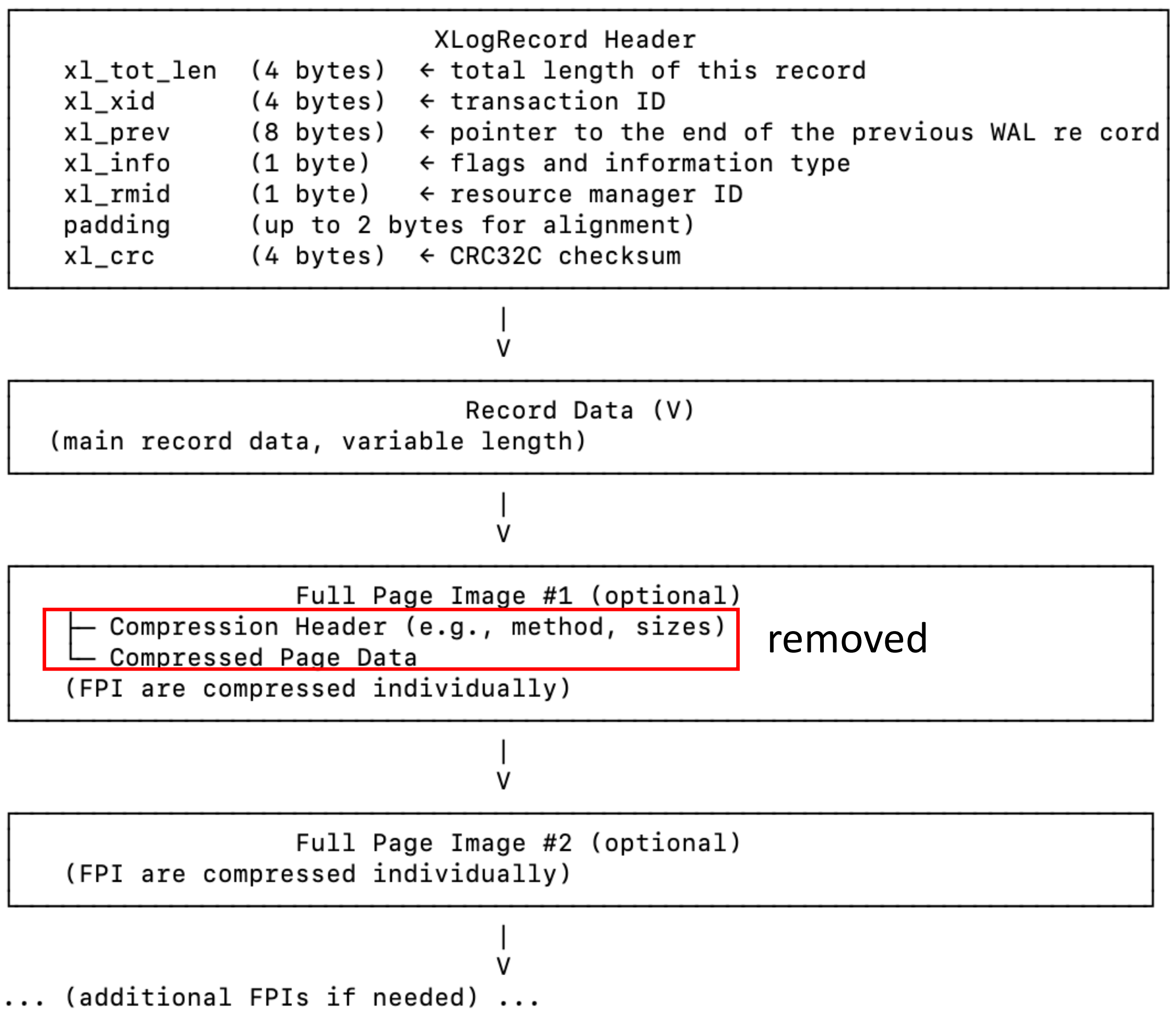
Compression of big WAL records

(path to wholesale WAL compression)

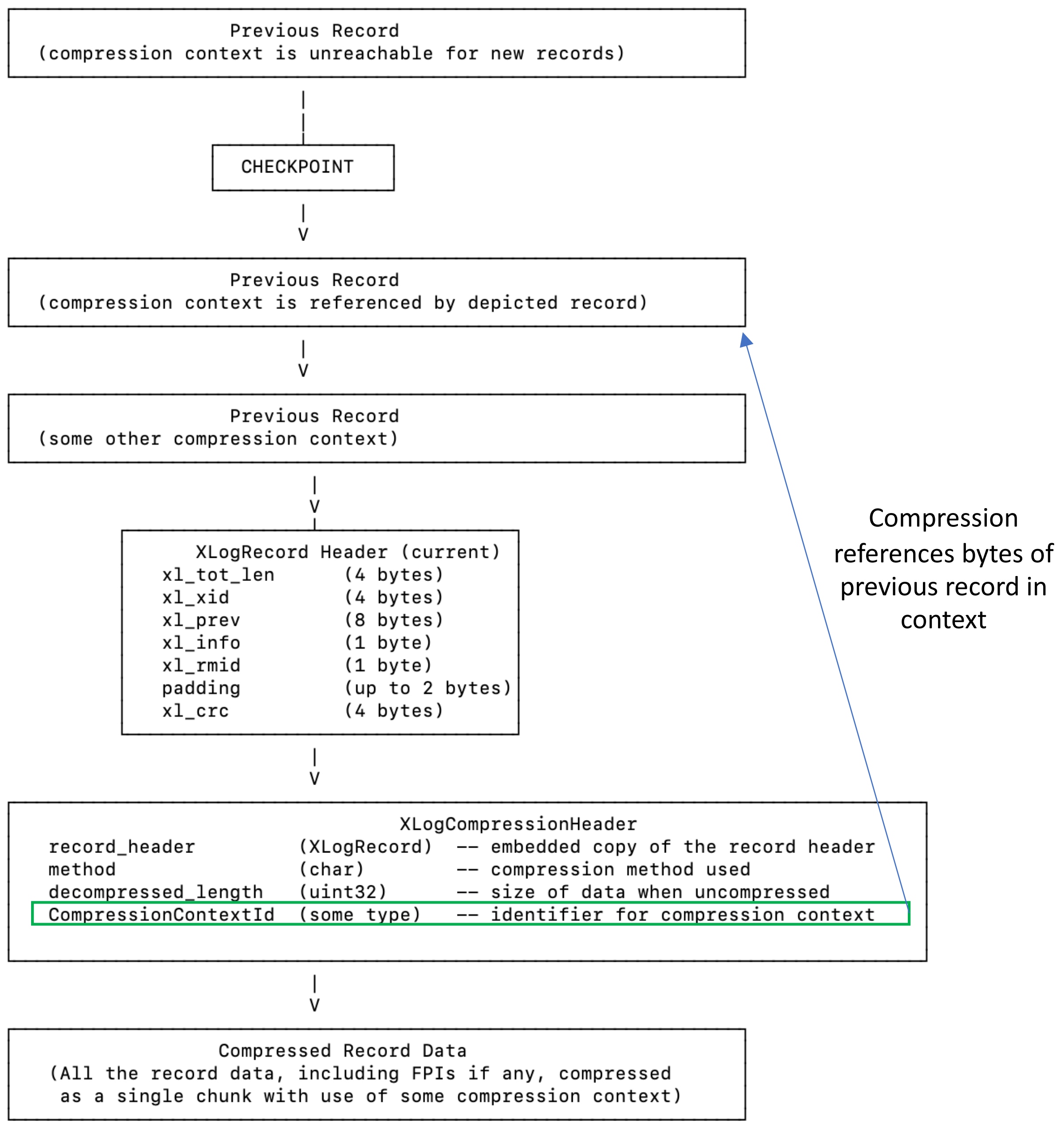
Andrey Borodin
amborodin@acm.org
Telegram: @x4mmm

Below is a diagram of the **current structure** of a WAL record (XLogRecord). Each Full Page Image (FPI) is compressed individually.



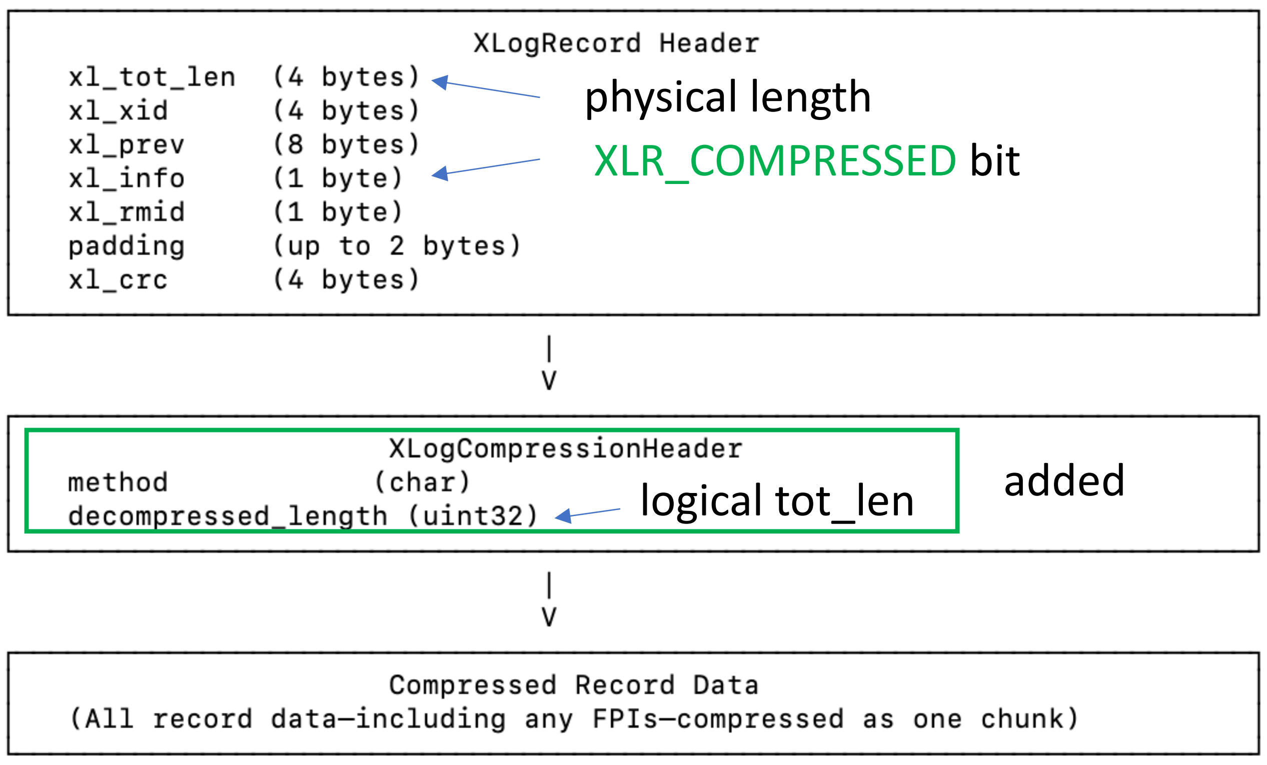
```
                         XLogRecord Header
   xl_tot_len  (4 bytes)  ← total length of this record
   xl_xid      (4 bytes)  ← transaction ID
   xl_prev     (8 bytes)  ← pointer to the end of the previous WAL re cord
   xl_info     (1 byte)   ← flags and information type
   xl_rmid     (1 byte)   ← resource manager ID
   padding     (up to 2 bytes for alignment)
   xl_crc      (4 bytes)  ← CRC32C checksum

                              |
                              V

                          Record Data (V)
  (main record data, variable length)

                              |
                              V

                    Full Page Image #1 (optional)
   ├─ Compression Header (e.g., method, sizes)
   └─ Compressed Page Data
   (FPI are compressed individually)

                              |
                              V

                    Full Page Image #2 (optional)
   (FPI are compressed individually)

                              |
                              V
... (additional FPIs if needed) ...
```

Future version can reduce **wal_compression_threshold** by retaining compression context between records. But this limits ability to to read WAL at random positions. Contexts cannot cross CHECKPOINT boundaries.



```
                 Previous Record
 (compression context is unreachable for new records)

                        |
                        |
                  CHECKPOINT
                        |
                        V

                 Previous Record
 (compression context is referenced by depicted record)

                        |
                        V

                 Previous Record
 (some other compression context)

                        |
                        V
        XLogRecord Header (current)
    xl_tot_len      (4 bytes)
    xl_xid          (4 bytes)
    xl_prev         (8 bytes)
    xl_info         (1 byte)
    xl_rmid         (1 byte)
    padding         (up to 2 bytes)
    xl_crc          (4 bytes)

                        |
                        V

                               XLogCompressionHeader
  record_header         (XLogRecord)  -- embedded copy of the record header
  method                (char)        -- compression method used
  decompressed_length   (uint32)      -- size of data when uncompressed
  CompressionContextId  (some type)   -- identifier for compression context

                        |
                        V

                Compressed Record Data
  (All the record data, including FPIs if any, compressed
   as a single chunk with use of some compression context)
```

Patch idea: compress body of big records, if it's bigger than some threshold (e.g. 860 bytes).



```
                    XLogRecord Header
   xl_tot_len   (4 bytes)
   xl_xid       (4 bytes)
   xl_prev      (8 bytes)
   xl_info      (1 byte)
   xl_rmid      (1 byte)
   padding      (up to 2 bytes)
   xl_crc       (4 bytes)

                         |
                         V

                  XLogCompressionHeader
   method                (char)
   decompressed_length (uint32)

                         |
                         V

                  Compressed Record Data
   (All record data—including any FPIs—compressed as one chunk)
```

Proposed patch converts individual FPI compression headers to one compression header per record. Minimal threshold for compression is controlled via GUC **wal_compression_threshold.**

Benchmarks of current patch version indicate substantial WAL reduction in cases, when single WAL records contains many FPIs. For example, B-tree index creation:

```
create table a as
    select random() from generate_series(1,1e7);
create index on a(random );
```

WAL footprint of creating index:

method	HEAD	patched
pglz	193 MB	193 MB
lz4	160 MB	132 MB
zstd	125 MB	97 MB

Open questions:

1. Memory allocations for compression\decompression buffers, that are hard to predict. Some are known only in critical sections.
2. Logical replication and WAL dump want to start at random LSN, not from CHECKPOINT.
3. Codecs do not play well with compression continuation: need a copy of previously compressed data.
4. Unknown unknowns.

Collaboration wanted:

1. Overall design review needed.
2. The design of compression context needs refinement.
3. Benchmarks and testing.

Commitfest item:

PGConf.dev 2025