Что нового в PostgreSQL 9.6

Александр Коротков,
Федор Сигаев
Postgres Professional

15 сентября 2016, ГАИШ МГУ

Postgres PROFESSIONAL

Что нового 9.6 ?

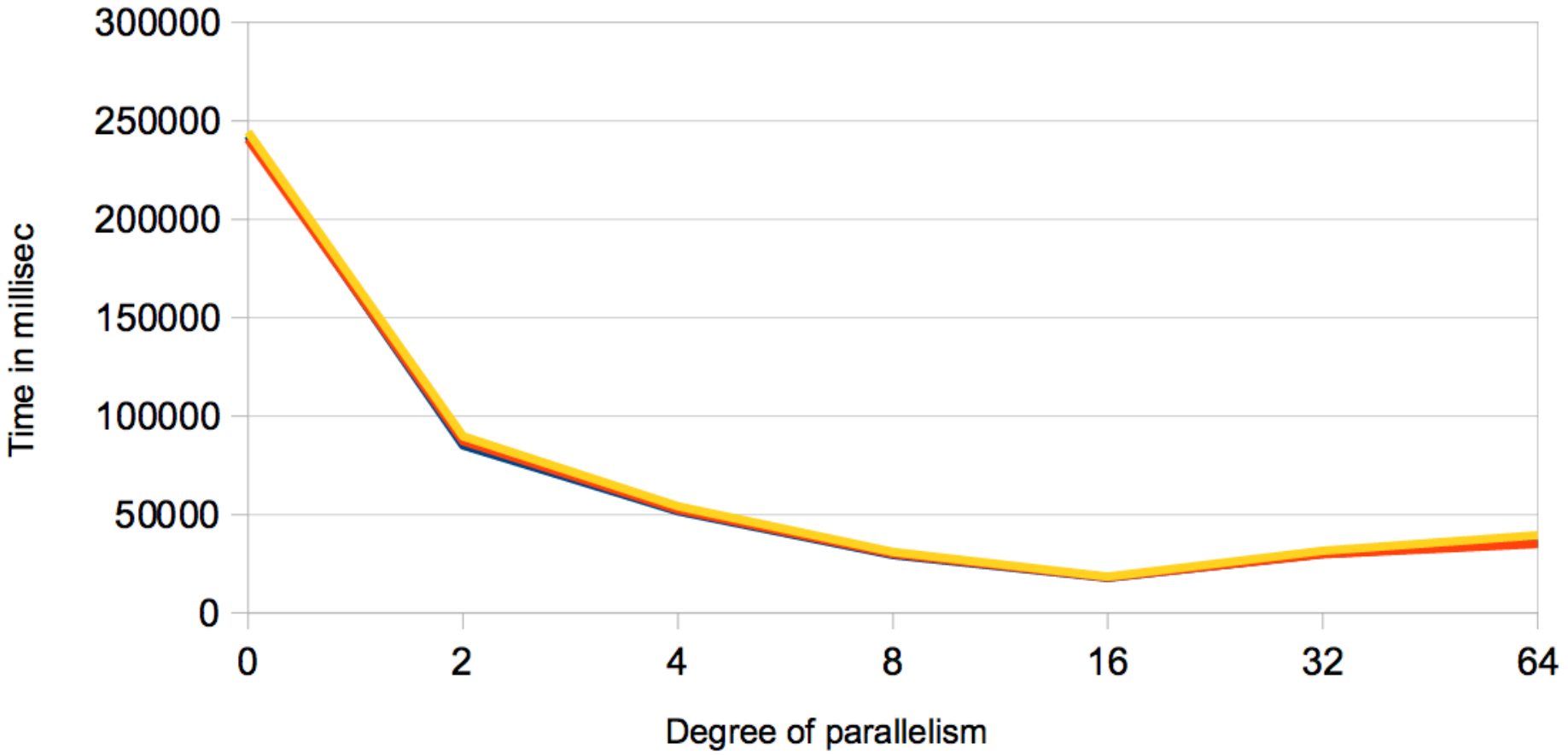
1. Параллельное исполнение запросов (sequence scan, join, aggregate)
2. Улучшение работы VACUUM с большими таблицами
3. Синхронная репликация на несколько серверов
4. Расширение таблиц и индексов сразу на несколько блоков
5. Улучшение FDW (join, order by, update, delete)
6. Индексирование box и polygon типов с помощью SP-GiST
7. CREATE ACCESS METHOD, GENERIC WAL
8. Улучшение полнотекстового поиска — поиск фраз, улучшена поддержка словарей, функции для манипулирования с tsvector
9. KNN для CUBE
10. Комбинирование агрегатов (несколько агрегатов – одно состояние)
11. IOS для частичных индексов
12. Wait monitoring
13. Масштабирование на большое число ядер
14. Ошибка snapshot too old
15. Разное

Полный список улучшений: https://www.postgresql.org/docs/devel/static/release-9-6.html

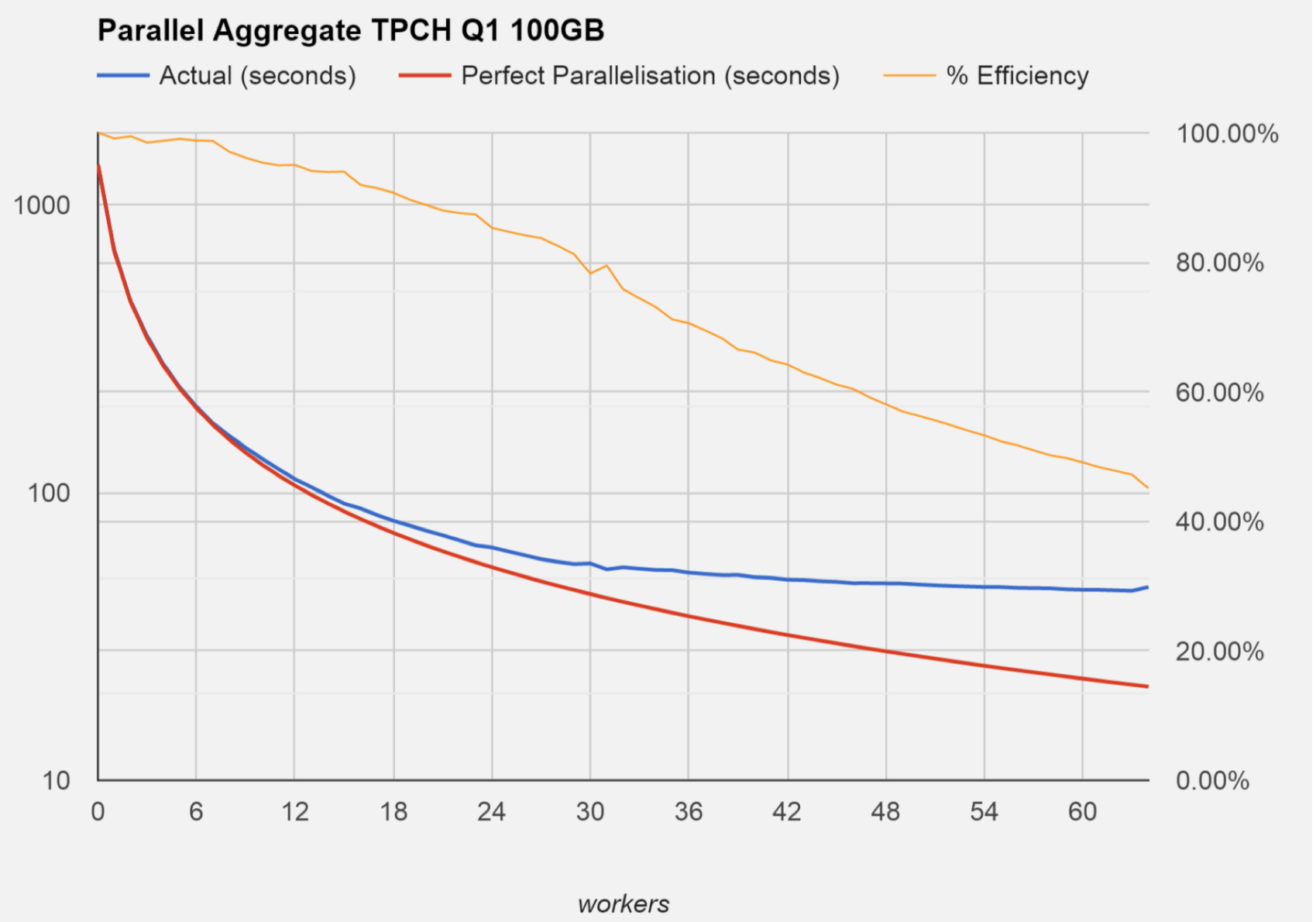
Параллельное выполнение запроса

```
# create table parallel_test as
  (select random() as val from generate_series(1,1000000));

# set max_parallel_workers_per_gather TO 0;
# explain analyze select count(*) from parallel_test where val < 0.1;
                                                          QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=17165.26..17165.27 rows=1 width=8) (actual time=86.284..86.284 rows=1
   ->  Seq Scan on parallel_test  (cost=0.00..16925.00 rows=96105 width=0) (actual time
         Filter: (val < '0.1'::double precision)
         Rows Removed by Filter: 900213
 Planning time: 0.051 ms
 Execution time: 86.305 ms

# set max_parallel_workers_per_gather TO 2;
# explain analyze select count(*) from parallel_test where val < 0.1;
                                                              QUERY PLAN
-------------------------------------------------------------------------------
 Finalize Aggregate  (cost=10733.66..10733.67 rows=1 width=8) (actual time=36.371..36.3
   ->  Gather  (cost=10733.44..10733.65 rows=2 width=8) (actual time=36.270..36.367 row
         Workers Planned: 2
         Workers Launched: 2
         ->  Partial Aggregate  (cost=9733.44..9733.45 rows=1 width=8) (actual time=33.
               ->  Parallel Seq Scan on parallel_test  (cost=0.00..9633.33 rows=40044 w
                     Filter: (val < '0.1'::double precision)
                     Rows Removed by Filter: 300071
 Planning time: 0.057 ms
 Execution time: 37.505 ms
```

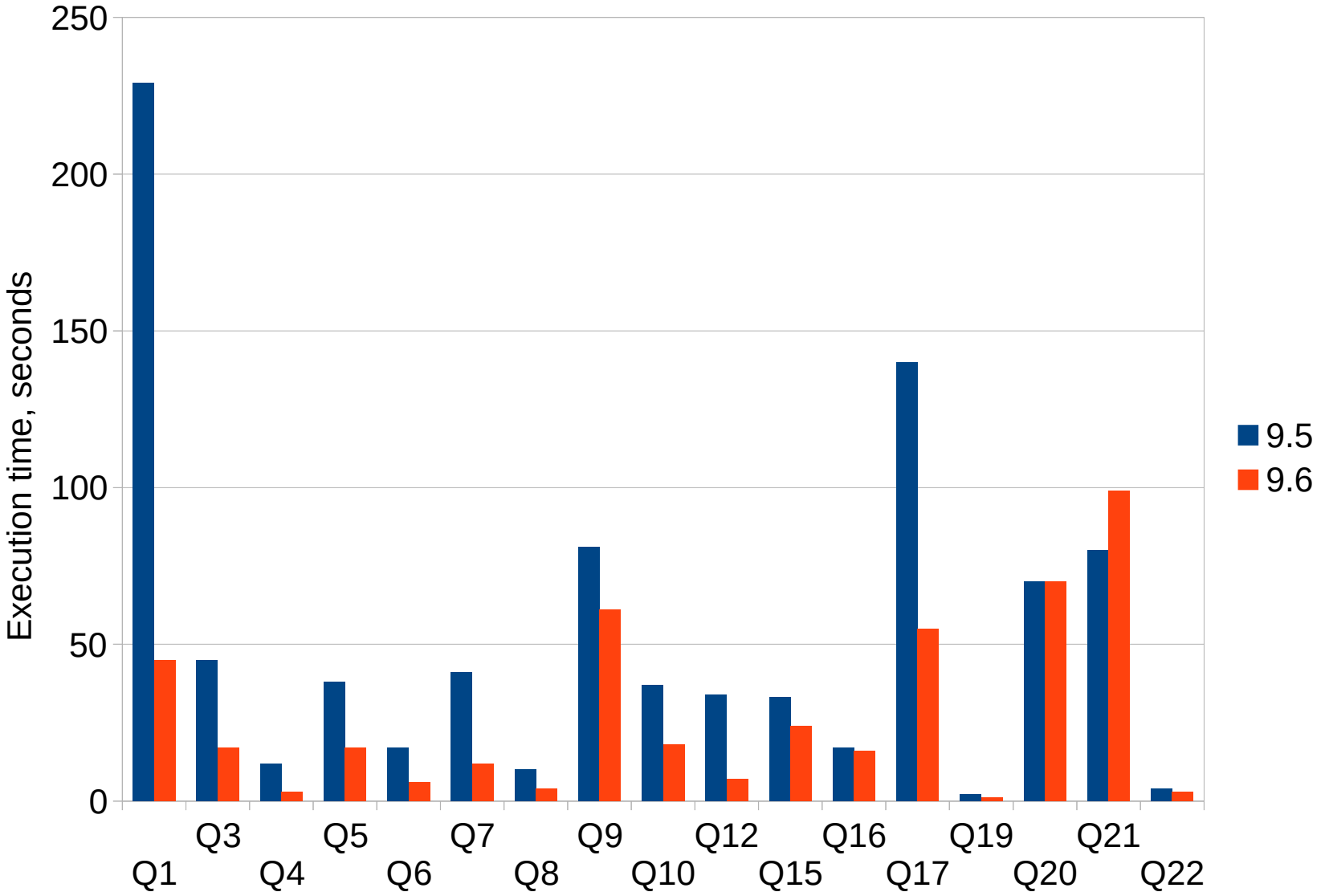
Параллельный seqscan



Параллельная агрегация



Параллельные запросы: TPC-H



VACUUM на больших таблицах

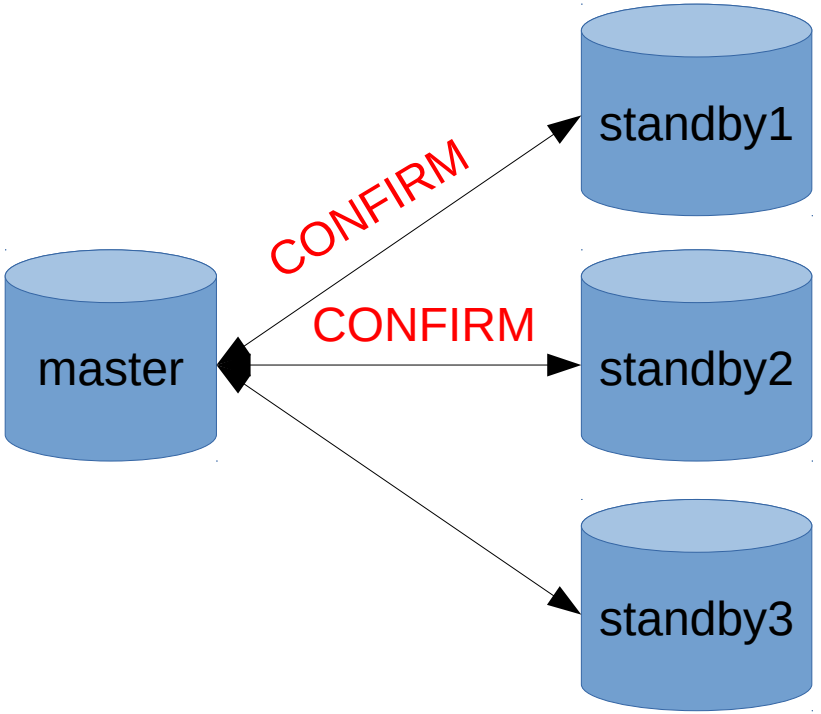
- 32Bit идентификатор транзакции, «закольцованы» 2 млрд
- Wraparound — надо обойти ВСЕ таблицы и ВСЕ записи
- Современность — каждые несколько недель
- (микро)оптимизация — помнить самый старый XID для каждой таблицы

VACUUM на больших таблицах

- Visibility map?!
- Добавим битик «зафрожено»
- Profit — читаем полностью только VM, 2 бита на страницу (предел - 1Гб)

Синхронная репликация на несколько серверов

- До 9.5 включительно synchronous_commit = on дожидается подтверждения от хотя бы одной синхронной реплики.
- Начиная с 9.6 в параметре synchronous_standby_names можно записать число реплик, от которых требуется подтверждение, например: «2 (standby1, standby2, standby3)».



Расширение таблиц

- Таблица с большим кол-вом инсертов — растет с хвоста, страница за страницей
- Фрагментация! ([posix_]fallocate)
- Борьба процессов за локи файла — для расширения

Расширение таблиц

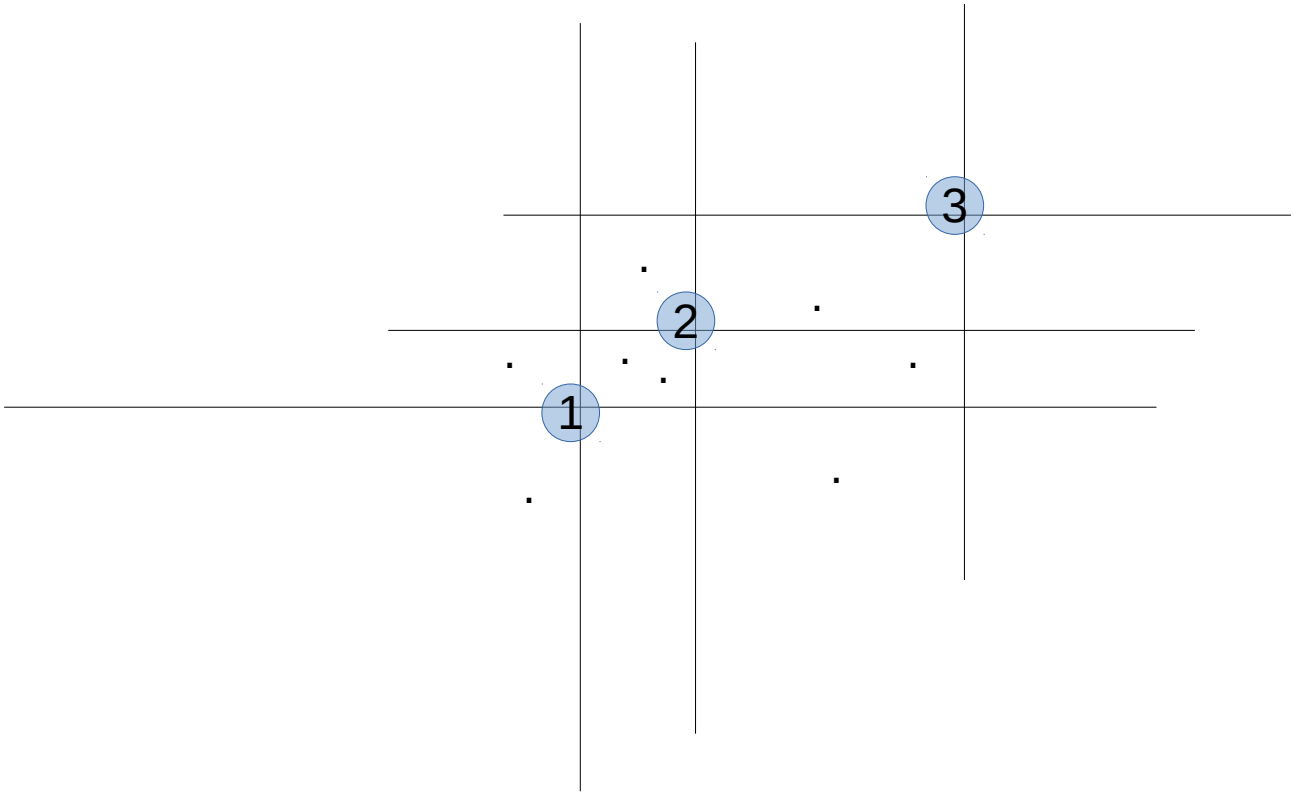
- А давайте увеличивать сразу на несколько страниц
- Количество Min(512, lockWaiters * 20)

FDW (sort, join, insert, update, delete)

```
# explain (analyze, verbose) select * from foreign_tab
                             order by id;
                                                                QUERY PLA
-----------------------------------------------------------------------------
 Foreign Scan on public.foreign_tab  (cost=100.00..155.68 rows=1365
   Output: id, val
   Remote SQL: SELECT id, val FROM public.local_tab ORDER BY id ASC
 Planning time: 0.088 ms
 Execution time: 1423.868 ms

# explain (analyze, verbose) update foreign_tab
                             set val = -1 where id = 1;
                                                               QUERY PLAN
-----------------------------------------------------------------------------
 Update on public.foreign_tab  (cost=100.00..140.35 rows=12 width=42
   ->  Foreign Update on public.foreign_tab  (cost=100.00..140.35 ro
         Remote SQL: UPDATE public.local_tab SET val = (-1)::numeric
 Planning time: 0.114 ms
 Execution time: 0.595 ms
```

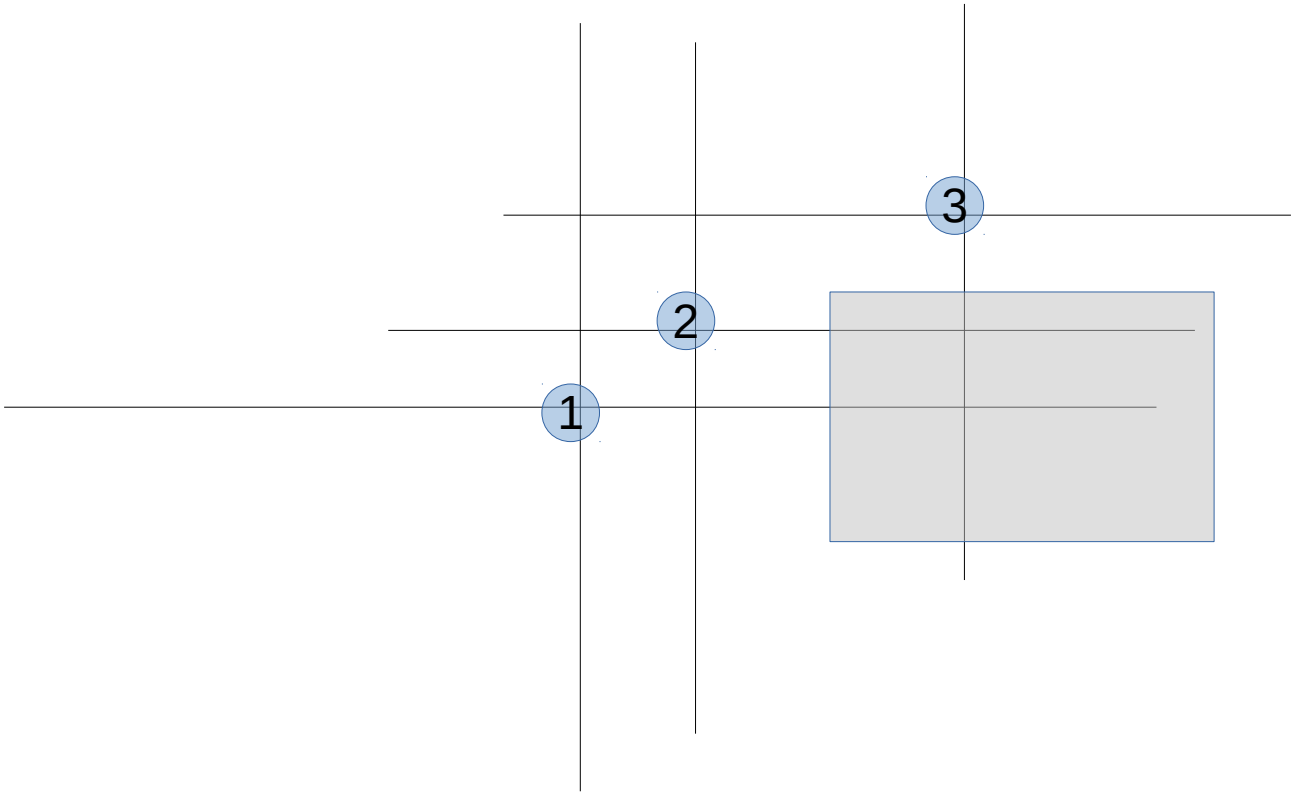
SP-GiST

- Space-Partitioned
- Для точек: quad-tree, k-d-tree



SP-GiST

- Прямоугольник — все плохо



Прямоугольник — точка?!

SP-GiST

- Да, точка. 4D
- Нарисовать даже не просите

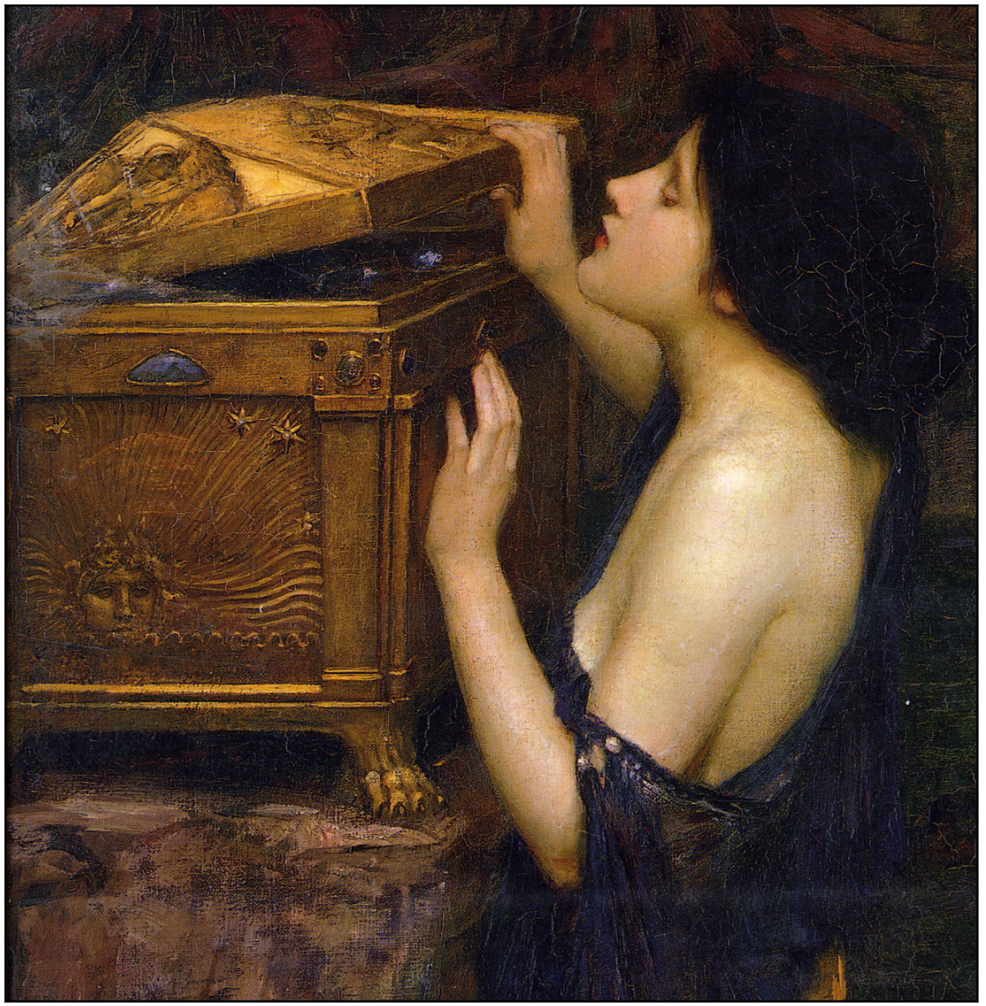
  by Alexander Lebedev, Postgres Professional

Расширяемость PostgreSQL

- Функции
- Типы данных
- Операторы
- Агрегаты
- Языки хранимых процедур (sql, pl/pgsql, pl/perl, pl/python, pl/tcl, pl/R, pl/java, …,pl/v8)
- Классы операторов (индексный доступ: btree, Hash, GiST, GIN, SP-GiST, BRIN)
- Foreign Data Wrappers – (практически ко всем СУБД)
- Custom Nodes (другие способы выполнения запроса, например на GPU)

9.6 opens «Pandora box»

Индексный метод доступа как расширение !



- Заложено в Postgres изначально, утрачено по пути развития.
- В 9.6 сделано:
  - Рефакторинг интерфейса;
  - «Легальная» команда CREATE ACCESS METHOD;
  - Generic WAL, позволяющий описывать разницу между страницами в общем виде.
- contrib/bloom в качестве примера (и не только примера).

by Alexander Korotkov, Teodor Sigaev
Postgres Professional

PROFESSIONAL Postgres

No positions in index !

Inverted Index in PostgreSQL

ENTRY TREE

Report Index

A

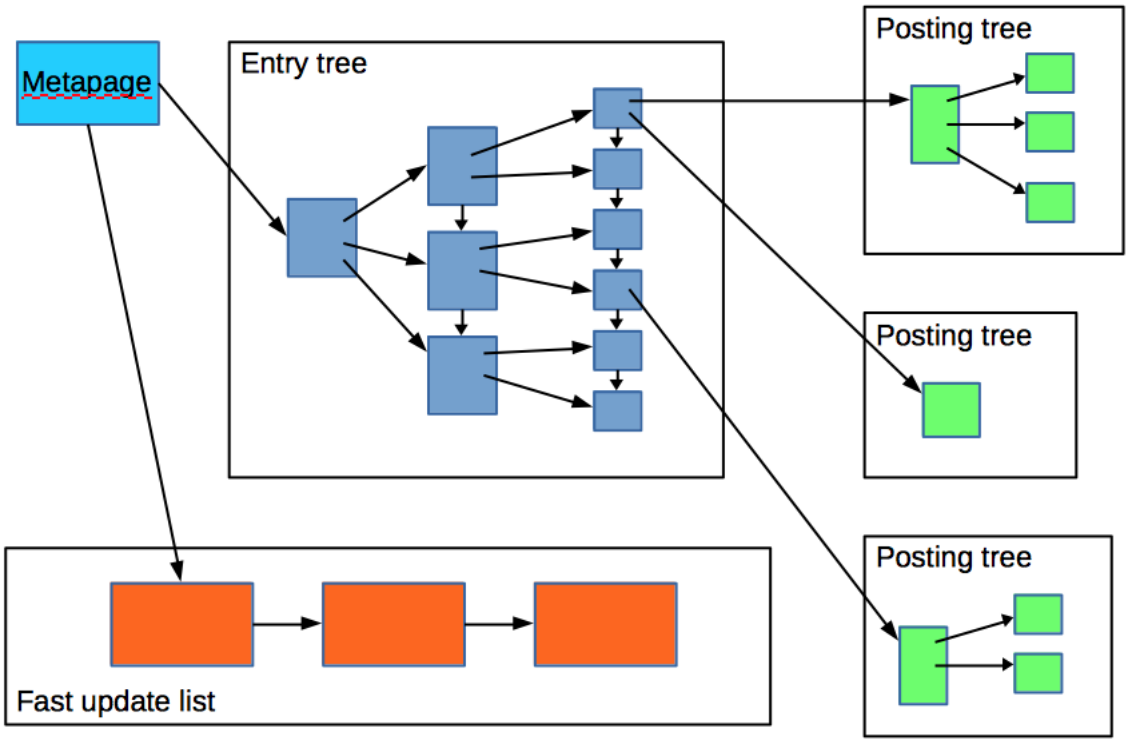
abrasives, 27
acceleration measurement, 58
accelerometers, 5, 10, 25, 28, 30, 36, 58, 59, 61, 73, 74
actuators, 4, 37, 46, 49
adaptive Kalman filters, 60, 61
adhesion, 63, 64
adhesive bonding, 15
adsorption, 44
aerodynamics, 29
aerospace instrumentation, 61
aerospace propulsion, 52
aerospace robotics, 68
aluminium, 17
amorphous state, 67
angular velocity measurement, 58
antenna phased arrays, 41, 46, 66
argon, 21
assembling, 22
atomic force microscopy, 13, 27, 35
atomic layer deposition, 15
attitude control, 60, 61
attitude measurement, 59, 61
automatic test equipment, 71
automatic testing, 24

Posting list
Posting tree

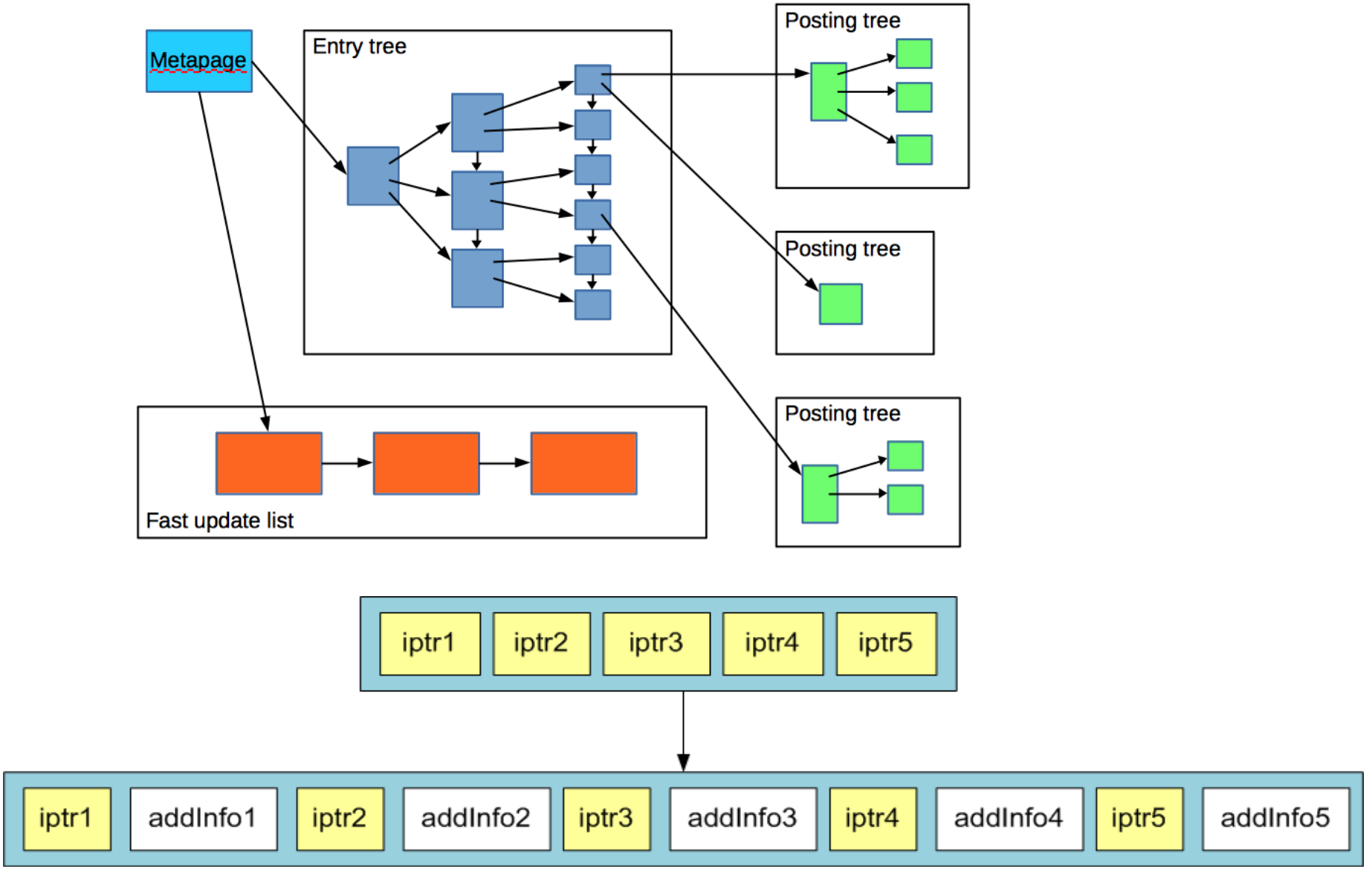
compensation, 30, 68
compressive strength, 54
compressors, 29
computational fluid dynamics, 23, 29
computer games, 56
concurrent engineering, 14
contact resistance, 47, 66
convertors, 22
coplanar waveguide components, 40
Couette flow, 21
creep, 17
crystallisation, 64

B

backward wave oscillators, 45



Improving GIN: RUM



https://github.com/postgrespro/rum

Поиск фраз

- Запросы 'A & B'::tsquery и 'B & A'::tsquery эквивалентны
- Поиск фраз – учитывает порядок слов в запросе
  Результаты 'A B' и 'B A' должны быть разные!
- Новый оператор: PHRASE (<->):
  - Порядок
  - Расстояние

a <n> b == a & b & (∃ i,j : pos(b)i – pos(a)j = n)

A <n> B - A «phrase n» B

Поиск фраз

- `TSQUERY phraseto_tsquery([CFG,] TEXT)`

```
select phraseto_tsquery('english','PostgreSQL can be extended by the user in many ways');
                    phraseto_tsquery
-----------------------------------------------------------------
 ( ( ( 'postgresql' <3> 'extend' ) <3> 'user' ) <2> 'mani' ) <-> 'way'
(1 row)
```

Стоп-слова опускаются!

Поиск фраз

- Сложнее чем просто И (~5%)
- GIN/GiST требую recheck (~2p)
- RUM не требует, все есть в индексе (~3%)

by Teodor Sigaev, Dmitry Ivanov, Oleg Bartunov

Postgres Professional

Редактирование tsvector

- Stas Kelvich (PostgresPro)
- setweight(tsvector, «char», text[]) - простановка меток лексемам

```
select setweight( to_tsvector('english', '20-th anniversary of PostgreSQL'),
'A',  '{postgresql,20}');
                   setweight
-----------------------------------------------
 '20':1A 'anniversari':3 'postgresql':5A 'th':2
(1 row)
```

- ts_delete(tsvector, text[]) - удаление лексем

```
select ts_delete( to_tsvector('english', '20-th anniversary of PostgreSQL'),
'{20,postgresql}'::text[]);
       ts_delete
------------------------
 'anniversari':3 'th':2
(1 row)
```

Tsvector editing functions

- unnest(tsvector)

```
select * from unnest( setweight( to_tsvector('english',
'20-th anniversary of PostgreSQL'),'A',  '{postgresql,20}'));
   lexeme    | positions | weights
-------------+-----------+---------
 20          | {1}       | {A}
 anniversari | {3}       | {D}
 postgresql  | {5}       | {A}
 th          | {2}       | {D}
(4 rows)
```

- tsvector_to_array(tsvector) — tsvector to text[]
  array_to_tsvector(text[])

```
select tsvector_to_array( to_tsvector('english',
'20-th anniversary of PostgreSQL'));
       tsvector_to_array
-------------------------------
 {20,anniversari,postgresql,th}
(1 row)
```

Tsvector editing functions

- ts_filter(tsvector,text[]) - фильтрация лексем по метке

```
select ts_filter($$'20':2A 'anniversari':4C 'postgresql':1A,6A 'th':3$$::tsvector,
'{C}');
    ts_filter
------------------
 'anniversari':4C
(1 row)

select ts_filter($$'20':2A 'anniversari':4C 'postgresql':1A,6A 'th':3$$::tsvector,
'{C,A}');
                   ts_filter
-----------------------------------------------
 '20':2A 'anniversari':4C 'postgresql':1A,6A
(1 row)
```

Non-9.6

- RUM (Generic WAL, CREATE ACCESS METHOD)
  - In index ranking
  - Filtering
  - Inverse FTS
- Словарь как расширение
- Предзагруженные словари

  Alexander Korotkov, Arthur Zakirov, Teodor Sigaev (PostgresPro)

KNN для CUBE

- N-мерные кубы....
- Поиск ближайшего!

Stas Kelvich (PostgresPro)

Комбинирование агрегатов

```
# create table agg_test as (select i, random() val
                            from generate_series(1,1000000) i);
# vacuum analyze agg_test;

# explain analyze select avg(val) from agg_test;
                                                          QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=17908.00..17908.01 rows=1 width=32) (actual time=2
   ->  Seq Scan on agg_test  (cost=0.00..15408.00 rows=1000000 width
 Planning time: 0.041 ms
 Execution time: 267.739 ms

# explain analyze select avg(val), sum(val) from agg_test;
                                                          QUERY PLAN
-------------------------------------------------------------------------------
 Aggregate  (cost=20408.01..20408.01 rows=1 width=64) (actual time=2
   ->  Seq Scan on agg_test  (cost=0.00..15408.00 rows=1000000 width
 Planning time: 0.044 ms
 Execution time: 259.963 ms
```

Раньше было (258.638 ms => 333.232 ms)

Два агрегата по цене одного!

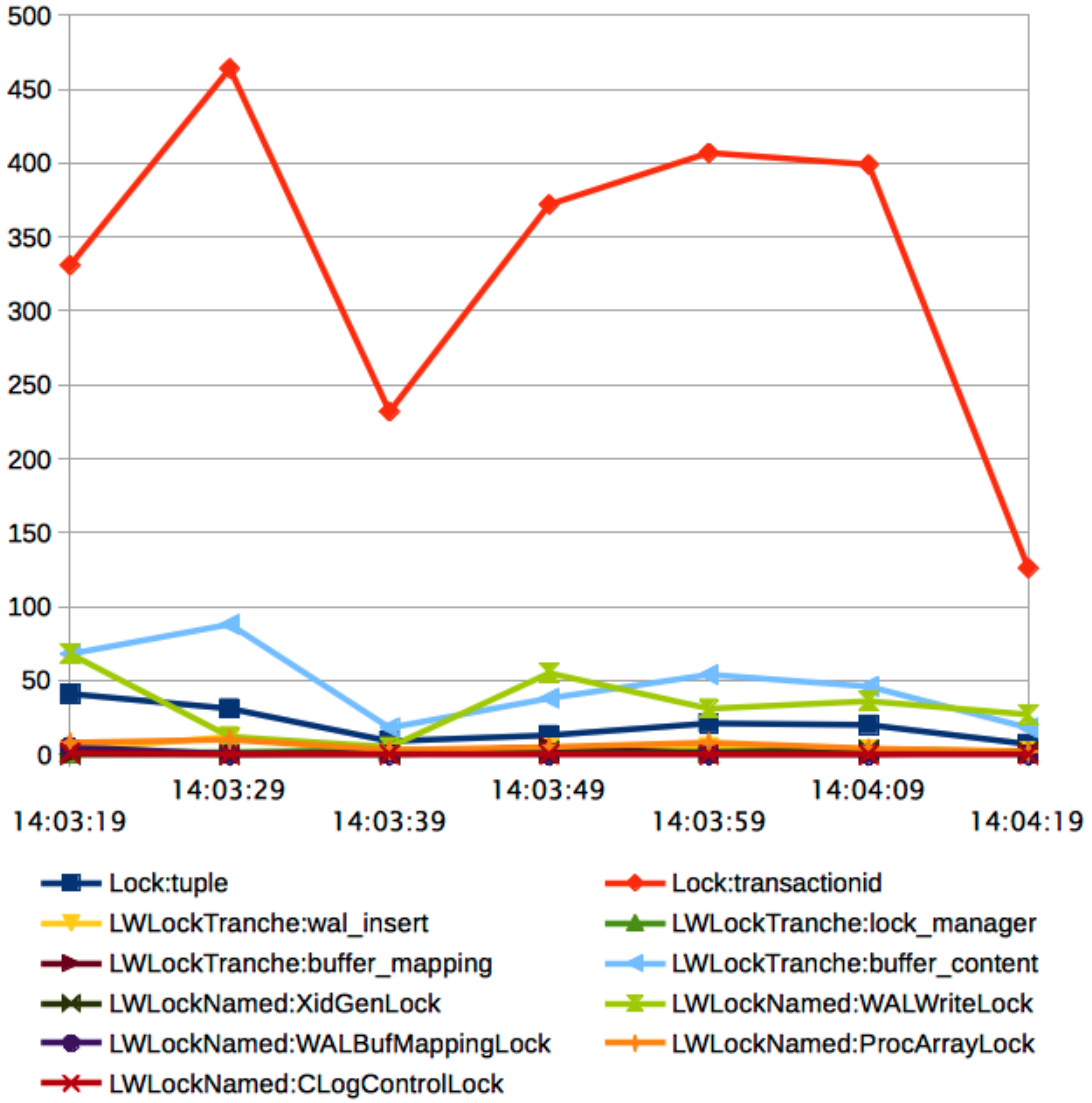
IOS по выражениям

```
# create table ios_test as (select i,
                                    random() val,
                                   (random()*2)::int::bool flag
                             from   generate_series(1,1000000) i);

# create index ios_test_i_val_true_idx on ios_test(i, val) where flag;

# vacuum analyze ios_test;

# explain analyze select i, val from ios_test
  where flag order by i limit 10;
                                                                                QUER
-------------------------------------------------------------------------------------
 Limit  (cost=0.42..0.73 rows=10 width=12) (actual time=0.020..0.024 rows=1
   ->  Index Only Scan using ios_test_i_val_true_idx on ios_test  (cost=0.4
         Heap Fetches: 0
 Planning time: 0.105 ms
 Execution time: 0.049 ms
```

Wait monitoring

- pg_wait_sampling

```
[local]:5432 postgres@postgres=# \d pg_stat_activit
          View "pg_catalog.pg_stat_activity"
      Column      |           Type           | Modi
------------------+--------------------------+-----
 datid            | oid                      |
 datname          | name                     |
 pid              | integer                  |
 usesysid         | oid                      |
 usename          | name                     |
 application_name | text                     |
 client_addr      | inet                     |
 client_hostname  | text                     |
 client_port      | integer                  |
 backend_start    | timestamp with time zone |
 xact_start       | timestamp with time zone |
 query_start      | timestamp with time zone |
 state_change     | timestamp with time zone |
 wait_event_type  | text                     |
 wait_event       | text                     |
 state            | text                     |
 backend_xid      | xid                      |
 backend_xmin     | xid                      |
 query            | text                     |
```



Масштабирование на большое число ядер

Перед тем, как «потрогать» любой блок данных нужно его «запинить». Pin/UnpinBuffer – очень частая операция.

Было

PinBuffer:

```
S_LOCK(bufHdr);
bufHdr->pinCount++;
S_UNLOCK(bufHdr);
```

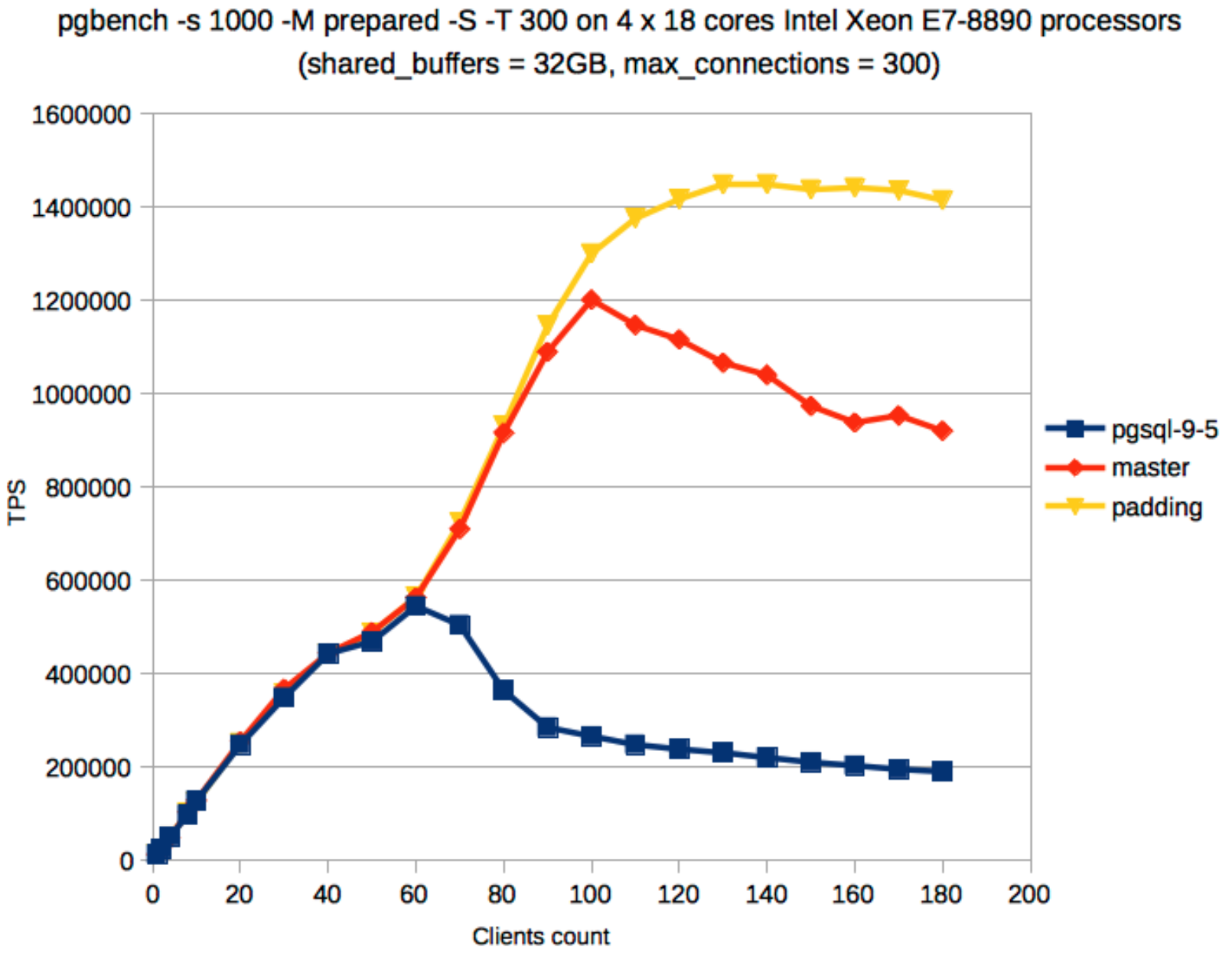
Стало

PinBuffer:

```
atomic_increment(buf_hdr->pinCount);
```

by Alexander Korotkov (PostgresPro), Andres Freund

Масштабирование на большое число ядер



Snapshot too old

- Oracle
- Bug to bug implementation of bug
- Задача — прибить больно старые транзакции
- old_snapshot_threshold = -1
  # 1min-60d; -1 disables; 0 is immediate

Рассыпуха

- Колонки Foreign Key для multicolumn joins (кросс-корреляция)
- Вычисление вычисление колонок ПОСЛЕ ORDER BY (Константин Книжник, PostgresPro)
- \ev, \sv — редактирование и показ представлений
- \crosstabview
- idle_in_transaction_session_timeout - решает проблему «idle in transaction»
- simple VACUUM progress reporting — можно следить за выполнением вакуума pg_stat_progress_vacuum
- per-tablespace effective_io_concurrency
- Новый режим синхронной репликации - 'remote_apply'
- Системная view pg_config

Postgres PROFESSIONAL

Что будет в 10.0 ?

- BDR — двунаправленная репликация
  http://2ndquadrant.com/en/resources/bdr/
- Pglogical (5x быстрее slony, londiste3)
  http://2ndquadrant.com/en/resources/pglogical/
- Declarative partitioning (+pg_pathman)
- Highly Available multimaster «из коробки»
- Инкрементальный бэкап на уровне блоков
- FDW pushdown aggregates
- In-memory хранилище
- Хранение временных объектов в памяти
  - временные таблицы на слейве

Postgres PROFESSIONAL

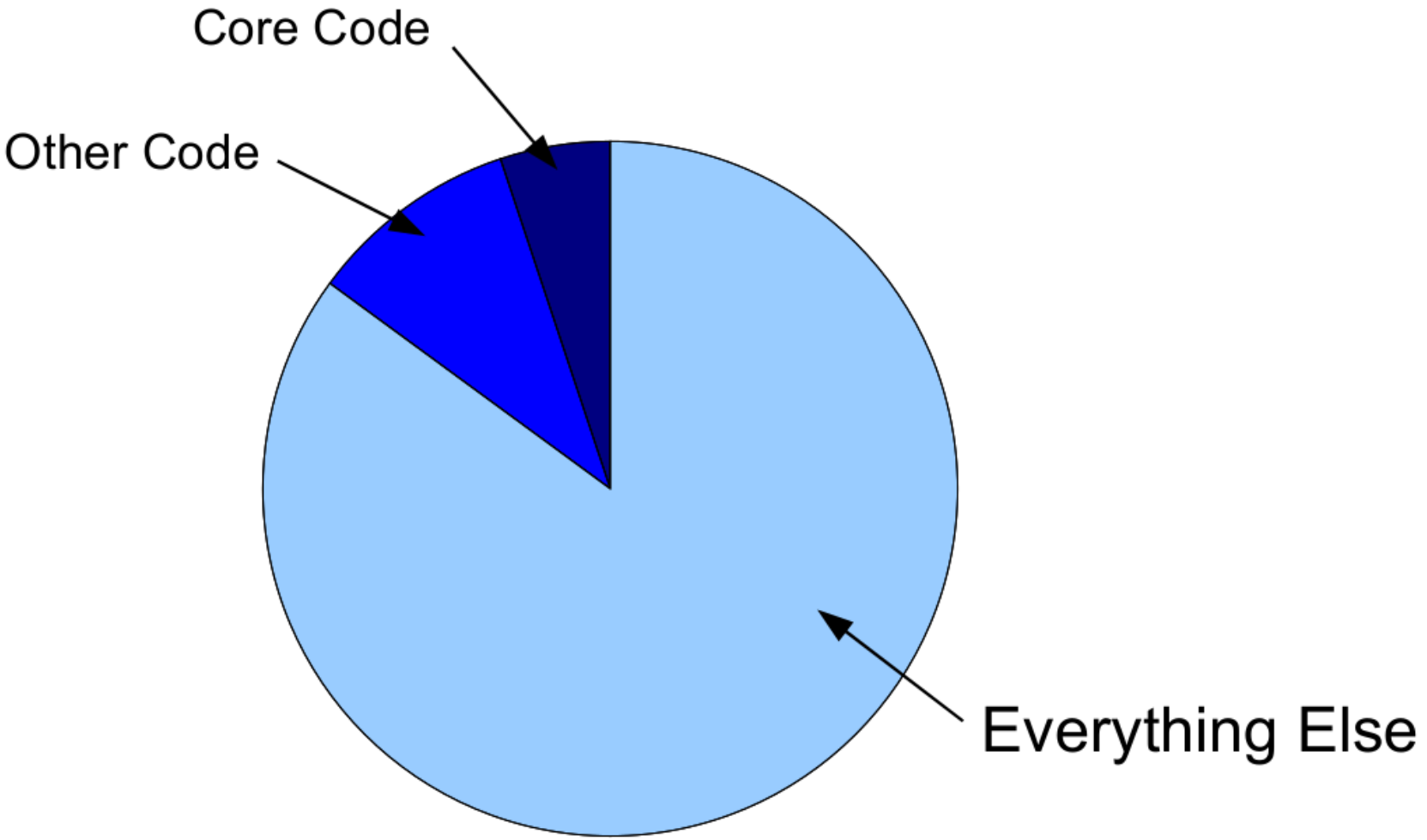
Мы ищем таланты

- Разработчики, инженеры, QA, PM, технические писатели, стажеры (студенты), люди науки
  - Работать в команде, «жить» в сообществе
  - Уметь и любить учиться
  - Любить вызовы
  - Держать цель
- Возможна удаленная работа

Российское сообщество

- Самое организованное — несколько тысяч человек
- Митапы при поддержки крупных компаний
- Крупнейшие в мире конференции по постгресу:
  - летом PGDay.ru в Санкт-Петербурге (2014, 2015, 2016)
  - Зимой PGConf.ru в Москве (2015, 2016)
- Секции и квартирники на крупнейших конференциях
  - Highload++, RIT, Codefest, Stachka
- Участвуем в международных конференциях
  - PGConf.EU, PGCon.org
- Свободные курсы DBA1, DBA2, «Hacking Postgres» от Postgres Professional

50 способов помочь сообществу



50 способов помочь сообществу

Ядро
Разработка, review, тестирование, reporting bugs

Экосистема
Расширения, драйверы, ORM, средства мониторинга… поддержка Pg в прикладном ПО
Создание дистрибутивов, пакетирование

Документация
Улучшение, перевод, публикация статей, книг, учебных, маркетинговых материалов…блоггинг!

Расскажите о своей истории с PostgreSQL!

Общение, образование
Создание локальных сообществ
Проведение конференций, митапов, семинаров, учебных курсов.

Внедрите PostgreSQL!
В Вашей компании. Запустите учебный курс в Вашем ВУЗе

Спонсорство
Спонсируйте разработку нужной Вам функциональности.
Спонсируйте мероприятие.

Если вы с Open Source

Свобода — необходимое условие для творчества.
Идеи рождаются в свободном творчестве.

Главные свойства

- Возможность бесплатного тиражирования
- Доступ к исходным кодам

Основные свободы по FSF/GNU:

- Выполнять программу
- Изучать и модифицировать программу
- Передавать копии программы
- Передавать копии модифицированной программы

«Безрассудная» свобода MIT/BSD:

- Создавать закрытую программу на основе открытой

Postgres PROFESSIONAL

Если вы захотели стать разработчиком OPEN SOURCE проекта

Что дает участие в Open Source

- Причастность к большому проекту, большому сообществу
- Реализация как разработчика
- Влияние на развитие проекта
- Независимость от компании, репутация в сообществе
- Карьера в сообществе коррелирует с карьерой в компании
- Возможность жить и работать в удобном месте — дОма (no Piter, no Moscow) !
- Удовлетворение — help the World !

Требования к разработчику

- Знание и владение основными инструментариями
  - Язык[и] программирования
  - Git, треккеры, вики, средства документирования
- Совместимость с сообществом
  - Знание английского языка (разные)
  - Умение вести переписку
  - Не пропадать надолго
  - Следовать стилю кодирования
  - Синхронизоваться с циклом разработки
  - Следовать принятым сценариям разработки
  - Принимать участие в жизни сообщества

Postgres PROFESSIONAL