EDB
Postgres® for the AI Generation

Incremental Backup in PostgreSQL 17

Robert Haas
VP, Chief Architect, Database Servers
PGCONF.EU | October 25, 2024

Feature Overview

- Incremental backup means that instead of copying the entire database when we make a backup, we copy only the parts of the database which have changed.
- Goal is to make backups smaller and faster (possibly at the expense of recovery time).
- We focus on *relation files*, which typically account for most of the database size. Currently, non-relation files are always copied in their entirety.
- This feature is *block-level*, which means that we make a separate decision for each 8kB block. This will produce smaller backups than a *file-level* feature, where a single byte change could cause you to copy an entire 1GB file.

Motivation

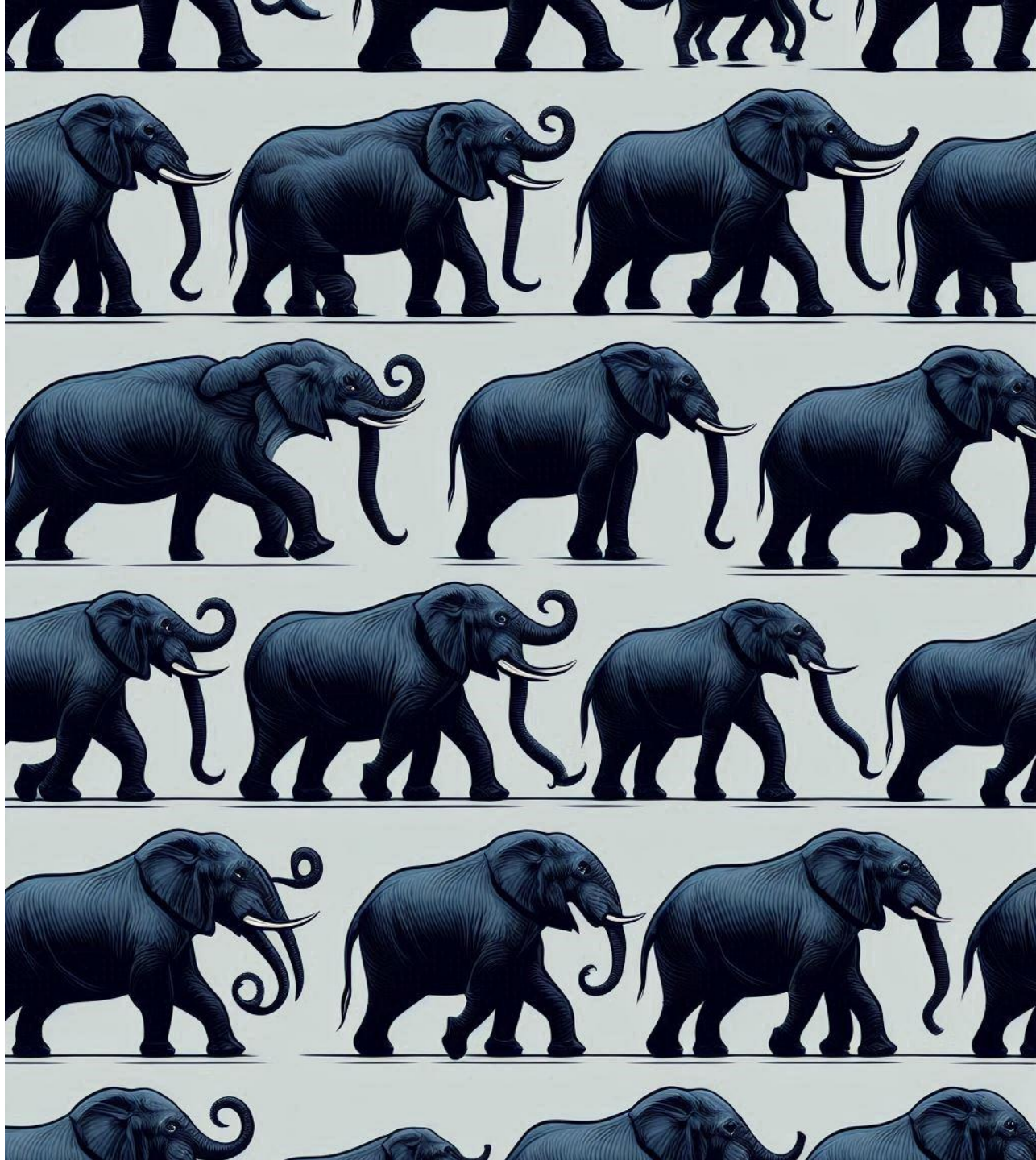
- Backing up large databases can be very challenging, with backup times sometimes exceeding one day.
- Users sometimes around this problem using specialized hardware with volume snapshot capabilities, or by using a cloud provider who solves the problem for them, but PostgreSQL should have a solution that works for self-hosted PostgreSQL without special hardware.
- Various backup tools for PostgreSQL already support some form of incremental backup, but each tool is different and all have their own bugs. This is open source, so a shared solution is better!

Disclaimer

- All code has bugs.
- Brand new code often has more bugs than old code.
- Any bugs in this code are likely to result in data loss.
- I think it works, so please try it!
- But also be careful.

EDB
Postgres® for the AI Generation

Taking an Incremental Backup



Basic Usage

- In `postgresql.conf` (or using `ALTER SYSTEM`), set `summarize_wal = on`.
  - Reload the configuration (or restart the server).

- `pg_basebackup -c fast -D sunday`
  - Full backup.
  - Use `-c fast` for testing to speed it up, but maybe not on a production system.

- `pg_basebackup -c fast -D monday --incremental sunday/backup_manifest`
  - Incremental backup based on Sunday's full backup.

- `pg_basebackup -c fast -D tuesday --incremental sunday/backup_manifest`
  `pg_basebackup -c fast -D tuesday --incremental monday/backup_manifest`
  - Incremental backup based on either on Sunday's full backup or Monday's incremental backup.

Optional: Configure `wal_summary_keep_time`

- When you take an incremental backup, how far back in time was the previous backup?
- `wal_summary_keep_time` should be greater than this amount of time.
- The default is 10 days, which I think should be more than enough in most cases.

Tooling

- Writing your own backup scripts is a bad idea!
- Use a quality backup tool that supports whatever you want to do.

EDB
Postgres® for the AI Generation

Architecture

Knowing What Has Changed: Requirements

- *Accurate*. If we think something has not changed when actually it did, then we will not include it in the backup and our data will be lost. If we think something has changed when it didn't really, that will not break anything but our backups will be larger.
- *Efficient*. It should be possible to determine what has changed without much effort.
- *Easy to Implement*. Reuse as much existing code as possible so that we don't have to write and debug too much new code.
- *Not Reliant on OS Features*. Especially, I like to avoid relying on things that work differently on different operating systems. Also, if something is entirely internal to PostgreSQL, it's easier to debug problems than if some of it is controlled at the OS level.

The Write-Ahead Log To The Rescue!

- PostgreSQL's write-ahead log contains all the information about which blocks have been modified.
- It's already used for many other purposes and has existing debugging tools like `pg_waldump` and `pg_walinspect`.
- However, the write-ahead log is very big, so we can't use it directly.
- Instead, we add a new WAL summarizer process which will read the WAL as it's generated and produce WAL summary files containing only the information that is required for incremental backup.
- These files are very small and cheap to generate.
- New `pg_walsummary` tool can be used to dump the information from the WAL summary files.

Filling in the Gaps With the Backup Manifest

- *Changed since when?* The backup manifest tells us at which position in the write-ahead log (LSN) the previous backup was taken.
  - From this, we know which WAL summary files are required.
  - We read all of the files starting at the LSN of the previous backup and up to the current time, and that tells us what has changed.

- It also gives us a list of the files that were present in the previous backup. If we see a file that according to the WAL summary has not been modified, then it's either:
  - a very old file that has never been modified, or else
  - a new file that was created after the current backup started.

EDB
Postgres® for the AI Generation

Restoring an Incremental Backup

Using pg_combinebackup

- Consider this example again:

```
pg_basebackup -c fast -D sunday
pg_basebackup -c fast -D monday --incremental sunday/backup_manifest
pg_basebackup -c fast -D tuesday --incremental monday/backup_manifest
```

- Everything that has changed between Monday and Tuesday is in the `tuesday` backup.
- Everything that has changed between Sunday and Monday is in the `monday` backup.
- Everything else is in the `sunday` backup.
- So we will need all three backups in order to restore:

```
pg_combinebackup sunday monday tuesday -o tuesday_full
```

Recovery Is Still Required!

- The output of `pg_combinebackup` is a full backup.
- When you start `postgres` on any full backup whatsoever, database recovery is required.
  - If the required WAL is present in the backup's `pg_wal` directory, then you can just start the server and it will perform recovery as normal.
  - Otherwise, you need to create `recovery.signal` or `standby.signal` and set `primary_conninfo` and/or `restore_command` just as you normally would.
- Incremental backup does not let you skip any step that would otherwise be required.
- Again, it's a good idea to leave this orchestration up to a well-written backup tool!

Restoring Even More Incrementally

- Consider this example again:

```
pg_basebackup -c fast -D sunday
pg_basebackup -c fast -D monday --incremental sunday/backup_manifest
pg_basebackup -c fast -D tuesday --incremental monday/backup_manifest
```

- Before we saw this:

```
pg_combinebackup sunday monday tuesday -o tuesday_full
```

- But this also works:

```
pg_combinebackup sunday monday -o monday_full
pg_combinebackup monday_full tuesday -o tuesday_full
```

Why Split Up `pg_combinebackup`?

- We can shorten the chain of backups that need to be combined in the event of a restore.
- Suppose we take an initial full backup and then an incremental backup every day. After 1 week:

```
pg_combinebackup sunday monday tuesday wednesday thursday \
   friday saturday -o saturday_full
```

- But:

```
pg_combinebackup sunday monday tuesday wednesday -o wednesday_full
rm -rf sunday monday tuesday wednesday
mv wednesday_full wednesday
```

- And then, if needed:

```
pg_combinebackup wednesday thursday friday saturday -o saturday_full
```

Evergreen

- Potentially, you can stop taking full backups altogether.
- Just take new incremental backups.
- Use the technique shown on the previous slide to keep the backup chains short.
- Potentially a big deal if the incremental backups are much faster than the full backups.

EDB
Postgres® for the AI Generation

Future Work

What's Next?

- Currently, `pg_combinebackup` can only work with plain-format backups, but tar-format backups may often be more convenient, especially for people relying on cloud storage.
  - I just committed a patch from Amul Sul to allow `pg_verifybackup` to work on tar-format backups, so maybe we'll even make further progress here in time for v18.
- Currently, `pg_combinebackup` can only work with backups stored locally, but wouldn't it be cool if it could access cloud storage directly?
- Performance work?
- Bug fixes?

Hacker Mentoring!

- One on one mentoring program for a limited number of people (applications not currently open)
- Monthly Zoom calls to discuss a topic of interest to inspiring hackers (see rhaas.blogspot.com for signup links)
- PostgreSQL Hacker Mentoring Discord to ask random questions about hacking (https://discord.gg/bx2G9KWyrY)

EDB
Postgres® for the AI Generation

Thank you!

Any questions?