Hacker's Look at PostgreSQL 13:

How Does It All Work, Where Can It Be Used?

13

PostgresPro

PostgreSQL

Webinar

October 29 6 PM CET

Anastasia Lubennikova

Sr.Database Developer, Postgres Professional

PostgresPro

Webinars

About Postgres Professional

- Founded in 2015 by PostgreSQL contributors Oleg Bartunov & Teodor Sigaev.
  - 24/7/365 support for PostgreSQL
  - Migrations to PostgreSQL
  - Remote DBA for PostgreSQL
  - HA PostgreSQL deployments
  - Database audits
- 2 supported PostgreSQL forks:
  - **Postgres Pro Standard** (early access to PostgreSQL features, 1-3 years prior to the official release)
  - **Postgres Pro Enterprise** (enterprise-ready version of Postgres)
- Custom feature development for PostgreSQL

About me

- PostgreSQL contributor since 2015
  - Index-only scan for GiST
  - Microvacuum for GiST
  - B-tree INCLUDE clause
  - B-tree deduplication
  - pg_probackup co-maintainer

- Tier 3 support for PostgreSQL and PostgresPro solutions

- Education and mentoring

Agenda for today's webinar

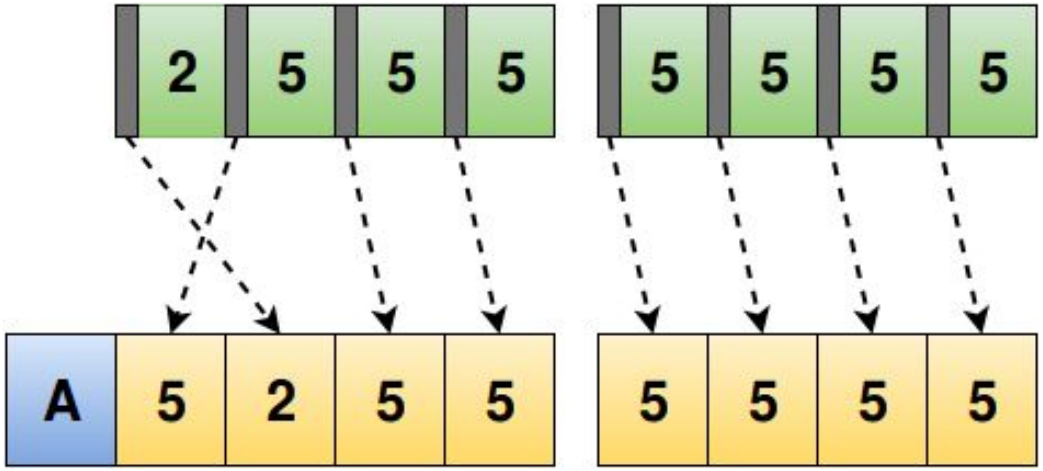
- Major features of the release
  - b-tree deduplication
  - incremental sorting
  - parallelized vacuum
  - enhanced partitioning
- Notable improvements for
  - backups & verification
  - security
  - many other areas
- Steps toward future improvements

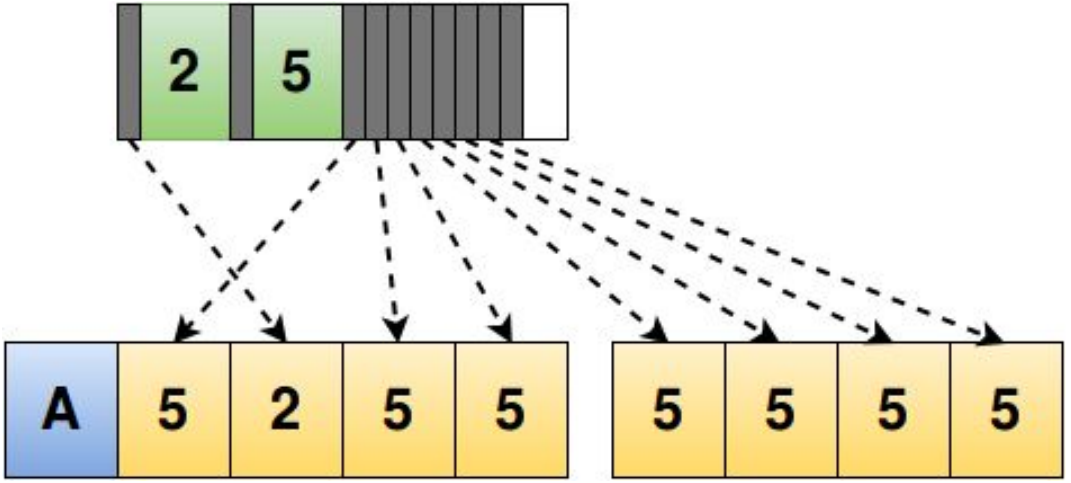
Backward compatibility

- wal_keep_segments → wal_keep_size
  - wal_keep_size = wal_keep_segments * wal_segment_size
  - New setting: max_slot_wal_keep_size
- effective_io_concurrency
  - Use formula from release notes to tune the value
  - New parameter: maintenance_io_concurrency.
- Wait events renamed to improve consistency
  - Hash/Batch/Allocating → HashBatchAllocate
  - ControlFileLock → ControlFile
  - clog → XactBuffer
  - AsyncCtlLock → NotifySLRU

https://postgresqlco.nf - PostgreSQL configuration for HUMANS

B-tree deduplication





B-tree deduplication

- Transparently makes indexes 2.5X — 5X smaller (on real data)
- On TPC-H benchmark it saves 40% of space
  - 5921 MB → 3576 MB
- New b-tree parameter: deduplicate_items
  - Enabled by default for all user indexes
- Opclass restrictions
  - Does not support numeric, float and container types
- Deduplication overhead is amortized across insertions
  - Only 2% overhead on append-only benchmark
- REINDEX after upgrade to make use of it

B-tree deduplication in UNIQUE index

- Allows to delay splits caused by MVCC copies
- In synergy with microvacuum, helps to avoid index bloat
- Benchmark with pgbench
  - Old index growth: 10.5 GiB → 19.4 GiB
  - New index growth: 10.5 GiB → 12.7 GiB
- REINDEX after upgrade to make use of it

Incremental sorting

- Optimizes multikey sorting when intermediate result is sorted on a prefix
  - Reduces memory consumption
  - Read less rows with LIMIT
- Useful for
  - ORDER BY
  - DISTINCT
  - GROUP BY
  - window functions (only in v14)
  - merge joins
- New parameter: enable_incrementalsort
  - Enabled by default

Incremental sorting. Example

```
CREATE TABLE test (id integer, data char(100));
CREATE INDEX on test (id);

INSERT INTO test SELECT i%1000, 'payload'
FROM generate_series(0,10000000) AS i;

table_size = 1.3 Gb, index_size = 66 Mb

EXPLAIN ANALYZE SELECT id, data FROM test
ORDER BY id, data LIMIT 1000;
```

Plan	**Time, ms**
Incremental Sort over Index Scan	**25**
Sort over Seq Scan with 2 parallel workers	**600**

Disk-based Hash Aggregation

- Optimized hash aggregation
  - Spills hash table to disk when it exceeds memory limit
  - Chooses HashAggregaton more often
- No more OOM killer because of the planner's mistakes
- New parameter: hash_mem_multiplier
  - Compute memory limit for hash tables as work_mem * hash_mem_multiplier
  - Default is 1
- Typical query:

```
SELECT count(*) FROM test GROUP BY id;
```

Parallel VACUUM

- Indexes are processed in parallel
  - 2 times faster with 3 workers
  - The table itself is still vacuumed by one process.
- New vacuum option: PARALLEL n_workers
  - Enabled by default
  - n_workers is limited by max_parallel_maintenance_workers and the number of indexes.
  - index size must be > min_parallel_index_scan_size
- Limitations:
  - not VACUUM FULL
  - not autovacuum

Improved autovacuum

- Now autovacuum runs for for append-only tables
  - It allows to use index-only scans for such tables
  - It prevents transaction id wraparound
- New parameters:

  autovacuum_vacuum_insert_threshold,

  autovacuum_vacuum_insert_scale_factor

Partition-wise join

- An exact match of partition bounds is no longer needed
  - i.e. table partitioned by days & table partitioned by weeks
- New parameter: enable_partitionwise_join
  - Disabled by default
- Limitations:
  - only works for equi-join (join on t1.a = t2.b)

Logical replication of partitioned tables

- Replicate partitioned table easily.
- Replicate regular table to a partitioned one.
  - Especially useful to run analytical queries on replicas.

Before ROW trigger for partitioned table

- Create BEFORE trigger for partitioned table easily.
- Limitation:
  - Trigger function cannot reroute tuple to another partition

Backups and verification

- pg_basebackup creates a “backup manifest”
  - Enabled by default
- pg_verifybackup
  - checks files in the backup
  - checks WAL files (relies on pg_waldump)
- pg_stat_progress_basebackup
  system view to report the progress of streaming base backups

Security

- Change authentication defaults for a new instance
  - peer (or md5) for local connections and md5 for external
  - Be aware that packaged Postgres can apply extra changes
- Add "password_protocol" connection parameter to libpq
  - Default is plaintext
- Raise default minimum TLS version from 1.0 to 1.2
  - ssl_min_protocol_version
- Show the ssl_passphrase_command setting only to superusers

More features of PG 13

- TRUSTED extensions
  - create extension without a superuser privileges
  - List of built-in trusted extensions is HERE
  - check recent CVE-2020-14349
- Online change for some parameters
  - primary_conninfo, primary_slot_name

    and wal_receiver_create_temp_slot
- Recovery will pause if PITR target not reached

More features of PG 13

- TOAST extraction and decompression improvement
- pg_dump for foreign tables
- Extended monitoring
  - log_statement_sample_rate
  - log_min_duration_sample
- Glossary in documentation

Contribute to PostgreSQL

- Vote for features
- Share your opinion on usability
- Test early and report bugs and inconsistent behavior
- Share performance results

https://commitfest.postgresql.org/

Postgres PROFESSIONAL

Thank you for attention!
Any questions?

a.lubennikova@postgrespro.com

PostgresPro

https://postgrespro.com/