Cloud friendly COPY

Copy from/to anything.

PostgreSQL Conference Europe 2022,
Berlin 2022.10.25

hannuk@google.com

Speaker introduction

Hannu Krosing
Cloud SQL / PostgreSQL
hannuk@google.com

Working with PostgreSQL since it was called `Postgres95` (and also played around with `Postgres 4.2` - without the "SQL" - a little before that).

My oldest *surviving* post on `postgresql-hackers@` mailing list archives is from January 1998, proposing using index for fast ORDER BY queries with LIMIT.

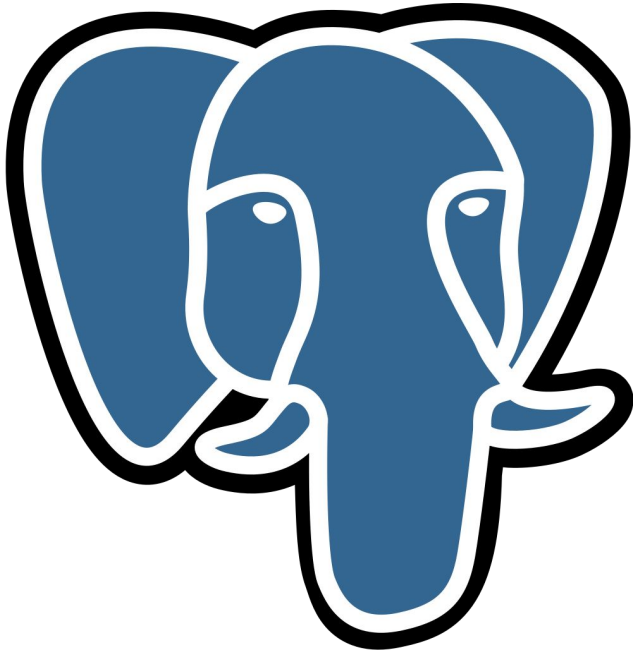
The first DBA at Skype, where I wrote patches for making **VACUUM** able to **work on more than one table in parallel** and invented the sharding and remote call language **`pl/proxy`** to make it easy to use PostgreSQL in an infinitely scalable way.

Have written books, **PostgreSQL 9 Admin Cookbook** and **PostgreSQL Server Programming**

After Skype I did 10+ years of PostgreSQL consulting all over the world as part of 2ndQuadrant.

For last few years he has been a PostgreSQL Database Engineer at Google working mostly with Cloud SQL.

PostgreSQL on local server and Cloud PostgreSQL



PostgreSQL on local server and Cloud PostgreSQL

- Runs on "a server"
- Fixed location
- Fixed number of CPUs
- Fixed amount of RAM
- Fixed size Disk

- Traffic to disk only through server

- Runs in a VM in the Cloud
- Moves around between physical servers
- Flexible nr of CPUs
- Flexible amount of RAM
- Disks grow and shrink as needed

- Disks are accessible directly
- Backups can happen entirely "on storage"
- Can snapshot live disk, then use in another VM

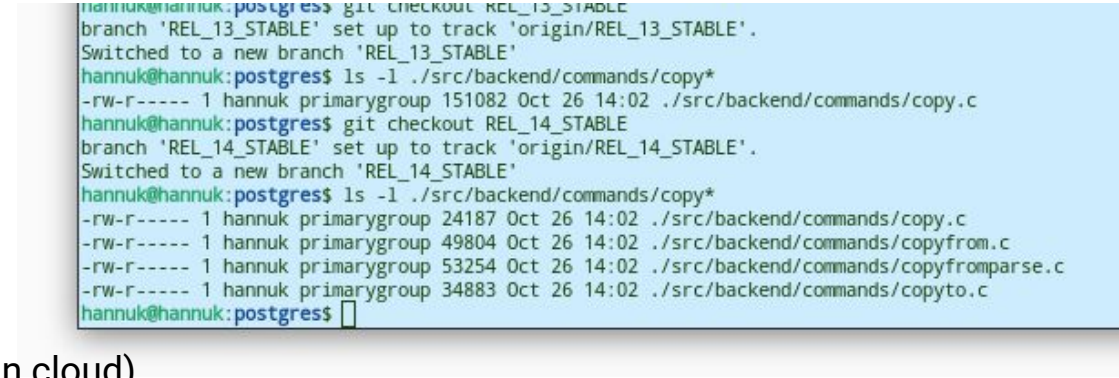
COPY statement now

- Hard-wired in syntax, flags are defined in grammar
- Monolithic code (was re-factored a little between Pg13 and Pg14)

- 3 “formats”
  - “Native”
  - CSV
  - “Binary”

- 3 “Transports”
  - STDIN/STDOUT
  - Local file (and this is disabled in cloud)
  - Pipe to local PROGRAM

- No compression or other stream manipulation
- No Parallelism
- No Index handling

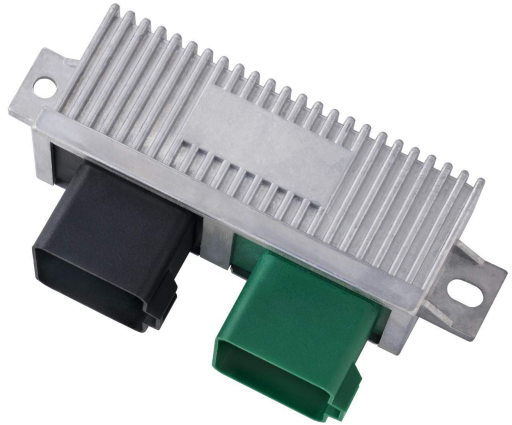


Cloud data now

- Tens of different data formats
  - avro, parquet, orc
  - xls
  - csv
- Tens of transports
  - GCS, S3, …
  - ftp, http(s), scp, …
  - NFS, CIFS
- Compression is widely used at rest and in transit
  - gzip, lz4, zstd, …
- Parallelism is often used

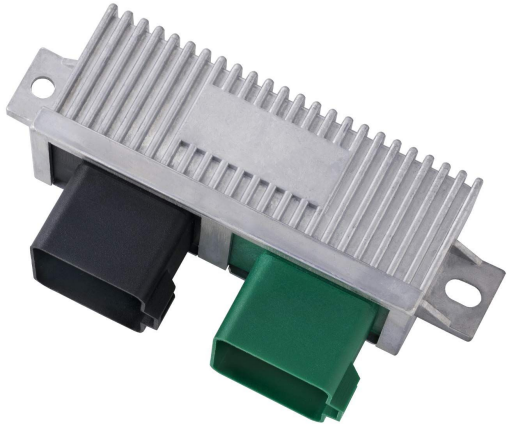
Extending PostgreSQL via pluggable modules

- PostgreSQL already uses **virtual function tables** like used in
  - Index access methods
  - Table access methods
  - logical decoding (CDC)
  - Foreign Data Wrappers (FDW)



This extensibility is needed for COPY too!

- 3 Dimensions
  - FORMAT
  - TRANSPORT
  - STREAM_PROCESSOR
    - (actually pre and post stream generation)
    - When still table format, then things like anonymisation
    - On external stream,
      - compression
      - Network control, throttling
      - encryption
- Internally more enhancements via full replacement hooks



COPY extended possibilities

- COPY mytable TO 'gs://mybucket/mybasefilename'
- COPY mytable TO gs://mybucket/mybasefilename

  WITH (CHUNKS 4, COMPRESSION ‘lz4’, …)

  uses parallel scans with direct storage to cloud via lz4 pipe

- COPY … WITH (INDEXES `defer`)

  copies first, then updates indexes

COPY full replacement extensions

- LOCAL COPY in database
  - COPY TABLE TO local://TABLE2
  - COPY SCHEMA S1 TO local://S2

  Can utilise FS level lopy

  Use reflink if available in FS (btrfs, XFS, …)
- COPY BETWEEN LOCAL DATABASES
- COPY TO fdw://…/
- COPY … WITH (CHUNKS 4, COMPRESSION ‘lz4’, …)
- COPY … WITH (FORMAT `split_binary`, INDEXES `defer`)

COPY extensions

- COPY … WITH (**CHUNKS 4**,...)
  - Set up parallel seqscan, but instead of berging results, stream directly to destination(s)
- COPY … WITH (COMPRSSION ‘lz4’, …)
  - As it says, compress the stream
- COPY … WITH (FORMAT `split_binary`)
  - There is a serious degradation in seqscan for huge tables with many TOASTED fields
  - Copying toast pointers as toast pointers and copying TOAST table separately should give huge speedup
- COPY … WITH (INDEXES `defer`, …)
  - Managing indexes during copy should be handled by COPY

COPY and file_fdw

- file_fdw directly reuses COPY internals for it working
- file_fdw will gain ability to work with GCS and other “cloud” storages
- And we should make sure that all other pg_file_***() functions also use COPY

COPY pg_dump

- Can pg_dump data directly to GCS
- Can use parallel dump, “directory” format
- dump large tables in chunks, to GCS, in parallel

How to get there

- Refactor COPY in gram.y to accept flexible flags
- Introduce virtual function arrays
- Refactor current formats and transports to use the above
- Add new ones,
  - at least GCS,
  - hopefully also postgres_fdw
- Parallel copy - if target is GCS, or FDW allow normal seqscan parallelization to use these as output sink
- Refactor “file functions” (pg_ls_dir(), pg_read_file(), pg_read_binary_file(), …) to use COPY

STEPS

- Refactor current formats and transports

```
foreach(cur, cstate->attnumlist)
{
  int     attnum = lfirst_int(cur);
  Datum   value = slot->tts_values[attnum - 1];
  bool    isnull = slot->tts_isnull[attnum - 1];

  if (!cstate->opts.binary)
  {...
  }

  if (isnull)
  {
    if (!cstate->opts.binary)
      CopySendString(cstate, cstate->opts.null_print_client);
    else
      CopySendInt32(cstate, -1);
  }
  else
  {
    if (!cstate->opts.binary)
    {
      string = OutputFunctionCall(&out_functions[attnum - 1],
                                  value);
      if (cstate->opts.csv_mode)
        CopyAttributeOutCSV(cstate, string,
                            cstate->opts.force_quote_flags[attnum - 1],
                            list_length(cstate->attnumlist) == 1);
      else
        CopyAttributeOutText(cstate, string);
    }
    else
    {
      bytea    *outputbytes;
```

Pluggable COPY Extension

Usually features are not back-ported to older mayor versions
… so we should provide a “CloudSQL COPY” extension

- Starts from copy of COPY code
- Rewritten for Pluggability
- Hooks UtilityCommand
- Has a set of cloudsqlcopy.<flag> flags to override COPY behaviour
- So when customer installs it DMS can
  - Copy data in using all the Pluggable COPY goodness
  - Parallel
  - Compressed
  - Chunked
  - SPLIT main and TOAST
  - With automatic index and FK managenet

Pluggable COPY Extension v0.2

Usually features are not back-ported to older mayor versions
… so we should provide a “CloudSQL COPY” extension

- Starts from copy of COPY code
- Rewritten for Pluggability
- **Enchanced UtilityCommand Hook, which also catches syntax error and can completely override or add new Utility Commands**
- Has a set of cloudsqlcopy.<flag> flags to override COPY behaviour **(still useful for non-modified apps, like pglogical)**
- So when customer installs it DMS can
  - Copy data in using all the Pluggable COPY goodness
  - Parallel, Chunked
  - Compressed
  - SPLIT main and TOAST
  - With automatic index and FK management **and non-cascading TRUNCATE option**

Cloud friendly COPY

Copy from/to anything.

PostgreSQL Conference Europe 2022,
Berlin 2022.10.25

hannuk@google.com