Что нам ждать от PostgreSQL 13?

Александр Коротков

Postgres Professional

2020

О докладе

- Только фичи, не баги.
- Commit messages на слайдах вы не разглядите, они для справки :)
- То, что закоммичено, войдёт в релиз с 99% вероятностью
- То, что не закоммичено, войдёт с 50% вероятностью :)

Performance (1/2)

- 3x ускорение read-only PL/PGSQL
- Частичный decompress TOAST
- Предвычисление immutable function использованных как таблица:

```
select rank(txtsample, q), *
from test_tsquery,
     to_tsquery('english', 'new') q
where txtsample @@ q
order by 1 desc;
```

Ускорение read-only PL/PGSQL

```
commit 73b06cf893c9d3bb38c11878a12cc29407e78b6c
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Fri Nov 22 15:02:18 2019 -0500

    Avoid taking a new snapshot for an immutable simple expression in plpgsql.

    We already used this optimization if the plpgsql function is read-only.
    But it seems okay to do it even in a read-write function, if the
    expression contains only immutable functions/operators.  There would
    only be a change of behavior if an "immutable" called function depends
    on seeing database updates made during the current plpgsql function.
    That's enough of a violation of the promise of immutability that anyone
    who complains won't have much of a case.

    The benefits are significant --- for simple cases like

      while i < 10000000
      loop
        i := i + 1;
      end loop;

    I see net performance improvements around 45%.  Of course, real-world
    cases won't get that much faster, but it ought to be noticeable.
    At the very least, this removes much of the performance penalty that
    used to exist for forgetting to mark a plpgsql function non-volatile.
```

Частичный TOAST decompress

```
commit 4d0e994eed83c845a05da6e9a417b4efec67efaf
Author: Stephen Frost <sfrost@snowman.net>
Date:   Tue Apr 2 12:35:32 2019 -0400

    Add support for partial TOAST decompression

    When asked for a slice of a TOAST entry, decompress enough to return the
    slice instead of decompressing the entire object.

    For use cases where the slice is at, or near, the beginning of the entry,
    this avoids a lot of unnecessary decompression work.

    This changes the signature of pglz_decompress() by adding a boolean to
    indicate if it's ok for the call to finish before consuming all of the
    source or destination buffers.

    Author: Paul Ramsey
    Reviewed-By: Rafia Sabih, Darafei Praliaskouski, Regina Obe
    Discussion: https://postgr.es/m/CACowWR07EDm7Y4m2kbhN_jnys\%3DBBf9A6768RyQdKm_\%3DNp
```

Предвычисление immutable function, использованных как таблица

```
commit 7266d0997dd2a0632da38a594c78e25ff21df67e
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Thu Aug 1 18:50:22 2019 -0400

    Allow functions-in-FROM to be pulled up if they reduce to constants.

    This allows simplification of the plan tree in some common usage
    patterns: we can get rid of a join to the function RTE.

    In principle we could pull up any immutable expression, but restricting
    it to Consts avoids the risk that multiple evaluations of the expression
    might cost more than we can save.  (Possibly this could be improved in
    future --- but we've more or less promised people that putting a function
    in FROM guarantees single evaluation, so we'd have to tread carefully.)

    To do this, we need to rearrange when eval_const_expressions()
    happens for expressions in function RTEs.  I moved it to
    inline_set_returning_functions(), which already has to iterate over
    every function RTE, and in consequence renamed that function to
    preprocess_function_rtes().  A useful consequence is that
    inline_set_returning_function() no longer has to do this for itself,
    simplifying that code.

    In passing, break out pull_up_simple_subquery's code that knows where
    everything that needs pullup replace_vars() processing is, so that
```

Performance (2/2)

- Ускорение jsonb функций
- Скоростной truncate() - удаление буферов за один просмотр
- old_snapshot_threshold vs индексы
- LISTEN менее дорогой (~300 слушателей)
- Avoid full index for GIN

Ускорение jsonb функций (1/3)

```
commit 1a2983231d9080bfa06cfbf38d5415b5d71eea91
Author: Alvaro Herrera <alvherre@alvh.no-ip.org>
Date:   Fri Sep 20 20:18:11 2019 -0300

    Split out code into new getKeyJsonValueFromContainer()

    The new function stashes its output value in a JsonbValue that can be
    passed in by the caller, which enables some of them to pass
    stack-allocated structs -- saving palloc cycles.  It also allows some
    callers that know they are handling a jsonb object to use this new jsonb
    object-specific API, instead of going through generic container
    findJsonbValueFromContainer.

    Author: Nikita Glukhov
    Discussion: https://postgr.es/m/7c417f90-f95f-247e-ba63-d95e39c0ad14@postgrespro.ru
```

Ускорение jsonb функций (2/3)

```
commit dbb9aeda9959d8a8f463e841b69dfa04afc67a3a
Author: Alvaro Herrera <alvherre@alvh.no-ip.org>
Date:   Fri Sep 20 19:18:24 2019 -0300

    Optimize get_jsonb_path_all avoiding an iterator

    Instead of creating an iterator object at each step down the JSONB
    object/array, we can just just examine its object/array flags, which is
    faster.  Also, use the recently introduced JsonbValueAsText instead of
    open-coding the same thing, for code simplicity.

    Author: Nikita Glukhov
    Discussion: https://postgr.es/m/7c417f90-f95f-247e-ba63-d95e39c0ad14@postgrespro.ru
```

Ускорение jsonb функций (3/3)

```
commit abb014a63106653f2872a3b1662a79ab80d4dcbb
Author: Alvaro Herrera <alvherre@alvh.no-ip.org>
Date:   Fri Sep 20 18:34:31 2019 -0300

    Refactor code into new JsonbValueAsText, and use it more

    jsonb_object_field_text and jsonb_array_element_text both contained
    identical copies of this code, so extract that into new routine
    JsonbValueAsText.  This can also be used in other places, to measurable
    performance benefit: the jsonb_each() and jsonb_array_elements()
    functions can use it for outputting text forms instead of their less
    efficient current implementation (because we no longer need to build
    intermediate a jsonb representation of each value).

    Author: Nikita Glukhov
    Discussion: https://postgr.es/m/7c417f90-f95f-247e-ba63-d95e39c0ad14@postgrespro.ru
```

Скоростной truncate()

```
commit 6d05086c0a79e50d8e91ed953626ec7280cd2481
Author: Fujii Masao <fujii@postgresql.org>
Date:   Tue Sep 24 17:31:26 2019 +0900

    Speedup truncations of relation forks.

    When a relation is truncated, shared_buffers needs to be scanned
    so that any buffers for the relation forks are invalidated in it.
    Previously, shared_buffers was scanned for each relation forks, i.e.,
    MAIN, FSM and VM, when VACUUM truncated off any empty pages
    at the end of relation or TRUNCATE truncated the relation in place.
    Since shared_buffers needed to be scanned multiple times,
    it could take a long time to finish those commands especially
    when shared_buffers was large.

    This commit changes the logic so that shared_buffers is scanned only
    one time for those three relation forks.

    Author: Kirk Jamison
    Reviewed-by: Masahiko Sawada, Thomas Munro, Alvaro Herrera, Takayuki Tsunakawa and F
    Discussion: https://postgr.es/m/D09B13F772D2274BB348A310EE3027C64E2067@g01jpexmbkw24
```

old_snapshot_threshold vs индексы

```
commit bf9a60ee3349a2f2dc5fe6d571a8d39cfc634371
Author: Kevin Grittner <kgrittn@postgresql.org>
Date:   Fri Jun 10 09:25:31 2016 -0500

    Fix interaction between CREATE INDEX and ”snapshot too old”.

    Since indexes are created without valid LSNs, an index created
    while a snapshot older than old_snapshot_threshold existed could
    cause queries to return incorrect results when those old snapshots
    were used, if any relevant rows had been subject to early pruning
    before the index was built.  Prevent usage of a newly created index
    until all such snapshots are released, for relations where this can
    happen.

    Questions about the interaction of ”snapshot too old” with index
    creation were initially raised by Andres Freund.

    Reviewed by Robert Haas.
```

LISTEN менее дорогой

```
commit 51004c7172b5c35afac050f4d5849839a06e8d3b
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Sun Sep 22 11:46:29 2019 -0400

    Make some efficiency improvements in LISTEN/NOTIFY.

    Move the responsibility for advancing the NOTIFY queue tail pointer
    from the listener(s) to the notification sender, and only have the
    sender do it once every few queue pages, rather than after every batch
    of notifications as at present.  This reduces the number of times we
    execute asyncQueueAdvanceTail, and reduces contention when there are
    multiple listeners (since that function requires exclusive lock).
    This change relies on the observation that we don't really need the tail
    pointer to be exactly up-to-date.  It's certainly not necessary to
    attempt to release disk space more often than once per SLRU segment.
    The only other usage of the tail pointer is that an incoming listener,
    if it's the only listener in its database, will need to scan the queue
    forward from the tail; but that's surely a less performance-critical
    path than routine sending and receiving of notifies.  We compromise by
    advancing the tail pointer after every 4 pages of output, so that it
    shouldn't get more than a few pages behind.

    Also, when sending signals to other backends after adding notify
    message(s) to the queue, recognize that only backends in our own
    database are going to care about those messages, so only such
```

Avoid full index for GIN

```
commit 4b754d6c16e16cc1a1adf12ab0f48603069a0efd
Author: Alexander Korotkov <akorotkov@postgresql.org>
Date:   Sat Jan 18 01:11:39 2020 +0300

    Avoid full scan of GIN indexes when possible

    The strategy of GIN index scan is driven by opclass-specific extract_query
    method.  This method that needed search mode is GIN_SEARCH_MODE_ALL.  This
    mode means that matching tuple may contain none of extracted entries.  Simple
    example is '!term' tsquery, which doesn't need any term to exist in matching
    tsvector.

    In order to handle such scan key GIN calculates virtual entry, which contains
    all TIDs of all entries of attribute.  In fact this is full scan of index
    attribute.  And typically this is very slow, but allows to handle some queries
    correctly in GIN.  However, current algorithm calculate such virtual entry for
    each GIN_SEARCH_MODE_ALL scan key even if they are multiple for the same
    attribute.  This is clearly not optimal.

    This commit improves the situation by introduction of "exclude only" scan keys.
    Such scan keys are not capable to return set of matching TIDs.  Instead, they
    are capable only to filter TIDs produced by normal scan keys.  Therefore,
    each attribute should contain at least one normal scan key, while rest of them
    may be "exclude only" if search mode is GIN_SEARCH_MODE_ALL.
```

Server features (1/2)

- Random UUID generation in core
- Corruption error code
- min/max для pg_lsn
- FTS для греческого языка
- CREATE DATABASE ... LOCALE = ...
- Parallel vacuum indexes
- ignore_invalid_pages GUC

Random UUID generation in core

```
commit 5925e5549890416bcf588334d9d0bc99f8ad6c7f
Author: Peter Eisentraut <peter@eisentraut.org>
Date:   Sun Jul 14 14:30:27 2019 +0200

    Add gen_random_uuid function

    This adds a built-in function to generate UUIDs.

    PostgreSQL hasn't had a built-in function to generate a UUID yet,
    relying on external modules such as uuid-ossp and pgcrypto to provide
    one.  Now that we have a strong random number generator built-in, we
    can easily provide a version 4 (random) UUID generation function.

    This patch takes the existing function gen_random_uuid() from pgcrypto
    and makes it a built-in function.  The pgcrypto implementation now
    internally redirects to the built-in one.

    Reviewed-by: Fabien COELHO <coelho@cri.ensmp.fr>
    Discussion: https://www.postgresql.org/message-id/6a65610c-46fc-2323-6b78-e8086340a3
```

Corruption error code

```
commit fd6ec93bf890314ac694dc8a7f3c45702ecc1bbd
Author: Peter Eisentraut <peter@eisentraut.org>
Date:   Thu Aug 1 11:05:08 2019 +0200

    Add error codes to some corruption log messages

    In some cases we have elog(ERROR) while corruption is certain and we
    can give a clear error code ERRCODE_DATA_CORRUPTED or
    ERRCODE_INDEX_CORRUPTED.

    Author: Andrey Borodin <x4mmm@yandex-team.ru>
    Discussion: https://www.postgresql.org/message-id/flat/25F6C686-6442-4A6B-BAF8-A6F7E
```

min/max для pg_lsn

```
commit 313f87a17155a6dbd27a3ce687cf59bd171fe75e
Author: Michael Paquier <michael@paquier.xyz>
Date:   Fri Jul 5 12:21:11 2019 +0900

    Add min() and max() aggregates for pg_lsn

    This is useful for monitoring, when it comes for example to calculations
    of WAL retention with replication slots and delays with a set of
    standbys.

    Bump catalog version.

    Author: Fabrízio de Royes Mello
    Reviewed-by: Surafel Temesgen
    Discussion: https://postgr.es/m/CAFcNs+oc8ZoHhowA4rR1GGCgG8QNgK_TOwPRVYQo5rYy8_PXzA@
```

FTS для греческого языка

```
commit 7b925e12703652fef63a2fbbb28d3407b2971d6e
Author: Peter Eisentraut <peter@eisentraut.org>
Date:   Thu Jul 4 13:10:41 2019 +0200

    Sync our Snowball stemmer dictionaries with current upstream

    The main change is a new stemmer for Greek.  There are minor changes
    in the Danish and French stemmers.

    Author: Panagiotis Mavrogiorgos <pmav99@gmail.com>
```

CREATE DATABASE ... LOCALE = ...

```
commit 06140c201b982436974d71e756d7331767a41e57
Author: Peter Eisentraut <peter@eisentraut.org>
Date:   Tue Jul 23 14:40:42 2019 +0200

    Add CREATE DATABASE LOCALE option

    This sets both LC_COLLATE and LC_CTYPE with one option.  Similar
    behavior is already supported in initdb, CREATE COLLATION, and
    createdb.

    Reviewed-by: Fabien COELHO <coelho@cri.ensmp.fr>
    Discussion: https://www.postgresql.org/message-id/flat/d9d5043a-dc70-da8a-0166-1e218
```

Parallel vacuum indexes

```
commit 40d964ec997f64227bc0ff5e058dc4a5770a70a9
Author: Amit Kapila <akapila@postgresql.org>
Date:   Mon Jan 20 07:57:49 2020 +0530

    Allow vacuum command to process indexes in parallel.

    This feature allows the vacuum to leverage multiple CPUs in order to
    process indexes.  This enables us to perform index vacuuming and index
    cleanup with background workers.  This adds a PARALLEL option to VACUUM
    command where the user can specify the number of workers that can be used
    to perform the command which is limited by the number of indexes on a
    table.  Specifying zero as a number of workers will disable parallelism.
    This option can't be used with the FULL option.

    Each index is processed by at most one vacuum process.  Therefore parallel
    vacuum can be used when the table has at least two indexes.

    The parallel degree is either specified by the user or determined based on
    the number of indexes that the table has, and further limited by
    max_parallel_maintenance_workers.  The index can participate in parallel
    vacuum iff it's size is greater than min_parallel_index_scan_size.

    Author: Masahiko Sawada and Amit Kapila
    Reviewed-by: Dilip Kumar, Amit Kapila, Robert Haas, Tomas Vondra,
    Mahendra Singh and Sergei Kornilov
```

ignore_invalid_pages GUC

```
commit 41c184bc642b25f67fb1d8ee290f28805fa5a0b4
Author: Fujii Masao <fujii@postgresql.org>
Date:   Wed Jan 22 11:56:34 2020 +0900

    Add GUC ignore_invalid_pages.

    Detection of WAL records having references to invalid pages
    during recovery causes PostgreSQL to raise a PANIC-level error,
    aborting the recovery. Setting ignore_invalid_pages to on causes
    the system to ignore those WAL records (but still report a warning),
    and continue recovery. This behavior may cause crashes, data loss,
    propagate or hide corruption, or other serious problems.
    However, it may allow you to get past the PANIC-level error,
    to finish the recovery, and to cause the server to start up.

    Author: Fujii Masao
    Reviewed-by: Michael Paquier
    Discussion: https://www.postgresql.org/message-id/CAHGQGwHCK6f77yeZD4MHOnN+PaTf6XiJf
```

Server features (2/2)

- log_min_duration_sample / log_statement_sample_rate
- .datetime() in JSONPATH
- ALTER STATISTICS [IF EXISTS] ... SET STATISTICS ...
- DROP DATABASE ... FORCE
- gcd(), lmc()
- jsonb_set_lax()
- ANALYZE progress report

log_min_duration_sample / log_statement_sample_rate

```
commit 6e3e6cc0e884a6091e1094dff29db430af08fb93
Author: Tomas Vondra <tomas.vondra@postgresql.org>
Date:   Mon Nov 4 01:57:45 2019 +0100

    Allow sampling of statements depending on duration

    This allows logging a sample of statements, without incurring excessive
    log traffic (which may impact performance).  This can be useful when
    analyzing workloads with lots of short queries.

    The sampling is configured using two new GUC parameters:

     * log_min_duration_sample - minimum required statement duration

     * log_statement_sample_rate - sample rate (0.0 - 1.0)

    Only statements with duration exceeding log_min_duration_sample are
    considered for sampling. To enable sampling, both those GUCs have to
    be set correctly.

    The existing log_min_duration_statement GUC has a higher priority, i.e.
    statements with duration exceeding log_min_duration_statement will be
    always logged, irrespectedly of how the sampling is configured. This
    means only configurations
```

.datetime() in JSONPATH

```
commit bffe1bd68457e43925c362d8728ce3b25bdf1c94
Author: Alexander Korotkov <akorotkov@postgresql.org>
Date:   Wed Sep 25 21:54:14 2019 +0300

    Implement jsonpath .datetime() method

    This commit implements jsonpath .datetime() method as it's specified in
    SQL/JSON standard.  There are no-argument and single-argument versions of
    this method.  No-argument version selects first of ISO datetime formats
    matching input string.  Single-argument version accepts template string as
    its argument.

    Additionally to .datetime() method itself this commit also implements
    comparison ability of resulting date and time values.  There is some difficulty
    because exising jsonb_path_*() functions are immutable, while comparison of
    timezoned and non-timezoned types involves current timezone.  At first, current
    timezone could be changes in session.  Moreover, timezones themselves are not
    immutable and could be updated.  This is why we let existing immutable functions
    throw errors on such non-immutable comparison.  In the same time this commit
    provides jsonb_path_*_tz() functions which are stable and support operations
    involving timezones.  As new functions are added to the system catalog,
    catversion is bumped.

    Support of .datetime() method was the only blocker prevents T832 from being
    marked as supported.  sql_features.txt is updated accordingly.
```

ALTER STATISTICS [IF EXISTS] ... SET STATISTICS ...

```
commit d06215d03b50c264a0f31e335b895ee1b6753e68
Author: Tomas Vondra <tomas.vondra@postgresql.org>
Date:   Tue Sep 10 20:09:27 2019 +0200

    Allow setting statistics target for extended statistics

    When building statistics, we need to decide how many rows to sample and
    how accurate the resulting statistics should be. Until now, it was not
    possible to explicitly define statistics target for extended statistics
    objects, the value was always computed from the per-attribute targets
    with a fallback to the system-wide default statistics target.

    That's a bit inconvenient, as it ties together the statistics target set
    for per-column and extended statistics. In some cases it may be useful
    to require larger sample / higher accuracy for extended statics (or the
    other way around), but with this approach that's not possible.

    So this commit introduces a new command, allowing to specify statistics
    target for individual extended statistics objects, overriding the value
    derived from per-attribute targets (and the system default).

      ALTER STATISTICS stat_name SET STATISTICS target_value;

    When determining statistics target for an extended statistics object we
    first look at this explicitly set value. When this value is -1, we fall
```

DROP DATABASE ... FORCE

```
commit 1379fd537f9fc7941c8acff8c879ce3636dbdb77
Author: Amit Kapila <akapila@postgresql.org>
Date:   Tue Nov 12 11:06:13 2019 +0530

    Introduce the 'force' option for the Drop Database command.

    This new option terminates the other sessions connected to the target
    database and then drop it.  To terminate other sessions, the current user
    must have desired permissions (same as pg_terminate_backend()).  We don't
    allow to terminate the sessions if prepared transactions, active logical
    replication slots or subscriptions are present in the target database.

    Author: Pavel Stehule with changes by me
    Reviewed-by: Dilip Kumar, Vignesh C, Ibrar Ahmed, Anthony Nowocien,
    Ryan Lambert and Amit Kapila
    Discussion: https://postgr.es/m/CAP_rwwmLJJbn70vLOZFpxGw3XD7nLB_7+NKz46H5EOO2k5H7OQ@
```

gcd(), lmc()

```
commit 13661ddd7eaec7e2809ff5c29fc14653b6161036
Author: Dean Rasheed <dean.a.rasheed@gmail.com>
Date:   Sat Jan 25 14:00:59 2020 +0000

    Add functions gcd() and lcm() for integer and numeric types.

    These compute the greatest common divisor and least common multiple of
    a pair of numbers using the Euclidean algorithm.

    Vik Fearing, reviewed by Fabien Coelho.

    Discussion: https://postgr.es/m/adbd3e0b-e3f1-5bbc-21db-03caf1cef0f7@2ndquadrant.com
```

jsonb_set_lax()

```
commit a83586b5543b948f9e621462537a7303b113c482
Author: Andrew Dunstan <andrew@dunslane.net>
Date:   Fri Jan 17 11:41:35 2020 +1030

    Add a non-strict version of jsonb_set

    jsonb_set_lax() is the same as jsonb_set, except that it takes and extra
    argument that specifies what to do if the value argument is NULL. The
    default is 'use_json_null'. Other possibilities are 'raise_exception',
    'return_target' and 'delete_key', all these behaviours having been
    suggested as reasonable by various users.

    Discussion: https://postgr.es/m/375873e2-c957-3a8d-64f9-26c43c2b16e7@2ndQuadrant.com

    Reviewed by: Pavel Stehule
```

ANALYZE progress report

```
commit a166d408eb0b35023c169e765f4664c3b114b52e
Author: Alvaro Herrera <alvherre@alvh.no-ip.org>
Date:   Wed Jan 15 11:02:09 2020 -0300

    Report progress of ANALYZE commands

    This uses the progress reporting infrastructure added by c16dc1aca5e0,
    adding support for ANALYZE.

    Co-authored-by: Álvaro Herrera <alvherre@alvh.no-ip.org>
    Co-authored-by: Tatsuro Yamada <tatsuro.yamada.tf@nttcom.co.jp>
    Reviewed-by: Julien Rouhaud, Robert Haas, Anthony Nowocien, Kyotaro Horiguchi,
            Vignesh C, Amit Langote
```

Client utilities

- \d pg_toast* - показ индексов и родительской таблицы (-E)
- \dA – table access methods
- Дополнение CREATE TYPE
- Дополнение CREATE OR REPLACE
- pgbench – partitioned account
- vacuumdb –parallel

\d pg_toast* - показ индексов и родительской таблицы (-E)

```
commit eb5472da9f83c2e432ac27a053929947e354d20c
Author: Tom Lane <tgl@sss.pgh.pa.us>
Date:   Tue Jul 23 15:25:56 2019 -0400

    Improve psql's \d output for TOAST tables.

    Add the name of the owning table to the footers for a TOAST table.
    Also, show all the same footers as for a regular table (in practice,
    this adds the index and perhaps the tablespace and access method).

    Justin Pryzby, reviewed by Fabien Coelho

    Discussion: https://postgr.es/m/20190422154902.GH14223@telsasoft.com
```

\dA — table access methods

```
commit d3a5fc17ebdbeaec81bec1f41e304485b3292da3
Author: Andres Freund <andres@anarazel.de>
Date:   Fri Mar 29 08:59:40 2019 -0700

    Show table access methods as such in psql's \dA.

    Previously we didn't display a type for table access methods.

    Author: Haribabu Kommi
    Discussion: CAJrrPGeeYOqP3hkZyohDx_8dot4zvPuPMDBmhJ=iC85cTBNeYw@mail.gmail.com
```

Дополнение CREATE TYPE

```
commit 71851e9ab7ac8409fabc6f64273149aa71fa29f5
Author: Michael Paquier <michael@paquier.xyz>
Date:   Mon Aug 19 16:33:24 2019 +0900

    Fix tab completion for CREATE TYPE in psql

    Oversight in 7bdc655.

    Author: Alexander Lakhin
    Discussion: https://postgr.es/m/5da8e325-c665-da95-21e0-c8a99ea61fbf@gmail.com
```

Дополнение CREATE OR REPLACE

```
    Add tab completion for CREATE OR REPLACE in psql.

    Author: Shenhao Wang
    Discussion: https://postgr.es/m/63580B24E208E3429D94153A03C68E0901AA8002D5@G08CNEXME
```

pgbench – partitioned account

```
commit b1c1aa53182372e907f3f7f090e7eb5f432a4c9a
Author: Amit Kapila <akapila@postgresql.org>
Date:   Tue Oct 1 09:50:26 2019 +0530

    pgbench: add --partitions and --partition-method options.

    These new options allow users to partition the pgbench_accounts table by
    specifying the number of partitions and partitioning method.  The values
    allowed for partitioning method are range and hash.

    This feature allows users to measure the overhead of partitioning if any.

    Author: Fabien COELHO
    Reviewed-by: Amit Kapila, Amit Langote, Dilip Kumar, Asif Rehman, and
    Alvaro Herrera
    Discussion: https://postgr.es/m/alpine.DEB.2.21.1907230826190.7008@lancre
```

vacuumdb –parallel

```
commit 47bc9ced0d0e96523e2c639c7066c9aede189ed7
Author: Amit Kapila <akapila@postgresql.org>
Date:   Wed Jan 29 11:08:50 2020 +0530

    Add --parallel option to vacuumdb command.

    Commit 40d964ec99 allowed vacuum command to leverage multiple CPUs by
    invoking parallel workers to process indexes.  This commit provides a
    '--parallel' option to specify the parallel degree used by vacuum command.

    Author: Masahiko Sawada, with few modifications by me
    Reviewed-by: Mahendra Singh and Amit Kapila
    Discussion: https://postgr.es/m/CAD21AoDTPMgzSkV4E3SFo1CH_x50bf5PqZFQf4jmqjk-C03BWg@
```

Минутка политкорректности

PostgreSQL окончательно освободился от термина “slave”.

- Replication: “slave” => “standby”.
- Partitioning: “master” => “root”, “slave” => “leaf”.

To be commited... Server (1/2)

- SQL/JSON: Standard clauses

  `https://commitfest.postgresql.org/27/1472/`

  `https://commitfest.postgresql.org/27/1473/`
- Incremental Materialized View Maintenance

  `https://commitfest.postgresql.org/27/2138/`
- Anycompatible (coalesce, greatest etc)

  `https://commitfest.postgresql.org/27/1911/`

To be commited... Server (2/2)

- INSERT .. SET ..

  https://commitfest.postgresql.org/27/2218/
- REINDEX .. [TABLESPACE]

  https://commitfest.postgresql.org/27/2269/
- CREATE OR REPLACE TRIGGER

  https://commitfest.postgresql.org/27/2307/

To be commited...
Performance (1/2)

- Incremental sort

  `https://commitfest.postgresql.org/27/1124/`

- Remove self join

  `https://commitfest.postgresql.org/27/1712/`

- Autoprepare

  `https://commitfest.postgresql.org/27/1747/`

- KNN-Btree

  `https://commitfest.postgresql.org/27/1804/`

- Btree deduplication

  `https://commitfest.postgresql.org/27/2202/`

To be commited...
Performance (2/2)

- Index Skip Scan

  `https://commitfest.postgresql.org/27/1741/`

- Hash group on disk

  `https://commitfest.postgresql.org/27/2201/`

- Empty materialize node

  `https://www.postgresql.org/message-id/flat/`

  `a373827a-260c-2b05-e7e6-32f4135ec093%40postgrespro.ru`

- GROUP BY optimization

  `https://www.postgresql.org/message-id/flat/`

  `7c79e6a5-8597-74e8-0671-1c39d124c9d6%40sigaev.ru`

To be commited...
Parititoning

- Join двух партиционированных таблиц

  `https://commitfest.postgresql.org/27/1553/`
- Join партицированной и не партицированной таблицы

  `https://commitfest.postgresql.org/27/2249/`
- Логическая репликация отдельных патиций

  `https://commitfest.postgresql.org/27/2301/`

To be commited... Interesting (1/2)

- Row filtering for logical replication

  `https://commitfest.postgresql.org/26/2270/`
- Generic type subscription

  `https://commitfest.postgresql.org/26/1062/`
- Global temporary table

  `https://commitfest.postgresql.org/26/2233/`
- Multiple extended statistics

  `https://commitfest.postgresql.org/27/2421/`

To be commited... Interesting (2/2)

- BRIN multi-range and bloom indexes

  https://www.postgresql.org/message-id/c1138ead-7668-f0e1-0638-c3be3237e812%402ndquadrant.com

- Opclass parameters

  https://www.postgresql.org/message-id/d22c3a18-31c7-1879-fc11-4c1ce2f5e5af%40postgrespro.ru

To be commited...
Miscellaneous

- pgbench — pseudo-random permutation

  `https://commitfest.postgresql.org/26/1736/`
- pgrename_temp()

  `https://commitfest.postgresql.org/26/2230/`
- Shared-memory stats collector

  `https://commitfest.postgresql.org/26/1708/`
- Page number in error context of vacuum

  `https://commitfest.postgresql.org/27/2361/`

Postgres PROFESSIONAL

Спасибо за внимание!