PGCONF.Online SPRINT:

Appendable bytea TOAST

Oleg Bartunov
Nikita Glukhov
Motivational example

• A table with 100 MB bytea (uncompressed):
  
  CREATE TABLE test (data bytea);
  ALTER TABLE test ALTER COLUMN data SET STORAGE EXTERNAL;
  INSERT INTO test SELECT repeat('a', 100000000)::bytea data;

• Append 1 byte to bytea:
  
  EXPLAIN (ANALYZE, BUFFERS, COSTS OFF)
  UPDATE test SET data = data || 'x':bytea;

  Update on test (actual time=1359.229..1359.232 rows=0 loops=1)
    Buffers: shared hit=238260 read=12663 dirtied=25189 written=33840
    -> Seq Scan on test (actual time=155.499..166.509 rows=1 loops=1)
      Buffers: shared hit=12665
  Planning Time: 0.127 ms
  Execution Time: 1382.959 ms

  >1 second to append 1 byte !!!
  Table size doubled to 200 MB, 100 MB of WAL generated.

• Thanks to Alexander ? who raised the problem of (non-effective) streaming into bytea at PGConf.Online !
Motivational example (explanation)

- Current TOAST is not sufficient for partial updates
- All data is deTOASTed before in-memory modification
- Updated data is TOASTed back after modification with new TOAST oid
Solution

- Special datum format: TOAST pointer + inline data
- Inline data serves as a buffer for TOASTing
- Operator || does not deTOAST data, it appends inline data producing datum in the new format
Solution

• When size of inline data exceeds 2 KB, TOASTer recognizes changes in old and new datums and TOASTs only the new inline data with the same TOAST oid

• Last not filled chunk can be rewritten with creation of new tuple version

• First unmodified chunks are shared
Results – motivational example

• Append 1 byte to bytea:

```
EXPLAIN (ANALYZE, BUFFERS, COSTS OFF)
UPDATE test SET data = data || 'x': bytea;
```

Update on test (actual time=0.060..0.061 rows=0 loops=1)
 Buffers: shared hit=2
  -> Seq Scan on test (actual time=0.017..0.020 rows=1 loops=1)
    Buffers: shared hit=1
Planning Time: 0.727 ms
Execution Time: 0.496 ms (was 1382 ms)

2750x speed up!

• Table size remains 100 MB
• Only 143 bytes of WAL generated (was 100 MB)
• No unnecessary buffer reads and writes
Results – query execution time

UPDATE test SET data = data || repeat('a', append_size)::bytea;

- $\infty$ OLD + NEW
- $\infty$ APPEND SIZE
Results – WAL traffic

UPDATE test SET data = data || repeat('a', append_size):bytea;

- `OLD + NEW`
- `INLINED OLD + NEW`
Quick Summary

• TODO
  • TOAST uses special snapshot and it is not ready for multiple versions of the same chunk. This needs to be fixed.
  • Support for strings, arrays, jsonb arrays (see our main “Jsonb internals” talk).
  • Prepend, truncate, insert, delete support.
  • Compression (now it is applicable only to uncompressed EXTERNAL attributes)
  • Generalized datum format for partial updates of plain data types.
  • Pluggable datum formats with custom TOASTers.

• We demonstrate significant (1000X) performance improvement
  • Branch in our repository (currently based on jsonb_shared_toast branch):
    https://github.com/postgrespro/postgres/tree/bytea_appendable_toast
  • Slides of this talk (PDF)
  • It’s not PG14 ready

• Contact obartunov@postgrespro.ru, n.gluhov@postgrespro.ru for collaboration.
Нам нужны Ваши кейсы (тестовые данные и запросы)!