PGConf.Сибирь 2023

PostgresPro

# Способы решения проблем PostgreSQL путём отладки и профилирования

Михаил Жилин

# План встречи

- Проблемы и что с ними делать?
- Пара слов о PostgreSQL и об OS
- Профилирование
- Примеры

# Проблемы

- Функциональные
  - Всё сломалось
  - Ошибки в логах
  - Ошибки в ответах системы
- Первые шаги
  - Код ошибки
  - Сообщение об ошибке
  - Stack traces

PostgresPro

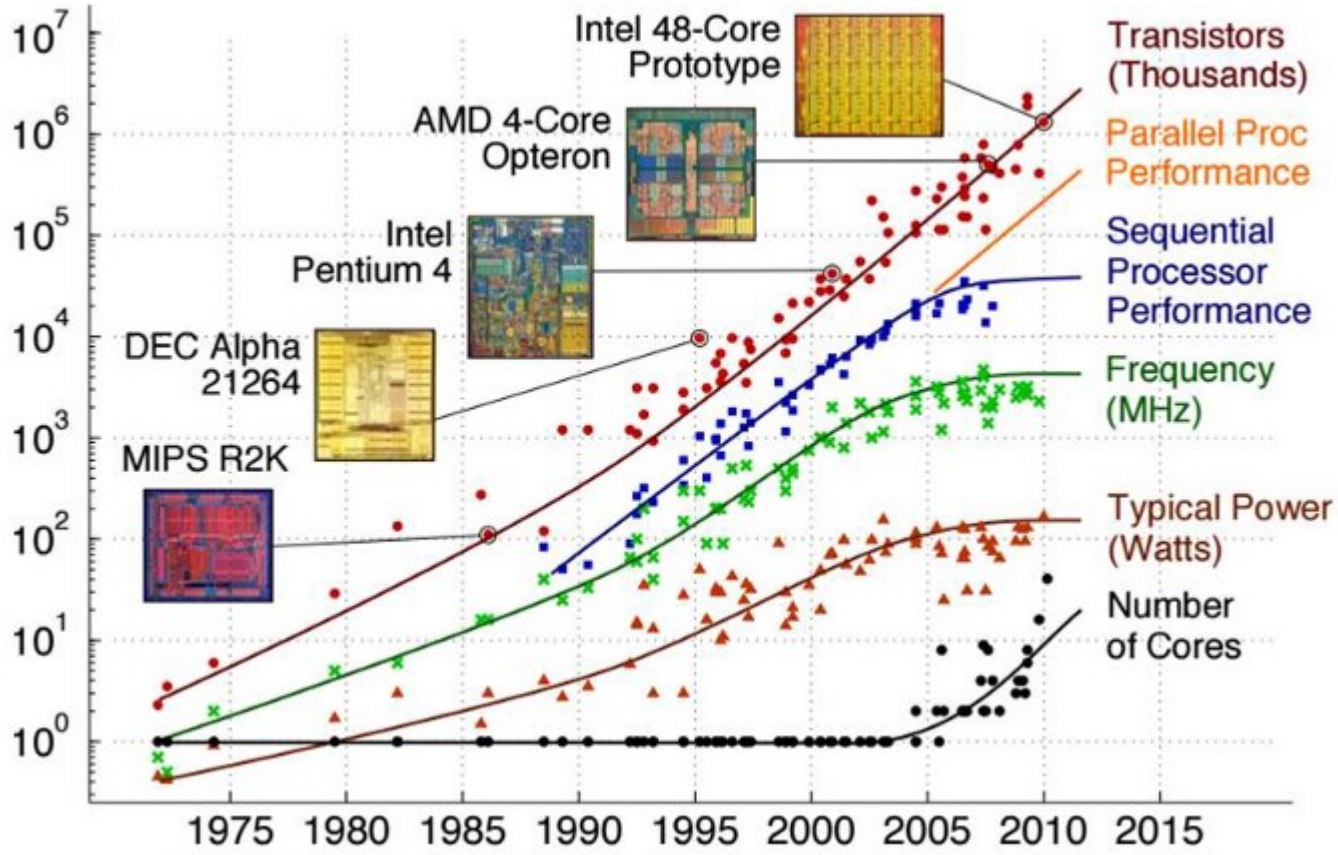
# Проблемы

- Нефункциональные
  - Тормоза
  - Нехватки ресурсов, частичные отказы системы
- Houston, we have problems with performance

# Проблемы

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

# Что делать?




Data partially collected by M. Horowitz, F. Labonte, O. Shacham, K. Olukotun, L. Hammond

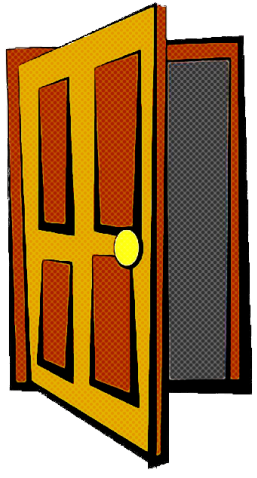
PostgresPro

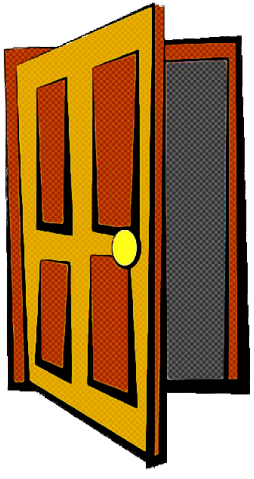
# Что делать?

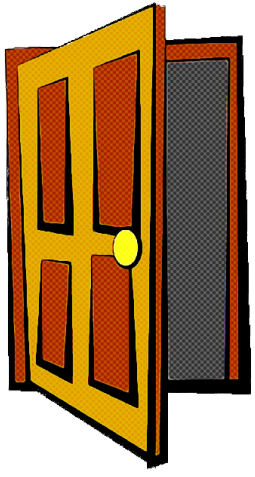
- Сложность растёт
  - SMT → SMP → NUMA
  - ISA → PCI → PCIx
  - Железо → HyperVizor
  - Local Storage → SAN → SDS
  - VLAN/Routing → Fabrics → SDN (VNF)
  - Standalone Database → Replication → Sharding

# Что делать?

- Узких дверей становится больше...







# Что делать?

- Узких дверей становится больше...

# Что делать?

- Узких дверей становится больше...

# Что делать?

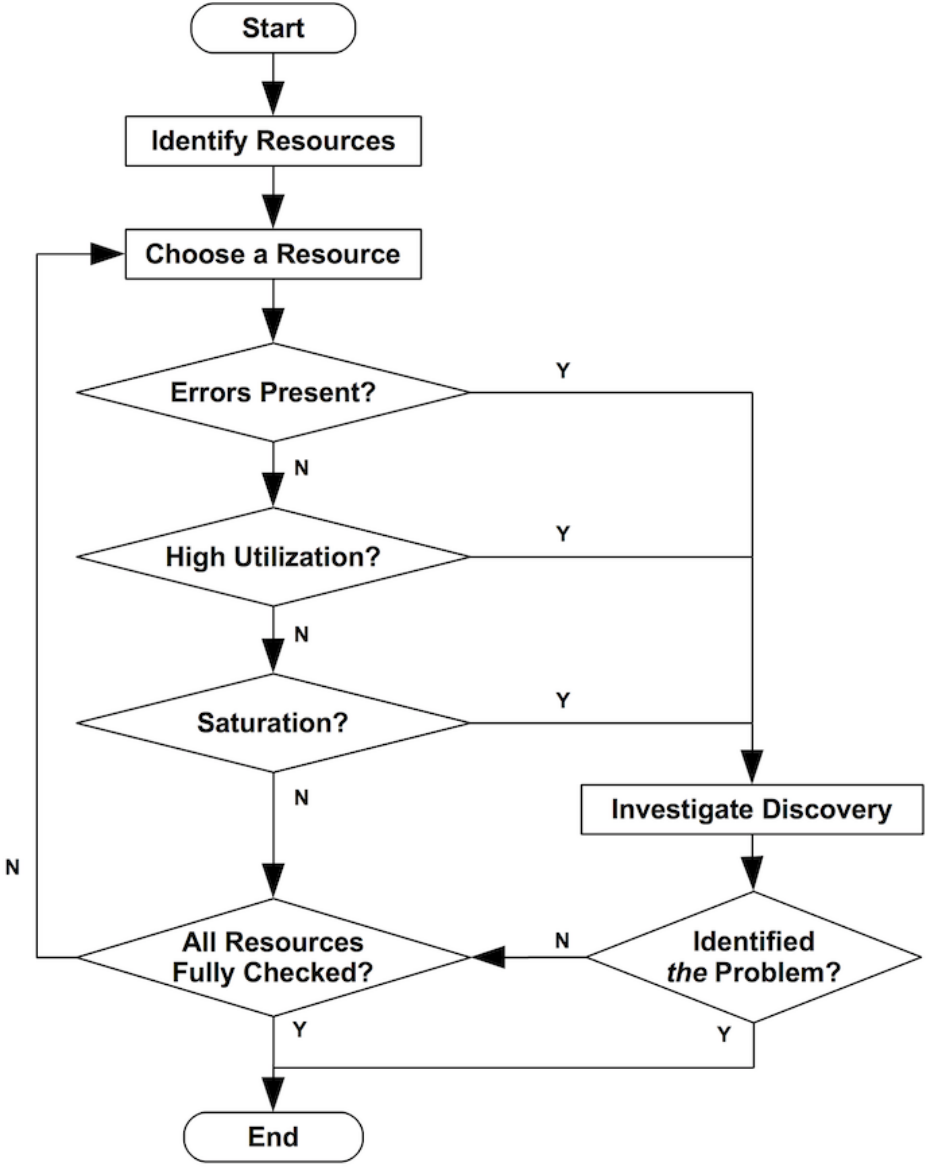
- Узких дверей становится слишком много...

PostgresPro

# Что делать?

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

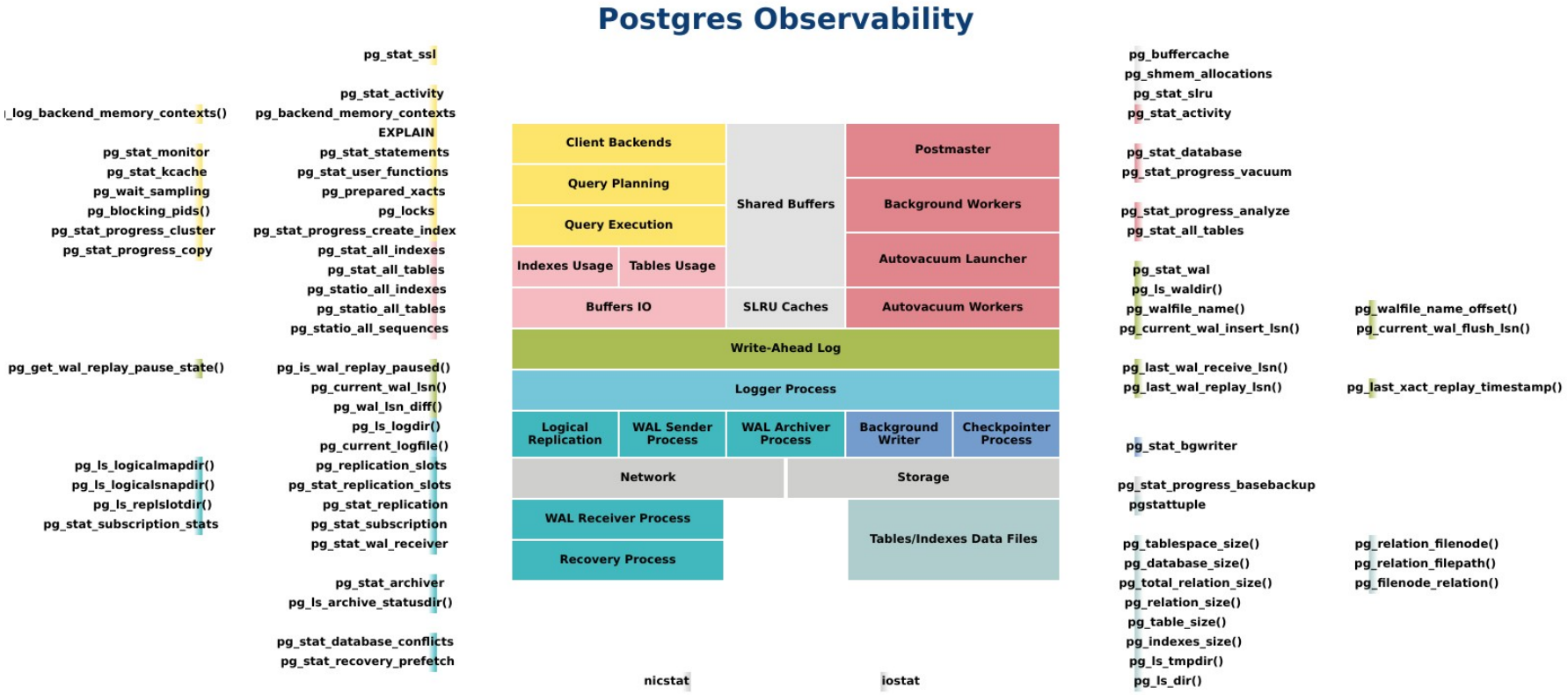
# Что делать?


Start
Identify Resources
Choose a Resource
Errors Present?
Y
N
High Utilization?
Y
N
Saturation?
Y
N
Investigate Discovery
N
All Resources Fully Checked?
N
Identified *the* Problem?
Y
Y
End

# Пара слов о PostgreSQL

- Собственные ресурсы
  - Shared Buffers
  - WAL
  - Autovacuum
- Ресурсы OS
  - Диск
  - CPU
  - Network
  - mem

# Пара слов о PostgreSQL




https://pgstats.dev/

# Пара слов о PostgreSQL

- **pg_stat_activity** – список сессий и их состояние
  - *state, wait_event* – блокировки / много ел / узкие двери?

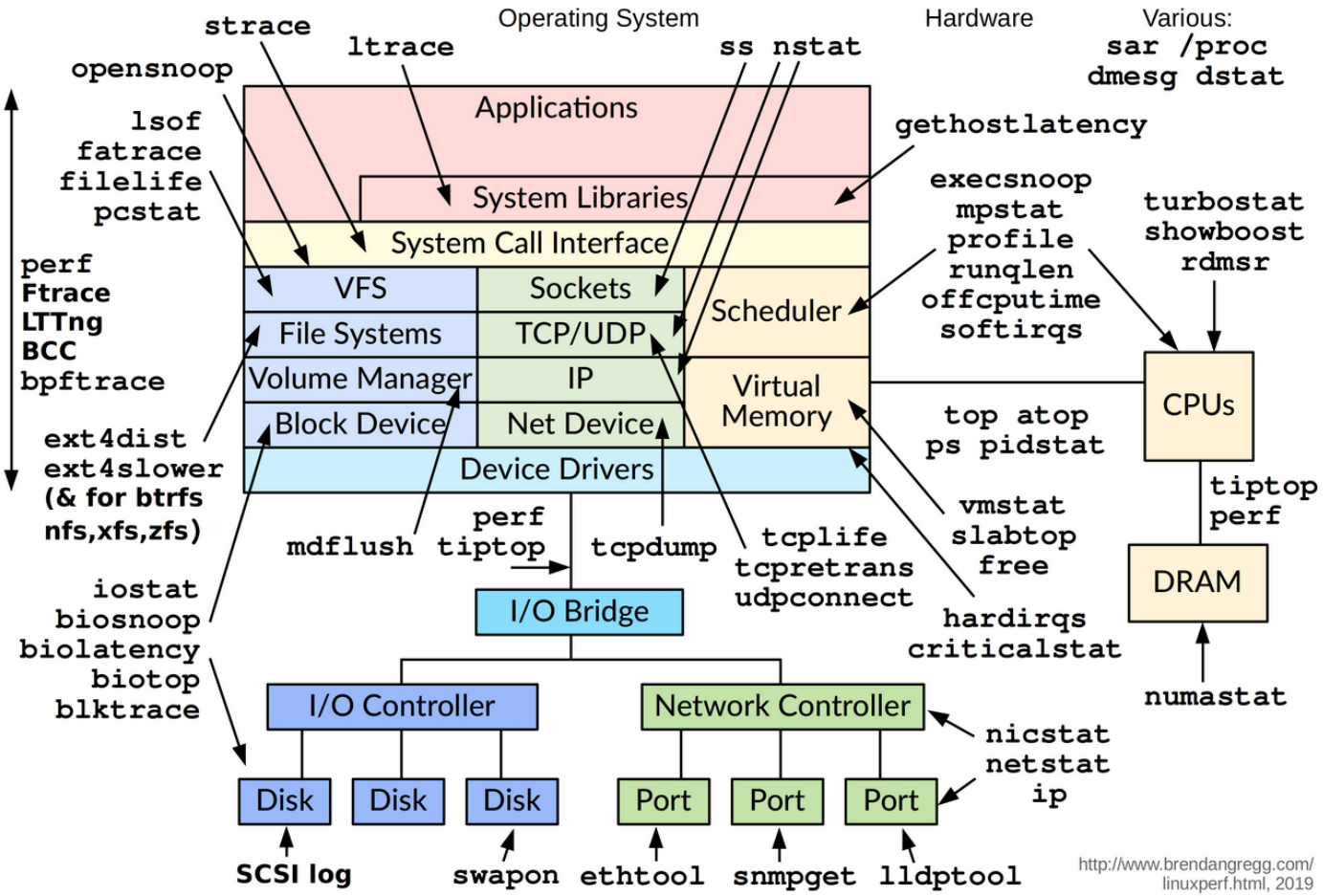
# Пара слов о PostgreSQL

- **pg_stat_statements** – статистика по отдельным запросам
  - *planning / execution time* – топ запросов
- **explain analyze / pgpro_stats** – получения плана запроса
  - *SQL Tuning*
  - https://use-the-index-luke.com/

USE THE INDEX, LUKE!

A Guide to Database Performance for Developers

# Пара слов об OS




http://www.brendangregg.com/linuxperf.html, 2019

# Пара слов об OS

- top -c
  - workload и утилизация системы
- iostat -x
  - workload и утилизация диска
- mpstat, vmstat, netstat
  - workload и утилизация процессора / памяти / сети

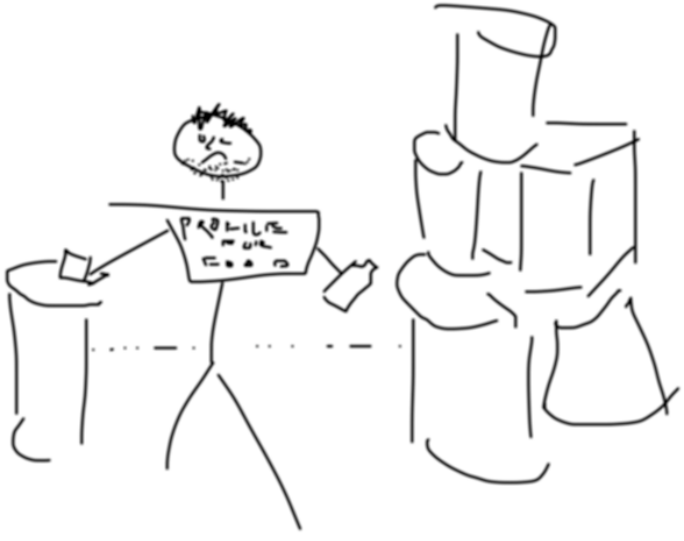
PostgresPro

# Промежуточный итог

- Формулировку причины
  - массовый поток запрос X
  - потребляет много ресурсов Y
- Увы, нам не помог SQL Tuning и тюнинг базы и OS
- Что не хватает?
  - Профилирования!

# Профилирование

- gdb (lldb)
  - attach подключиться к процессу
  - bt стек вызова
  - print вывод данных из памяти
- Poor man’s profiler



# Профилирование

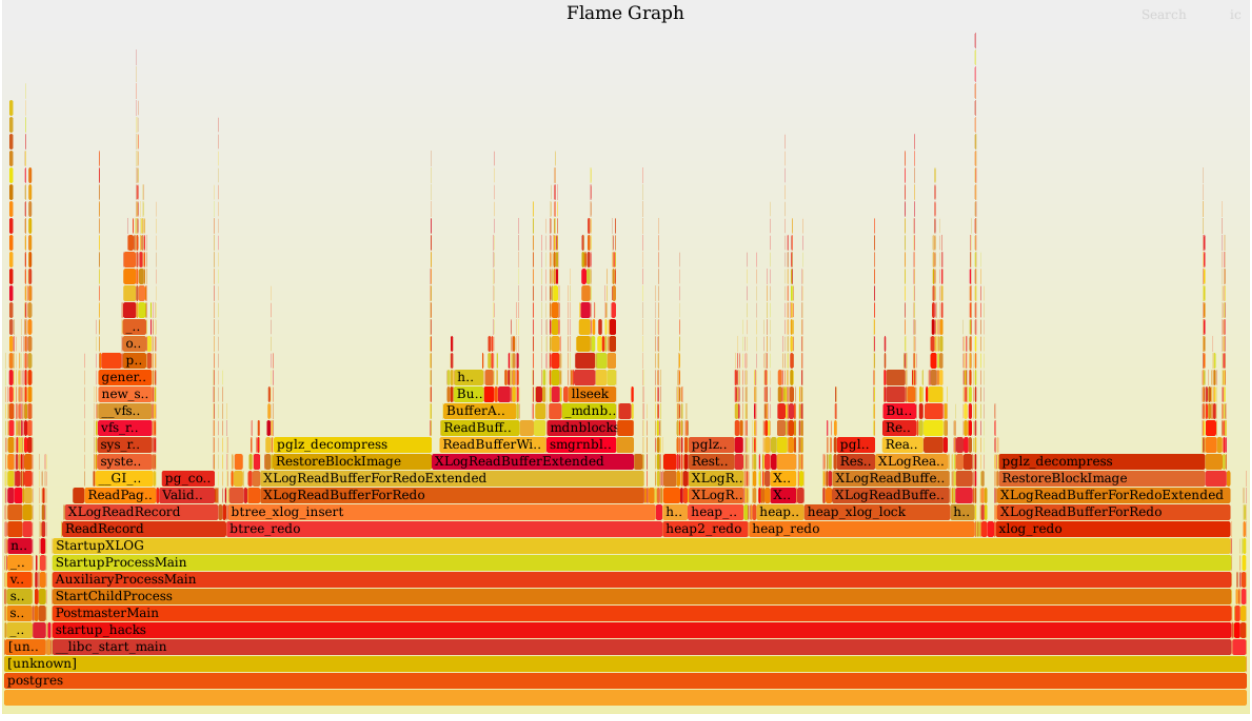
- Инструменты
  - Execution (CPU): DTrace, perf , eBPF
  - Disk (Block devices): blktrace, ioprof, eBPF
  - Memory: Valgrind, perf, eBPF
  - Network: tcpdump, wireshark, eBPF
- Способы
  - sampling – снимки с определенной частотой
  - timing (call/exit) – замер времени

# 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)

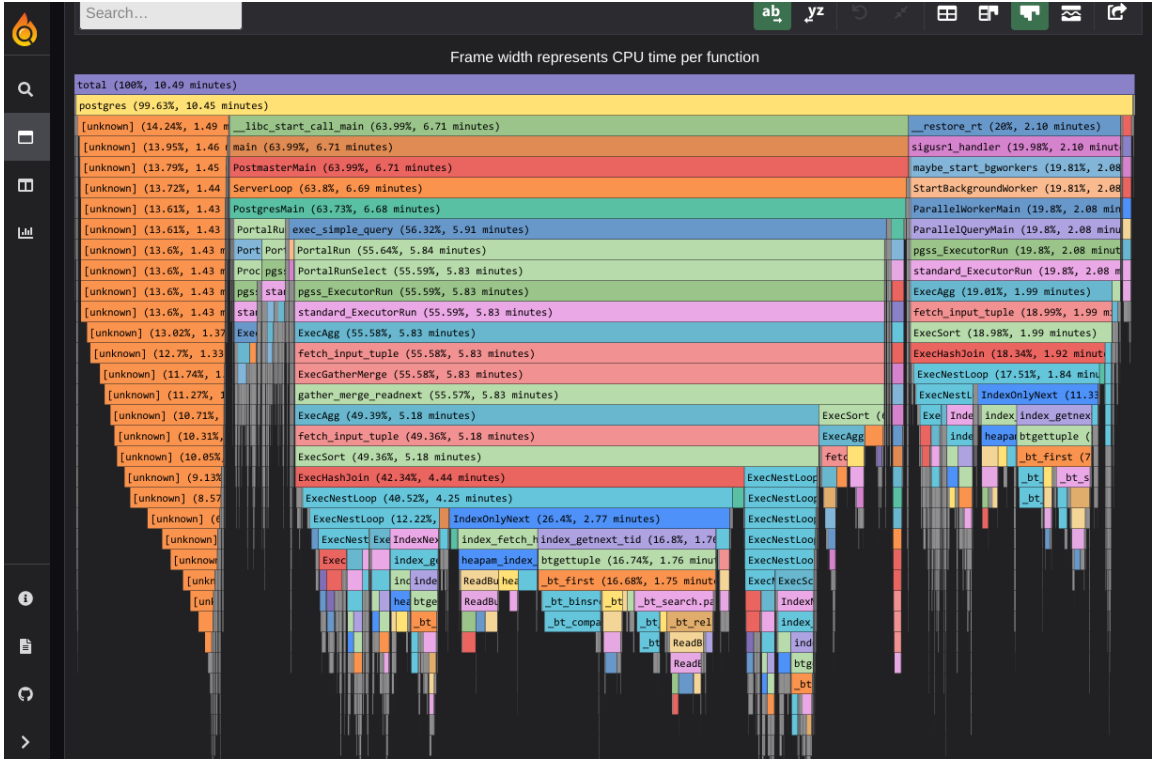
# Профилирование Perf + CPU

- FlameGraph



# Профилирование Perf + CPU

- Grafana-like - https://pyroscope.io/



# Примеры

# Пример №1: Unique vs Group By

- **select distinct X** или **select X group by X**
- Денис Смирнов (darthunix) в pgsql-hackers от 30 августа 2023:
  - Обсуждение https://www.postgresql.org/message-id/flat/3[illegible] EC-4930-A4EE-F06DC1F12C96%40gmail.com
  - Патч https://commitfest.postgresql.org/44/4531/

# Пример №1: Unique vs Group By

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

# Пример №1: Unique 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

# Пример №1: Unique vs Group By

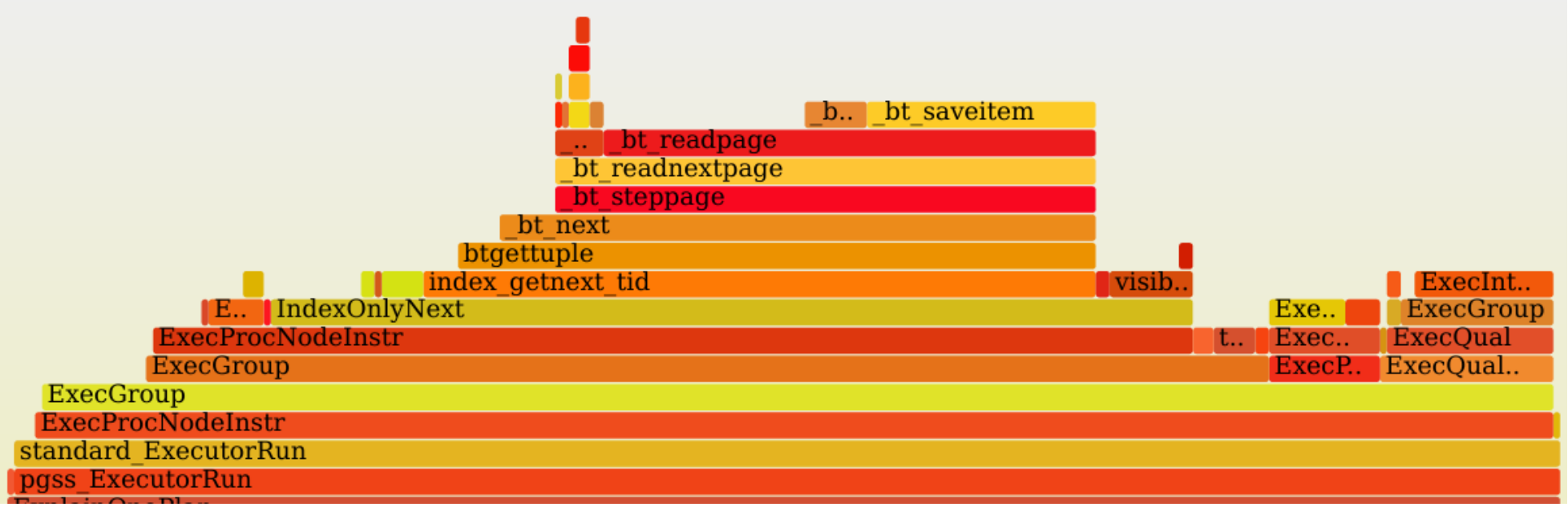
```
-- Unique
explain analyze select distinct a from t;
latency average = 737.680 ms

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

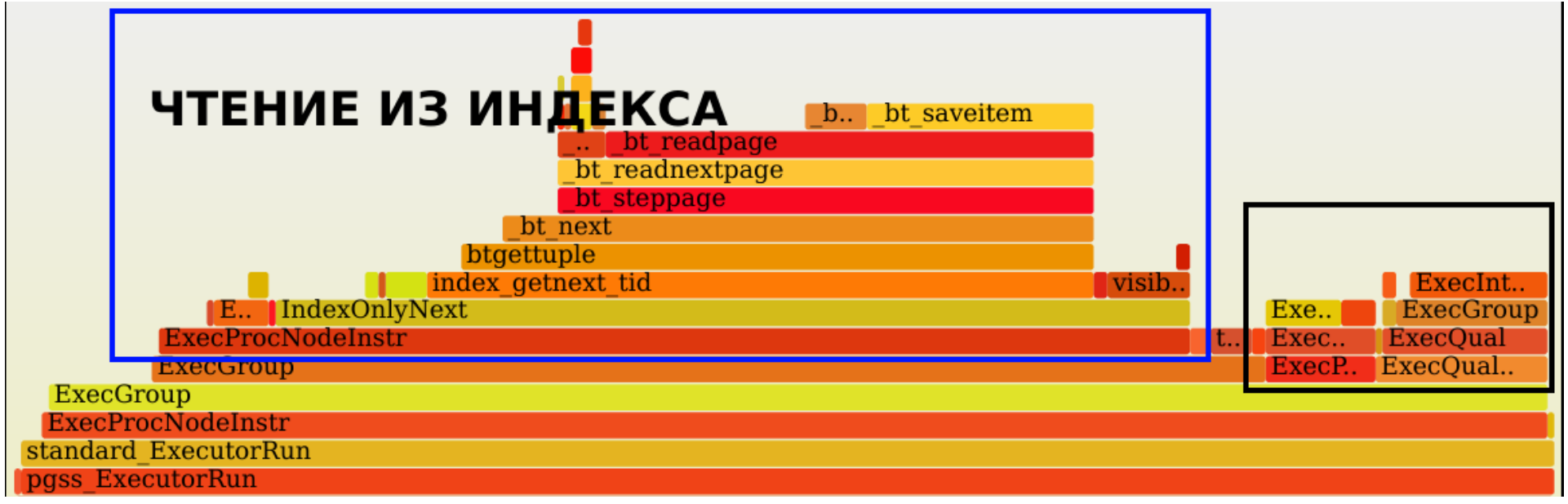
# Пример №1: Unique vs Group By

```
-- Unique
explain analyze select distinct a from t;
latency average = 737.680 ms
-- Group By
explain analyze select a from t group by a
latency average = 633.737 ms
```

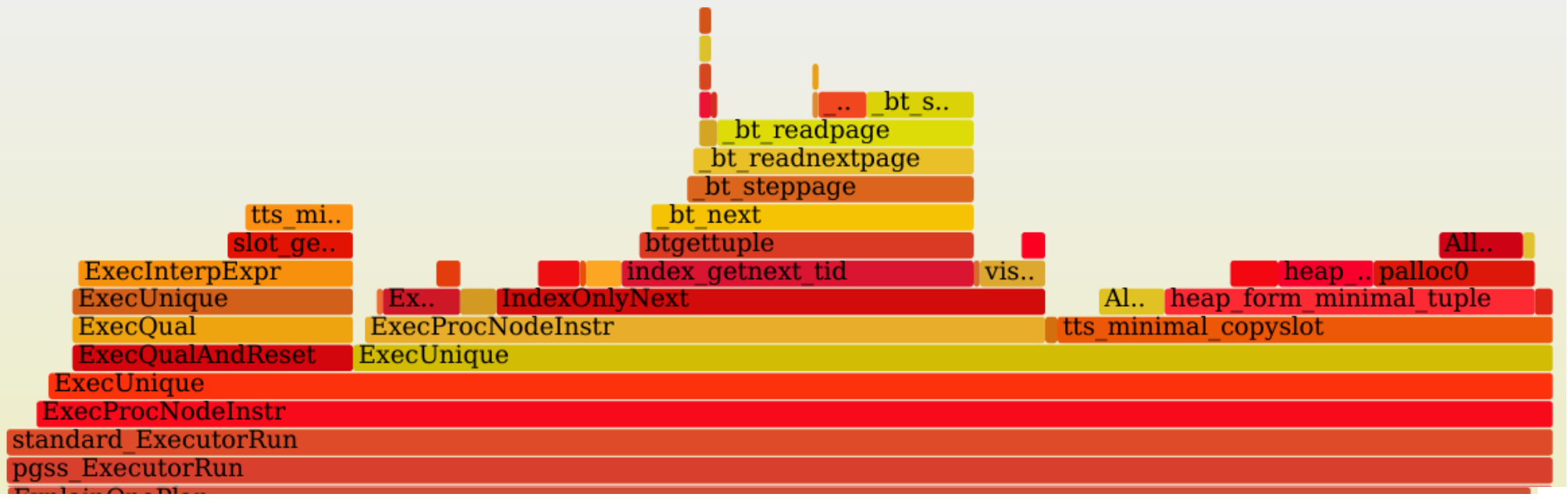
# Пример №1: Group By



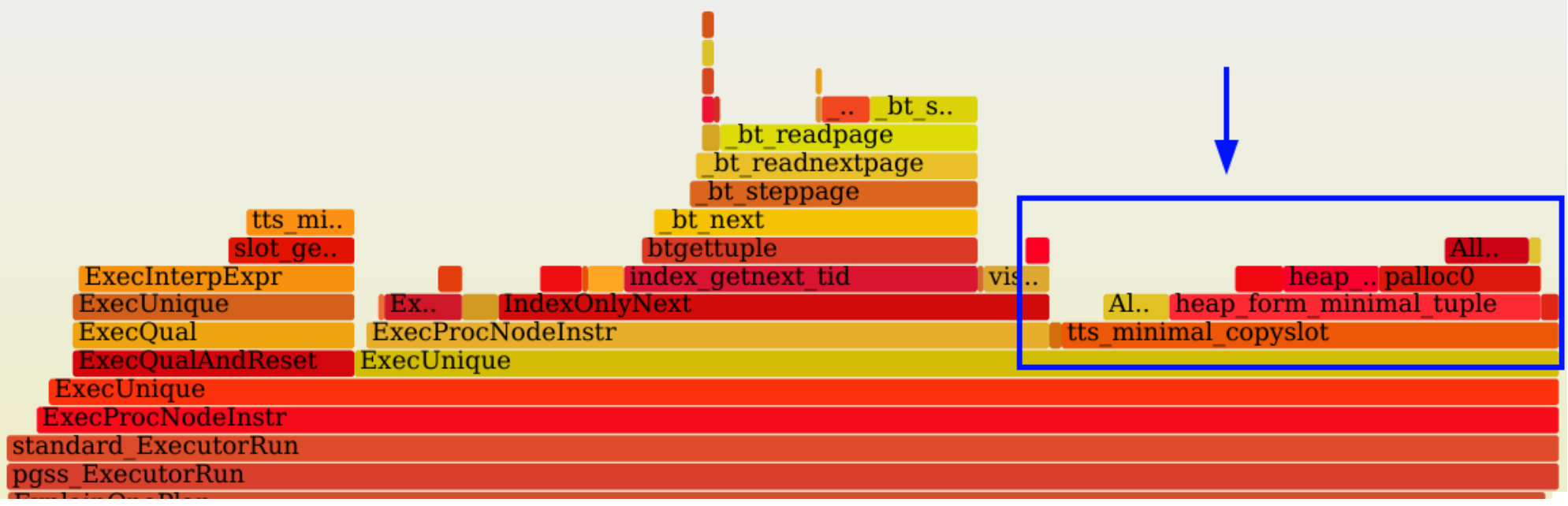
# Пример №1: Group By



# Пример №1: Unique



# Пример №1: Unique



# Пример №1: Unique 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;
```

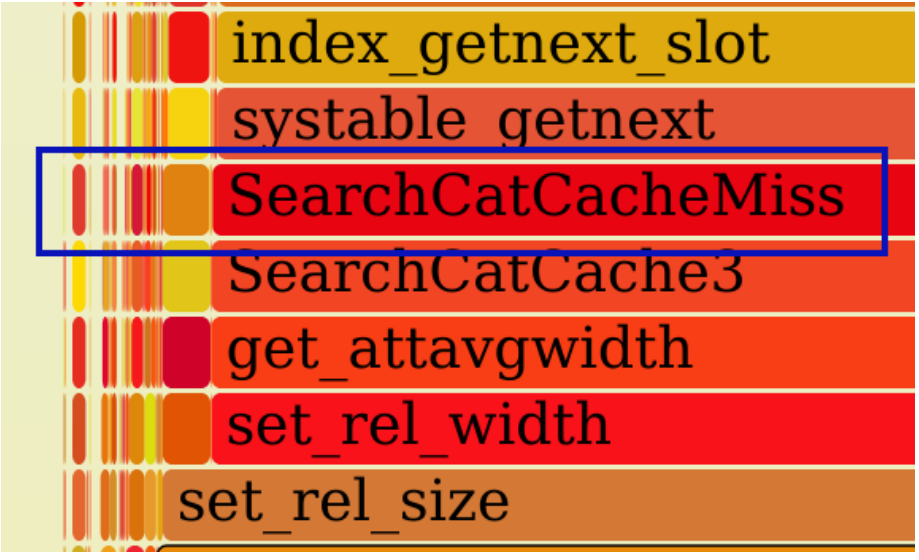
# Пример №2: SysCache

```
load average: 71.76, 72.36, 72.54
0 stopped,   0 zombie
.1 wa,  0.0 hi,  0.9 si,  0.0 st
304 used, 10989072+buff/cache
308 used. 49603376 avail Mem

 %CPU %MEM     TIME+ COMMAND
 70.0  1.1   0:00.38 postgres:(21047) BIND
 65.0  0.5   0:00.14 postgres:(21449) BIND
 60.0  0.9   0:00.19 postgres:(21391) BIND
 60.0  0.5   0:00.12 postgres:(21459) idle
 60.0  0.5   0:00.12 postgres:(21461) idle
 55.0  0.5   0:00.13 postgres:(21421) BIND
 55.0  0.5   0:00.14 postgres:(21431) BIND
```

- BIND – это обработка сетевого пакета со связанными переменными запроса, включая **иногда планирование**

# Пример №2: SysCache




- В кэше КАТАЛОГА что-то не нашлось…
- Но оно там наверняка было секунду назад…
- Куда же оно делось…

# Пример №2: SysCache

- Инвалидация кэша
  - DDL- и utility-команды
  - on-commit рассылает **всем** процессам в виде сообщений
- Отключение кэша
  - debug_discard_caches (DEVELOPER GUC)
- Мониторить инвалидации с помощью pgpro_stats:
  - https://postgrespro.ru/docs/postgrespro/13/pgpro-stats#PGPRO-STATS-INVAL-MSGS

# Пример №3: Commit

```
-- создаём 1000 временных таблиц каждая по 1000 строк
```

# Пример №3: Commit

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

# Пример №3: Commit

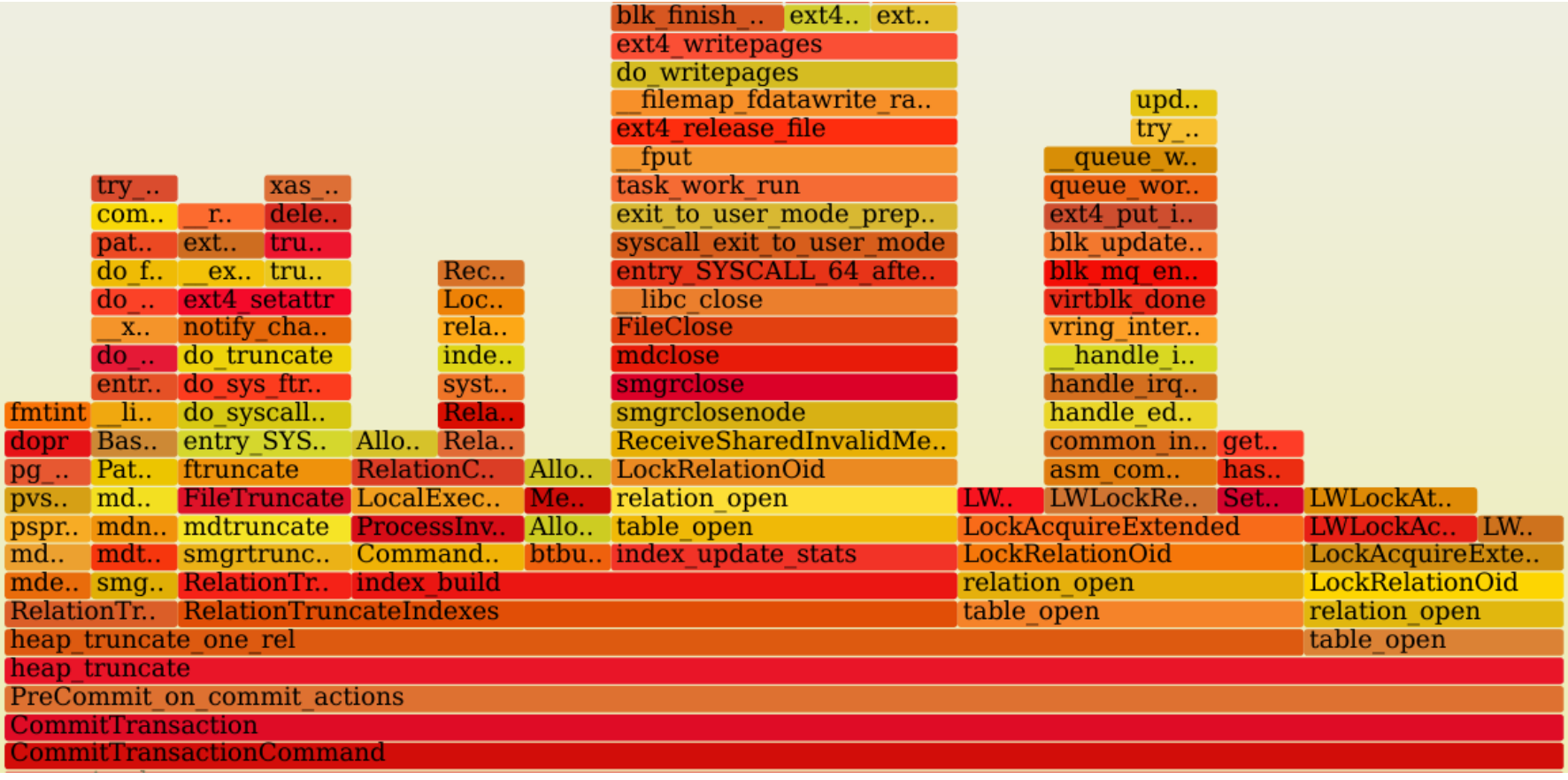
```
postgres=# begin;
Time: 0.230 ms
postgres=*# select 1 from t_tab_100;
Time: 0.317 ms
postgres=*# commit;
Time: 180.267 ms
```

# Пример №3: Commit

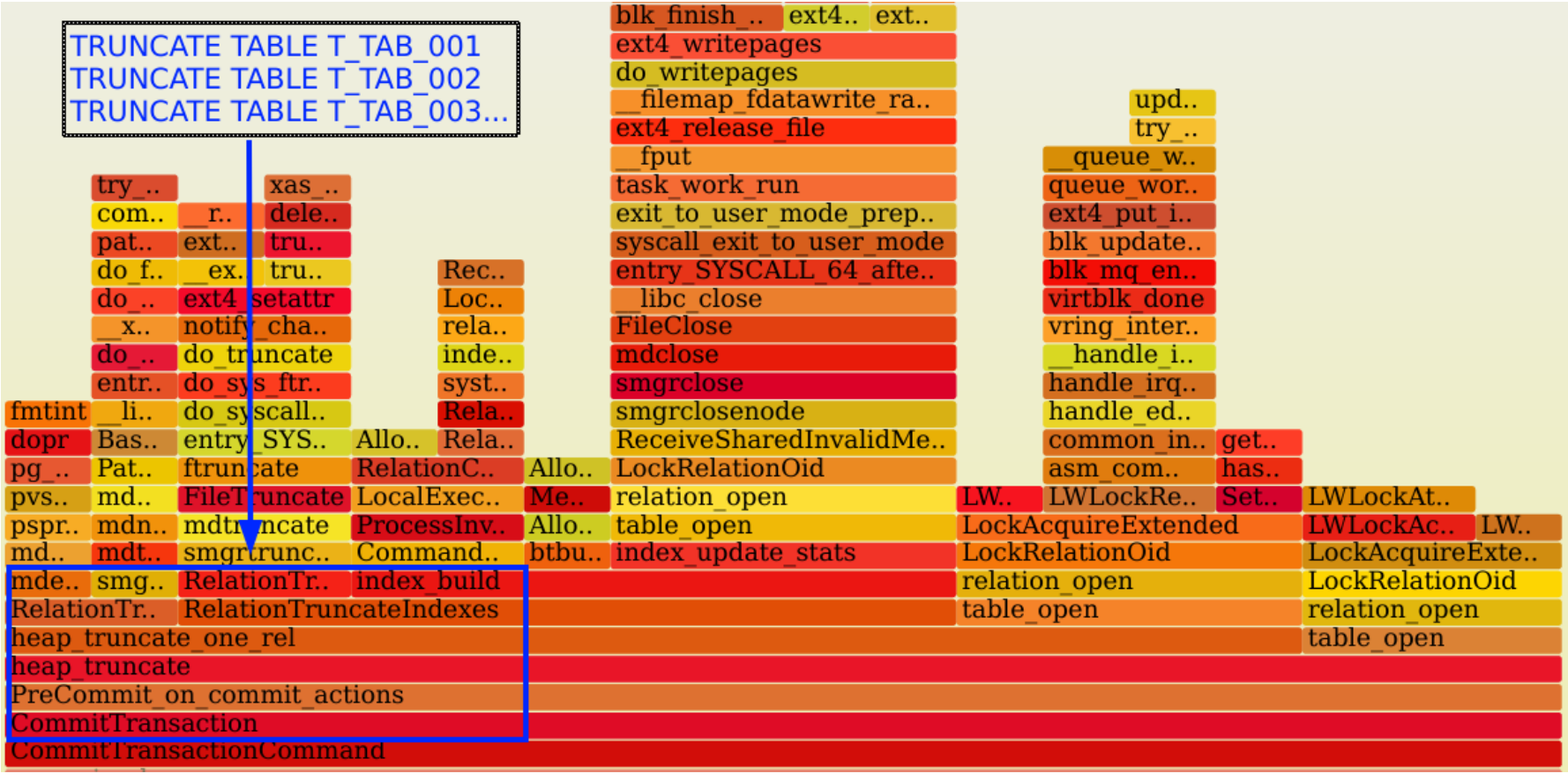
```
postgres=# begin;
Time: 0.230 ms
postgres=*# select 1 from t_tab_100;
Time: 0.317 ms
postgres=*# commit;
Time: 180.267 ms
```

**Диск не тормозит, commit потребляет CPU**

# Пример №3: Commit



# Пример №3: Commit



# Пример №3: Commit

```
on commit delete rows

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

# Пример №3: Commit

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

PostgresPro

# Выводы / советы

- Сначала посмотрите на метрики, найдите проблему в них
- Не надо профилировать CPU, если тормозит дисковая система и наоборот
- Анализируйте! Метрики, flamegraph-ы – лишь инструменты анализа
- Обогащайте свои знания чтением документации и иногда исходного кода

PostgresPro

# Спасибо!