Incremental Backup

Feature preview for PostgreSQL 17

Robert Haas
VP, Chief Database Scientist
EDB

EDB™

Disclaimer: Unreleased Feature Preview

- This feature was committed to the PostgreSQL master branch in December of 2023.
- If not reverted, it will appear in PostgreSQL 17, expected in September of 2024.
- The PostgreSQL community can decide to change or remove features at any time, but especially when they’re not yet released.
- This particular feature is at elevated risk because it is complicated and could eat your data.

Basic Idea of Incremental Backup

- We do not want to back up the data which has not changed since the previous backup.
- If we can avoid this, we can make the backups *smaller* and *faster*.

Knowing What Has Changed

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.

Knowing What Has Changed: Chosen Solution

- 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.

Knowing What Has Changed: User Interface

- Configure `summarize_wal = on` and reload the configuration
- Optionally, adjust `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.
- New debugging tool `pg_walsummary` allows us to look at the contents of WAL summary files so, in case of any problem, we can compare the WAL summary files against the original WAL.

Taking an Incremental Backup

Must Specify a Reference Backup

- If we want to copy only what has changed since some previous backup, how do we identify *which* previous backup we care about?

- Most of the time, you probably want to refer to the most recent backup, but maybe not always.

- For example:
  - full backup on Sunday
  - incremental backup on Monday containing changes since Sunday
  - incremental backup on Tuesday containing changes since { Sunday | Monday } ???

EDB

Specify Reference Backup via Backup Manifest

- `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.

What Information Do We Get From the Manifest?

- 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™

Restoring an Incremental Backup

Need the Whole Chain of Backups

- 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`

Warning: 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!

pg_combinebackup can be done in multiple steps

- 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`

Potentially, Full Backups Are Not Required!

- Suppose that every day you take a new incremental backup based on the previous day’s backup.
- To restore, you need to run `pg_combinebackup` starting with your full backup and including all of the incrementals after that up to the one that you want to restore.
- Eventually, this will be too many backups.
- But you can avoid this by turning an older incremental backup into a full backup and then erasing the older backups:
  - `pg_combinebackup sunday monday -o monday_full`
  - `rm -rf sunday monday && mv -f monday_full monday`

Restoring After Consolidation

- Before:

```
pg_combinebackup sunday monday tuesday wednesday -o wednesday_full
```

- After:

```
pg_combinebackup monday tuesday wednesday -o wednesday_full
```

Debugging

- What happens when it all goes wrong?
- `pg_combinebackup --debug`

Future Work

Future Work

- Backup utilities like pgbarman and pgBackRest need to be extended to support incremental backup.
  - Important to note: incremental backup affects retention policies!

- pg_combinebackup has a few scenarios that we could support but currently don't.
  - For example, it supports directory-format backups but not tar-format backups.

Thank You!

Any questions?

EDB™