PostgresPro

Примеры атипичных сценариев при нагрузочном тестировании PostgreSQL

Михаил Жилин
Руководитель группы производительности Postgres Professional

PostgresPro

План встречи

- Проблемы и их причины
- Ресурсы и нагрузки PostgreSQL
- Профилирование
- Примеры

PostgresPro

Проблемы тут, проблемы там...

PostgresPro

Что мы хотим?

- Проверить чтобы всё:
  - Работало
  - Без ошибок
  - Быстро
  - Практически идеально
- Но...

PostgresPro

Проблемы и их причины

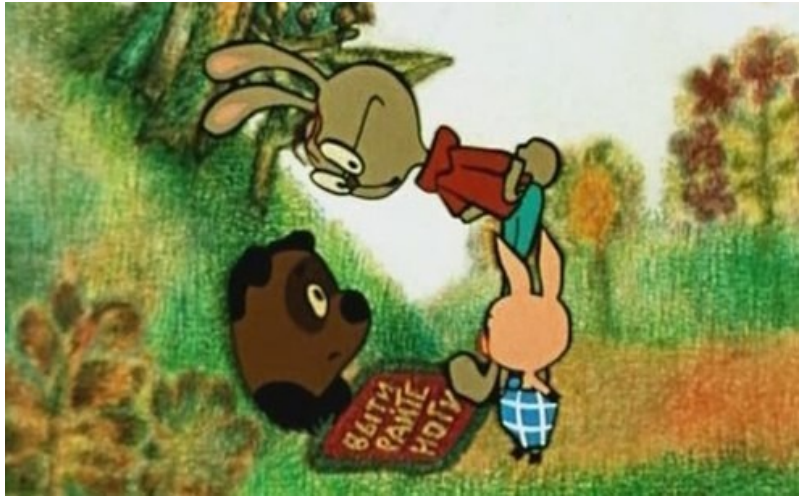
- Иногда случаются
  - Ошибки в ответах системы
  - Ошибки в логах
  - Всё сломалось
- Что делаем?
  - Смотрим логи
  - Пытаемся воспроизвести
  - Извлекаем stack trace-ы
  - Читаем код, документацию, снова код…

Проблемы и их причины

- А бывают тормоза…
  - Растёт время отклика системы
  - Тайм-ауты, жалобы
  - Массовые
- «Houston, we have problems»

Проблемы и их причины

- У кого-то слишком узкие двери!
- Нет! Всё потому, что кто-то слишком много ест!



Проблемы и их причины

- Много лет тому назад...

Проблемы и их причины

- Много лет тому назад...

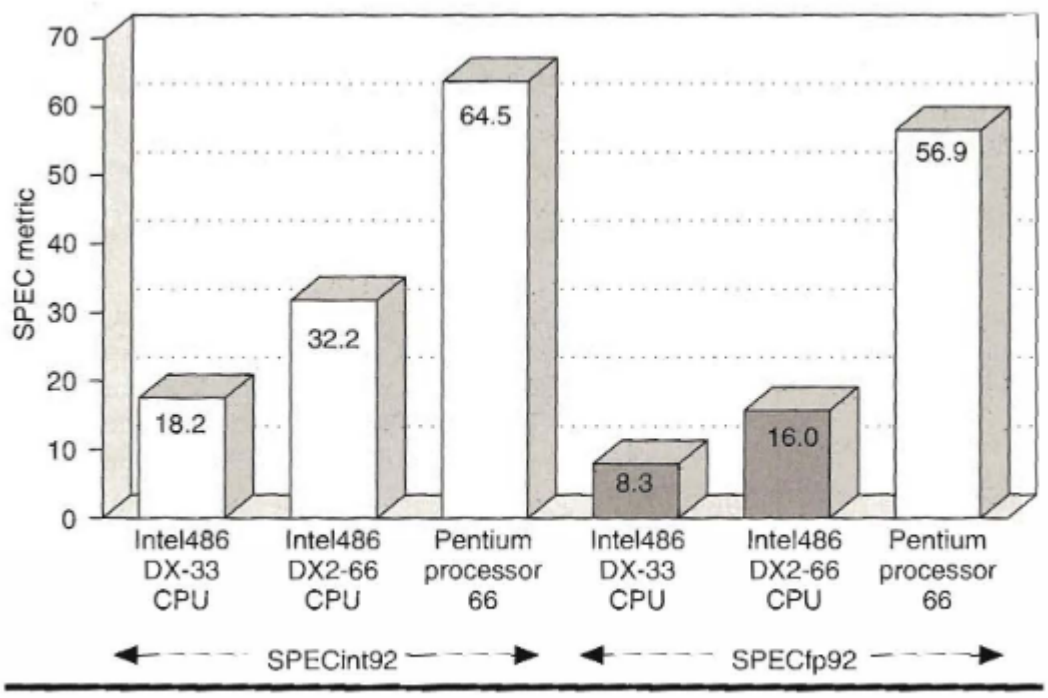
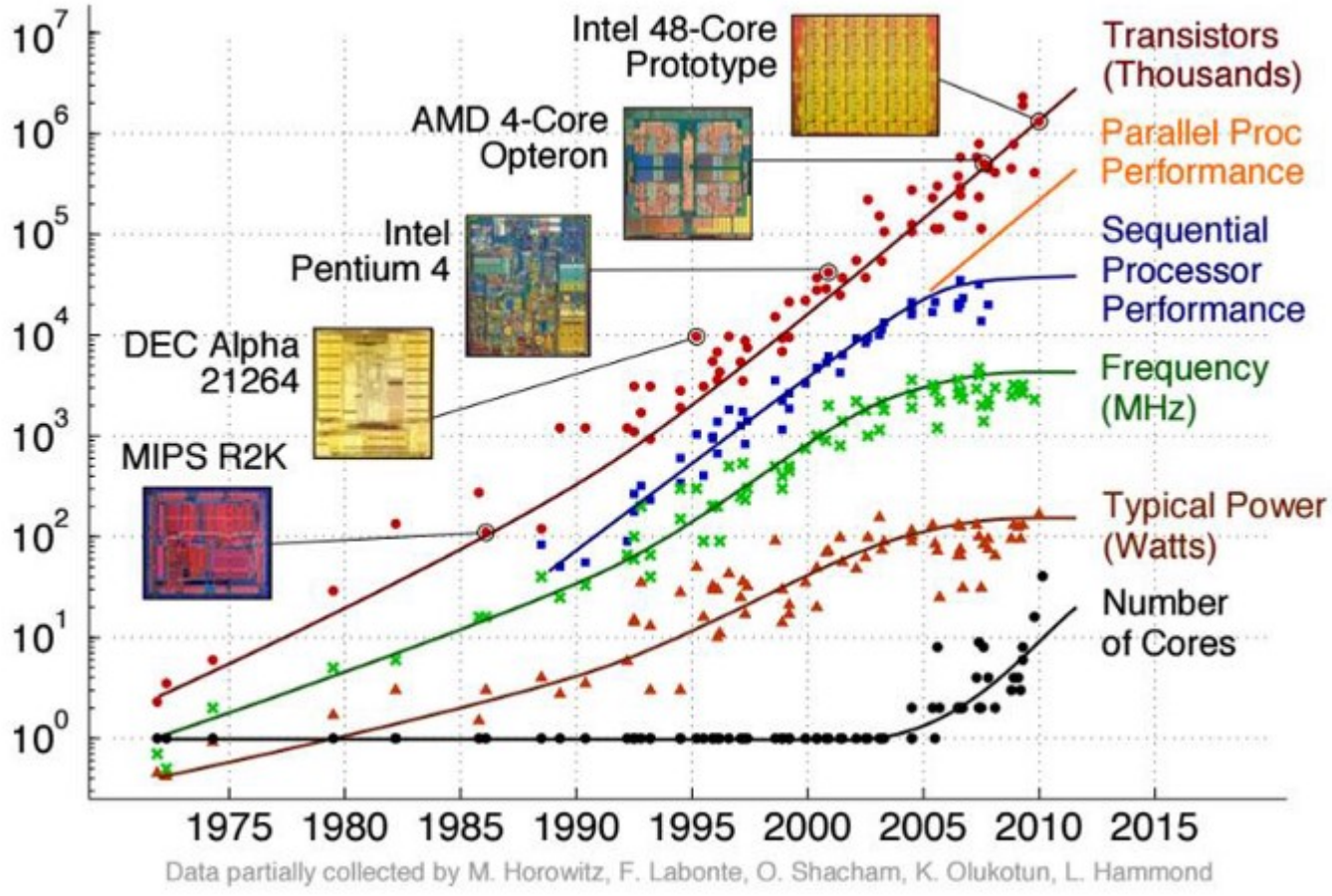


Figure 11. Pentium processor and i486 CPU performance for SPEC benchmarks.

Проблемы и их причины



Проблемы и их причины

- Сложность растёт
  - SMT → SMP → NUMA

Проблемы и их причины

- Сложность растёт
  - SMT → SMP → NUMA
  - Local Disks → SAN → SDS

Проблемы и их причины

- Сложность растёт
  - SMT → SMP → NUMA
  - Local Disks → SAN → SDS
  - VLAN/Routing → Fabrics → SDN (VNF)

Проблемы и их причины

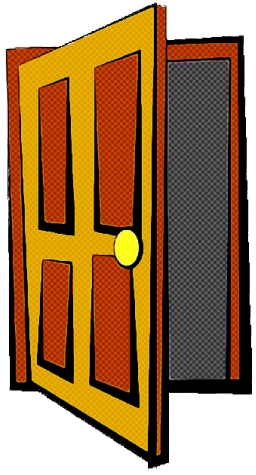
- Сложность растёт
  - SMT → SMP → NUMA
  - Local Disks → SAN → SDS
  - VLAN/Routing → Fabrics → SDN (VNF)
  - Железо → HyperVizor и Containers

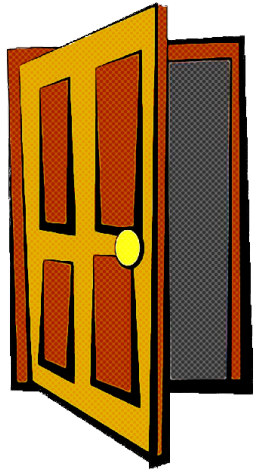
Проблемы и их причины

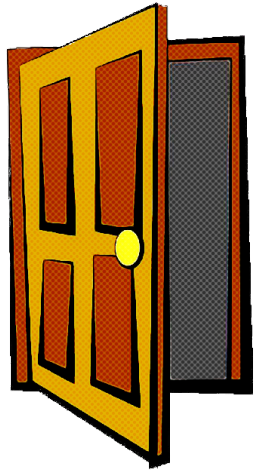
- Сложность растёт
  - SMT → SMP → NUMA
  - Local Disks → SAN → SDS
  - VLAN/Routing → Fabrics → SDN (VNF)
  - Железо → HyperVizor и Containers
  - Standalone Database → Replication → Sharding

Проблемы и их причины

- Узких дверей становится больше...







Проблемы и их причины

- Узких дверей становится больше...

Проблемы и их причины

- Узких дверей становится больше...

Проблемы и их причины

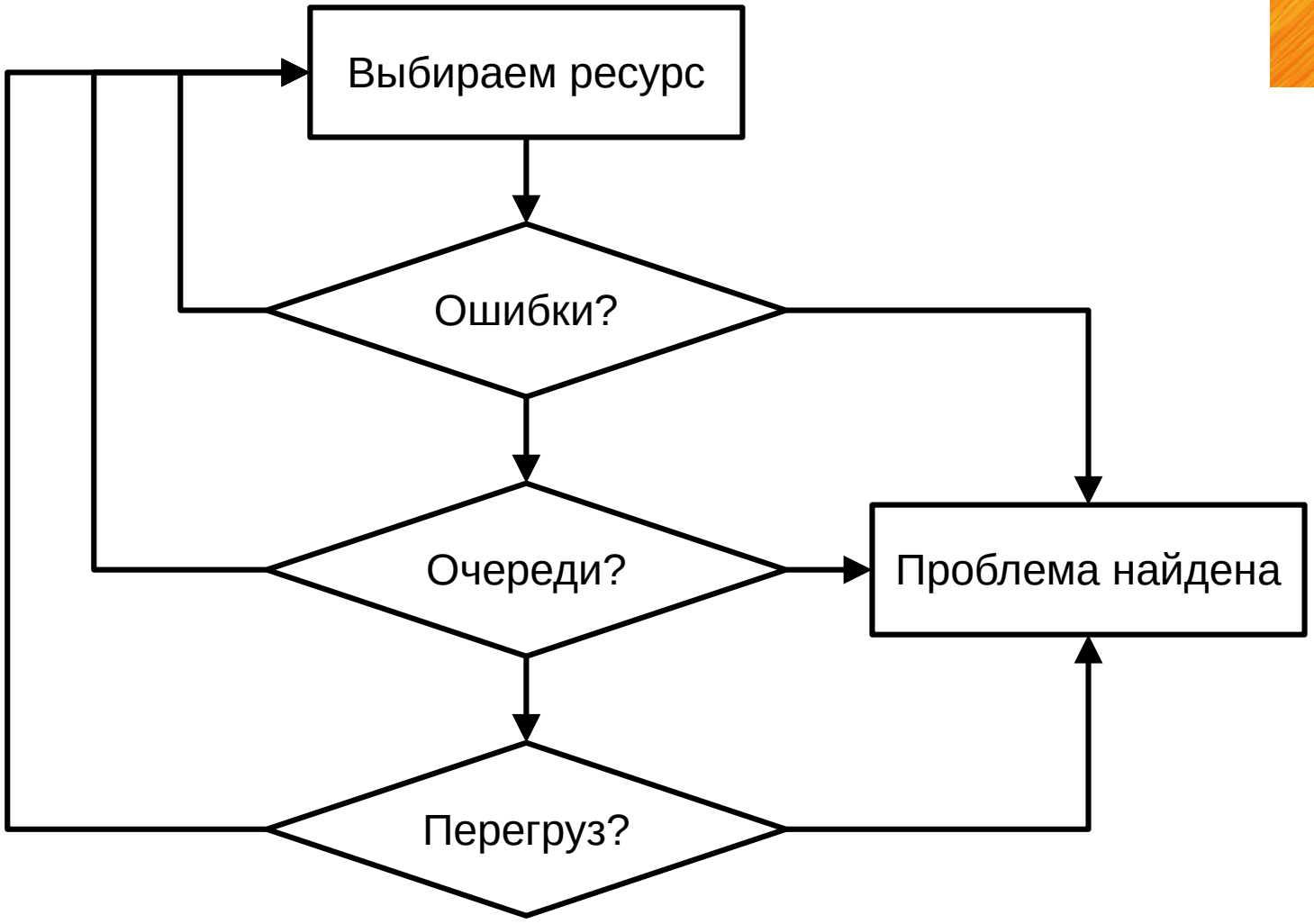
- Узких дверей становится слишком много...

Проблемы и их причины

- Поиск узкой двери
  - Перебор всех компонент
  - Медленно
  - Трудозатратно
- Замерить всё невозможно
- Но можно делать check list-ы

Проблемы и их причины

- Чек-лист ресурсов
  - Utilization, Saturation, Errors (USE)
- Чек-лист нагрузки
  - Rate, Errors, Durations (RED)
- Смешанный вариант
  - Latency, Traffic, Errors, Saturation (LTES)


Выбираем ресурс
Ошибки?
Очереди?
Проблема найдена
Перегруз?

PostgresPro

Ресурсы и нагрузки в PostgreSQL

Ресурсы и нагрузки в PostgreSQL

- Подключения

Ресурсы и нагрузки в PostgreSQL

- Подключения
  - Состояние (idle, active)

Ресурсы и нагрузки в PostgreSQL

- Подключения
  - Состояние (idle, active)
  - Блокировка (Lock, LWLock)

Ресурсы и нагрузки в PostgreSQL

- Подключения
  - Состояние (idle, active)
  - Блокировка (Lock, LWLock)
- Запросы

Ресурсы и нагрузки в PostgreSQL

- Подключения
  - Состояние (idle, active)
  - Блокировка (Lock, LWLock)
- Запросы
  - Частота

Ресурсы и нагрузки в PostgreSQL

- Подключения
  - Состояние (idle, active)
  - Блокировка (Lock, LWLock)
- Запросы
  - Частота
  - Время выполнения

Ресурсы и нагрузки в PostgreSQL

- Подключения
  - Состояние (idle, active)
  - Блокировка (Lock, LWLock)
- Запросы
  - Частота
  - Время выполнения
- Служебные процессы

Ресурсы и нагрузки в PostgreSQL

- **pg_stat_activity** – список сессий и их состояние
  - *state, wait_event* – состояния / блокировки

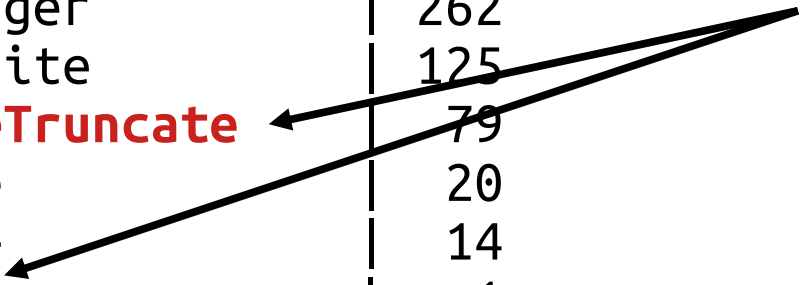
Ресурсы и нагрузки в PostgreSQL

```
select state, wait_event, count(1) as cnt
  from pg_stat_activity
 group by state, wait_event order by 1, 3;

 state    |     wait_event       | cnt
----------+----------------------+-----
 active   | LockManager          | 262
 active   | SInvalWrite          | 125
 active   | DataFileTruncate     |  79
 active   | WALWrite             |  20
 active   |                      |  14
 active   | WALSync              |   1
 inactive | ClientRead           |   3
          | AutoVacuumMain       |   1
          | LogicalLauncherMain  |   1
```

Ресурсы и нагрузки в PostgreSQL

```
select state, wait_event, count(1) as cnt
  from pg_stat_activity
 group by state, wait_event order by 1, 3;
```

```
 state    |     wait_event       | cnt
----------+----------------------+-----
 active   | LockManager          | 262
 active   | SInvalWrite          | 125
 active   | DataFileTruncate     |  79
 active   | WALWrite             |  20
 active   | < NULL >             |  14
 active   | WALSync              |   1
 inactive | ClientRead           |   3
          | AutoVacuumMain       |   1
          | LogicalLauncherMain  |   1
```



Disk

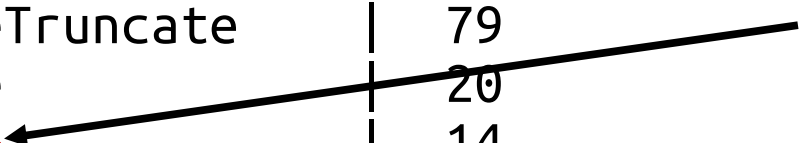
CPU

Блокировки

Ресурсы и нагрузки в PostgreSQL

```
select state, wait_event, count(1) as cnt
  from pg_stat_activity
 group by state, wait_event order by 1, 3;
```

```
 state    |     wait_event       | cnt
----------+----------------------+-----
 active   | LockManager          | 262
 active   | SInvalWrite          | 125
 active   | DataFileTruncate     |  79
 active   | WALWrite             |  20
 active   | < NULL >             |  14
 active   | WALSync              |   1
 inactive | ClientRead           |   3
          | AutoVacuumMain       |   1
          | LogicalLauncherMain  |   1
```



Disk

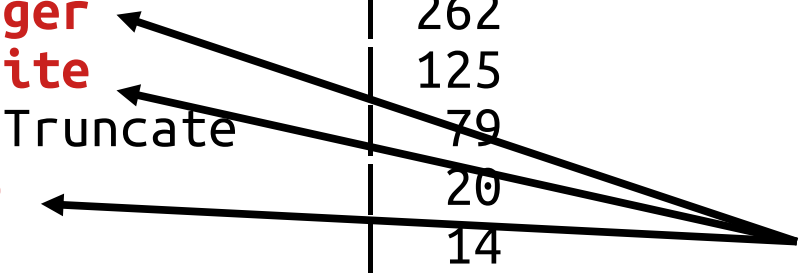
CPU

Блокировки

Ресурсы и нагрузки в PostgreSQL

```
select state, wait_event, count(1) as cnt
  from pg_stat_activity
 group by state, wait_event order by 1, 3;
```

```
 state    |     wait_event       | cnt
----------+----------------------+-----
 active   | LockManager          | 262
 active   | SInvalWrite          | 125
 active   | DataFileTruncate     |  79
 active   | WALWrite             |  20
 active   | < NULL >             |  14
 active   | WALSync              |   1
 inactive | ClientRead           |   3
          | AutoVacuumMain       |   1
          | LogicalLauncherMain  |   1
```



Disk

CPU

Блокировки

Ресурсы и нагрузки в PostgreSQL

- Описание всех возможных ожиданий PostgreSQL:

  https://bit.ly/3rmWAhu

Ресурсы и нагрузки в PostgreSQL

- **pg_stat_statements** – статистика по отдельным запросам

Ресурсы и нагрузки в PostgreSQL

- **pg_stat_statements** – статистика по отдельным запросам

  (+) planning / execution time

Ресурсы и нагрузки в PostgreSQL

- **pg_stat_statements** – статистика по отдельным запросам

  (+) planning / execution time

  (−) суммарная статистика за всё время

Ресурсы и нагрузки в PostgreSQL

- **pg_stat_statements** – статистика по отдельным запросам
  - (+) planning / execution time
  - (−) суммарная статистика за всё время
- **pg_profile** – исторический мониторинг

PostgresPro

Ресурсы и нагрузки в PostgreSQL

- **pg_stat_statements** – статистика по отдельным запросам
  - (+) planning / execution time
  - (−) суммарная статистика за всё время
- **pg_profile** – исторический мониторинг
- *SQL Tuning*
  - https://use-the-index-luke.com/

USE THE INDEX, LUKE!

A Guide to Database Performance for Developers

PostgresPro

Ресурсы и нагрузки в PostgreSQL

- **Андрей Зубков** — автор

  pg_profile / pgpro_pwr

- Доклад 16 октября в 12:15 (Зал 2)

  pg_profile — утилита стратегического мониторинга PostgreSQL

PostgresPro

Но бывают атипичные случаи...

PostgresPro

Но бывают атипичные случаи...

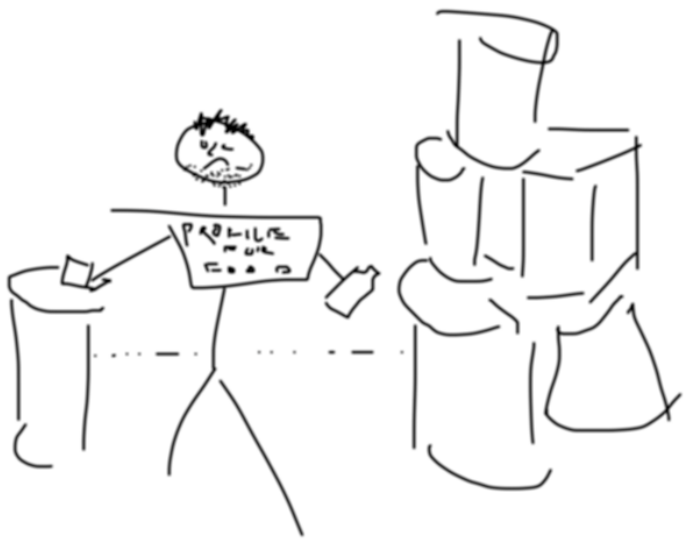
- Система тормозит
- Ресурсы израсходованы
- Запросы тормозят
  - Планы нормальные
- Не помогает тюнинг базы и OS
- Что не хватает? Куда бежать?

PostgresPro

Профилирование

Профилирование

- Давным давно...
- gdb (lldb)
  - attach подключиться к процессу
  - bt стек вызова
  - print вывод данных из памяти
- http://poormansprofiler.org/
- А сейчас...



Профилирование

- Инструменты
  - Execution (CPU): DTrace, perf, eBPF
  - Disk (Block devices): blktrace, ioprof, eBPF
  - Memory: Valgrind, perf, eBPF
  - Network: tcpdump, wireshark, eBPF
- Способы
  - sampling – снимки с определённой частотой
  - timing (call/exit) – замер времени

PostgresPro

Профилирование Perf + CPU

- Sampling
- Быстрый вариант
  - perf top
- Шпаргалка
  - perf record -F 99 -a -g --call-graph=dwarf sleep 2
  - perf script --header --fields comm,pid,tid,time,event,ip,sym,dso
- Требует установки debuginfo для PostgreSQL
- Требует perf-сборку с libunwind (проблема с RHEL-based)

Профилирование Perf + CPU

```
56272bd202b1 PGSemaphoreLock (bin/postgres)
56272bdac6db LWLockAcquire (bin/postgres)
56272bda8987 LockAcquireExtended (bin/postgres)
56272bda686e LockRelationOid (bin/postgres)
56272bac6ebc relation_open (bin/postgres)
56272bb10466 index_open (bin/postgres)
56272bb0fed3 systable_beginscan (bin/postgres)
56272bb81e82 ApplySetting (bin/postgres)
56272bee3d84 process_settings (inlined)
56272bee45d8 process_settings (inlined)
56272bee45d8 InitPostgres (bin/postgres)
56272bdbfaae PostgresMain (bin/postgres)
56272bd34c6d BackendRun (inlined)
56272bd34c6d BackendStartup (inlined)
56272bd35e89 ServerLoop (inlined)
56272bd35e89 PostmasterMain (bin/postgres)
56272bab7f44 main (bin/postgres)
7f94a1cded09 __libc_start_main (inlined)
56272bab7fe9 _start (bin/postgres)
```

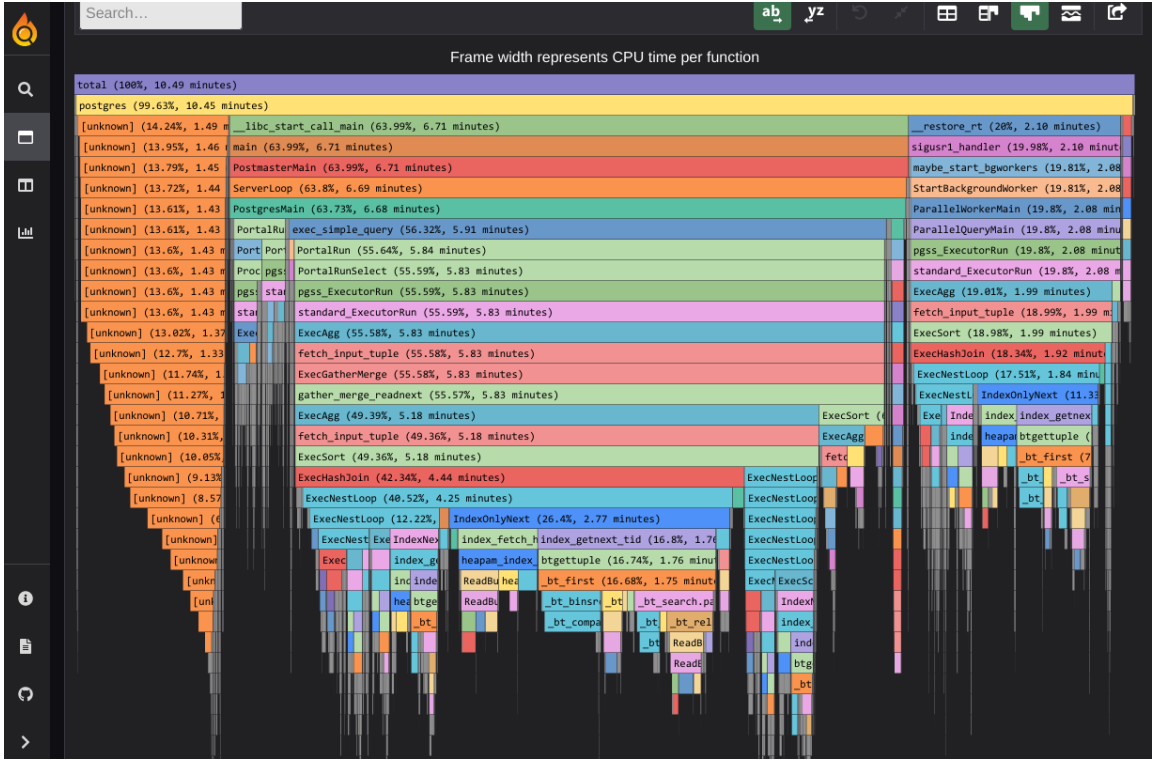
Профилирование Perf + CPU

- FlameGraph

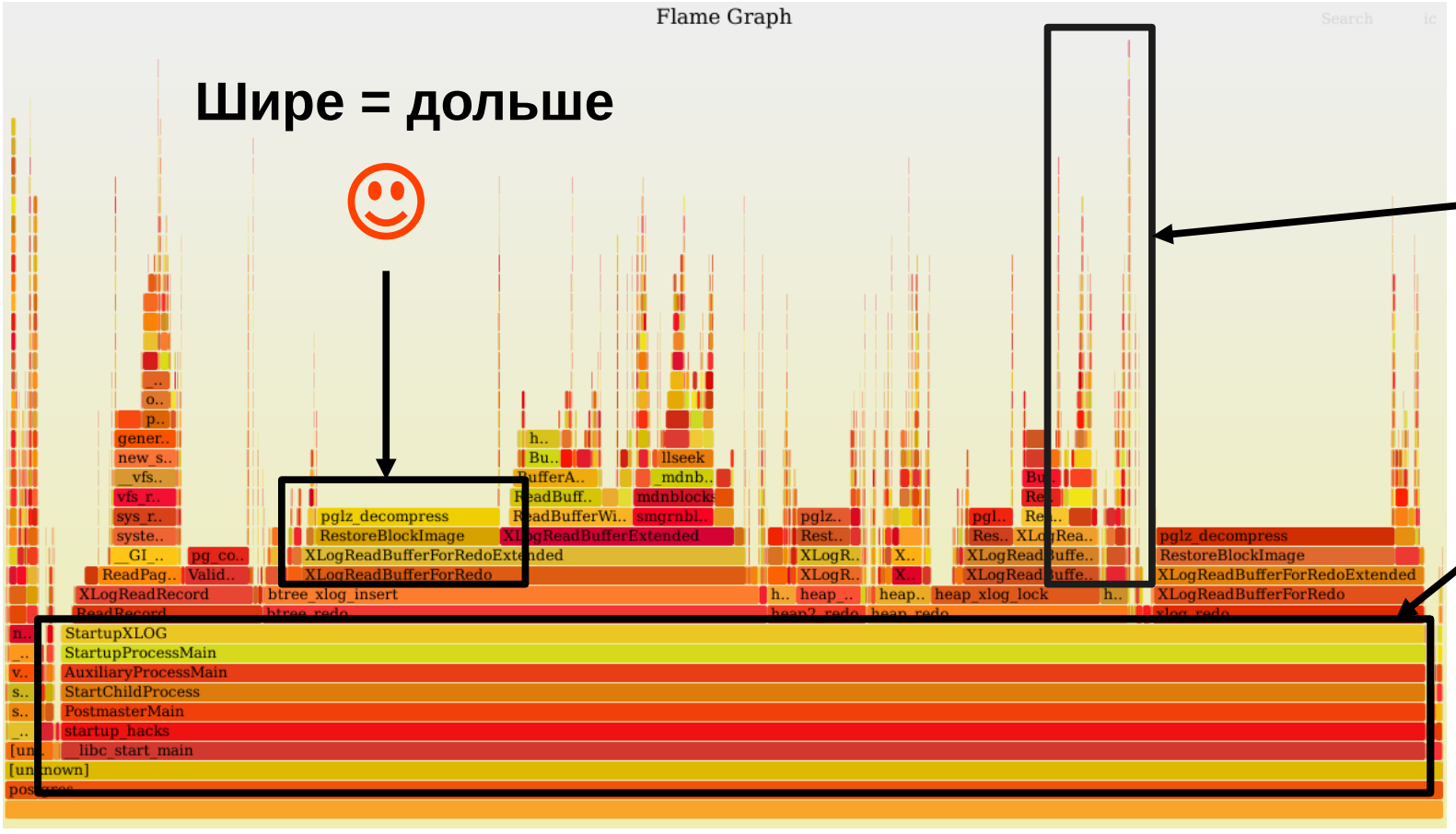
```
# stackcollapse-perf.pl $1 > $1.collapse
# flamegraph.pl $1.collapse > $1.svg
```

Профилирование Perf + CPU

- Continuous Profiling: https://pyroscope.io/

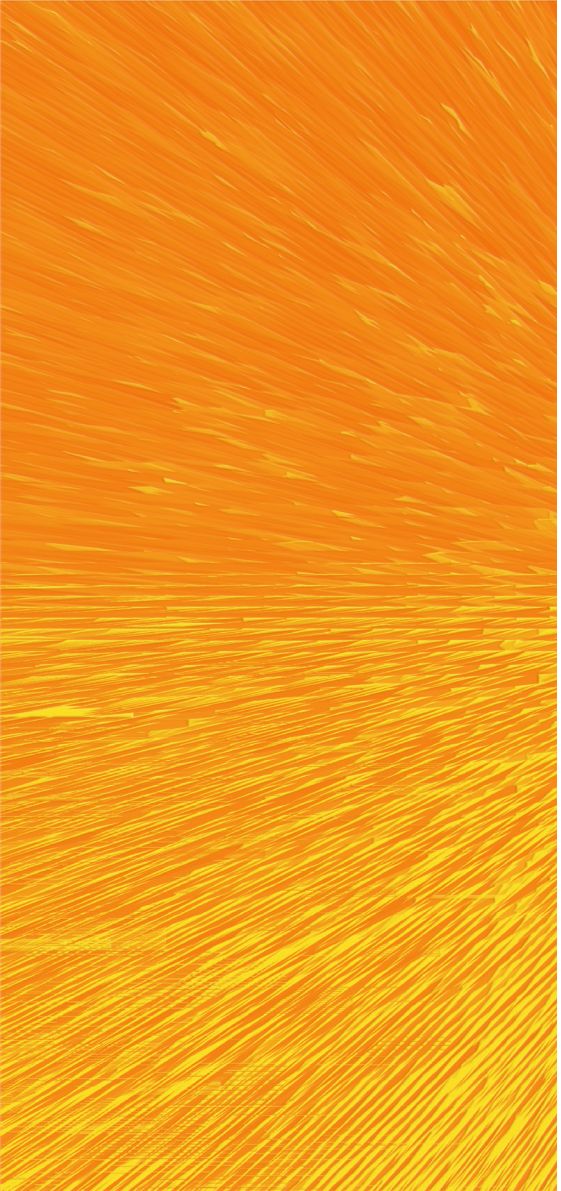


Профилирование Perf + CPU



PostgresPro

Примеры



Distinct
vs
Group By

PostgresPro

Distinct vs Group By

- **select distinct X** или **select X group by X**
- Денис Смирнов (darthunix) в pgsql-hackers от 30.08.2023:

  — Обсуждение

  https://www.postgresql.org/message-id/flat/388BA879-CAEC-4930-A4EE-F06DC1F12C96%40gmail.com

  — Патч

  https://commitfest.postgresql.org/44/4531/

Distinct vs Group By

```
-- создаём таблицу из 1 числовой колонки
create table t(a int, primary key(a));
```

Distinct vs Group By

```
create table t(a int, primary key(a));
-- вставляем несколько миллионов случайных чисел
insert into t select random() * 5000000
from generate_series(1, 5000000)
on conflict do nothing;
```

PostgresPro

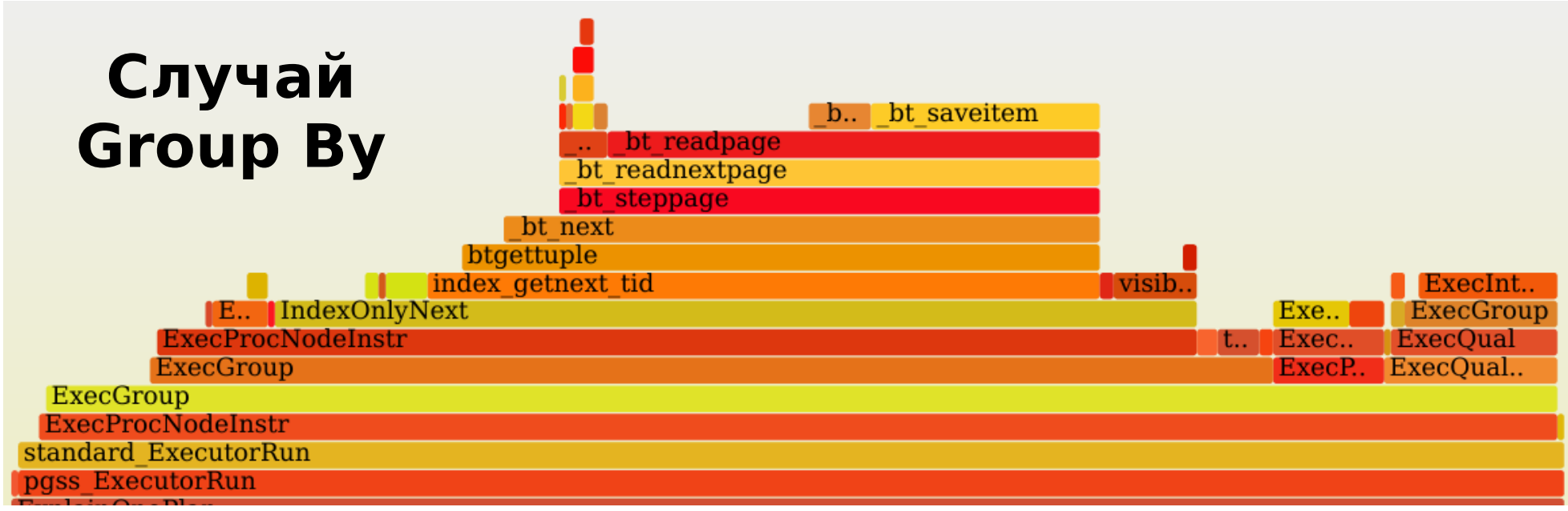
Distinct vs Group By

```
explain analyze select distinct a from t;
latency average = 737.680 ms

pgbench -n -c 1 -j 1 -T 60 -P 1
  -f unique.sql
```

Distinct vs Group By

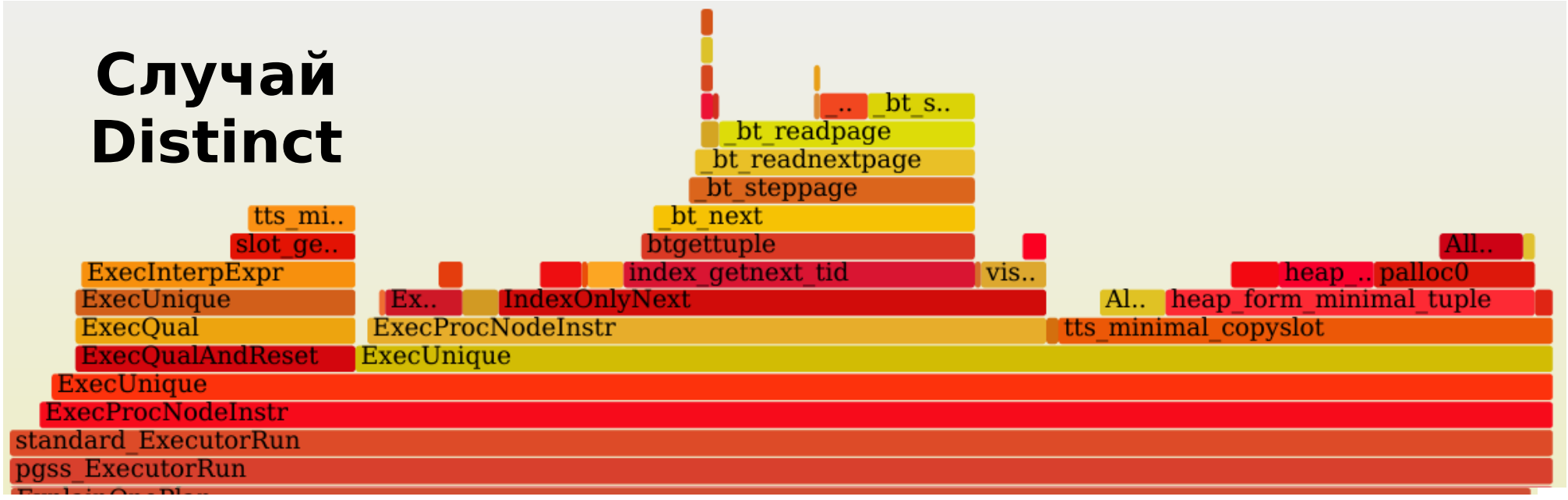
```
explain analyze select distinct a from t;
latency average = 737.680 ms

explain analyze select a from t group by a;
latency average = 633.737 ms
```

Distinct vs Group By

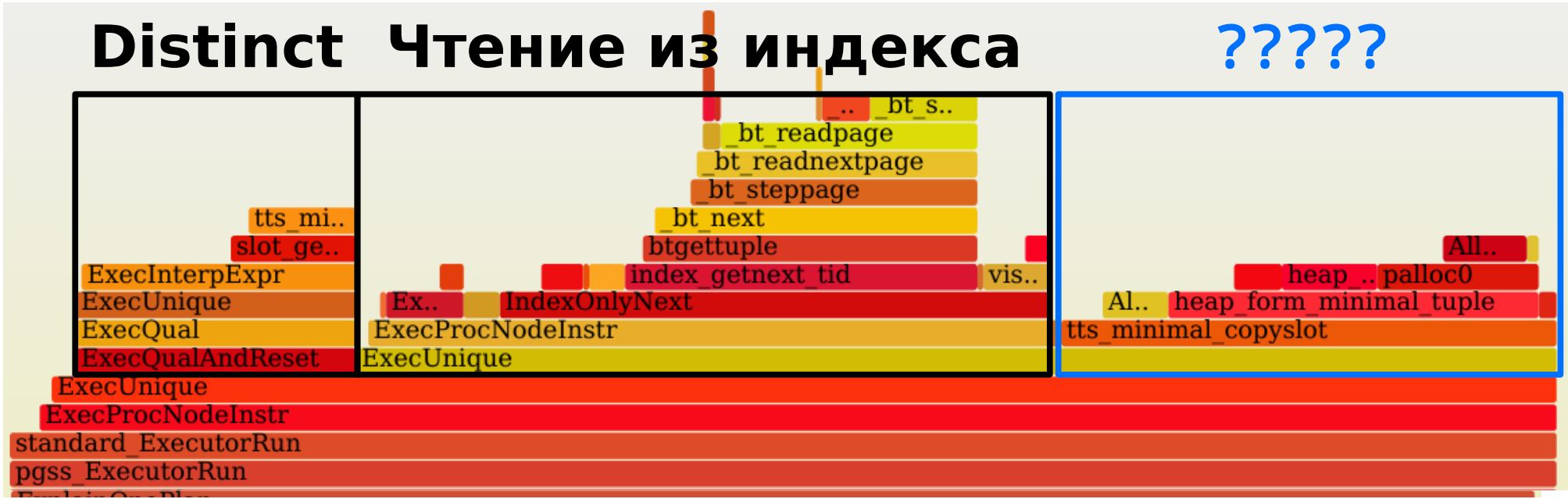


Distinct vs Group By

Distinct vs Group By



Distinct vs Group By



Distinct vs Group By

```
tts_minimal_copyslot: execTuples.c

const TupleTableSlotOps TTSOpsMinimalTuple = {
<...>
        .copyslot = tts_minimal_copyslot,
<...>
};
```

Замечание: данный код создаёт дубль строки в памяти

Distinct vs Group By

```
const TupleTableSlotOps TTSOpsVirtual = {
<...>
        .copyslot = tts_virtual_copyslot,
<...>
};
```

Замечание: данный код создаёт только ссылки

Distinct vs Group By

- Патч от Дениса Смирнова

```
diff --git a/src/backend/executor/nodeUnique.c b/src/backend/executor/nodeUnique.c
index 45035d74fa..c859add6e0 100644
--- a/src/backend/executor/nodeUnique.c
+++ b/src/backend/executor/nodeUnique.c
@@ -141,7 +141,7 @@ ExecInitUnique(Unique *node, EState *estate, int eflags)
         * Initialize result slot and type. Unique nodes do no projections, so
         * initialize projection info for this node appropriately.
         */
-       ExecInitResultTupleSlotTL(&uniquestate->ps, &TTSOpsMinimalTuple);
+       ExecInitResultTupleSlotTL(&uniquestate->ps, &TTSOpsVirtual);
        uniquestate->ps.ps_ProjInfo = NULL;
```

PostgresPro

Медленный Commit

Медленный Commit

```
postgres=# begin;
Time: 0.230 ms
```

Медленный Commit

```
postgres=# begin;
Time: 0.230 ms
postgres=*# select 1 from t_tab_100;
Time: 0.317 ms
```

Медленный Commit

```
postgres=# begin;
Time: 0.230 ms
postgres=*# select 1 from t_tab_100;
Time: 0.317 ms
postgres=*# commit;
Time: 180.267 ms
```

Медленный Commit

несколько минут ранее...

Медленный Commit

несколько минут ранее...
создали 1000 временных таблиц

Медленный Commit

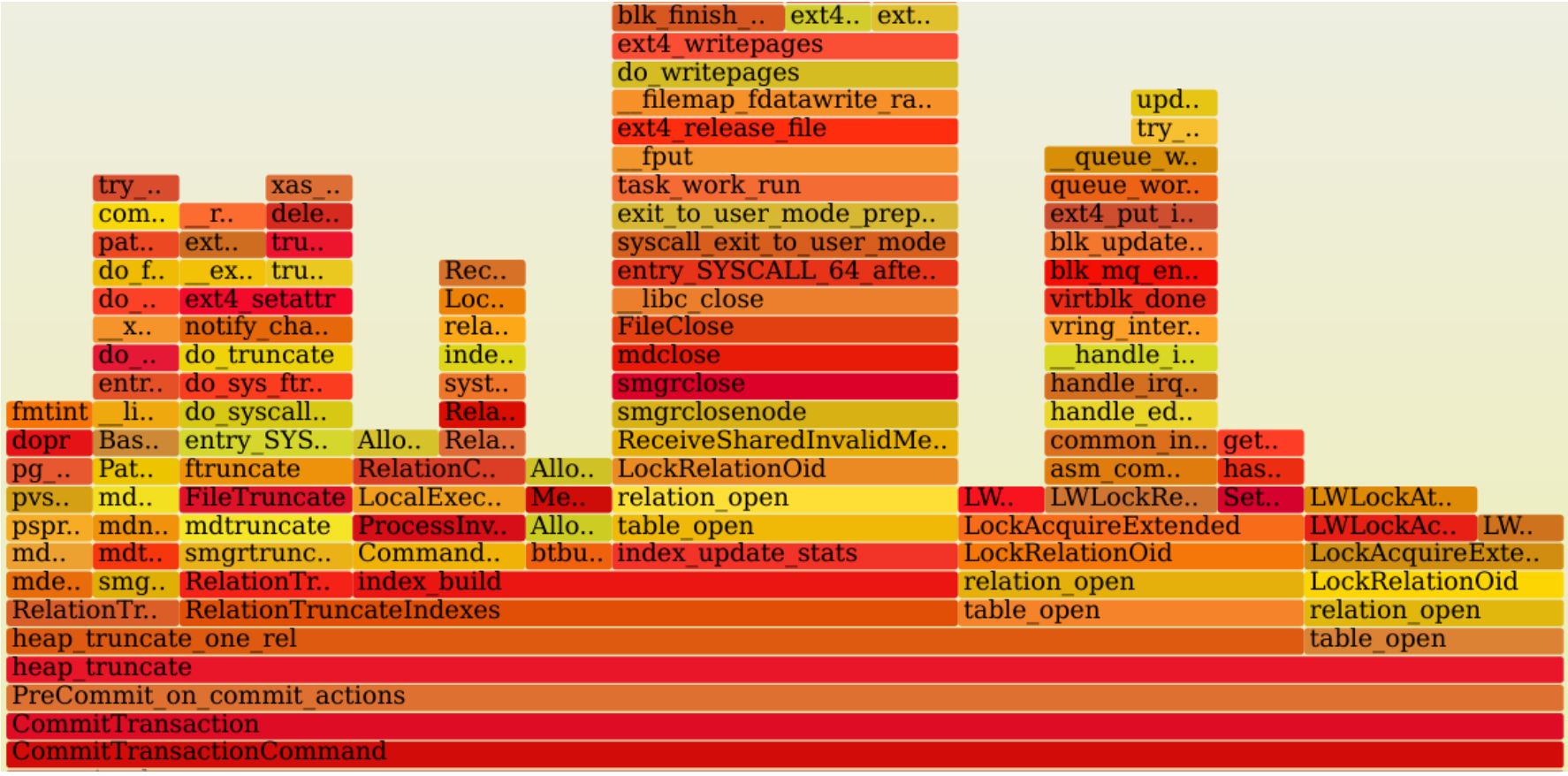
```
-- 1000 временных таблиц
create temp table t_tab_%s
   on commit delete rows
as
  select id::bigint, repeat('x', 1023)
    from generate_series(1,1000) id;
...
```

Медленный Commit

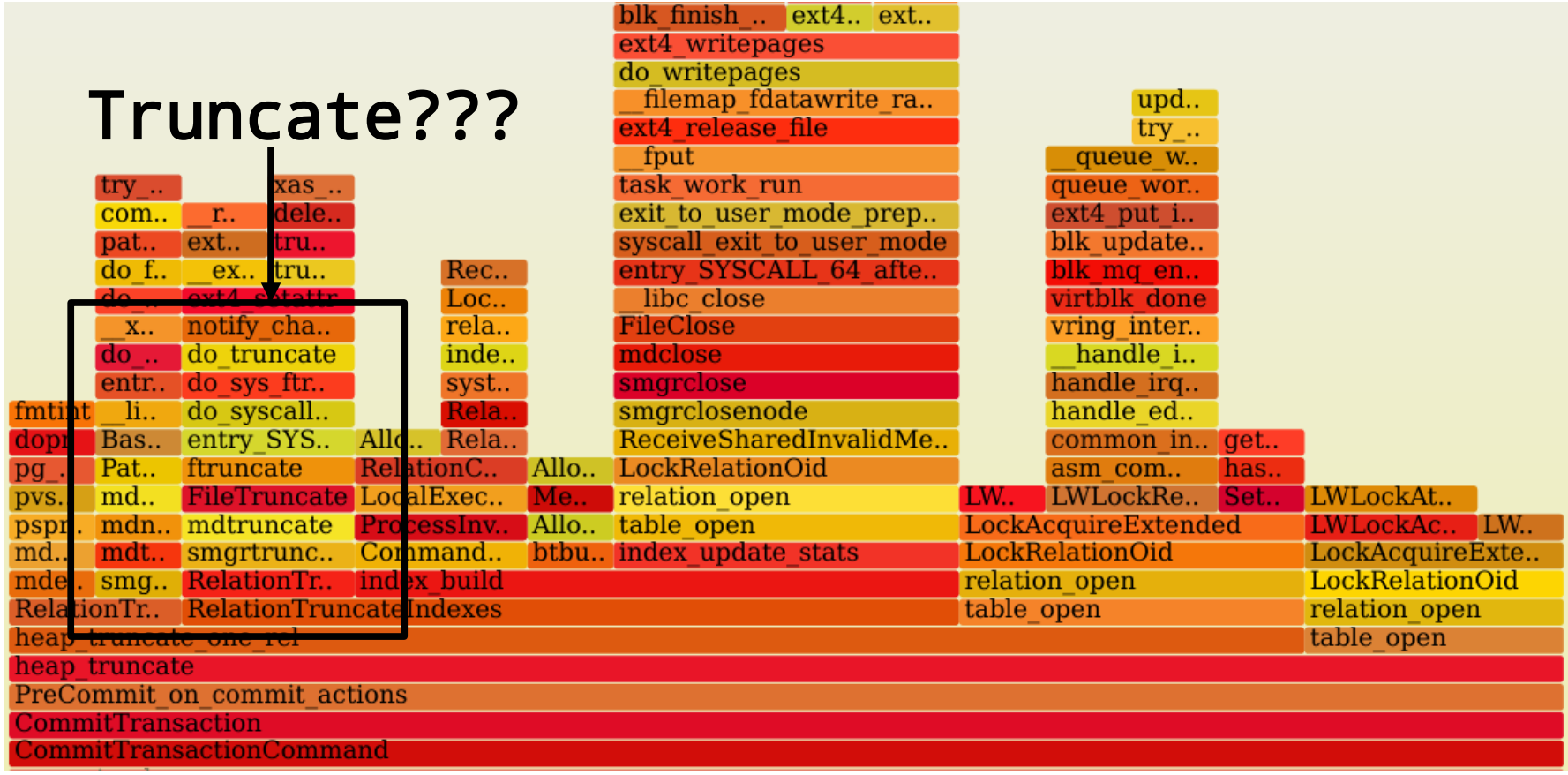
```
postgres=# begin;
Time: 0.230 ms
postgres=*# select 1 from t_tab_100;
Time: 0.317 ms
postgres=*# commit;
Time: 180.267 ms
```

Disk??? Нет, CPU!

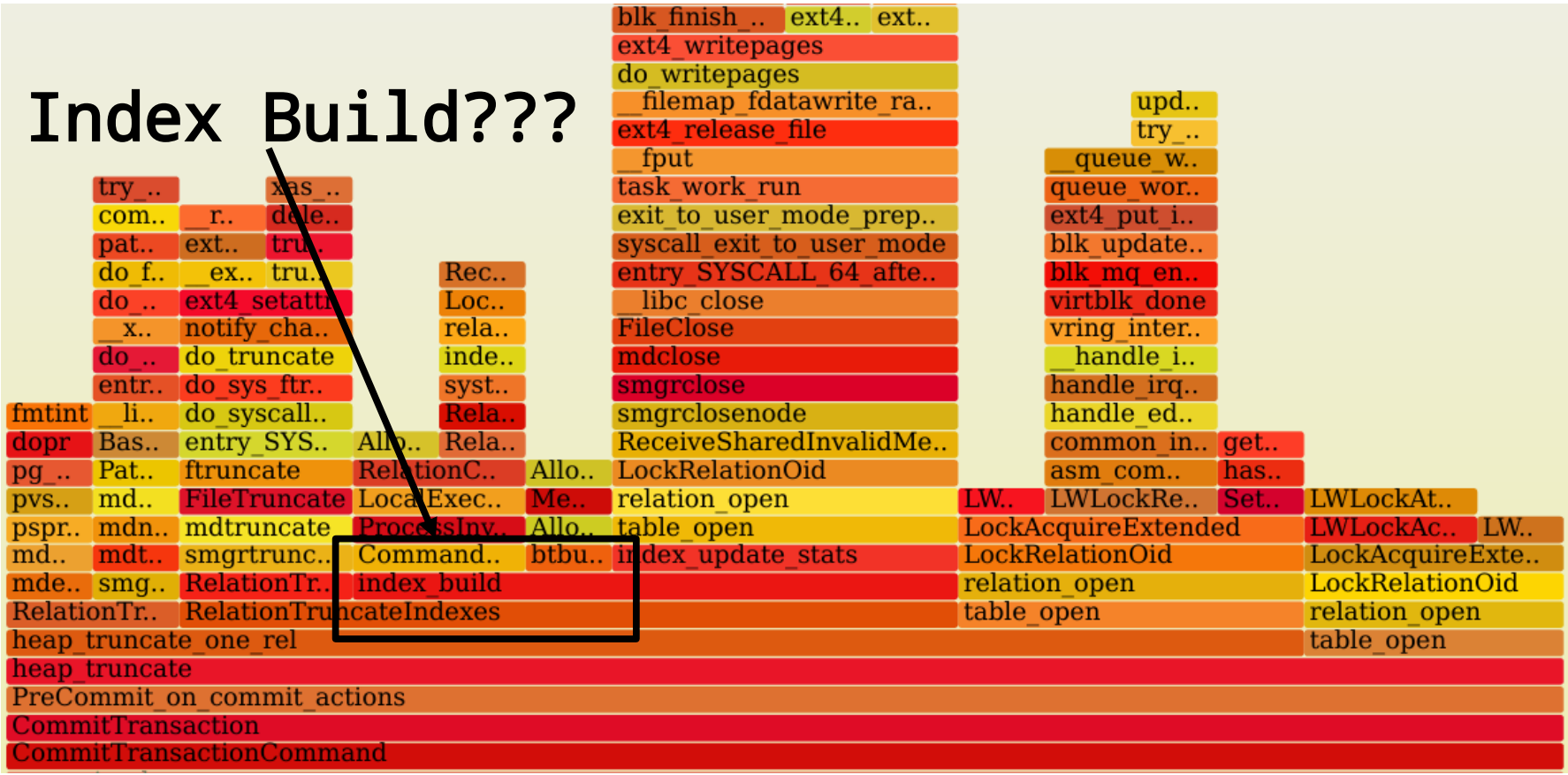
Медленный Commit



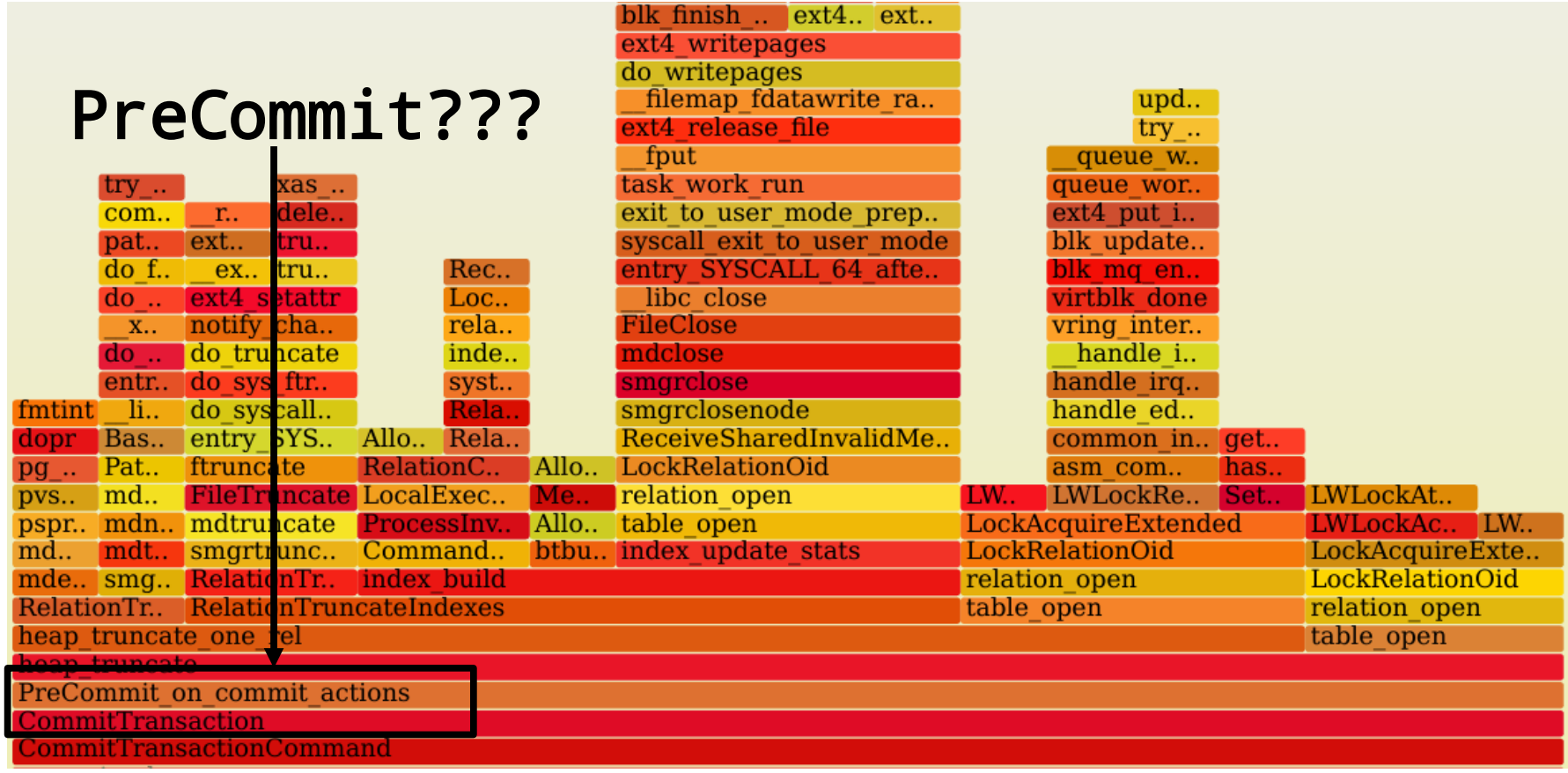
Медленный Commit



Медленный Commit



Медленный Commit



Медленный Commit

1000 tables on commit delete rows

```
precommit:
for (i = 0; i < 1000; i++) {
   truncate table t_tab_%i
}
```

Вот и причина...

PostgresPro

1,000,000 таблиц

1,000,000 таблиц

- pgio: Silly Little Oracle Benchmark (SLOB) for PostgreSQL

  https://github.com/therealkevinc/pgio

1,000,000 таблиц

- pgio: Silly Little Oracle Benchmark (SLOB) for PostgreSQL
  https://github.com/therealkevinc/pgio
- По началу побежало всё быстро

1,000,000 таблиц

- pgio: Silly Little Oracle Benchmark (SLOB) for PostgreSQL
  https://github.com/therealkevinc/pgio
- По началу побежало всё быстро
- Через пару часов всё затупило

1,000,000 таблиц

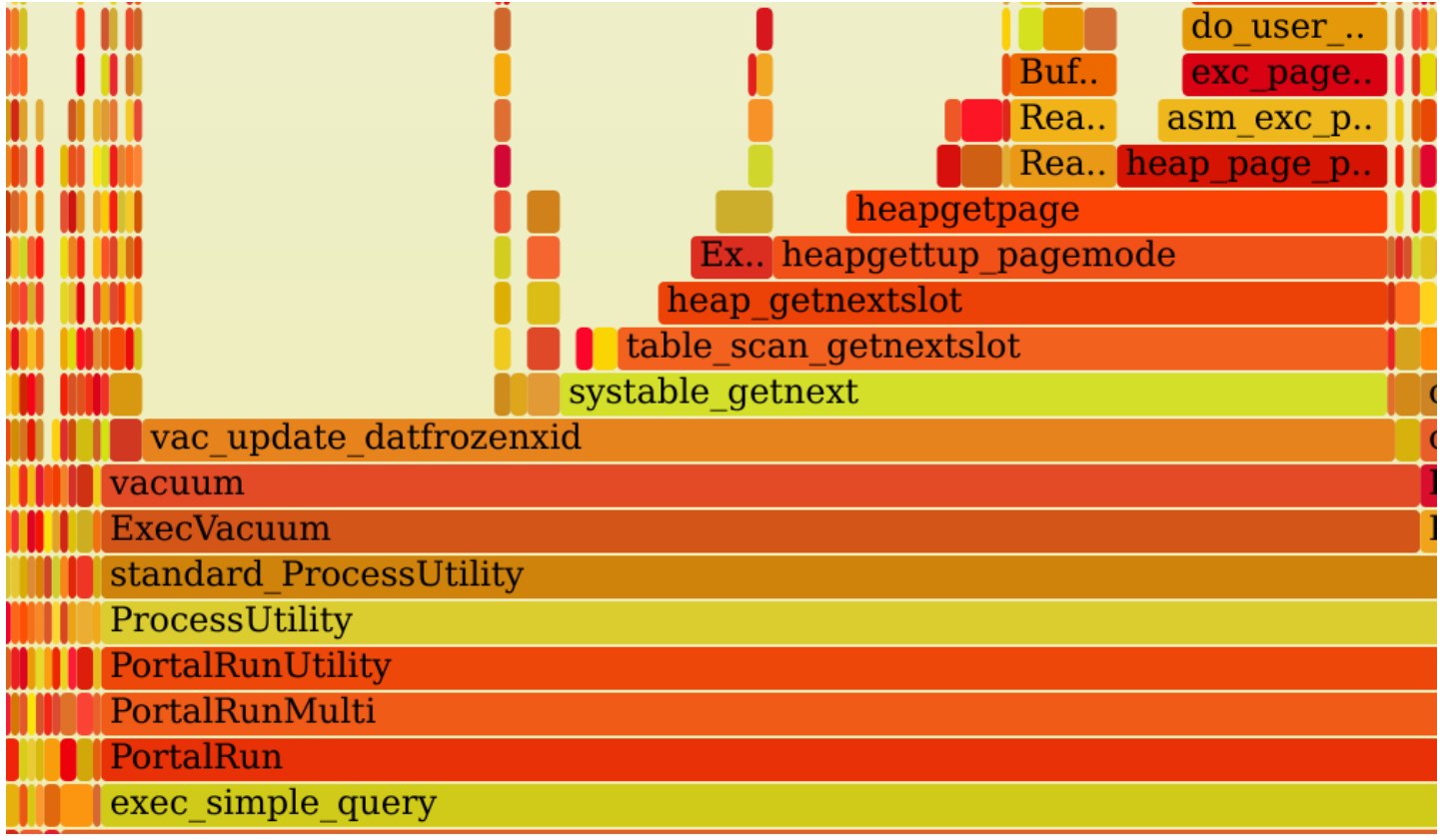
- pgio: Silly Little Oracle Benchmark (SLOB) for PostgreSQL
  https://github.com/therealkevinc/pgio
- По началу побежало всё быстро
- Через пару часов всё затупило
- Через сутки всё встало колом

1,000,000 таблиц

pg_stat_activity:

```
      wait_event       |                         query
-----------------------+-----------------------------------------------
 AutoVacuumMain        |
 LogicalLauncherMain   |
 frozenid              | autovacuum: ANALYZE public.pgio88012
 frozenid              | autovacuum: ANALYZE public.pgio551406
 frozenid              | autovacuum: ANALYZE public.pgio115257
 frozenid              | vacuum (analyze) pgio88029;
 frozenid              | vacuum (analyze) pgio88030;
 frozenid              | vacuum (analyze) pgio88031;
...
 frozenid              | vacuum (analyze) pgio115271;
 frozenid              | vacuum (analyze) pgio115273;
 frozenid              | vacuum (analyze) pgio115274;
 frozenid              | vacuum (analyze) pgio551418;
 BgWriterMain          |
 CheckpointWriteDelay  |
 WalWriterMain         |
(81 rows)
```

1,000,000 таблиц



1,000,000 таблиц

- vac_update_datfrozenxid
  - вызывается на каждый vacuum
  - вычисляет минимальное значение frozenxid по всем таблицам в базе
  - делает полное сканирование таблицы pg_class
- Фикс скоро будет ☺

PostgresPro

Index Scan в NestedLoop

Index Scan в Nested Loop

- Допустим, есть маленькая таблица (20 строк) и индекс

```
create table t (id bigint, value text);
create index t_idx on t (id);


insert into t
select i as id, md5(i*i || 'HASH') as value
  from generate_series(1,20) i;
```

Index Scan в Nested Loop

- Есть запрос с Nested Loop (5 миллионов циклов) по ней:

```
set enable_hashjoin = off;
set enable_mergejoin = off;
set work_mem = 100000000;


select t.value
  from t, generate_series(1,5000000) i
 where t.id = i + 1000
 limit 10;
```

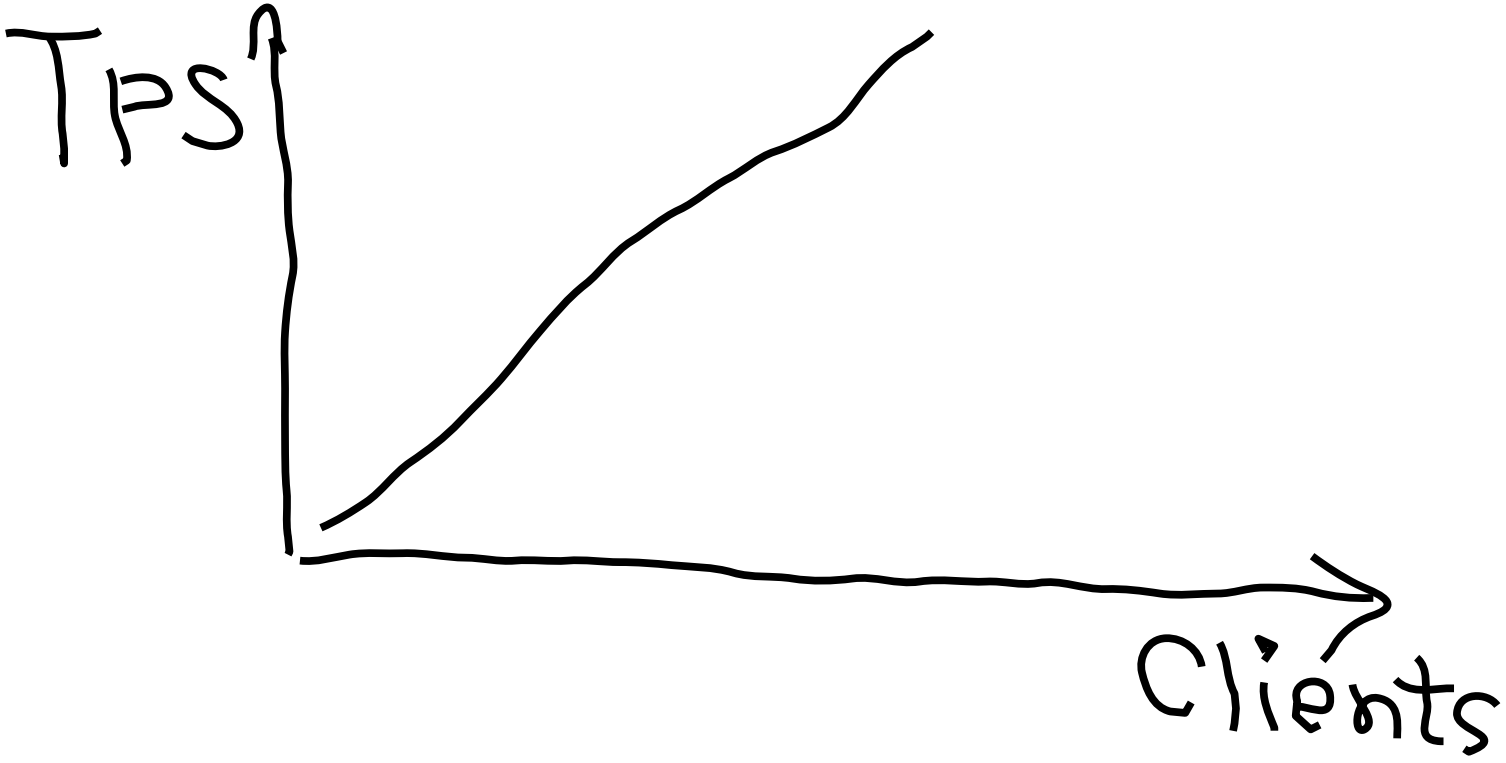
Index Scan в Nested Loop

```
Limit (actual rows=0 loops=1)
  -> Nested Loop (actual rows=0 loops=1)
     -> Function Scan on generate_series i (actual rows=5000000 loops=1)
     -> Index Scan using t_pk on t (actual rows=0 loops=5000000)
          Index Cond: (id = (i.i + 1000))
 Planning Time: 0.150 ms
 Execution Time: 2717.768 ms
```

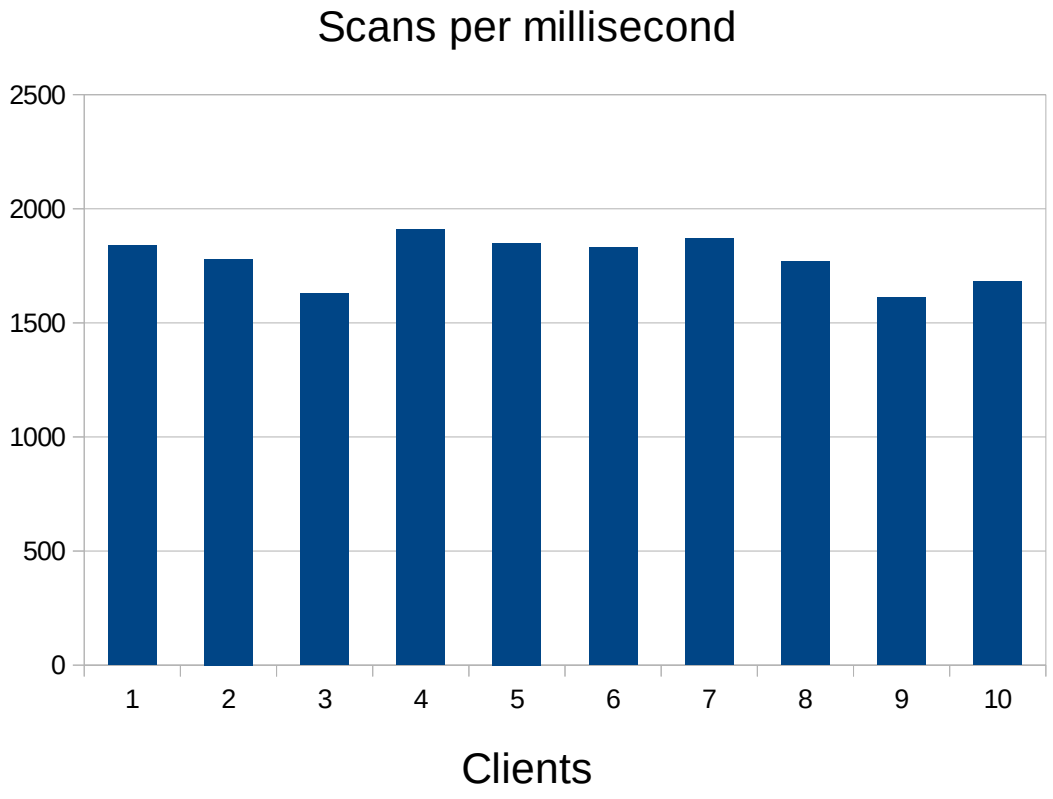
Index Scan в Nested Loop

```
5,000,000 / 2717 =    1840 итераций за 1 миллисекунду
                      1.8 итераций за 1 микросекунду
                      550 наносекунд на 1 итерацию выборки из индекса!
```

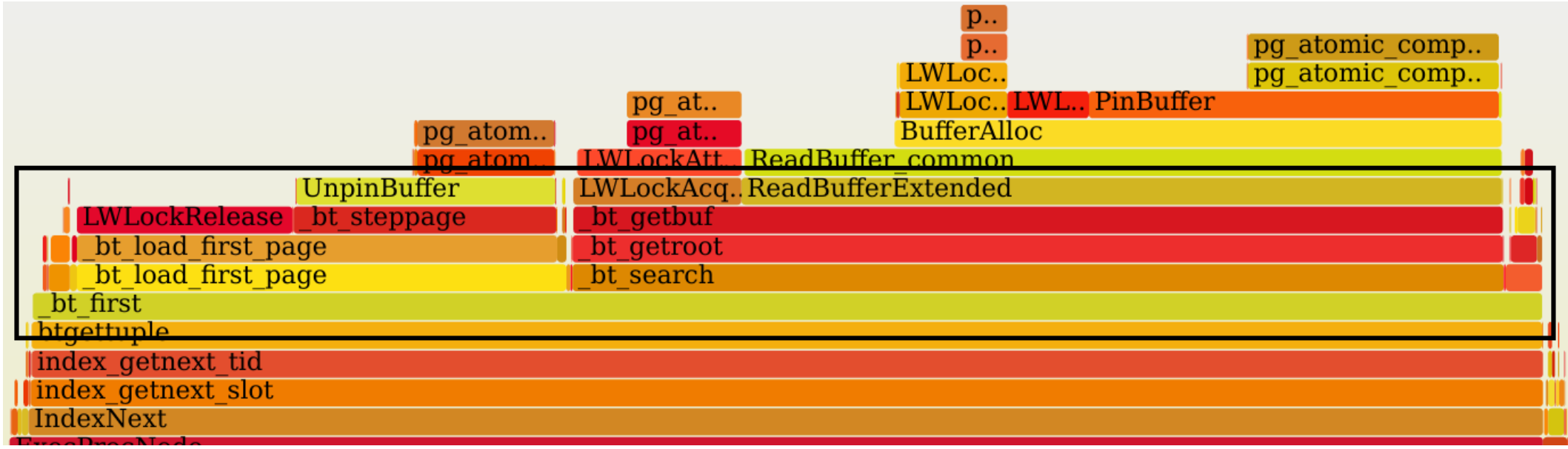
Index Scan в Nested Loop



Index Scan в Nested Loop

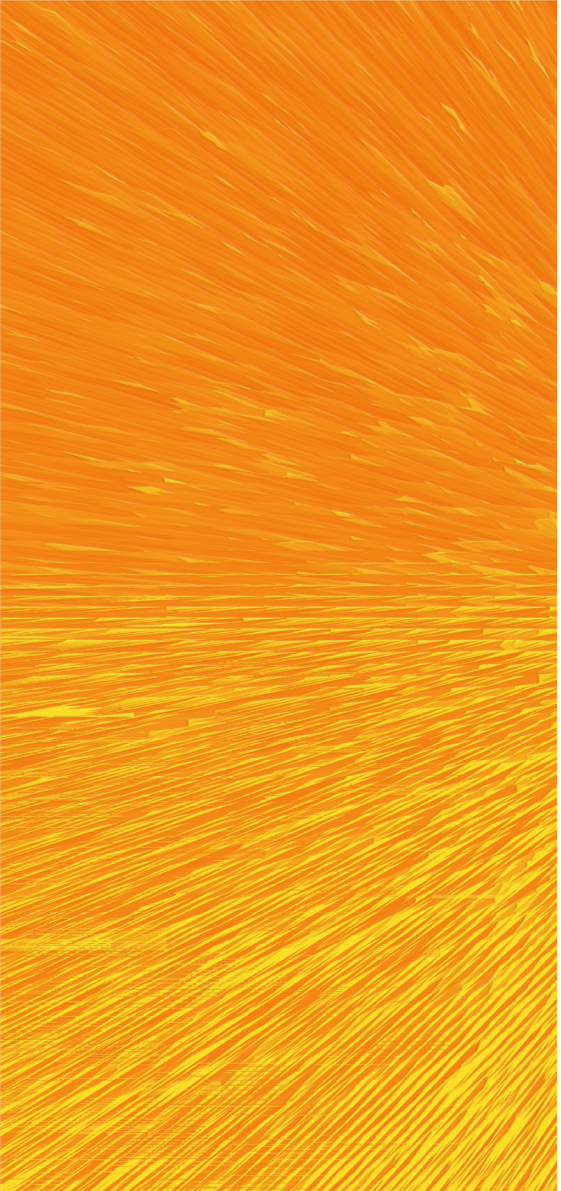


Index Scan в Nested Loop



Index Scan в Nested Loop

- _bt_load_first_page / _bt_get_root
  - корневая страница B-Tree дерева
  - она же и единственная
  - высокая конкуренция
- Фикс… пока сложно
- Проще удалить индекс



Всё?

Итоги

- Тесты без анализа — время на ветер!
- Используйте исторический мониторинг
- Не бойтесь профилировать и докопаться до правды
- Метрики, flamegraph-ы – лишь инструменты анализа

PostgresPro

Михаил Жилин

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