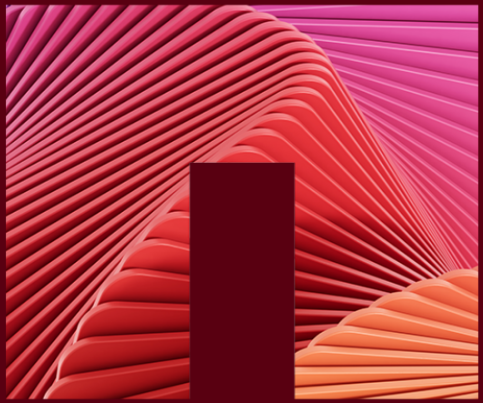
NetApp

PostgreSQL 17

New features



Josef Machytka <josef.machytka@netapp.com>
NetApp Open Source Services
2024-11-12 - NetApp workshop

Josef Machytka

- Professional Service Consultant - PostgreSQL specialist at NetApp Open Source Services / credativ
- 30+ years of experience with different databases.
- PostgreSQL (12y), BigQuery (7y), Oracle (15y), MySQL (12y), Elasticsearch (5y), MS SQL (5y).
- 10+ years of experience with Data Ingestion pipelines, Data Analysis, Data Lake and Data Warehouse
- 2 years of practical experience with different LLMs / AI including their architecture and principles.
- From Czechia, living now 11 years in Berlin.

- LinkedIn: linkedin.com/in/josef-machytka
- ResearchGate: researchgate.net/profile/Josef-Machytka
- Academia.edu: netapp.academia.edu/JosefMachytka
- Medium: medium.com/@josef.machytka
- sessionize: sessionize.com/josefmachytka

PostgreSQL 17 New Features

- PostgreSQL 17 released on September 26, 2024
- With many significant improvements in:

- Administration
- Monitoring and Performance
- Developer Features
- Incremental Backups
- Backup and Restore
- Logical Replication
- JSON Functions

AI image created
by the author of this talk
using DeepDreamGenerator

Administration

- Vacuum memory consumption improvements
- Removed 1 GB limit
- Vacuum can store more tuple references in memory
- Uses adaptive radix trees
- Multiple index scans on autovacuum no longer needed

- New *allow_alter_system* configuration parameter
- Allows/forbids ALTER SYSTEM command
- Must be changed in postgresql.conf, configuration reloaded

Administration

- New *transaction_timeout* parameter
- Cancels transactions (idle or active) after configured time
- Recommended only for session settings
- Not for global settings !

- New EXPLAIN options
- *SERIALIZE* shows time/memory spent deTOASTing data
- Has option TEXT and BINARY, TEXT is default
- *MEMORY* reports memory usage in planning phase

Administration

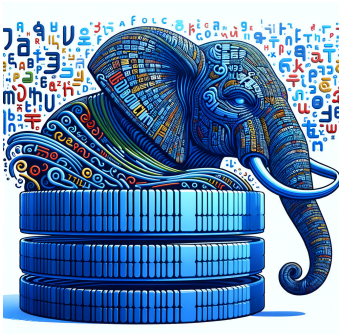
- *MAINTAIN* permission / *pg_maintain* predefined role
- For VACUUM, ANALYZE, REINDEX,
  REFRESH MATERIALIZED VIEW, CLUSTER and LOCK TABLE

- New event triggers for REINDEX and login
- New *event_triggers = [on|off]* parameter
- Switches event triggers on/off
- Useful in case of errors in some event trigger
- Does not require restart

Administration

- *COPY* command now has error handling
- *WITH (FORMAT csv, ON_ERROR 'ignore')*
- New *LOG_VERBOSITY verbose* setting
- PostgreSQL reports details about errors in COPY

- Identity space for IDENTITY column is now properly shared across partitions
- Values are now unique across partitions



Administration

- Wait event descriptions in new *pg_wait_events* view

```
SELECT a.pid, a.state, a.wait_event_type, a.wait_event, w.description, a.query
FROM pg_stat_activity a, pg_wait_events w
WHERE state = 'active' AND a.wait_event = w.name \gx

-[ RECORD 1 ]---+-----------------------------------------------------
pid             | 126315
state           | active
wait_event_type | Timeout
wait_event      | PgSleep
description     | Waiting due to a call to pg_sleep or a sibling function
query           | SELECT pg_sleep(3600);
```


Monitoring and Statistics

- Columns related to checkpointer moved from *pg_stat_bgwriter* to *pg_stat_checkpointer*
- Involves *buffers_clean, maxwritten_clean, buffers_alloc, stats_reset* columns

- *pg_stat_vacuum_progress* shows how many indexes have been vacuumed already

Monitoring and Statistics

- *pg_stat_statements* improvements:
- Old columns *blk_read_time, blk_write_time* split to multiple columns for better granularity
- New columns are *local_blk_read_time, local_blk_write_time, shared_blk_read_time, shared_blk_write_time*
- Additional columns *stats_since, minmax_stats_since* show when statistics were reset
- Normalization of parameters in CALL and SAVEPOINT queries for better grouping

Performance

- SLRU (Simple Least Recently Used) caches separated into multiple banks with dedicated locking for each
- SLRU types have been named better
- *commit_timestamp_buffers, multixact_member_buffers, notify_buffers, serializable_buffers*, etc.
- Helpful for high concurrency workloads
- Sizes can be configured individually, but changes require restart
- Columns *blks_hit, blks_read* in *pg_stat_slru* allows to estimate which cache sizes should be adjusted

Developer Features

- Improved query performance of IN clauses that use B-tree indexes
- In older versions data fetched by separate sub-plan
- Now subquery of IN clause is handled as join -> much more efficient

- Added MERGE [...] WHEN NOT MATCHED BY SOURCE
- New MERGE RETURNING
- Parallel CREATE INDEX for BRIN indexes
- Query plan optimizations for NOT NULL constraints
- Improvements for CTEs (WITH queries)

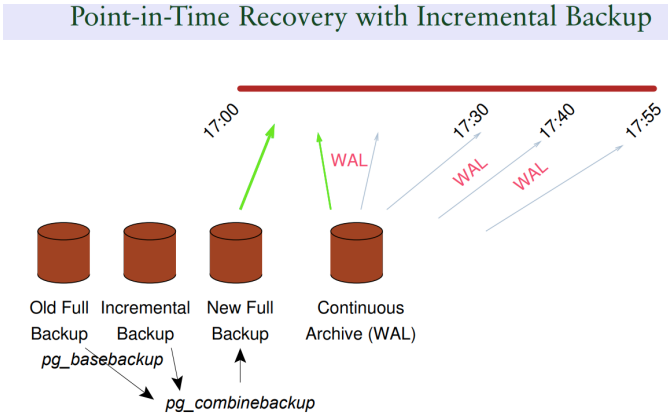
Incremental Base Backups

- *pg_basebackup* can do full or incremental backup
- Incremental backups stores only changed blocks, not whole data files
- Added new background process WAL Summarizer - must be enabled using *summarize_wal = on*
- New parameters *summarize_wal = on* and *wal_summary_keep_time = '1d'*
- New tool *pg_combinebackup* creates valid backup from a full and incremental backup(s)

```
$ pg_basebackup -D full/backup1
$ pg_basebackup --incremental=full/backup1/backup_manifest -D incr/backup2
$ du -sh full/backup1 incr/backup2
1.9G  full/backup1
25M   incr/backup2

$ pg_combinebackup -o comb/backup3 full/backup1 incr/backup2
$ du -sh comb/backup3
1.9G  comb/backup3
```

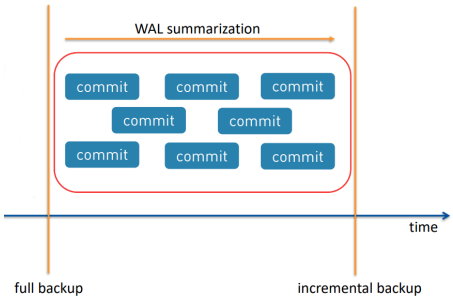

Incremental Base Backups



(Image from slides Major Features: Postgres 17 by Bruce Momjian (EDB))

Incremental Base Backups

- WAL summaries can be inspected with ”pg_wal_summary_contents”
- For relfilenode shows relblocknumber of changed blocks
- State of summarizer process can be checked with ”pg_get_wal_summarizer_state”



(Image from slides PostgreSQL 17 Highlights by Daniel Westermann (dbi))

Incremental Base Backups

- Incremental backup can reference full backup or another incremental backup
- If incremental backups references previous incremental backup, pg_combinebackup must use all of them
- Example pattern: weekly full backup, daily incremental backups
- Even with multiple incremental backups restore is still quicker than applying all WAL files
- We can create a new full backup from the last full backup and all incremental backups
- And then use it as a new base for incremental backups

Incremental Base Backups

- Backups contain ”backup manifest” file with metadata

```
postgres@pgbox:/home/postgres/ [pgxxx] cat backups/bkpfull_1/backup_manifest
{ "PostgreSQL-Backup-Manifest-Version": 2,
"System-Identifier": 7350636235843627793,
"Files": [
{ "Path": "backup_label", "Size": 225, "Last-Modified": "2024-03-26 11:48:13 GMT",
"Checksum-Algorithm": "CRC32C", "Checksum": "b92de9a2" },
{ "Path": "global/1262", "Size": 8192, "Last-Modified": "2024-03-26 11:42:46 GMT",
"Checksum-Algorithm": "CRC32C", "Checksum": "b0545bb5" },
{ "Path": "global/2964", "Size": 0, "Last-Modified": "2024-03-26 11:42:46 GMT", "ChecksumAlgorithm": "↩
 CRC32C", "Checksum": "00000000" },
…
{ "Path": "global/pg_control", "Size": 8192, "Last-Modified": "2024-03-26 11:48:13 GMT",
"Checksum-Algorithm": "CRC32C", "Checksum": "43872087" }
],
"WAL-Ranges": [
{ "Timeline": 1, "Start-LSN": "0/20000D8", "End-LSN": "0/20001D0" }
],
"Manifest-Checksum": "6cedb1657cd68e79c962fc8c46a1d963dfd1c41c161fcc7b0cceb5acf17d8488"}
```


Backup and Restore

- Directory format in pg_dump with *--sync-method=syncfs* much faster on Linux with lots of files

- *pg_dump/pg_restore* include/exclude lists can now be read from other file via *--filter* parameter
- This way we can omit include/exclude command line parameters
- Syntax is very simple

```
include table mytable*
exclude table mytable2
```


Backup and Restore

- Large object handling was improved in pg_dump
- Still requires *--large-objects* switch to include them
- Dump is now smaller

```
lotest=# SELECT lo_create(id) FROM generate_series(1,100000) AS id;

$ for version in 16 17; do echo -n "$version: "; /usr/bin/time -f '%E %Mk mem' \
> pg_dump --cluster $version/main -Fd -f lotest-$version.dir lotest
> ls -lh lotest-$version.dir/toc.dat | awk '{print $5 " " $9}'; done

16: 3:49.95 381248k mem
94M lotest-16.dir/toc.dat

17: 3:49.20 114728k mem
3.4M lotest-17.dir/toc.dat
```


Logical Replication

- Logical replication can now be continued after a failover
- Configurable via CREATE / ALTER SUBCRIPTION or via create replication slot
- We can specify ”failover” mode ”true” to continue replication after failover
- *sync_replication_slots* parameter must be set to ”on”
- Starts new background worker on standby which synchronizes replication slots with primary
- In case of failover, new primary will have the same replication slots in the same state

Logical Replication

- pg_upgrade will now migrate subscriptions and logical replication slots in upgrades from v17
- On older versions logical replication slots are ignored

- New program *pg_createsubscriber* allows to switch a physical standby to a logical replica
- It creates publication and subscription objects for all databases on standby
- Currently its usage seems to be limited with specific chain of steps and requirements

- Hash indexes can now be used for the REPLICA IDENTITY to identify rows during logical replication

JSON Functions

- Improved JSONPATH support, added new operators
- Nwe SQL/JSON functions
- JSON_TABLE(), JSON_QUERY(), JSON_VALUE(), JSON_EXISTS()
- Special ON ERROR clauses to handle errors in query
- JSON_EXISTS() True/False if JSON path exists
- JSON_QUERY() returns JSON object on JSON path
- JSON_VALUE() returns SCALAR value based on JSON path

JSON_TABLE

- JSON_TABLE() function helps to extract data from JSON documents
- Requires COLUMNS clause with columns definitions
- Can force UTF8 encoding for extracted data
- Can add row numbers (ordinality column), nested paths have its own counters
- Allows ON ERROR clause to handle errors in scope of query
- Each column definition must have a name, data type and a path to the data in JSON document

```
SELECT *
FROM JSON_TABLE('{"key1": "val1", "key2": "val2"}',
'$[*]' COLUMNS (key1 text PATH '$.key1', key2 text PATH '$.key2'));

 key1 | key2
------+------
 val1 | val2
```


THANK YOU

- **Questions?**

NetApp

