FUJITSU

shaping tomorrow with you

Data compression in PostgreSQL and its future

October 16, 2019

Aya Iwata
Fujitsu Limited

Who am I?

FUJITSU

AYA IWATA

- Develop
  FUJITSU Software Enterprise Postgres
  (PostgreSQL-based product)
- Support open source PostgreSQL in various products
- Steering Committee Member of PGConf.ASIA 2018

Outline

FUJITSU

- Why is data compress needed?
- Compression feature in current PostgreSQL
- Features under development
- What are other missing pieces?
- My idea: Row table compression

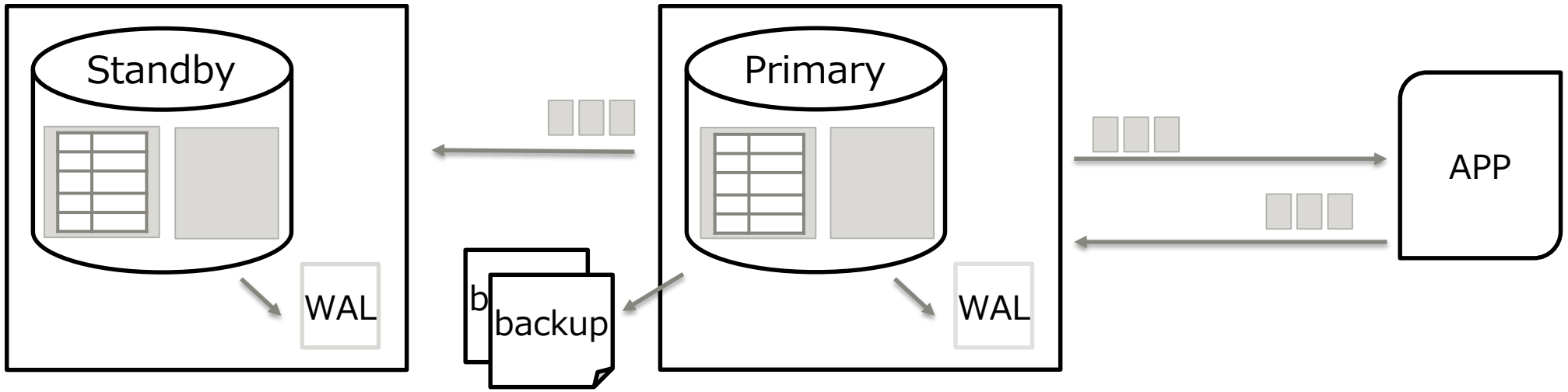
Why is data compression needed?

FUJITSU

- Reduce costs for growing data volumes
- Example :downsize AWS resources
  - Workload: needs 48 vCPU, 200 GB of DB cache
  - EC2: m5.24xlarge (96 vCPU, 384 GB RAM) → m5.12xlarge (48 vCPU, 192 GB RAM)
  - EBS: General SSD (gp2) 1TB →500GB
  - 3-years total cost: $124,785 → $62,349 (-50 %)
- Improve performance by reducing I/O and memory scan

What to compress?

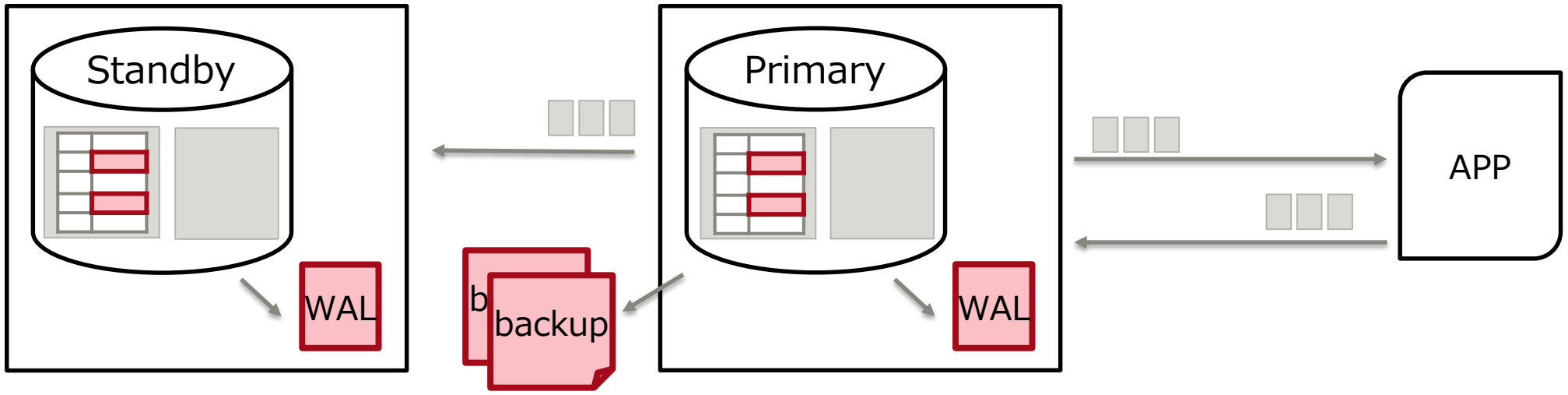
- Table and Index data
- WAL (Write Ahead Log)
- Backup (pg_basebackup)
- Export file (pg_dump, COPY)
- Network traffic (for replication across WAN)



Compression features in current PostgreSQL

FUJITSU

- TOAST
- WAL compression
- Export file compression
  - pg_dump –compress
  - COPY mytable FROM/TO PROGRAM ‘gzip …’



TOAST

FUJITSU

- Compress large data due to page size limitations
- TOAST compresses only variable-length data
  - varchar, text, jsonb, hstore, tsvector etc.
- Target data size : bigger than page size / 4 (default 2KB)
- Compression ratio of typical HTML documents is 2x

WAL Compression

FUJITSU

- Since v9.5
- Works by setting wal_compression = on in postgresql.conf
- Compress WAL page using PGLZ compression method
- Compress full page image in WAL
- Compression ratio is 3.5x with update-heavy pgbench (*)
- Less effective when checkpoint is infrequent
- Use extra CPU during WAL logging and replay

(*) Kaarel Moppel, https://www.cybertec-postgresql.com/en/postgresql-underused-features-wal-compression/

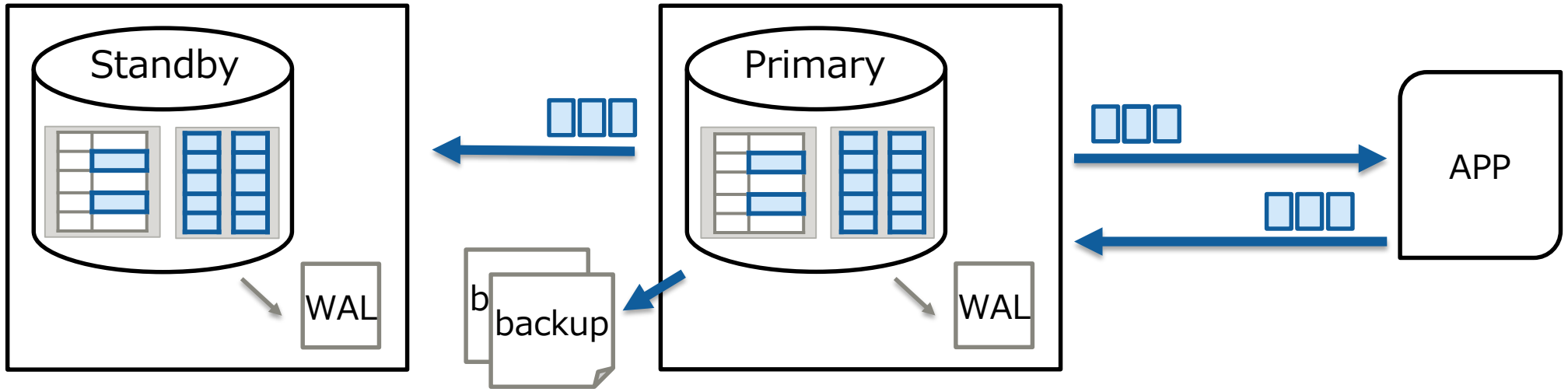
Export file compression

FUJITSU

- pg_dump –compress
  - “custom” format and “directory” format are compressed by default
  - --compress=0..9 parameter specify compression level
  - Uses zlib library (the algorithm is DEFLATE)
- COPY mytable FROM/TO PROGRAM ‘gzip …’
  - Any program can be used to compress/decompress data

Features under development

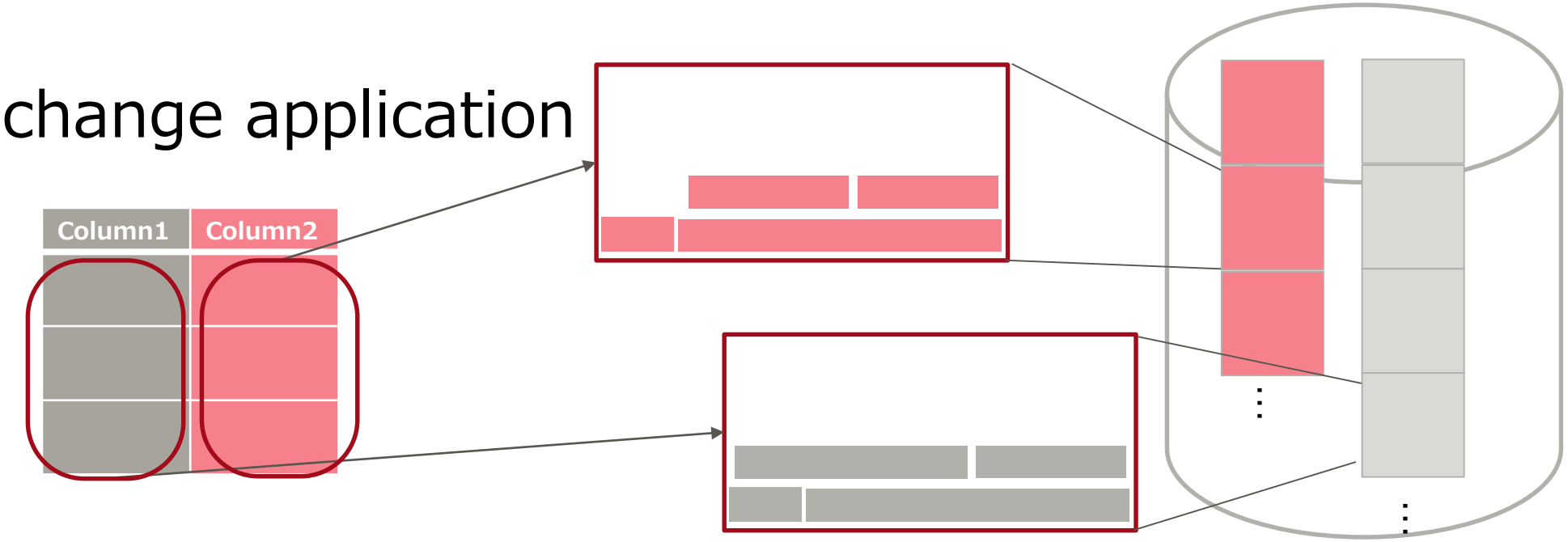
- Columnar storage with compression (Zedstore)
- Custom compression for TOAST
- Enhancements to TOAST
- Network traffic compression in libpq



Columnar compression in Zedstore

FUJITSU

- Columnar storage that implements table AM interface
  - Table AM interface is new in v12
- Store each column of all rows in a consecutive area
- Compression efficiency increases because similar data is collected
- No need to change application



Zedstore: Compression ratio

FUJITSU

- pgbench -i -s 1000

	PG v10	PG v12 + Zedstore	ratio
pgbench_accounts	1,281 MB	264 MB	-80%
pgbench_branches	8,192 bytes	56 kB	+600%
pgbench_tellers	72 kB	64 kB	-11%

- SELECT AVG() on 1 million rows: 4,679.0 ms → 379.7 ms
- Loading pgbench_accounts data: 50.3 s → 26.2 s

https://www.postgresql.org/message-id/101f8490-a7bc-230e-cb38-730e26ca81bd%40catalyst.net.nz

TOAST custom compression

FUJITSU

- Choose a compression algorithm for each TOASTable column by setting a compression access method
- Supports pglz and zlib by default
- Syntax:
  - CREATE ACCESS METHOD compression_am TYPE COMPRESSION handler_function;
  - ALTER TABLE tbl_name ALTER [COLUMN] column_name SET COMPRESSION compression_am
- No need to change application

TOAST custom compression: benchmark result

FUJITSU

- Load the community mailing list archives
- Compression ratio is higher with lz4 than with the default pglz

algorithm	Size(MB)
pglz(default)	1,637
Suitable algorithm for tsvector	1,521
lz4	1,487

Partial TOAST decompression (1/2)

FUJITSU

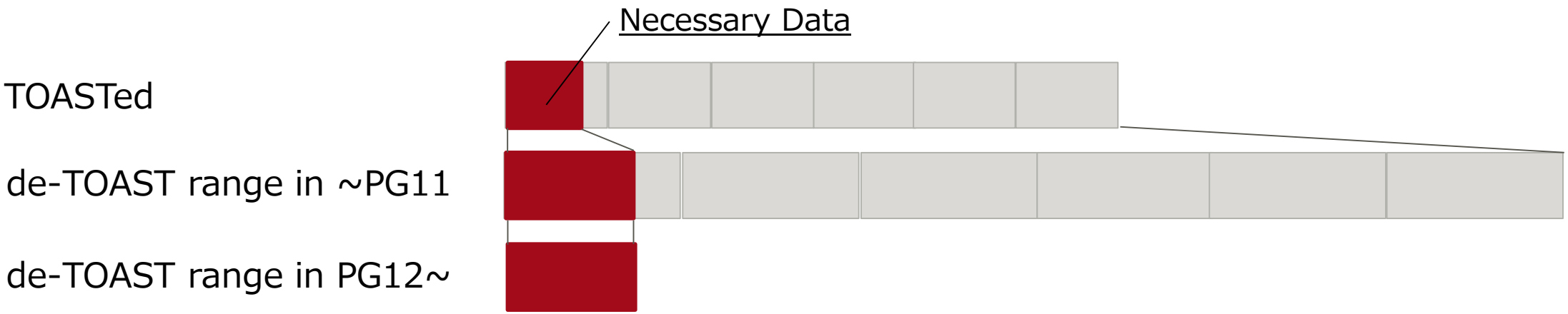
- Faster extraction of the first N bytes of TOASTed data
- This is useful when you want to get only the header of HTML data
- Ex. When needing access to only first 20 bytes of 59kB text columns in 10,000 rows
  - SELECT sum(length(substr(a, 0, 20))) on 10,000 TOASTed data: 1,427 ms → 47 ms (30x speedup)
- Available since PostgreSQL 12

- Further optimization of partial TOAST decompression is under way
- Ex. When needing access to only first 20 bytes of 38 MB text columns in 100 rows
  - SELECT sum(length(substr(a, 0, 20))) on 100 TOASTed data: 28.1 ms → 2.3 ms (10x speedup)

Partial TOAST decompression (2/2)

FUJITSU

- Faster extraction of TOASTed data whose length is unknown
- It is effective when we want to extract data from the beginning to </ header> of HTML data
- The performance of SELECT position() is 3x better
  - 100 TOASTed data is compressed, pattern is at beginning: 4,4 ms →1,5 ms



libpq network compression

FUJITSU

- Especially effective for:
  - Access to large result sets
  - Data import & export with COPY and pg_dump
  - Streaming & logical replication: good fit for narrow WAN bandwidth
- Works by setting connection parameter: "compress =1”
- Compress with zstd or zlib
- When compressing with zstd, configure the build with “—with-zstd ”
- zlib compression can be used with older versions of server
- No need to change application

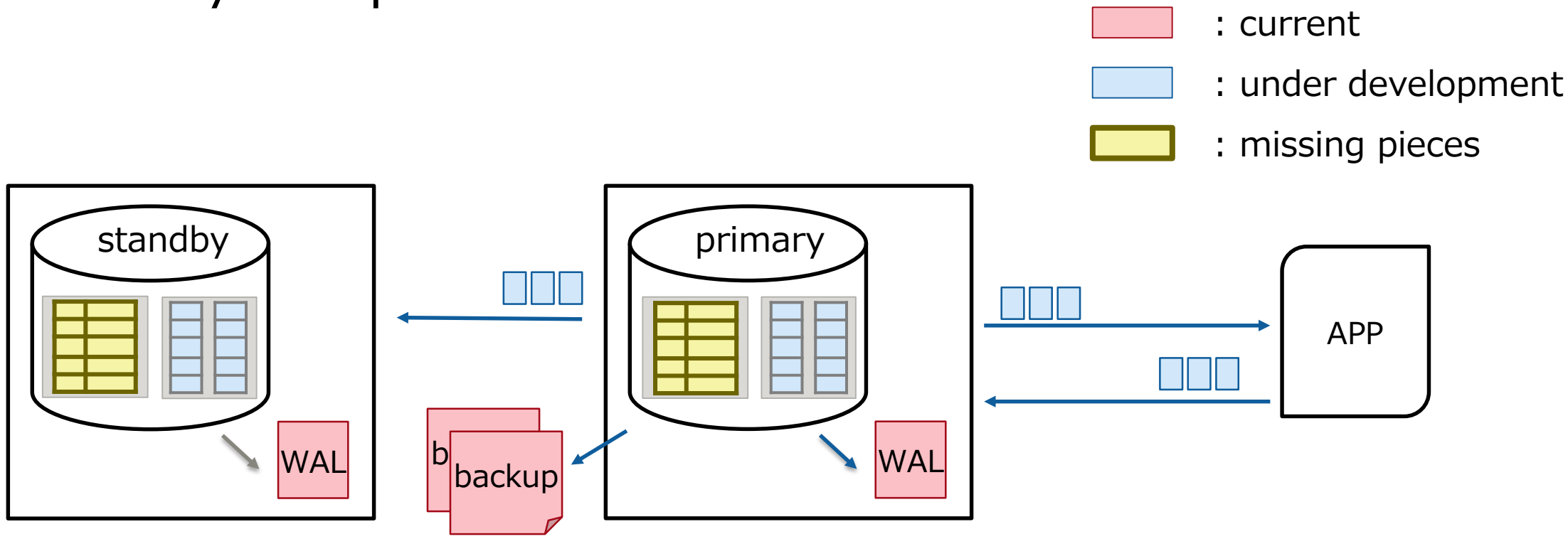
libpq network compression: compression ratio

FUJITSU

- Data size becomes 1.5%
  - pgbench -i -s 10
  - Data size : 16.2 MB → 0.3 MB
- Small impact on performance
  - pgbench -i -s 10
    no compression 1.55 s
    libz (level=1) 1.57 s
    libzstd (level=1) 1.61 s
  - pgbench -t 100000 -S
    no compression 4.48 s
    libz (level=1) 4.92 s
    libzstd (level=1) 4.87 s

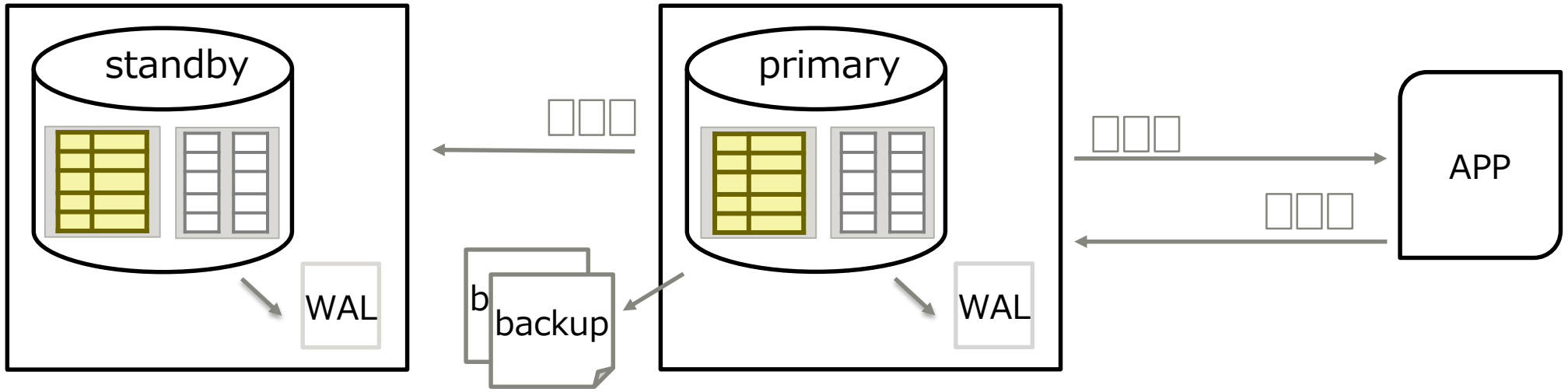
What are other missing pieces?

- Row-oriented table compression for un-TOASTed data
- Index key compression



What are other missing pieces?

- **Row-oriented table compression for un-TOASTed data**
- Index key compression

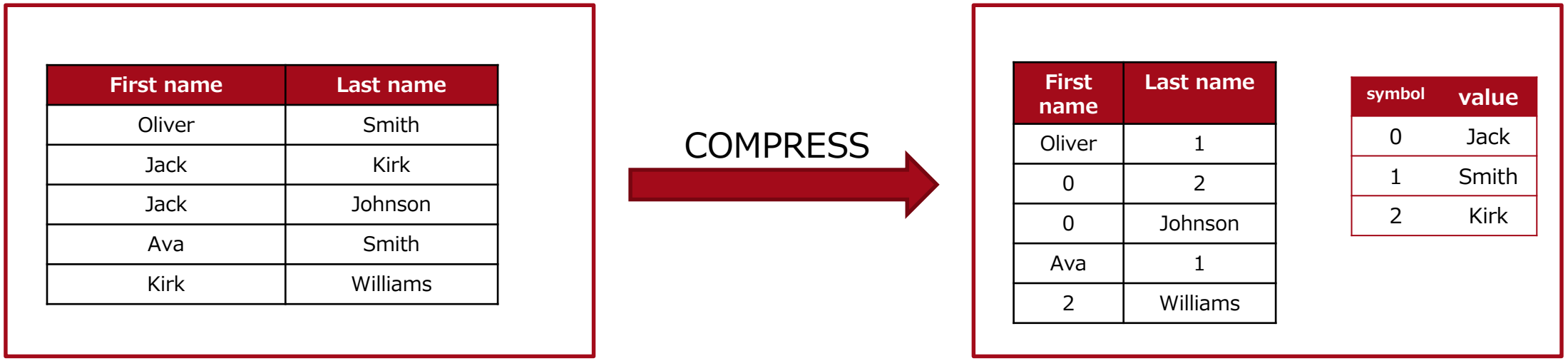


Row table compression: Overview

FUJITSU

Key concept: deduplication in page

- Replace duplicate values across columns with a symbol
- A symbol is a pointer to an entry in the symbol table
- Symbol table and compressed data are stored in the same page
- No extra I/O for symbol lookup



Row table Compression : Usage

FUJITSU

- Compress a new table

CREATE TABLE …COMPRESS BY DEDUPLICATION;

- Compress an existing table

ALTER TABLE ….COMPRESS BY DEDUPLICATION;

  - Only new full pages are compressed, existing ones are not
  - CLUSTER and VACUUM FULL compress all page (that have duplicate values)

- pg_get_compression_ratio()
  - The function to check the effect of compression
- Set the default compression for tables in a tablespace

CREATE TABLESPACE … COMPRESS BY DEDUPLICATION;

  - Eliminates the need to specify compression for each table
  - No need to modify application's SQL scripts

Row table compression: Compression processing

FUJITSU

What data to compress?

- Decide whether to compress by appearance count and length
  - (m-1)(n-2) > 6
  - m: duplicate times, n: column length

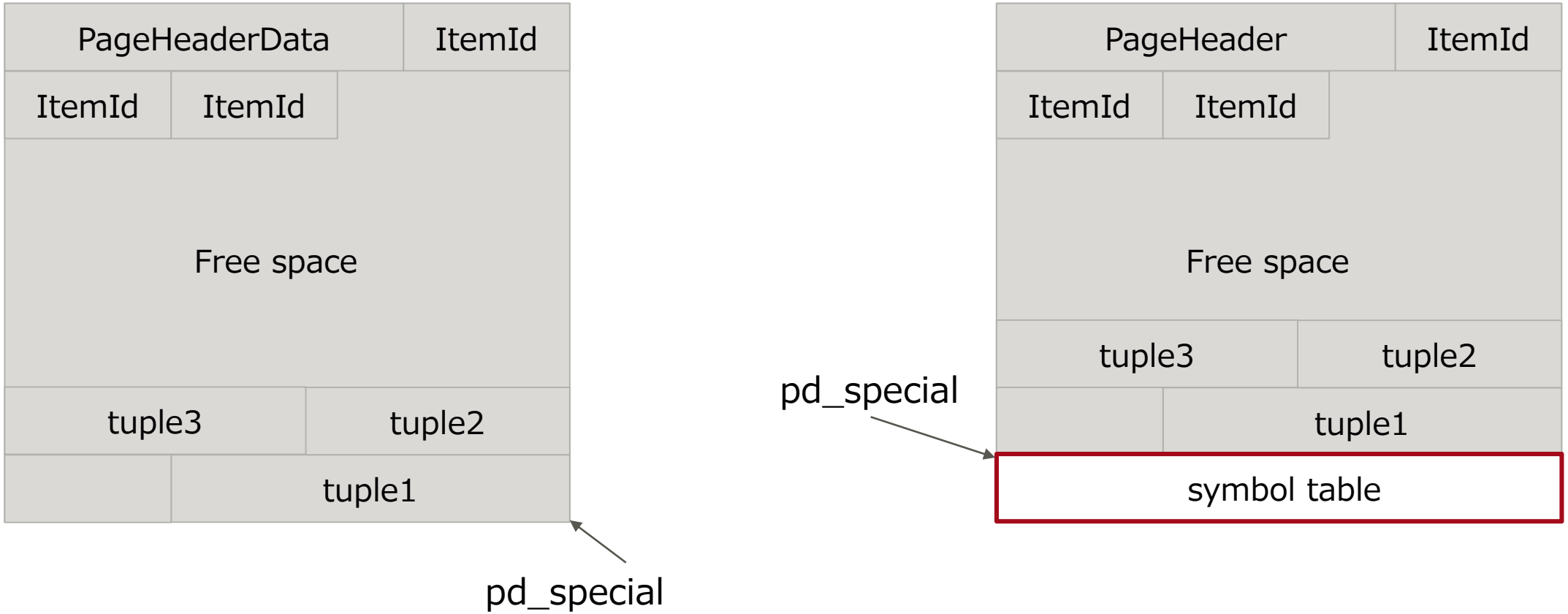
When and how to compress?

- Page gets full
- Scan the whole page to find duplicate values
- Create the symbol table, and replace column values with symbols

Page data structure

FUJITSU

- Store the symbol table in special space
  - Page header's pd_special points to the start of symbol table



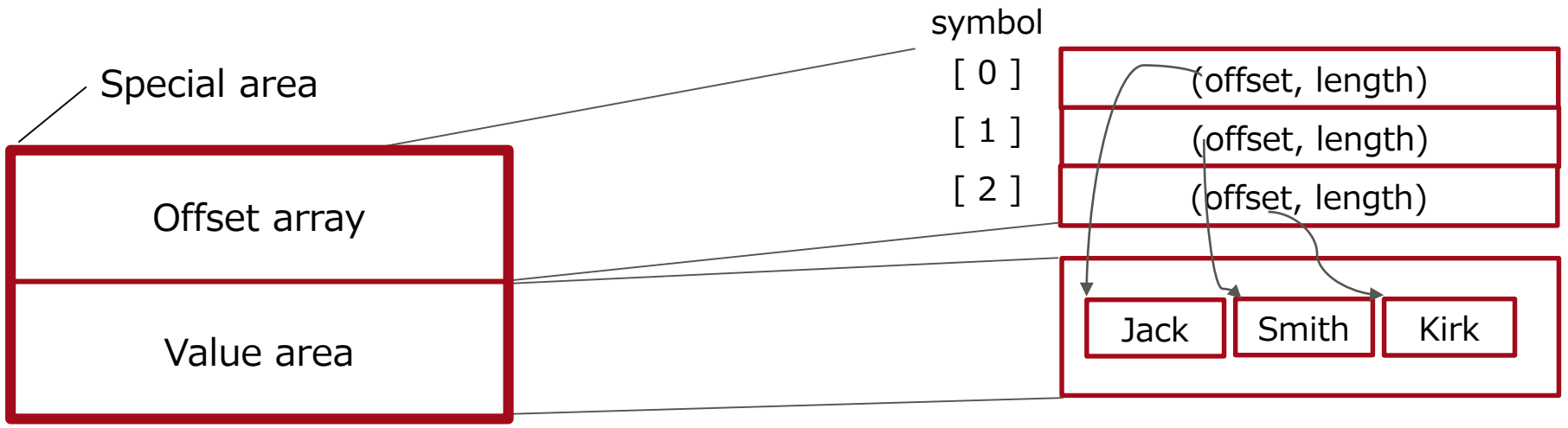
Symbol table data structure

FUJITSU

Key: quick access to original values when decoding

Consists of the following data:

- Array of item identifiers pointing to the original values
- Original values



Tuple data structure

FUJITSU

- Introduce t_compressbits bit map
- HEAP_HASCOMPRESS is set in t_infomask
  - To check whether data is compressed

Compression ratio

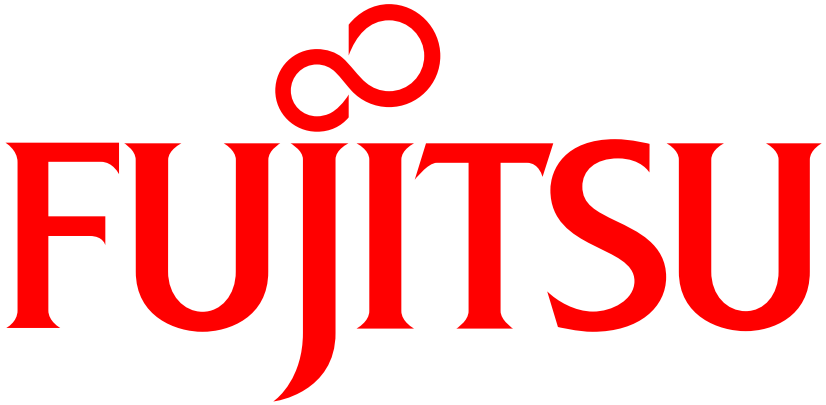
FUJITSU

- Ex. JpetStore
  - web shopping application
- Orders table which is stored temporary data for orders
- Data size becomes about 50%
  - Data size: 21 KB → 12 KB (-44%)
- Less effective for tables that do not have many duplicate values
- Ex. inventory table, item table

We want your ideas!

FUJITSU

- Any idea/wish comment as a user is welcome
- You may contact me
  iwata.aya@fujitsu.com



shaping tomorrow with you