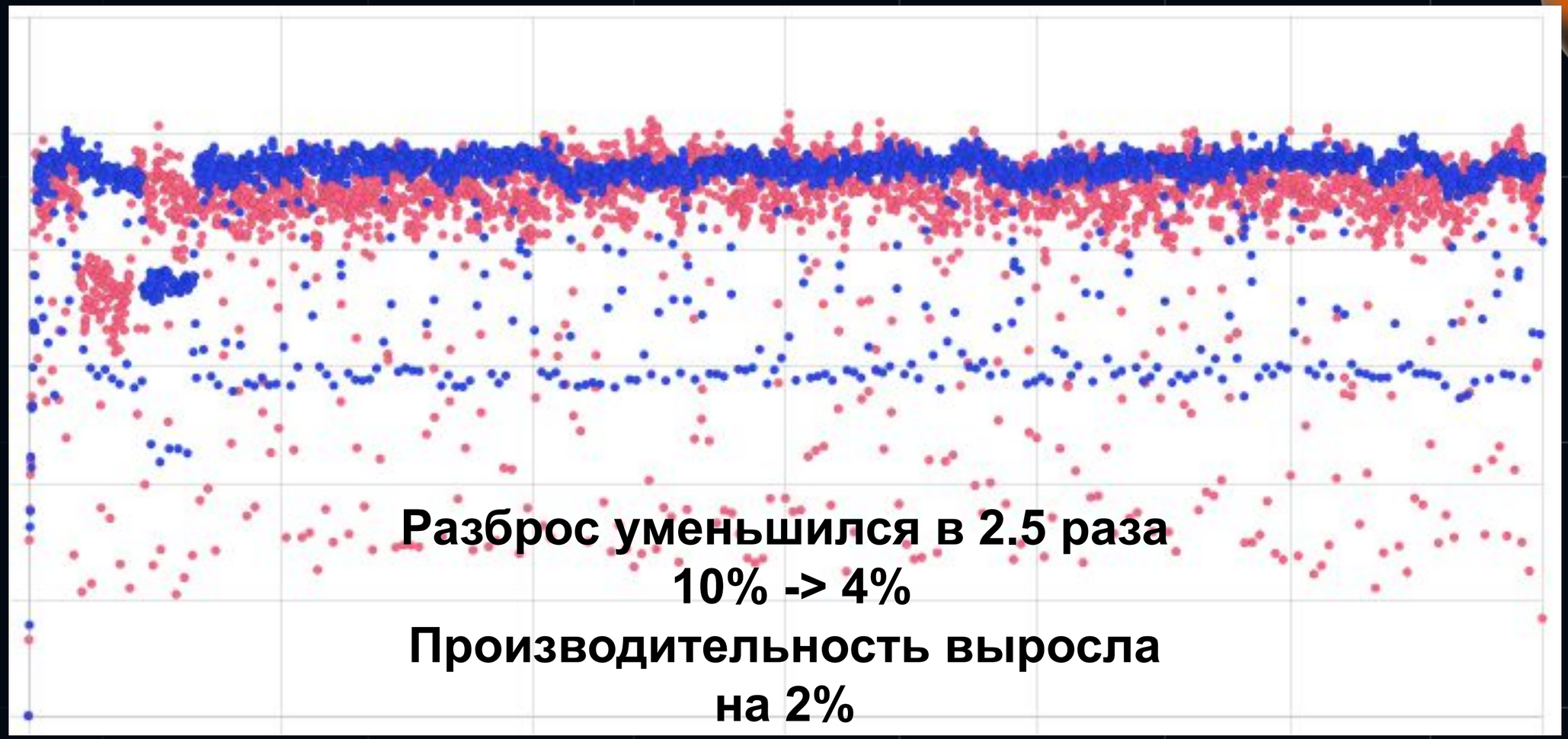
- Linux 5.13+ (sysctl -> sysfs)
  - cd /sys/kernel/debug/sched
  - echo 50000000 > migration_cost_ns
- Не понятно как восстановить значение после перезагрузки

[2] Simple Update: HyperVisor

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

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

[2] Simple Update: Результат



Разброс уменьшился в 2.5 раза
10% -> 4%
Производительность выросла
на 2%

[2] Simple Update: HyperVisor

- Мониторинг
  - утилизация ядер: **node_exporter**
  - NUMA, L* caches, IPC: **intel/opmc**

[2] Simple Update: Выводы

- **Стабилизируйте тесты!**
- Инфраструктура на MaxPerformance
  - C-State, P-State, Frequency
- Минимизировать миграции процессов
- CPU Pinning
- Бонус! Ansible playbook - http://bit.ly/3KZ4Yek

[3] Сложно…

- Время…
  - Установка, настройка

[3] Сложно…

- Время…
  - Установка, настройка: 1час - 5 дней

[3] Сложно…

- Время…
  - Установка, настройка: 1час - 5 дней
  - Человеческие ошибки:

[3] Сложно…

- Время…
  - Установка, настройка: 1час - 5 дней
  - Человеческие ошибки: ∞

[3] Сложно…

- Время…
  - Установка, настройка: 1час - 5 дней
  - Человеческие ошибки: ∞
  - Экспертиза…

[3] Сложно…

- Время…
  - Установка, настройка: 1час - 5 дней
  - Человеческие ошибки: ∞
  - Экспертиза… **$$$**

[3] Сложно…

- Автоматизировать
  - Нельзя запомнить все 100500 настроек
  - Систематизация знания
  - Возможность запусков 24 на 7
  - Удобство
  - Бывает полезно не только для бенчмарков

[3] Сложно… Автоматизируй!

- Наш путь
  - Terraform
  - Ansible
  - Proxmox
  - Home-made обёртки и доработки

[3] Сложно… Автоматизируй!

- Terraform
  - Создание и удаление окружений
  - Поддержка от VirtualBox до сложных ресурсов облаков
  - Open-Source, баги, патчи, романтика…

[3] Сложно… Автоматизируй!

- Ansible
  - shell/bash со стеройдами
  - расширяем (роли, модули на Python)
  - error handling
- От настроек OS до запусков тестов

[3] Сложно… Автоматизируй!

- Proxmox
  - Оркестрация виртуальных машин
  - Веб морда
  - Разграничение по пользователям
  - API, Open Source, активное community
  - Плагины (Perl)

[3] Сложно… Автоматизируй!

- Home-made обёртки и доработки
  - env {create,delete,list}
  - db generate
  - load test

[3] Сложно… Автоматизируй!

- Результаты
  - Локальный архив
  - Web морда

Test ID	DB Engine	Scenario	Results	Owner	CPU	Load	Duration	Volume
2023-04-13T16:42	sdm-14.7.1	hammerdb	74708 NOPM	mike	8	100	20	400
2023-04-13T16:35	sdm-14.7.1	hammerdb	101499 NOPM	mike	8	100	2	400
2023-04-13T15:47	pgproee-13.6.1	hammerdb	25906 NOPM	mike	16	20	10	10
2023-04-13T15:20	pgproee-13.6.1	hammerdb	34653 NOPM	mike	16	20	10	10
2023-04-13T15:02	pgproee-13.6.1	hammerdb	20698 NOPM	mike	16	20	10	10
2023-04-13T14:43	pgproee-13.6.1	hammerdb	62854 NOPM	mike	16	20	10	10
2023-04-13T14:27	pgproee-13.6.1	hammerdb	97184 NOPM	mike	16	20	10	10

[3] Сложно… Автоматизируй!

[3] Сложно… Автоматизируй!

```
export BM_BACKEND=mn23-g5
export BM_ENV=mz-heisen
export BM_NODES=1
export BM_SCHEMA=standalone
export BM_SCALE=c8-16-200
export BM_OS=debian-11.2
export BM_DBENGINE=postgresql-14.7

bm env create

-- 2-3 mins
```

[3] Сложно… Автоматизируй!

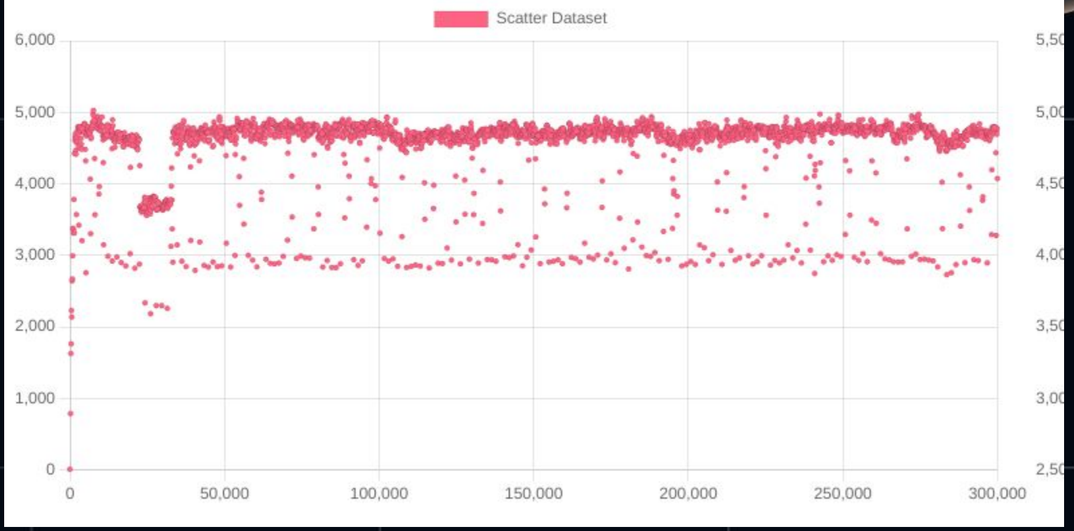
```
export BM_SCENARIO=hammerdb
export BM_LOAD=20
export BM_VOLUME=10

bm db generate
bm load test -duration 15

-- 20-30 mins
```

[3] Сложно… Автоматизируй!

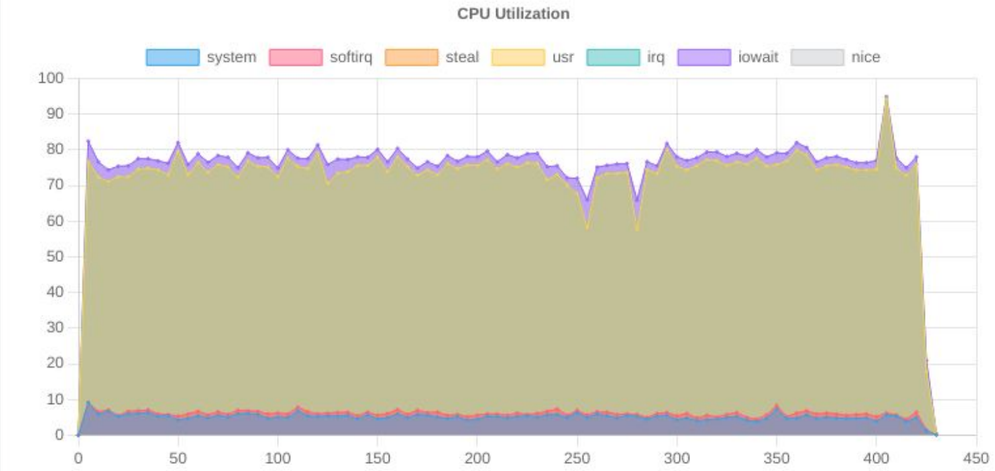
Overview | Settings | **Backup Details** | Database Hardware | S3 Hardware | Files

Test ID	Tool	Name	Type	Duration	Ratio	Start LSN	Stop LSN
2023-03-30T13:32	probackup	RSBXBS	FULL	00:28:08	55.62	2F/21000060	2F/23000728
2023-03-30T13:32	probackup	RSBYX3	PTRACK	00:06:13	20.8	2F/71006EE0	2F/95D67D88
2023-03-30T13:32	probackup	RSBZHQ	PTRACK	00:09:16	26.03	2F/ED008570	30/233FED08
2023-03-30T13:32	probackup	RSC07K	PTRACK	00:10:25	28.97	30/7B009610	30/B7B4B8A0

Test ID	DB Engine	Test	Results	Hostname	Owner
2023-03-01T01:44	pgproee-14.6-p0	jmeter	45472.0/s	loadgenerator0	Michael Zhilin



Overview | Settings | **Database Hardware** | Files



[3] Сложно… Выводы

К запуску benchmark-ов готовы!


СПАСИБО ЗА ВНИМАНИЕ!