PGConf Europe 2024

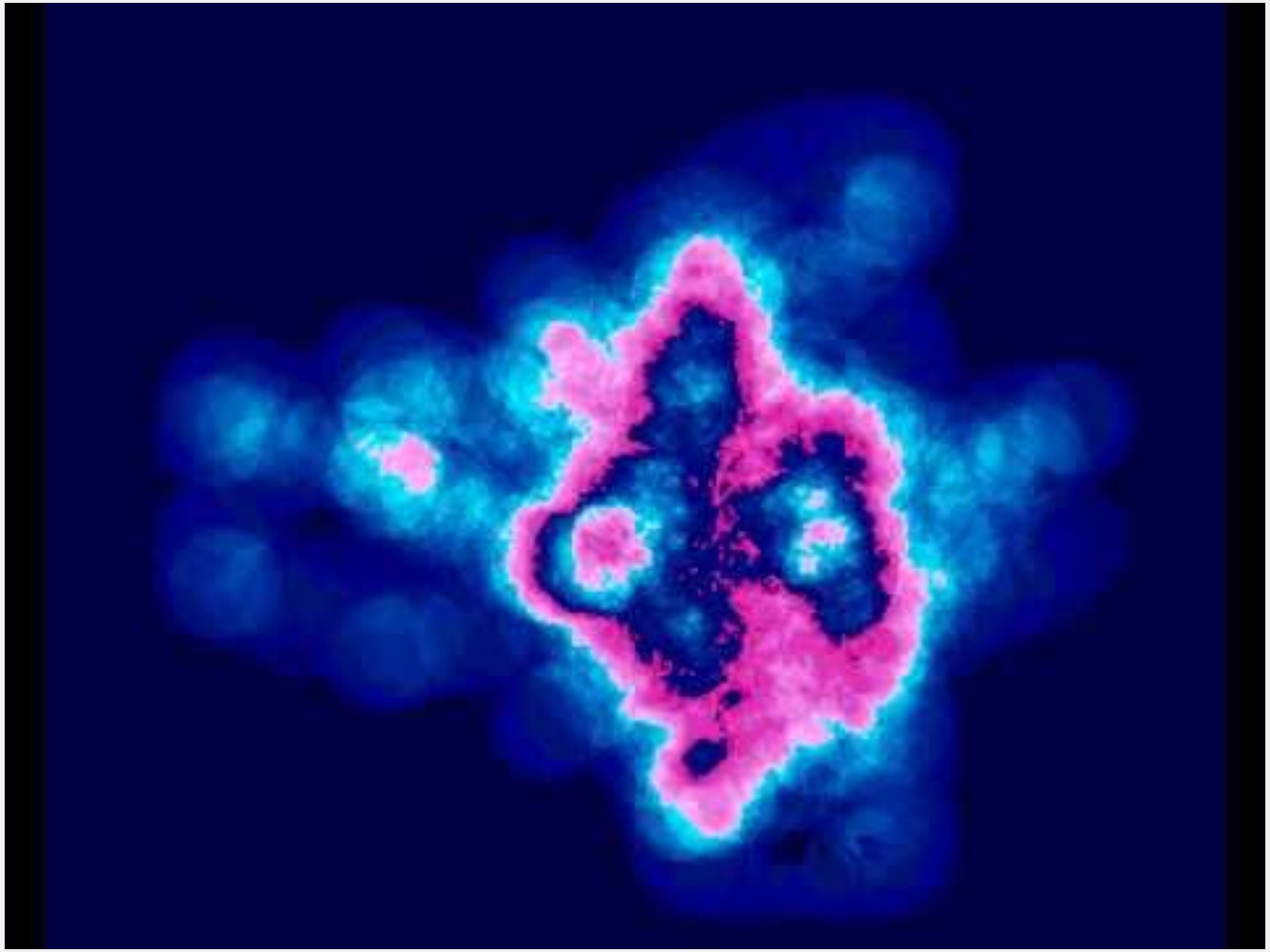
Are you collecting the right metrics?

frederic.delacourt@data-bene.io

DataBene

Frédéric Delacourt

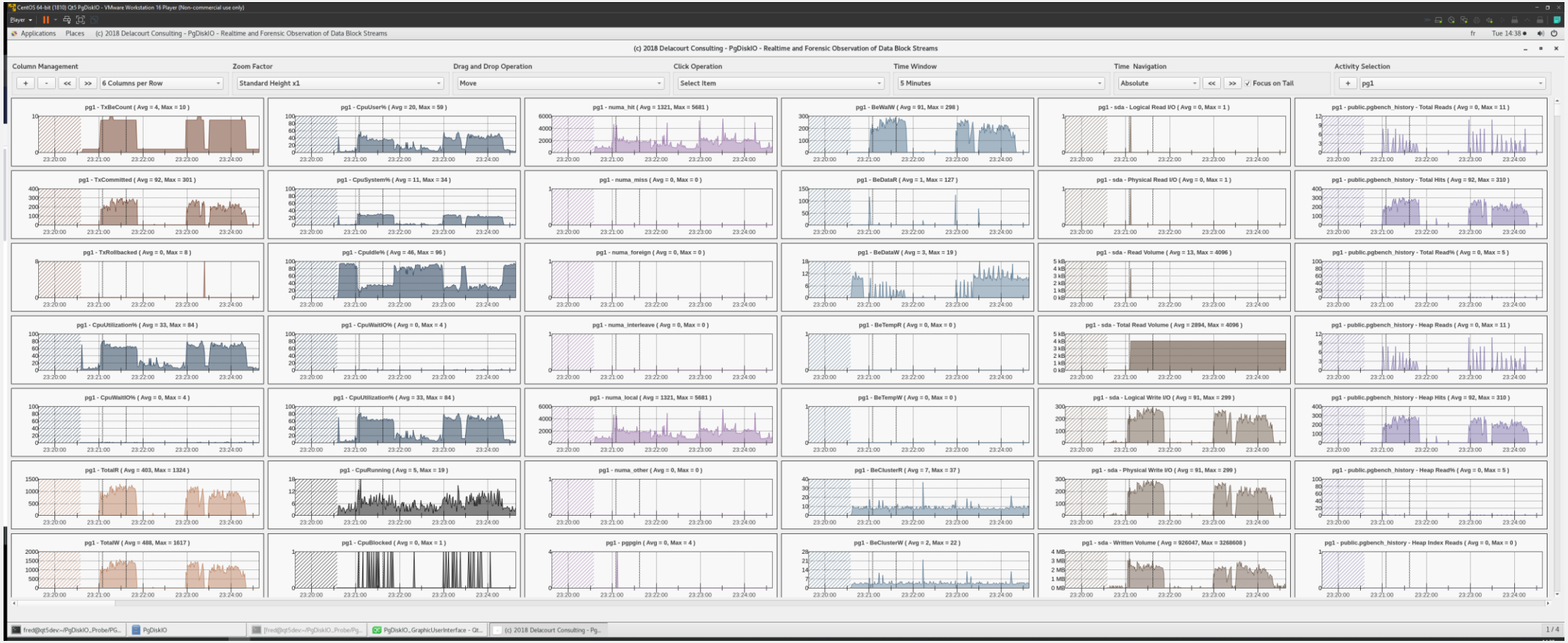
Demomaker
on Amiga 500



DataBene

Frédéric Delacourt

Creator of PgDiskIO



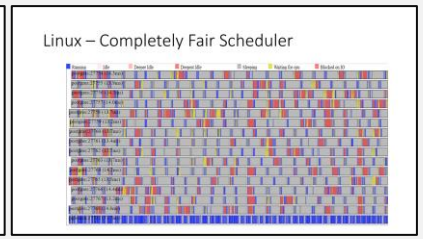
DataBene

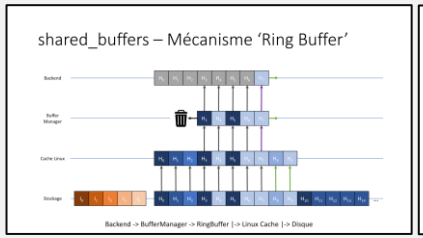
PostgreSQL

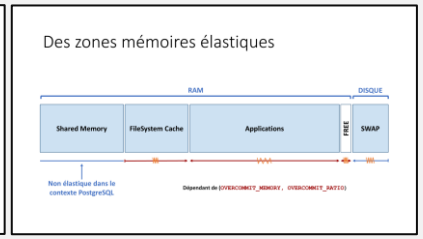
- Sinces 2006 (PostgreSQL 7.2) in the Telecoms
  - Database Architect (+support +dev)

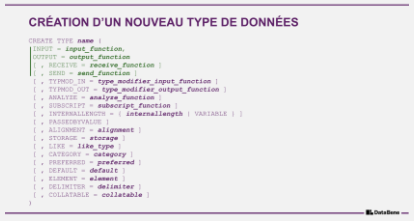
- Since 2017
  - Delacourt Consulting
  - 2ndQuadrant
  - Data Bene
  - Consultant Expert PostgreSQL

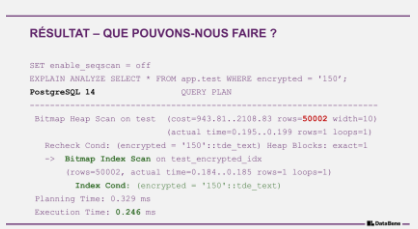
Conferences

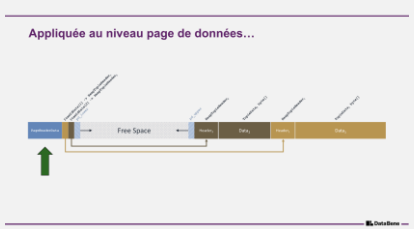


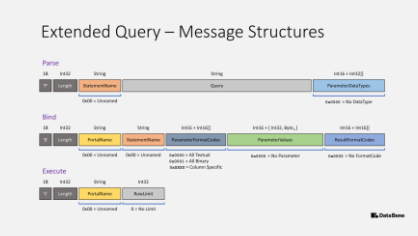


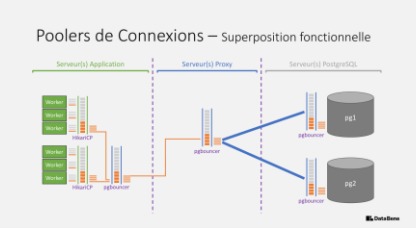


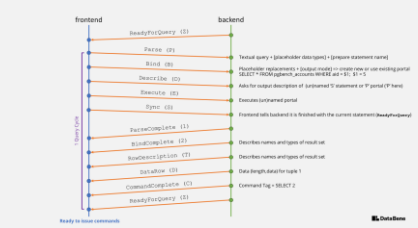


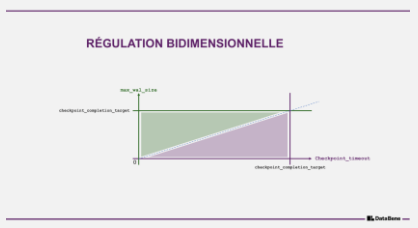


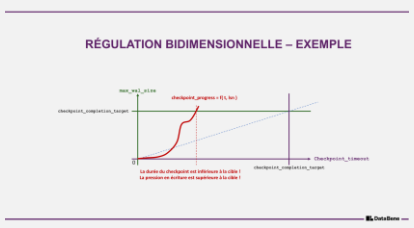












DataBene

What about you?

- Who are you?
- How are you related to PostgreSQL?
- What do you expect from this session?

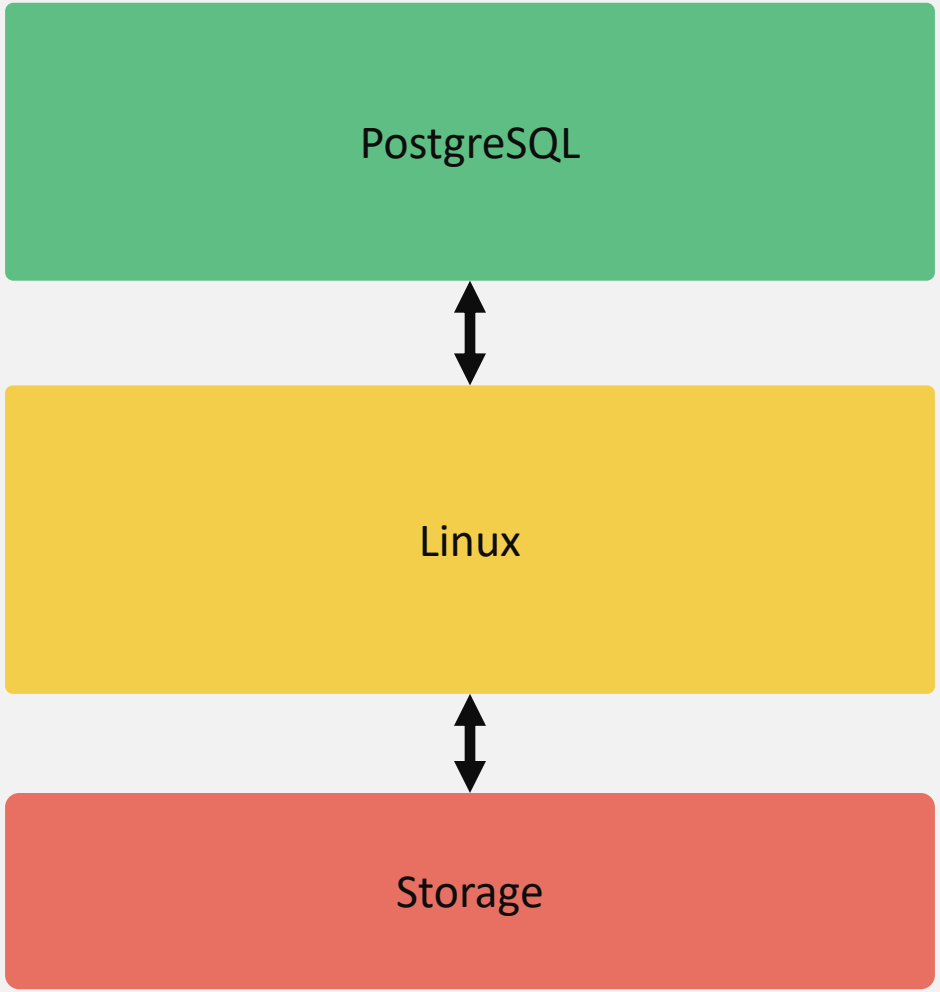
Why collecting metrics?

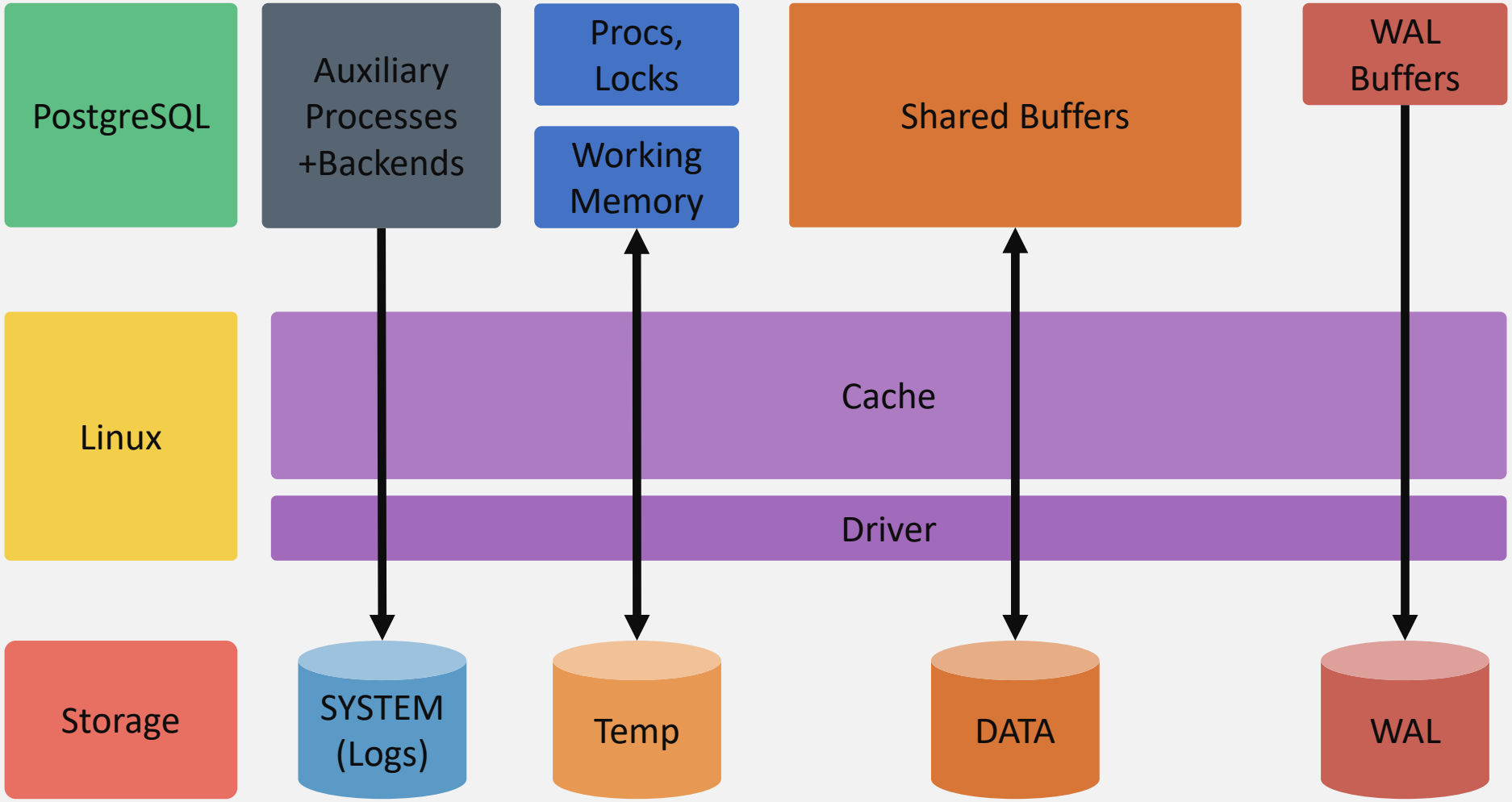
- Workload Analysis
  - Baseline when the system works fine
  - Allows unexpected pattern detections
- Trends
  - Capacity Planning
- Alerting
  - When things are broken
  - When things are breaking or about to break
- Detail Analyzing
  - Troubleshooting
  - Learning behaviours

PostgreSQL Diagrams

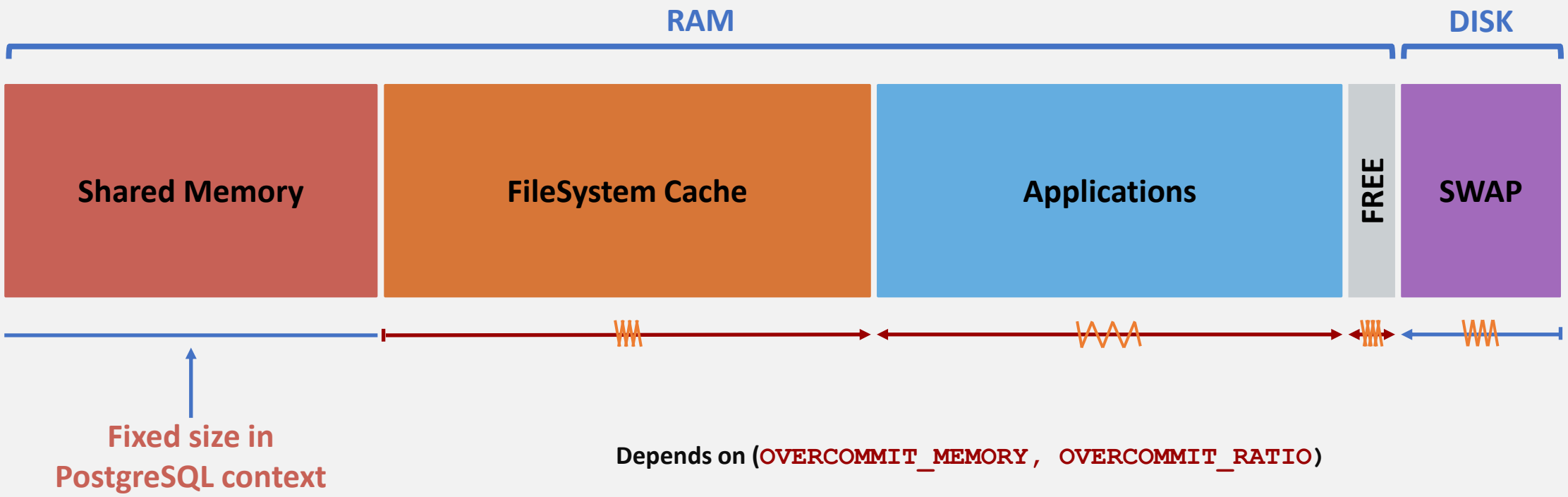
A Graphic Description of PostgreSQL Internals.

DataBene


PostgreSQL
Linux
Storage


PostgreSQL
Auxiliary Processes +Backends
Procs, Locks
Working Memory
Shared Buffers
WAL Buffers
Linux
Cache
Driver
Storage
SYSTEM (Logs)
Temp
DATA
WAL

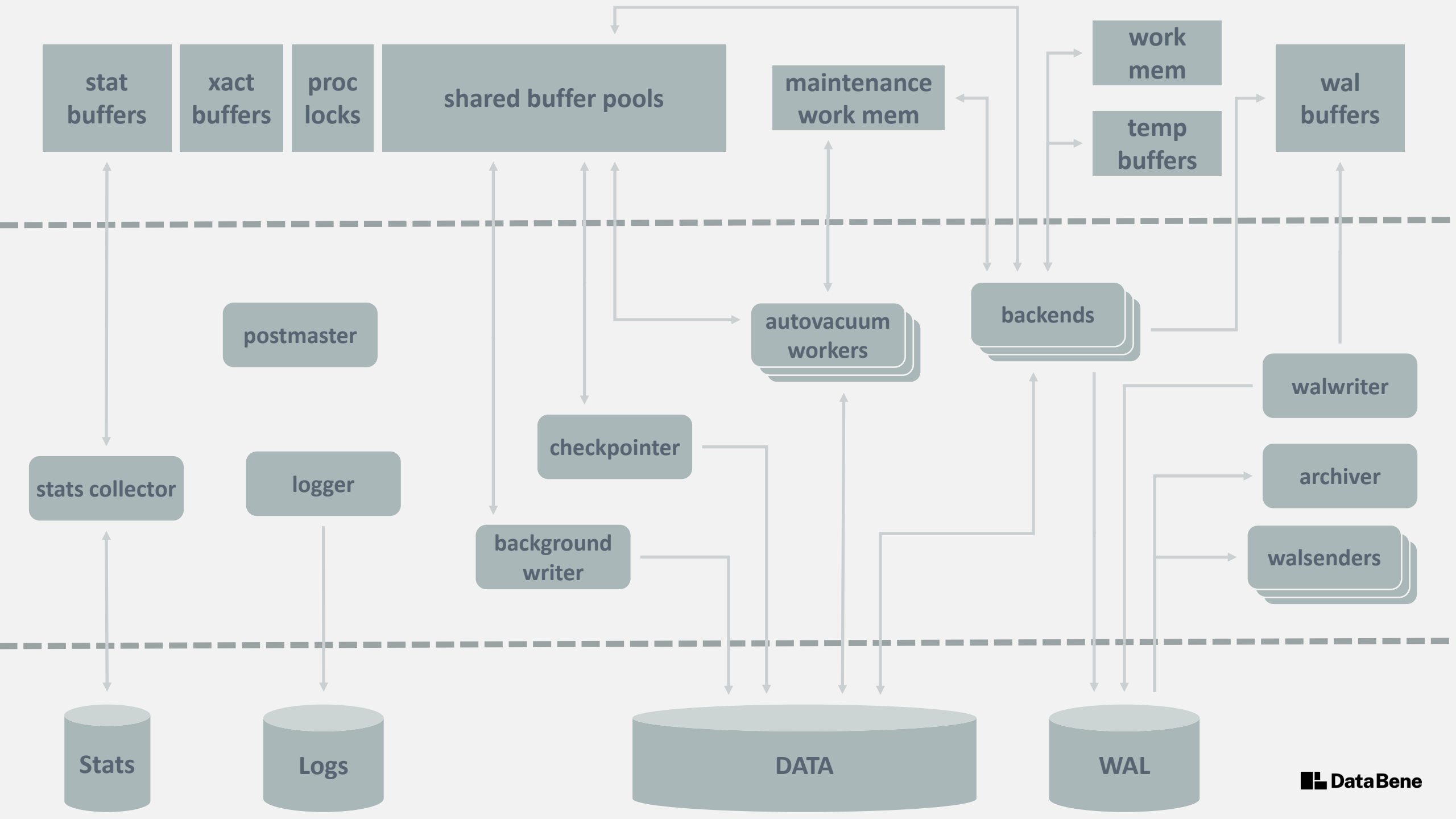
Memory Zones

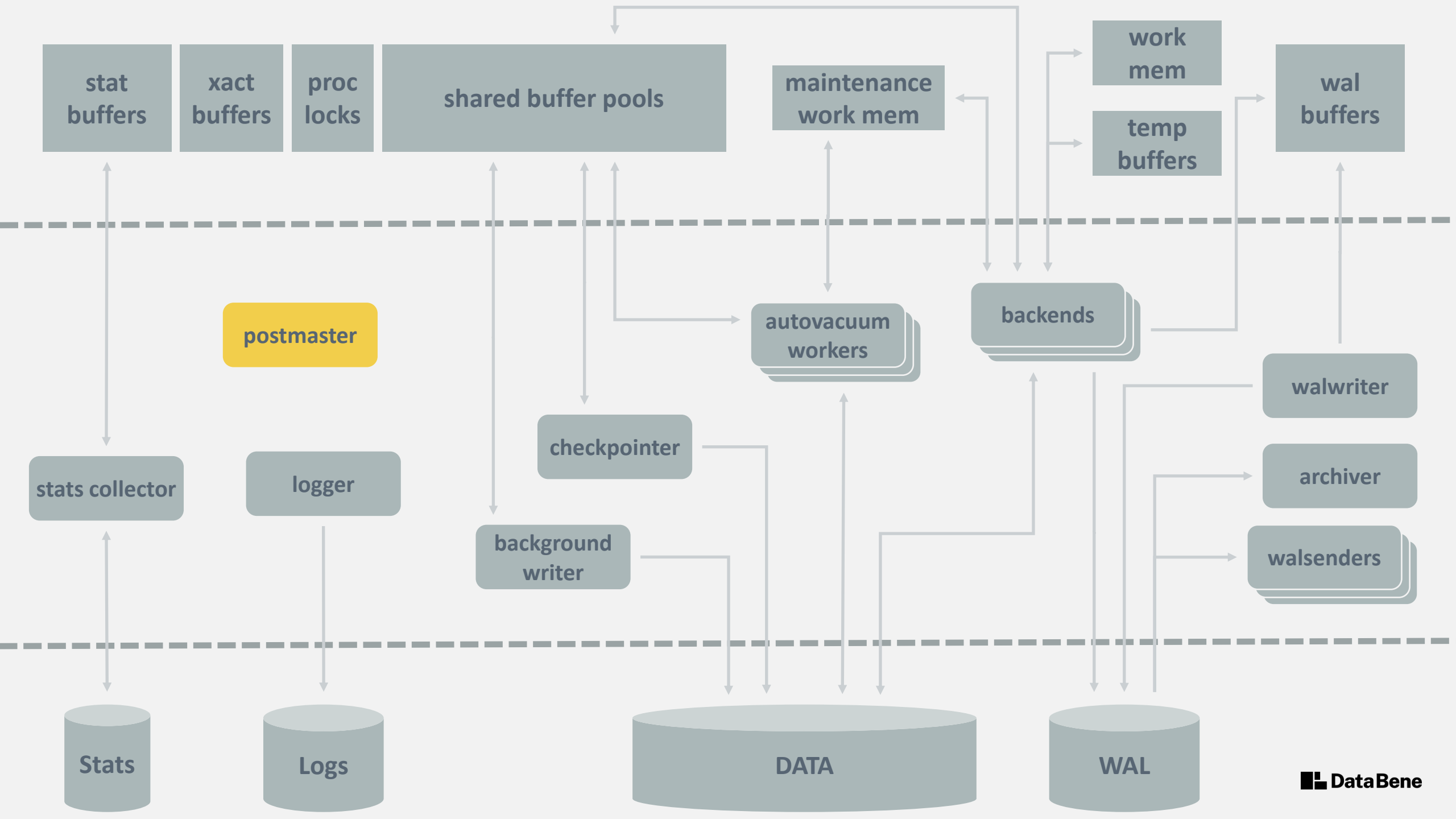


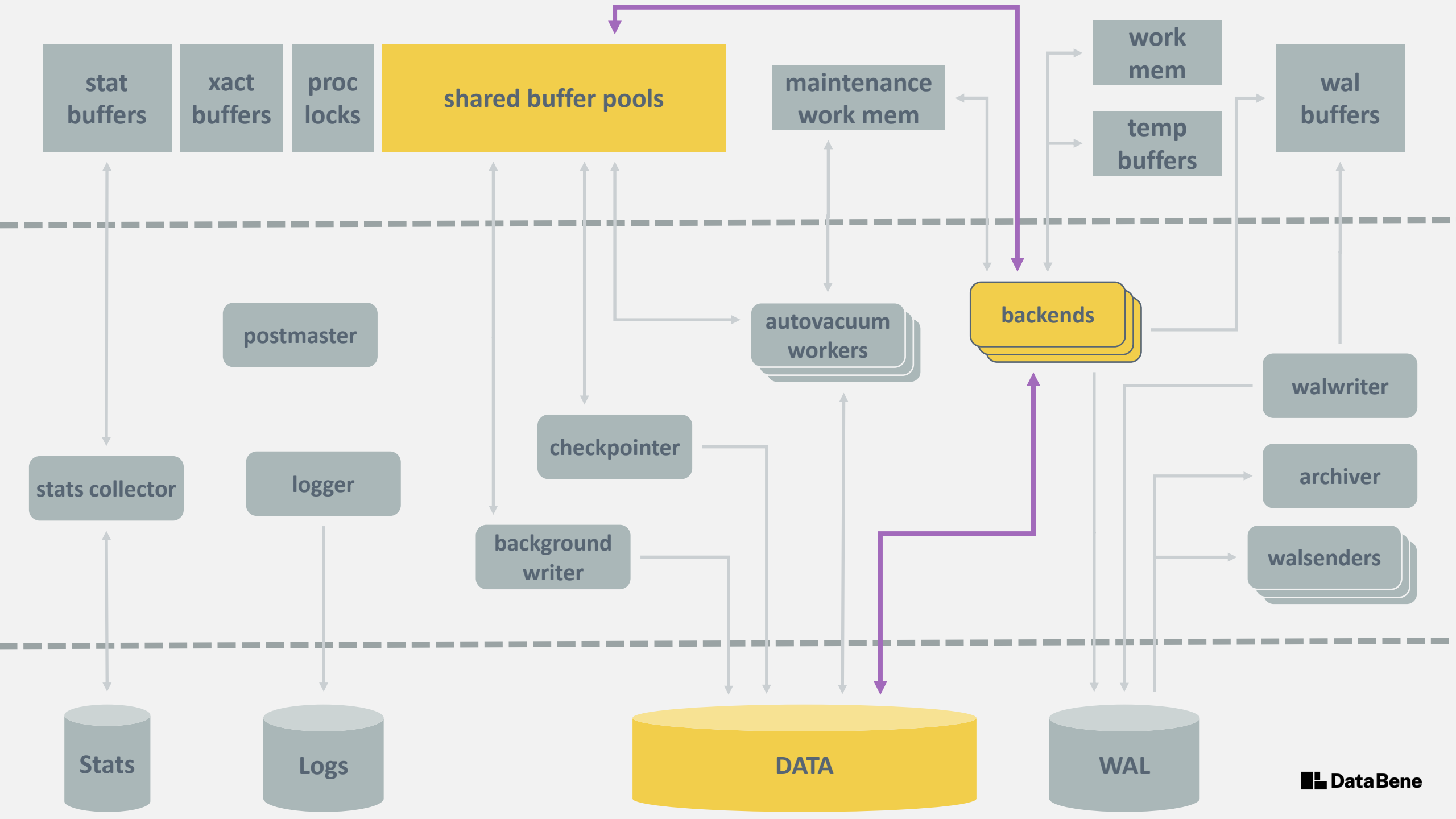
PostgreSQL Architecture

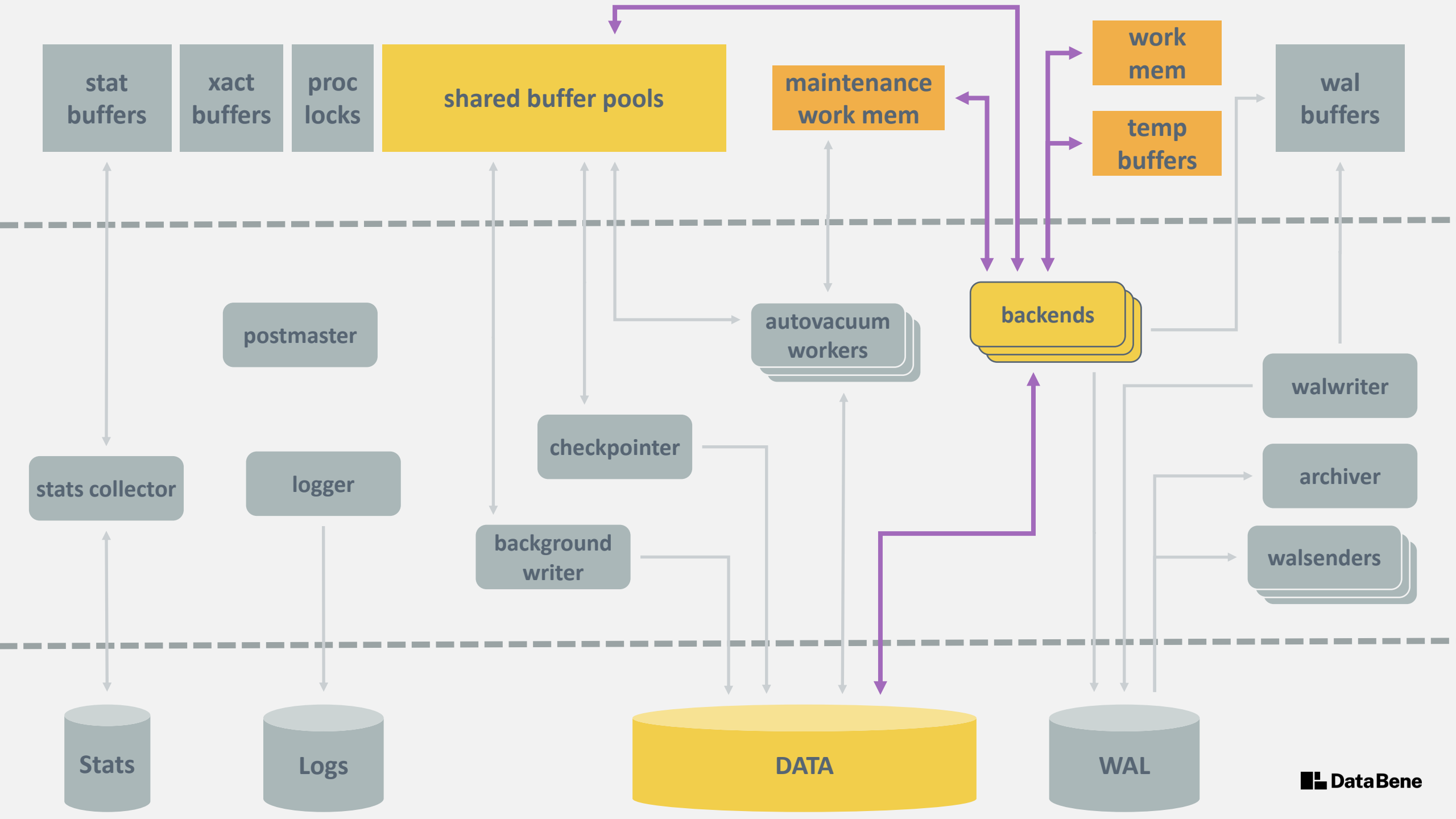
Memory – Process - Storage

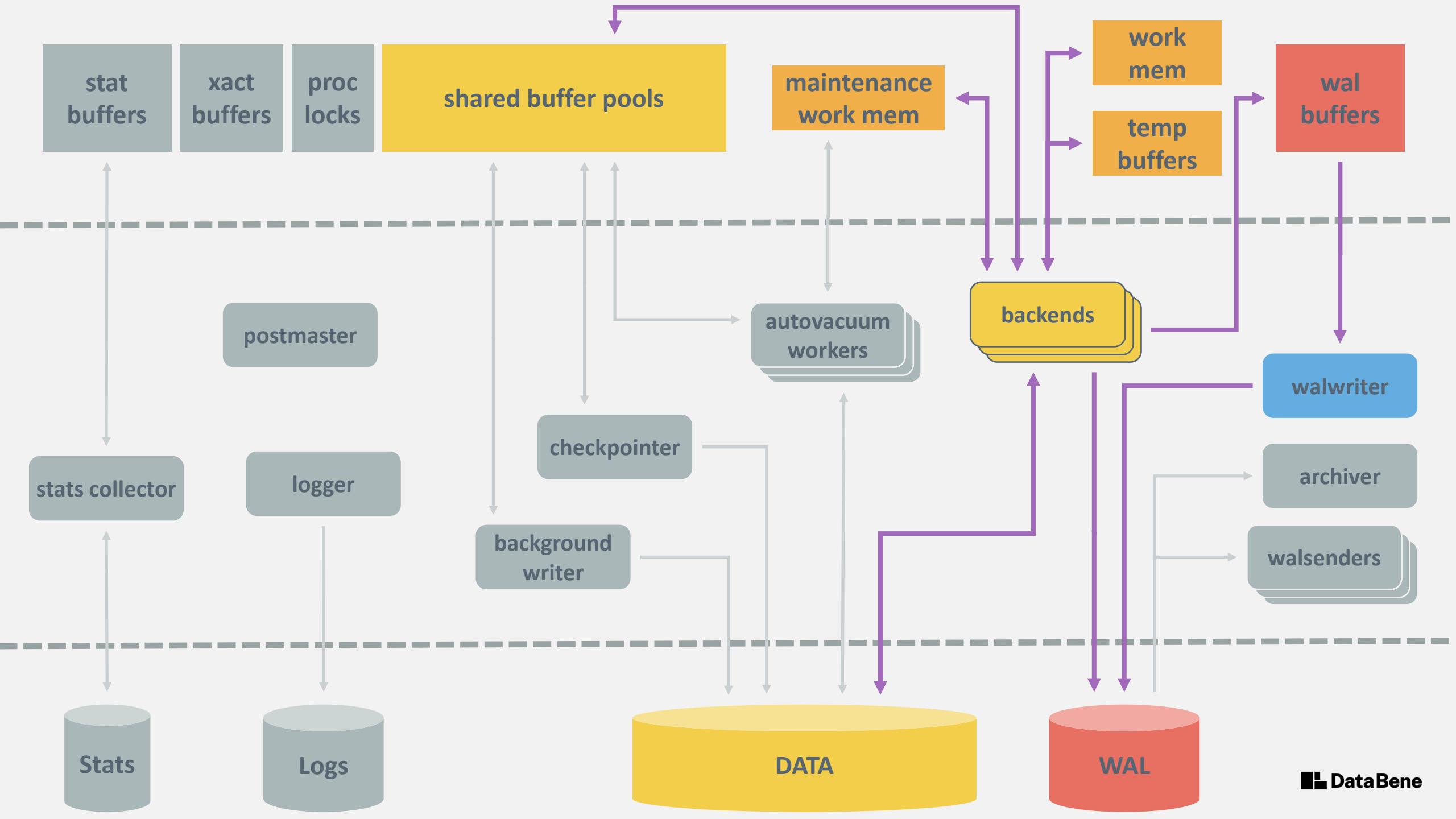
DataBene

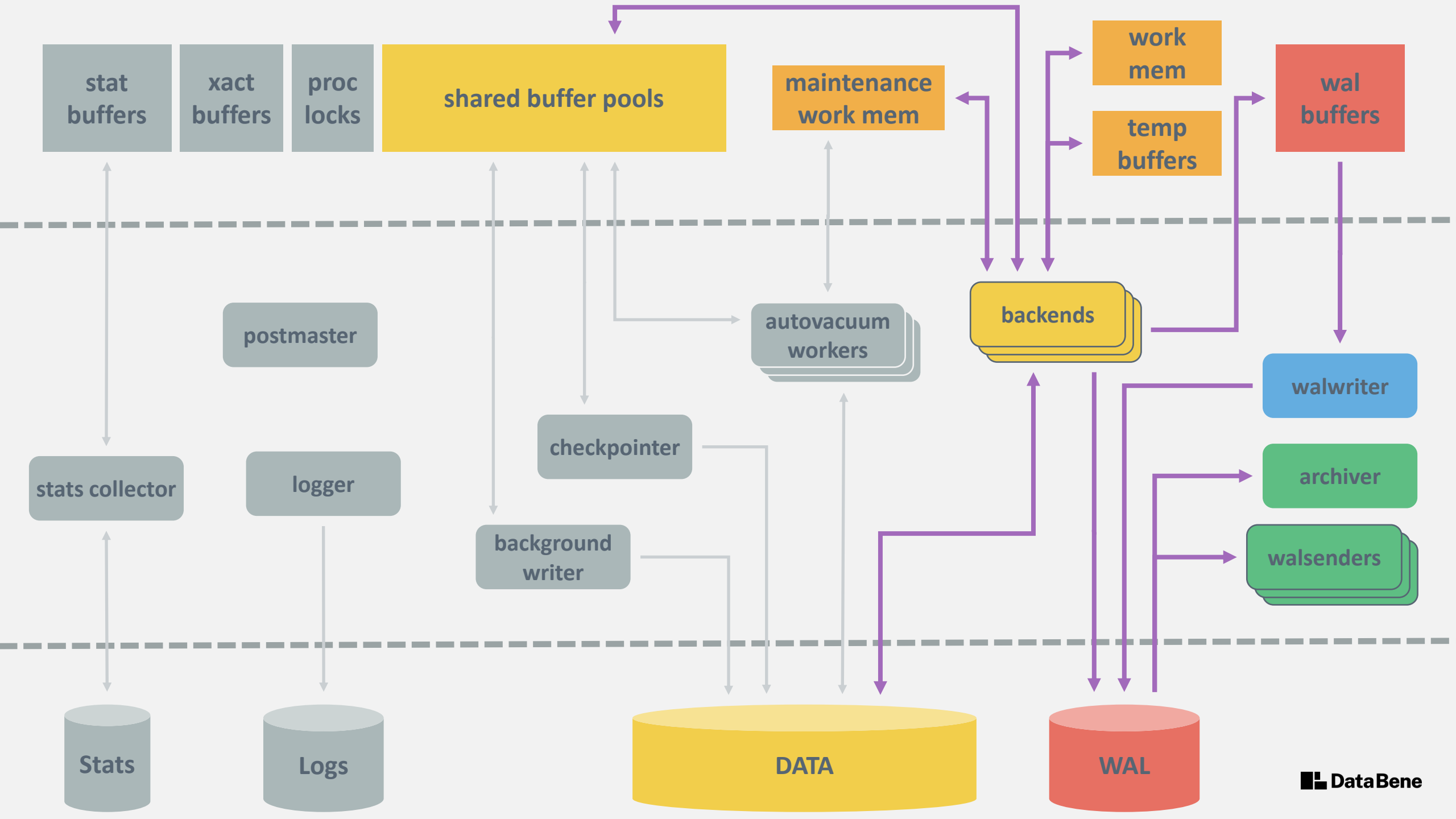

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

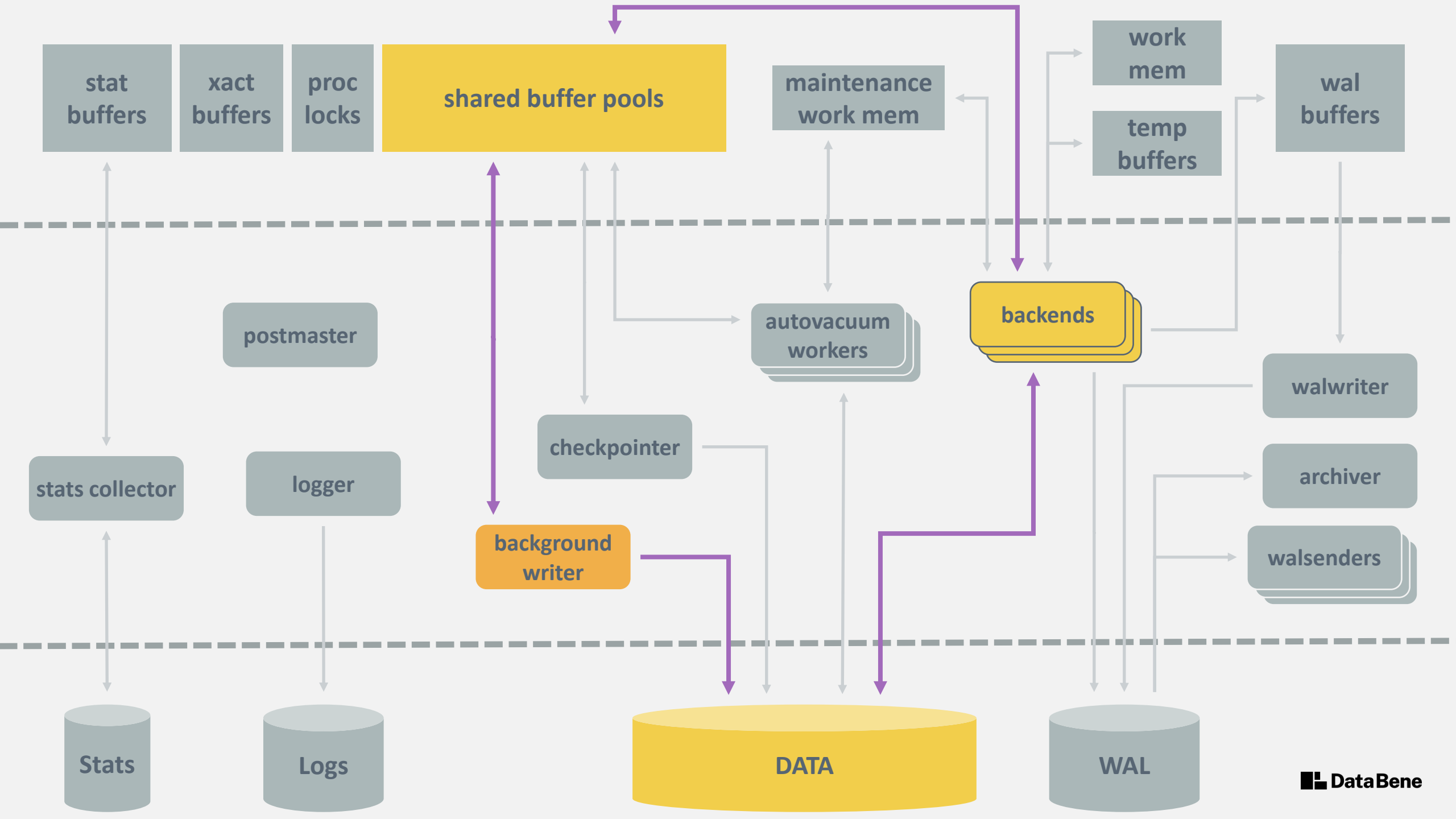

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

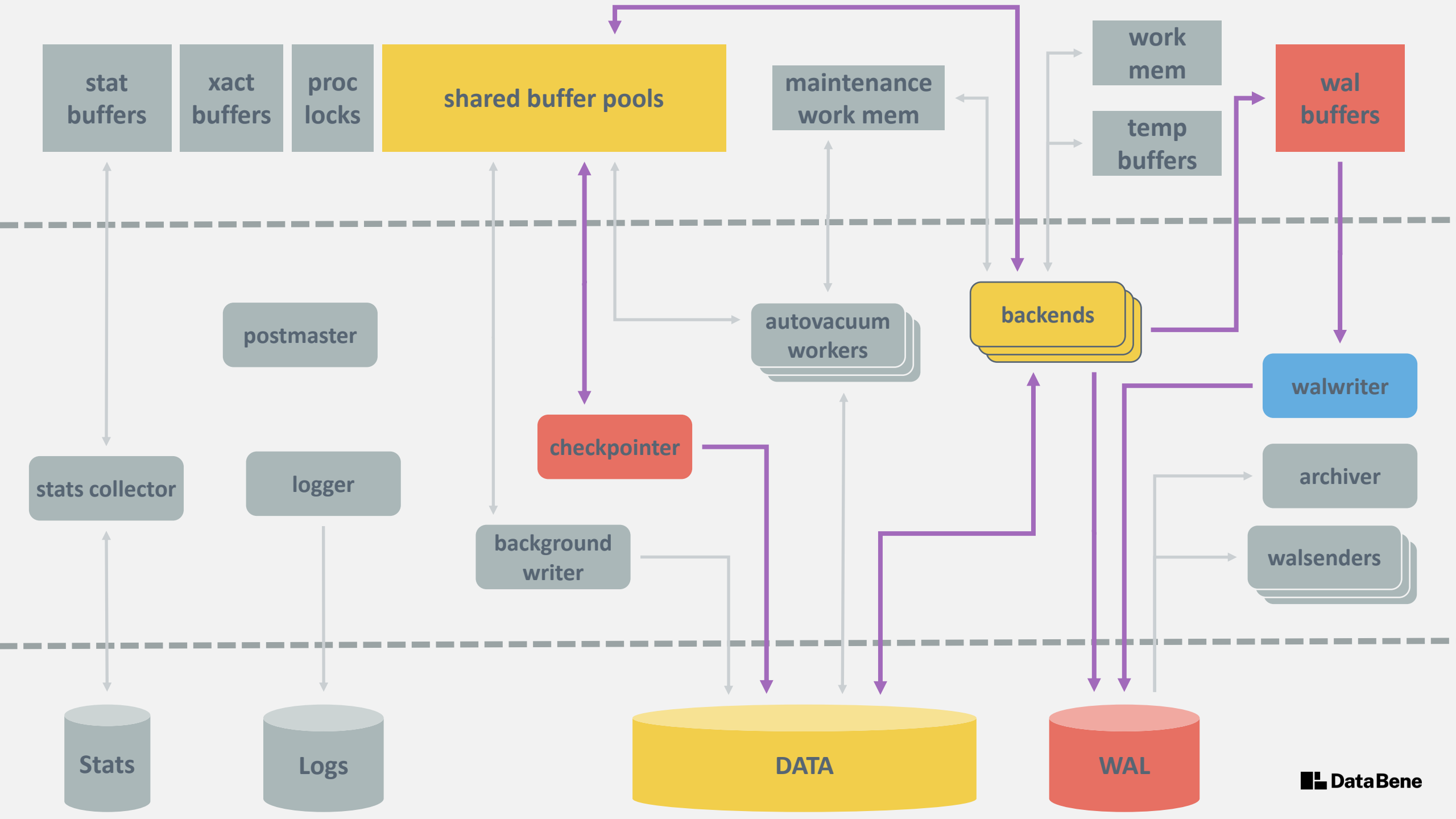

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

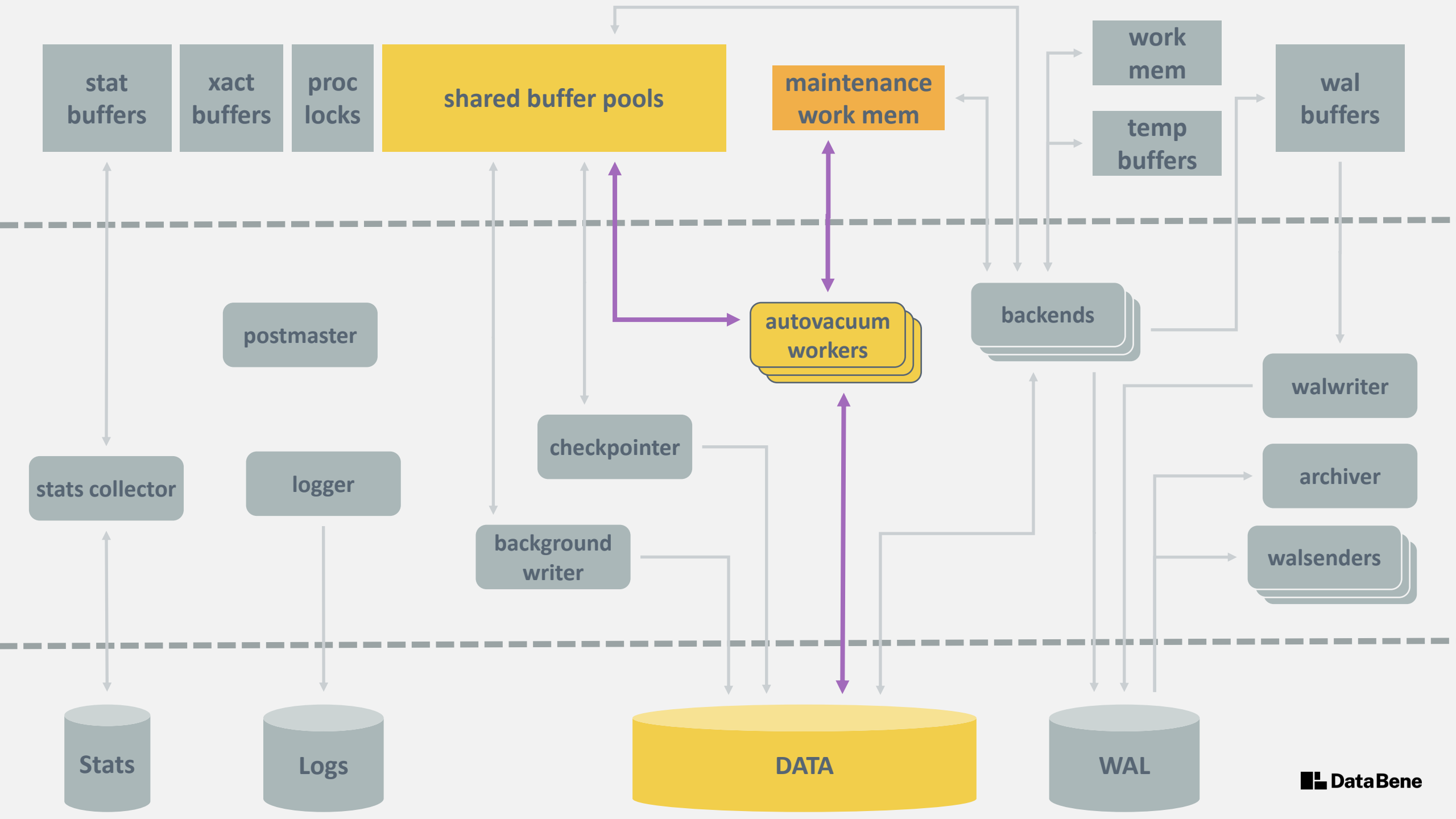

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

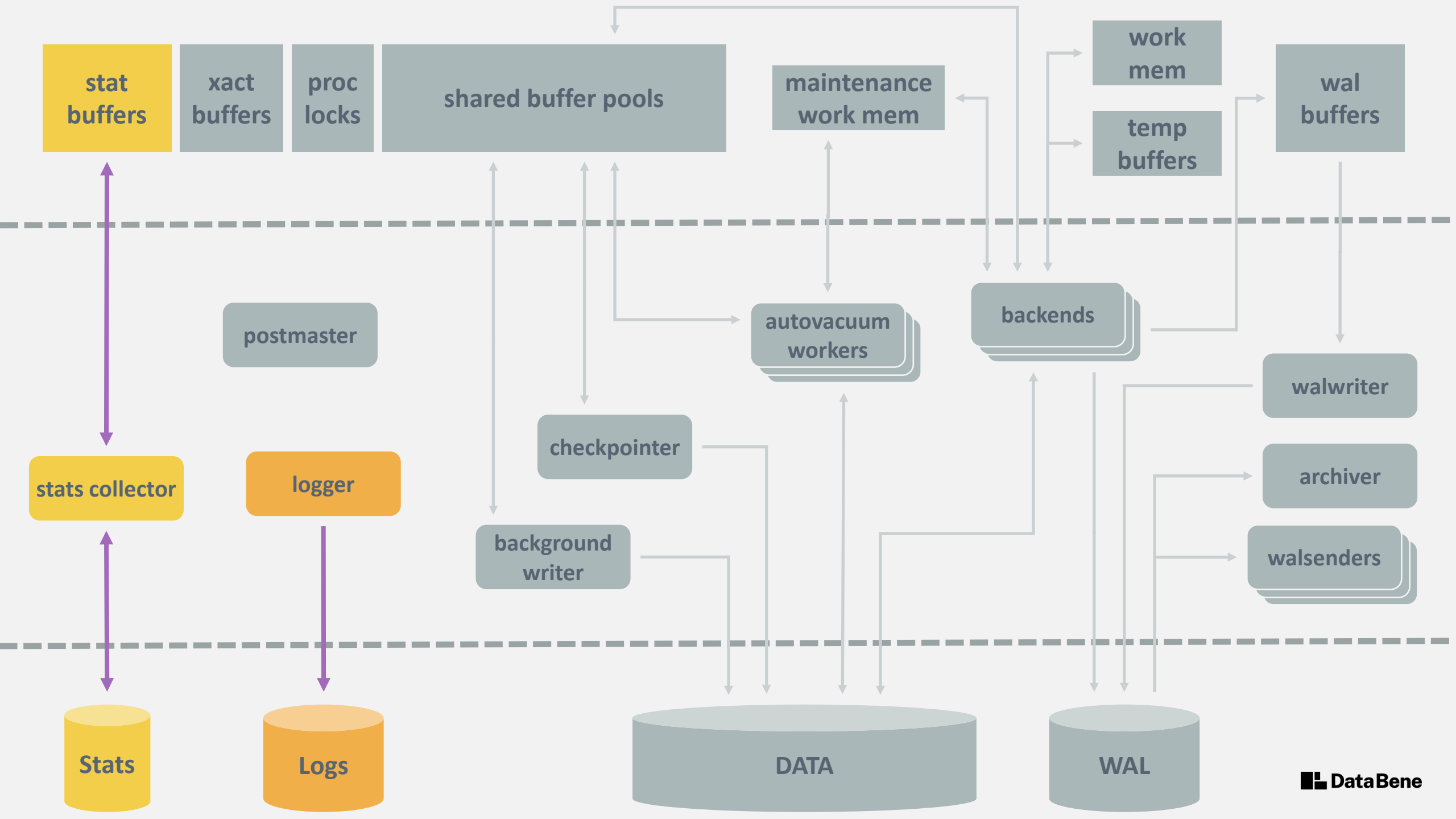

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

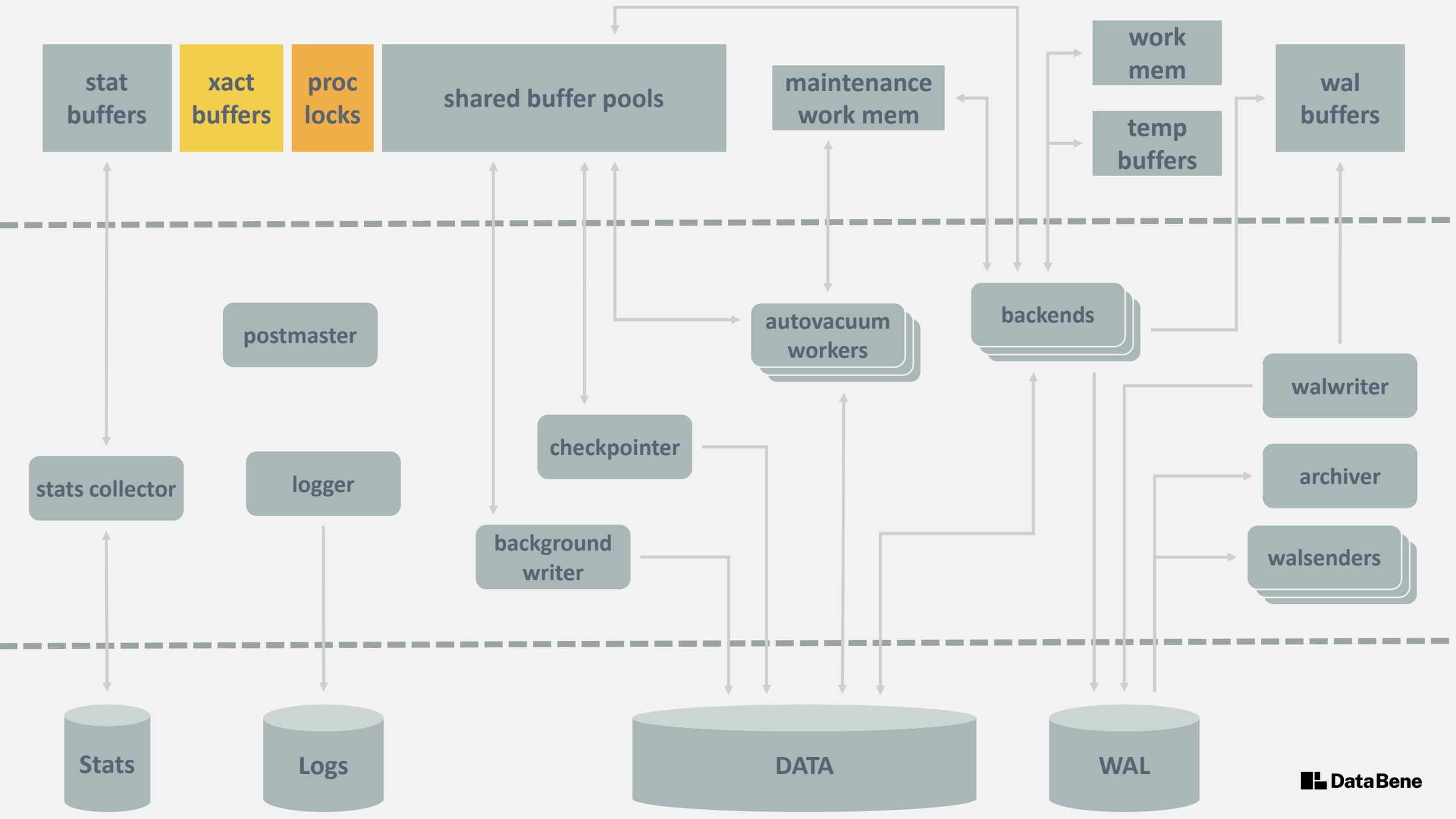

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene


stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene


stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
archiver
stats collector
logger
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

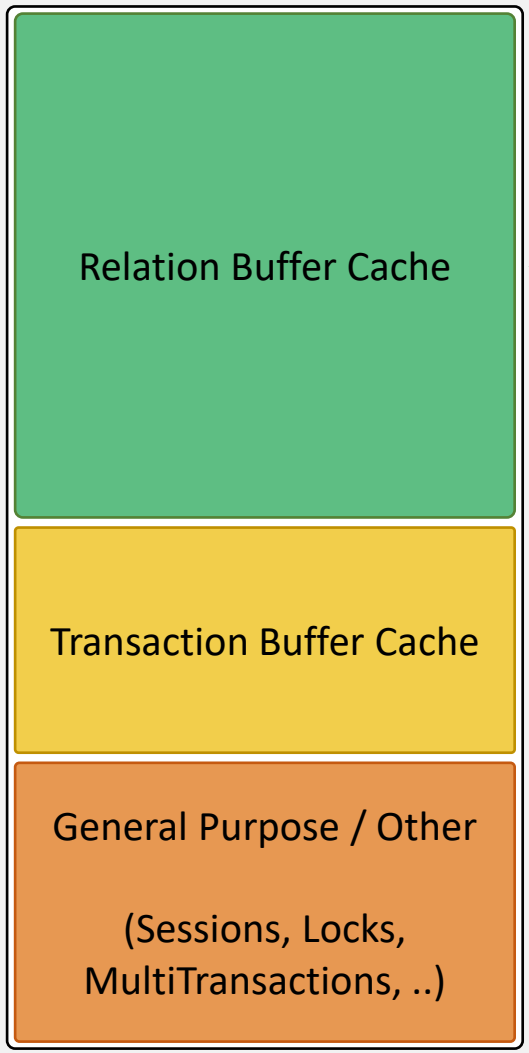

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene


stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
archiver
stats collector
logger
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

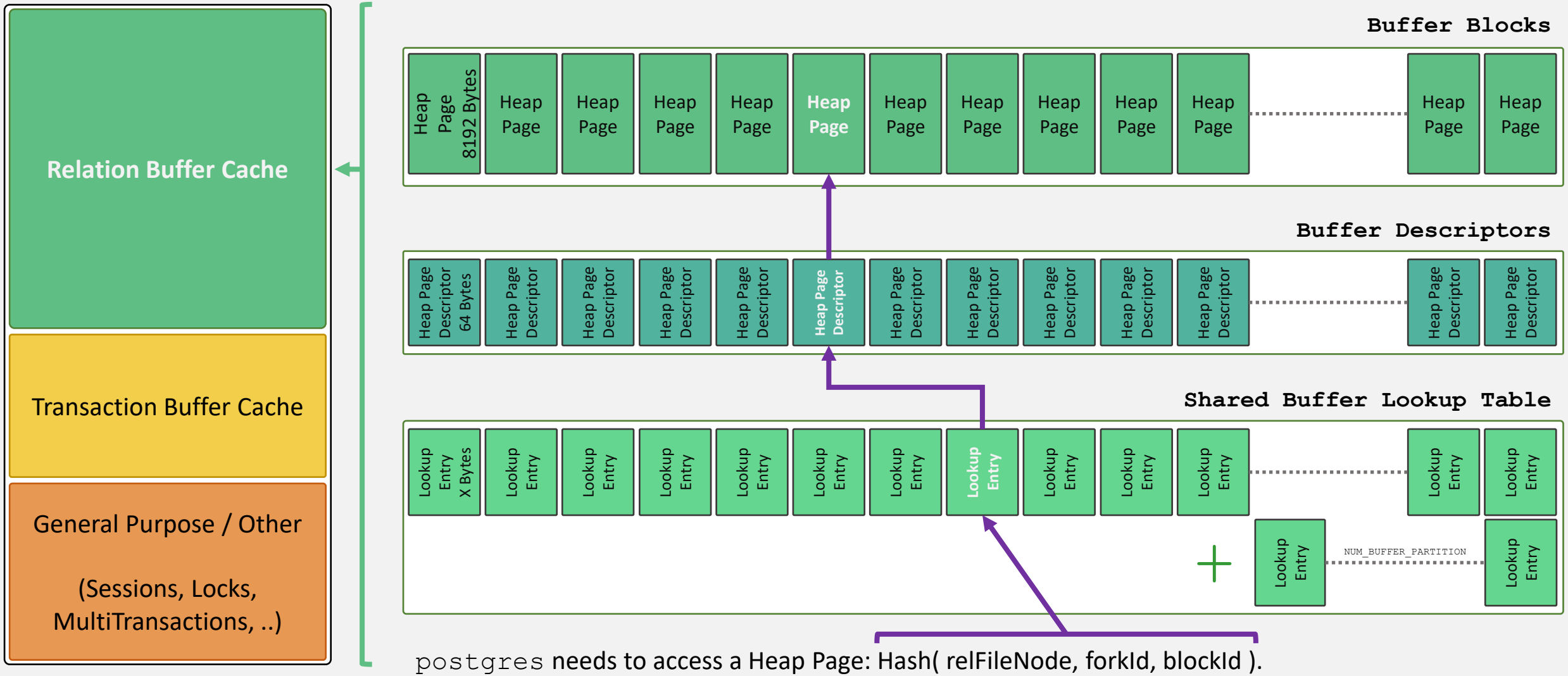

stat buffers
xact buffers
proc locks
shared buffer pools
maintenance work mem
work mem
temp buffers
wal buffers
postmaster
autovacuum workers
backends
walwriter
checkpointer
stats collector
logger
archiver
background writer
walsenders
Stats
Logs
DATA
WAL
DataBene

Shared Buffers

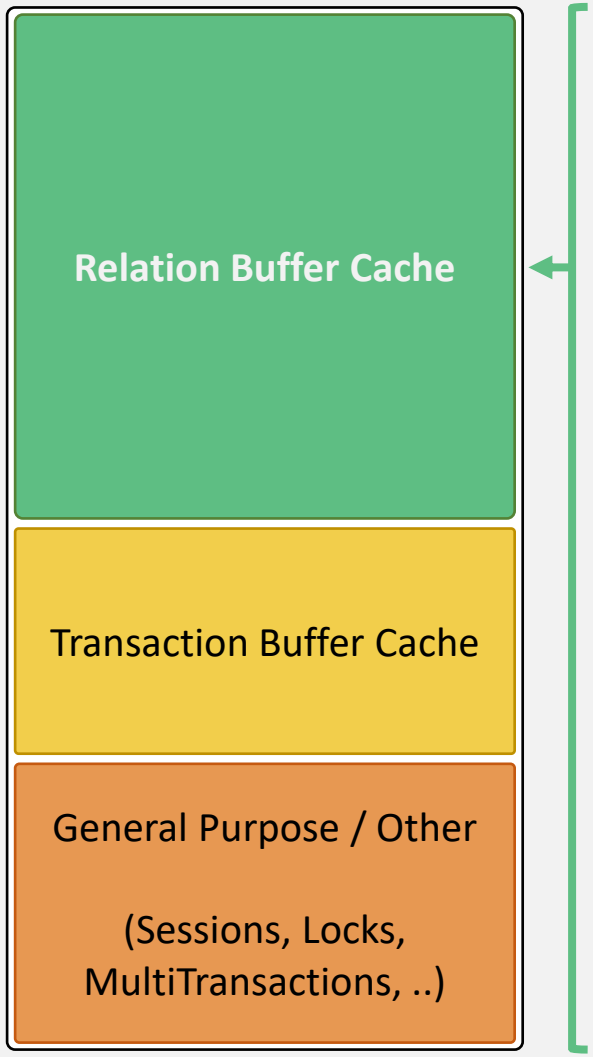
Structure – Usage – Processing

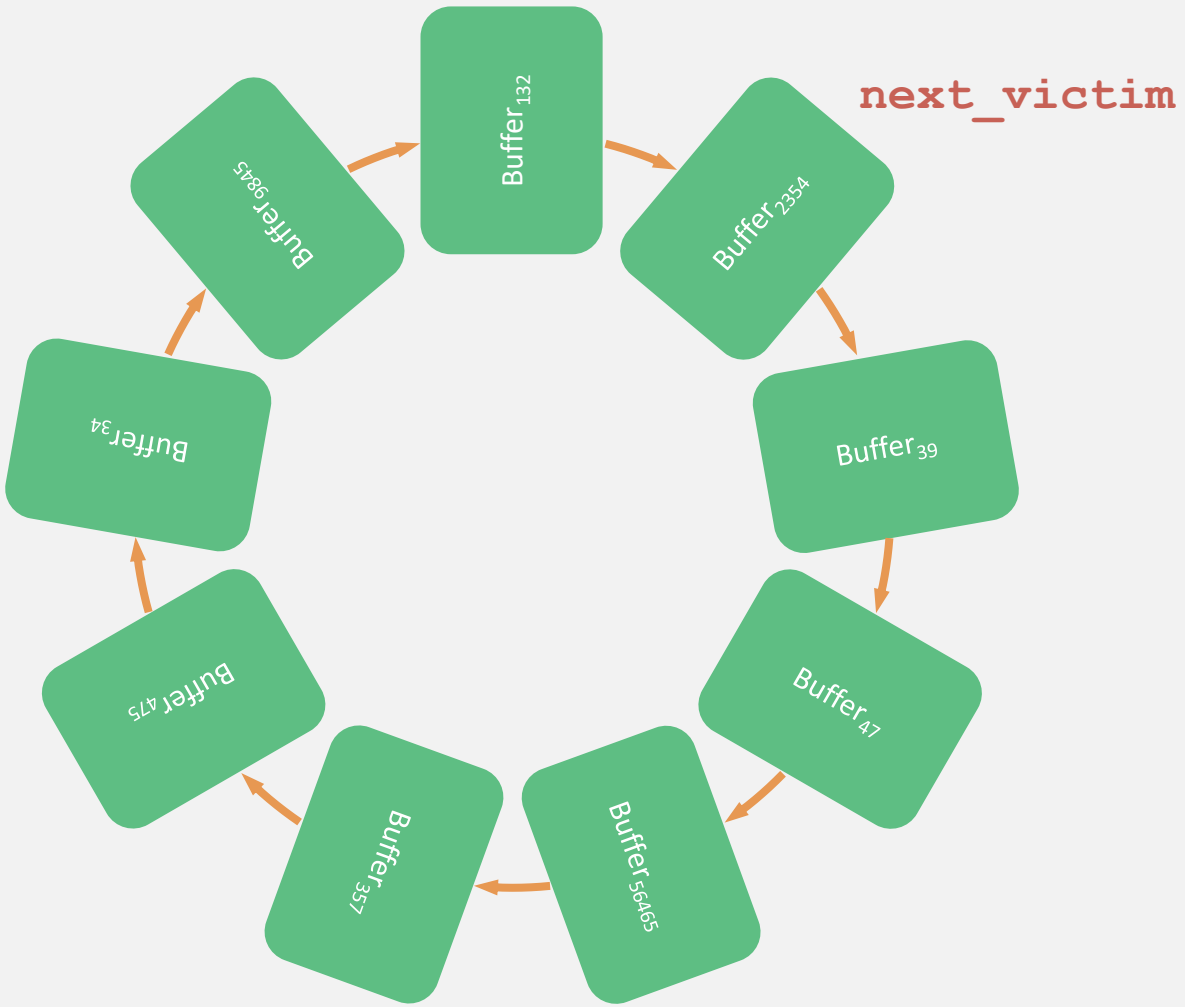


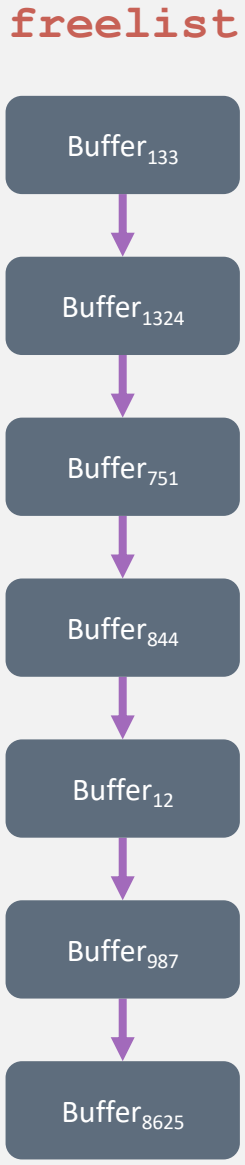
- Stored Objects
  - Tables, Indexes, Materialized Views
  - Free Space Map and Visibility Map.
- Used by Processes
  - `postgres`, `writer`, `checkpointer`.
- Modifications (INSERT, UPDATE, DELETE)
  - Are always done in the cache (`postgres`),
  - Written on Disk later (`checkpointer`, `writer`, `postgres`).


Relation Buffer Cache
Transaction Buffer Cache
General Purpose / Other
(Sessions, Locks, MultiTransactions, ..)
Buffer Blocks
Heap Page 8192 Bytes
Heap Page
Buffer Descriptors
Heap Page Descriptor 64 Bytes
Heap Page Descriptor
Shared Buffer Lookup Table
Lookup Entry X Bytes
Lookup Entry
NUM_BUFFER_PARTITION
postgres needs to access a Heap Page: Hash(relFileNode, forkId, blockId).

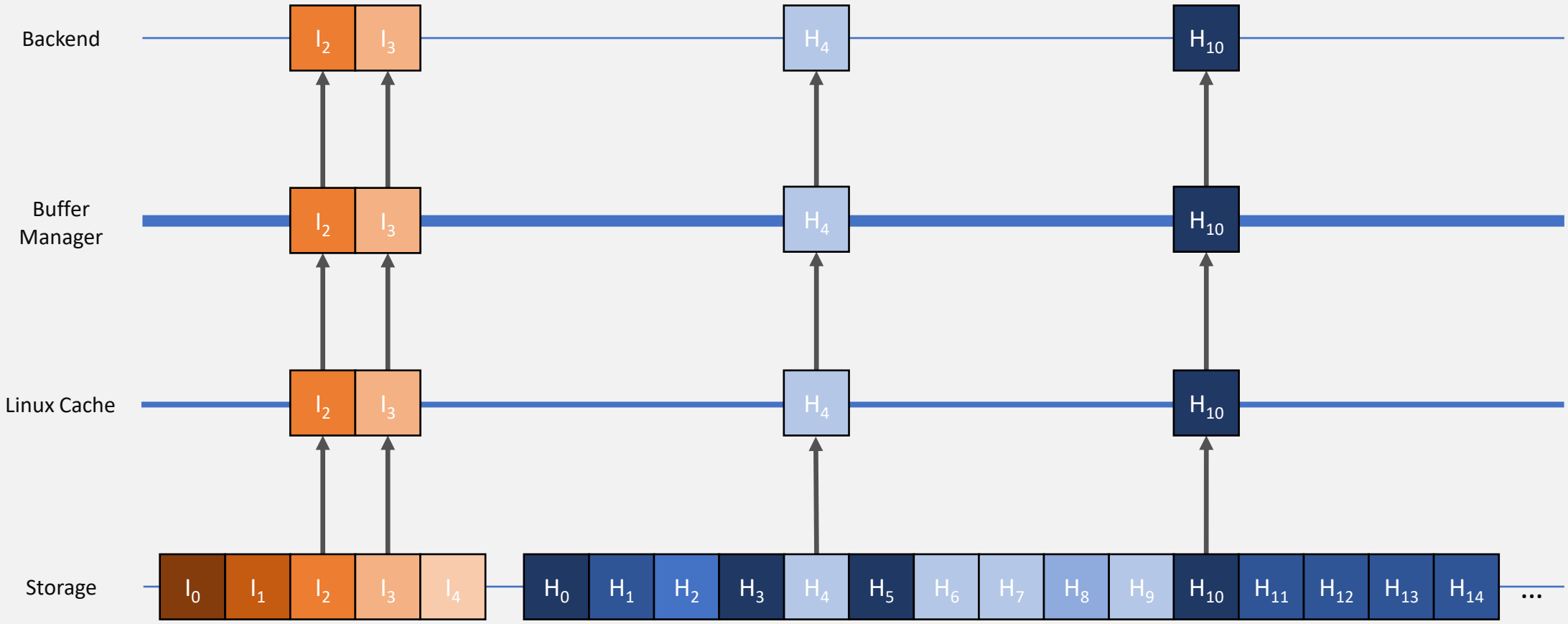
DataBene


Relation Buffer Cache
Transaction Buffer Cache
General Purpose / Other
(Sessions, Locks, MultiTransactions, ..)

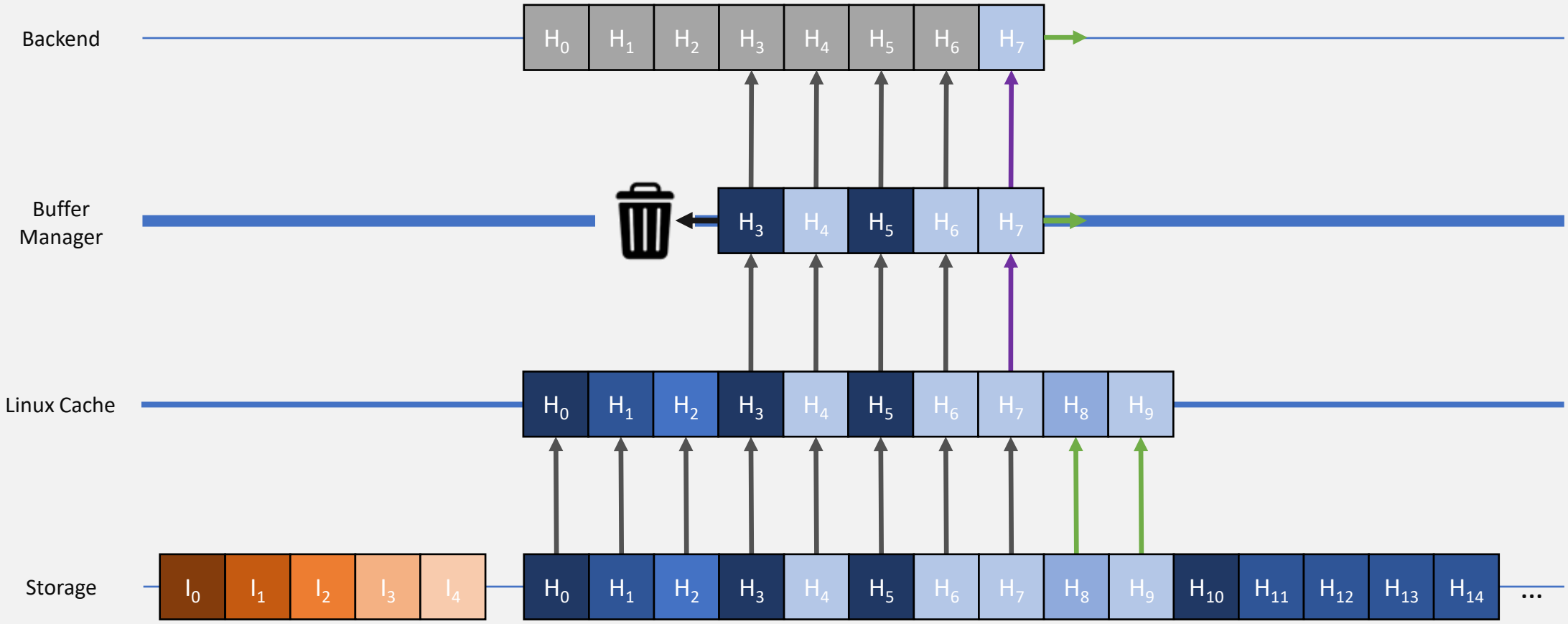

next_victim
Buffer132
Buffer2354
Buffer39
Buffer47
Buffer56465
Buffer357
Buffer475
Buffer34
Buffer9845


freelist
Buffer133
Buffer1324
Buffer751
Buffer844
Buffer12
Buffer987
Buffer8625

DataBene



Backend -> BufferManager -> BufferPool |-> Linux Cache |-> Storage



Backend -> BufferManager -> RingBuffer |-> Linux Cache |-> Storage

Index Level Statistics

pg_catalog.pg_stat_[all|sys|user]_indexes

Column	Type
relid	oid
indexrelid	oid
schemaname	**name**
relname	**name**
indexrelname	**name**
idx_scan	bigint
last_idx_scan	timestamp with time zone
idx_tup_read	bigint
idx_tup_fetch	bigint

pg_catalog.pg_stat_[all|sys|user]_indexes

Column	Type
relid	oid
indexrelid	oid
schemaname	name
relname	name
indexrelname	name
idx_scan	**bigint**
last_idx_scan	**timestamp with time zone**
idx_tup_read	bigint
idx_tup_fetch	bigint

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

$$rate_{idx\_scan} = \frac{\Delta idx\_scan}{\Delta T}$$

Old event when $last\_idx_{scan} \leq now() - k\Delta T; k \in [0.5\,;3[$

DataBene

pg_catalog.pg_stat_[all|sys|user]_indexes

Column	Type
relid	oid
indexrelid	oid
schemaname	name
relname	name
indexrelname	name
idx_scan	bigint
last_idx_scan	timestamp with time zone
idx_tup_read	**bigint**
idx_tup_fetch	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

- idx_tup_read is the number of returned entries by the index scan
- idx_tup_fetch is the number of entries which produced reads in heap
- idx_tup_fetch ≤ idx_tup_read

DataBene

pg_catalog.pg_stat_[all|sys|user]_indexes

Column	Type
relid	oid
indexrelid	oid
schemaname	name
relname	name
indexrelname	name
idx_scan	**bigint**
last_idx_scan	timestamp with time zone
idx_tup_read	**bigint**
idx_tup_fetch	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

$$ratio_{fetch} = \frac{\Delta idx\_tup\_fetch}{\Delta idx\_tup\_read}$$ (index_onlyscan)

$$\text{avg}_{index\_by\_scan} = \frac{\Delta idx\_tup\_read}{\Delta idx\_scan}$$ (for watch/baseline)

DataBene

pg_catalog.pg_statio_[all|sys|user]_indexes

Column	Type
relid	oid
indexrelid	oid
schemaname	name
relname	name
indexrelname	name
idx_blks_read	**bigint**
idx_blks_hit	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

$$\mathbf{cache_{idx\_hit}} = \frac{\Delta idx\_bkls\_hit}{\Delta idx\_bkls\_hit + \Delta idx\_bkls\_read} \geq \mathbf{0.98}$$

$$\mathbf{cache_{idx\_miss}} = \frac{\Delta idx\_blks\_read}{\Delta idx\_blks\_hit + \Delta idx\_blks\_read}$$

DataBene

Table Level Statistics

pg_catalog.pg_stat_[all|sys|user]_tables

Column	Type
relid	oid
schemaname	**name**
relname	**name**
seq_scan	bigint
last_seq_scan	timestamp with time zone
seq_tup_read	bigint
idx_scan	bigint
last_idx_scan	timestamp with time zone
idx_tup_fetch	bigint
n_tup_ins	bigint
n_tup_upd	bigint
n_tup_del	bigint
n_tup_hot_upd	bigint
n_tup_newpage_upd	bigint

Column	Type
n_live_tup	bigint
n_dead_tup	bigint
n_mod_since_analyze	bigint
n_ins_since_vacuum	bigint
last_vacuum	timestamp with time zone
last_autovacuum	timestamp with time zone
last_analyze	timestamp with time zone
last_autoanalyze	timestamp with time zone
vacuum_count	bigint
autovacuum_count	bigint
analyze_count	bigint
autoanalyze_count	bigint

pg_catalog.pg_stat_[all|sys|user]_tables

Column	Type
relid	oid
schemaname	name
relname	name
seq_scan	**bigint**
last_seq_scan	**timestamp with time zone**
seq_tup_read	**bigint**
idx_scan	bigint
last_idx_scan	timestamp with time zone
idx_tup_fetch	bigint

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

$$\text{avg}_{tuples\_by\_scan} = \frac{\Delta seq\_tup\_read}{\Delta seq\_scan}$$ (for watch/baseline)

Old event when $last\_ * \_scan \leq now() - k\Delta T; k \in [0.5\,;3[$

DataBene

pg_catalog.pg_stat_[all|sys|user]_tables

Column	Type
relid	oid
schemaname	name
relname	name
seq_scan	bigint
last_seq_scan	timestamp with time zone
seq_tup_read	bigint
idx_scan	**bigint**
last_idx_scan	**timestamp with time zone**
idx_tup_fetch	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

$$\text{avg}_{tuples\_by\_scan} = \frac{\Delta idx\_tup\_fetch}{\Delta idx\_scan}$$ (for watch/baseline)

Old event when $last\_ * \_scan \leq now() - k\Delta T; k \in [0.5\,;3[$

DataBene

pg_catalog.pg_stat_[all|sys|user]_tables

Column	Type
relid	oid
schemaname	name
relname	name
n_tup_ins	**bigint**
n_tup_upd	**bigint**
n_tup_del	**bigint**
n_tup_hot_upd	**bigint**
n_tup_newpage_upd	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

$$ratio_{hot} = \frac{\Delta n\_tup\_hot\_upd}{\Delta n\_tup\_upd}$$

$$ratio_{newpage} = \frac{\Delta n\_tup\_newpage\_upd}{\Delta n\_tup\_upd}$$

$$ratio_{rw} = \frac{\Delta seq\_tup\_read + \Delta idx\_tup\_fetch}{\Delta n\_tup\_ins + \Delta n\_tup\_upd + \Delta n\_tup\_del}$$

DataBene

pg_catalog.pg_stat_[all|sys|user]_tables

Column	Type
relid	oid
schemaname	name
relname	name
n_mod_since_analyze	**bigint**
!last_analyze	**timestamp with time zone**
!last_autoanalyze	**timestamp with time zone**
analyze_count	bigint
autoanalyze_count	bigint

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

Make sure table received (auto)ANALYZE

$$n\_mod\_since\_analyze \geq analyze\_threshold + analyze\_scale\_factor * pg\_class.reltuples$$

DataBene

pg_catalog.pg_stat_[all|sys|user]_tables

Column	Type
relid	oid
schemaname	name
relname	name
n_live_tup	**bigint**
n_dead_tup	**bigint**
n_ins_since_vacuum	**bigint**
!last_vacuum	**timestamp with time zone**
!last_autovacuum	**timestamp with time zone**
vacuum_count	bigint
autovacuum_count	bigint

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

Make sure table received (auto)VACUUM

$$n\_ins\_since\_vacuum \geq vacuum\_insert\_threshold + vacuum\_insert\_scale\_factor * pg\_class.reltuples$$
$$n\_dead\_tup \geq vacuum\_threshold + vacuum\_scale\_factor * pg\_class.reltuples$$

DataBene

pg_catalog.pg_statio_[all|sys|user]_tables

Column	Type
relid	oid
schemaname	name
relname	name
heap_blks_read	**bigint**
heap_blks_hit	**bigint**
idx_blks_read	**bigint**
idx_blks_hit	**bigint**
toast_blks_read	**bigint**
toast_blks_hit	**bigint**
tidx_blks_read	**bigint**
tidx_blks_hit	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 15\, mins)$$

$$cache_{hit} = \frac{\Delta x\_hit}{\Delta x\_hit + \Delta x\_read}$$

$$cache_{miss} = \frac{\Delta x\_read}{\Delta x\_hit + \Delta x\_read}$$

DataBene

Database Level Statistics

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
numbackends	integer
xact_commit	bigint
xact_rollback	bigint
blks_read	bigint
blks_hit	bigint
tup_returned	bigint
tup_fetched	bigint
tup_inserted	bigint
tup_updated	bigint
tup_deleted	bigint
conflicts	bigint
temp_files	bigint
temp_bytes	bigint

Column	Type
deadlocks	bigint
checksum_failures	bigint
checksum_last_failure	timestamp with time zone
blk_read_time	double precision
blk_write_time	double precision
session_time	double precision
active_time	double precision
idle_in_transaction_time	double precision
sessions	bigint
sessions_abandoned	bigint
sessions_fatal	bigint
sessions_killed	bigint
stats_reset	timestamp with time zone

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	**oid**
datname	**name**

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
numbackends	**integer**

The number of currently connected backends to this database!

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
stats_reset	**timestamp with time zone**

The cumulated statistic counters started accumulating since **stats_reset**.

pg_catalog.pg_stat_database

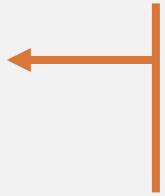
Column	Type
datid	oid
datname	name
xact_commit	**bigint**
xact_rollback	bigint

Number of COMMITTED transactions since **stats_reset**.

SELECT are accounted!

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
xact_commit	**bigint**
xact_rollback	bigint



$$tps_{committed} = \frac{\Delta xact\_commit}{\Delta T}$$

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
xact_commit	**bigint**
xact_rollback	bigint

$$tps_{committed} = \frac{\Delta xact\_commit}{\Delta T}$$

Should we alert when $tps_{committed}$ is outside upper and lower boundaries?

- **Absolute boundaries**
- **$\leq 0.5x$ *or* $\geq 2x$ window averaged**
- **Compared to expected timely activity**

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
xact_commit	**bigint**
xact_rollback	bigint

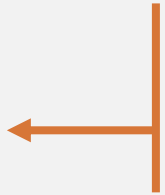
$$tps_{committed} = \frac{\Delta xact\_commit}{\Delta T}$$

Should we up/down-scale the system?

DataBene

pg_catalog.pg_stat_database

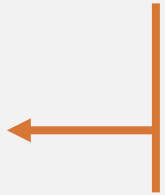
Column	Type
datid	oid
datname	name
xact_commit	bigint
xact_rollback	**bigint**



Should be kept as low as possible!

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
xact_commit	bigint
xact_rollback	**bigint**

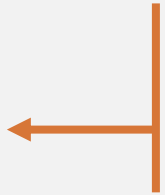


Too much rollbacks means:

- concurrency issues (or bad design)
- producing dead tuples!

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
xact_commit	bigint
xact_rollback	**bigint**



$$tps_{rollacked} = \frac{\Delta xact\_rollback}{\Delta T}$$

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
xact_commit	**bigint**
xact_rollback	**bigint**

$$tps_{committed} = \frac{\Delta xact\_commit}{\Delta T}$$

$$tps_{rollacked} = \frac{\Delta xact\_rollback}{\Delta T}$$

$$\text{ratio}_{rollacked} = \frac{\Delta xact\_rollback}{\Delta xact\_committed}$$

Should we alert when $\text{ratio}_{rollbacked}$ is higher than a threshold?

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
xact_commit	**bigint**
xact_rollback	**bigint**

$$tps_{committed} = \frac{\Delta xact\_commit}{\Delta T}$$

$$tps_{rollacked} = \frac{\Delta xact\_rollback}{\Delta T}$$

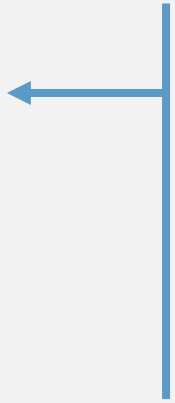
$$\text{ratio}_{rollacked} = \frac{\Delta xact\_rollback}{\Delta xact\_committed}$$

You just degraded performance and power efficiency by $\text{ratio}_{\text{rollbacked}}$ of your PostgreSQL server over the last observed period.

DataBene

pg_catalog.pg_stat_database

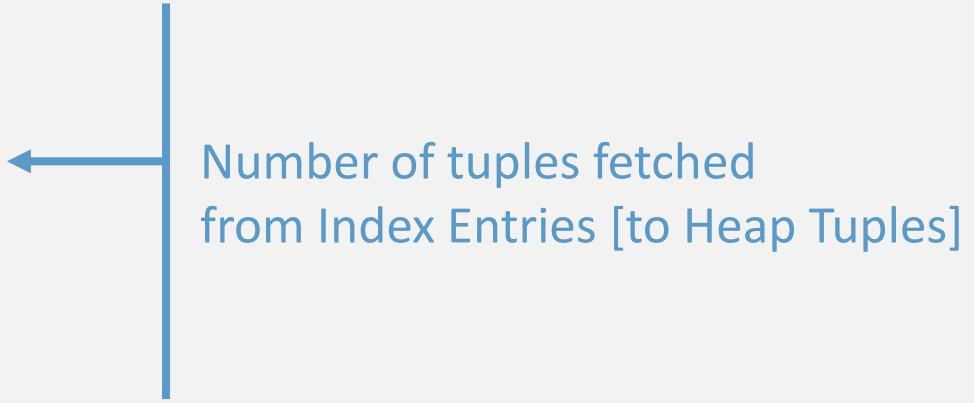
Column	Type
datid	oid
datname	name
tup_returned	**bigint**
tup_fetched	bigint
tup_inserted	bigint
tup_updated	bigint
tup_deleted	bigint



Number of visible tuples manipulated in SELECT, UPDATE, DELETE, VACUUM… excluding INSERT

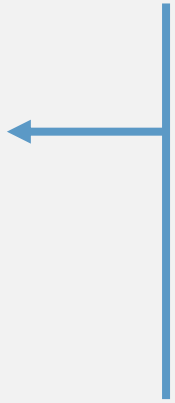
pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
tup_returned	bigint
tup_fetched	**bigint**
tup_inserted	bigint
tup_updated	bigint
tup_deleted	bigint



pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
tup_returned	**bigint**
tup_fetched	**bigint**
tup_inserted	bigint
tup_updated	bigint
tup_deleted	bigint



$$\text{index\_scan}\_ratio = \frac{\Delta tup\_fetched}{\Delta tup\_returned}$$

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
tup_returned	bigint
tup_fetched	bigint
tup_inserted	**bigint**
tup_updated	**bigint**
tup_deleted	**bigint**

Data Manipulation Language
- INSERT
- UPDATE
- DELETE

Produce dead tuples (no tracking at db level)
- INSERT + ROLLBACK | UPDATE + ROLLBACK
- UPDATE + COMMIT | DELETE + COMMIT

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
tup_returned	**bigint**
tup_fetched	**bigint**
tup_inserted	**bigint**
tup_updated	**bigint**
tup_deleted	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T}$$

$$rate_{DML} = \frac{\Delta inserted + \Delta updated + \Delta deleted}{\Delta T}$$

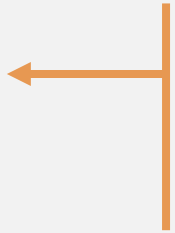
$$ratio_{rw} = \frac{rate_{tup\ returned}}{\bar{rate}_{DML}}$$

Do you find it useful?

DataBene

pg_catalog.pg_stat_database

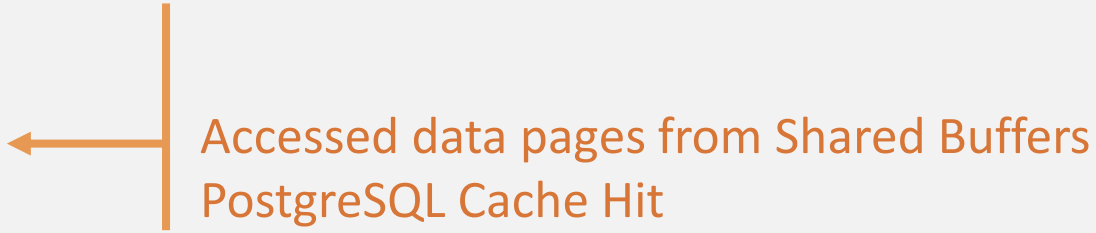
Column	Type
datid	oid
datname	name
blks_read	**bigint**
blks_hit	bigint
temp_files	bigint
temp_bytes	bigint
blk_read_time	double precision
blk_write_time	double precision



Missed data pages from Shared Buffers
PostgreSQL Cache Miss

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
blks_read	bigint
blks_hit	**bigint**
temp_files	bigint
temp_bytes	bigint
blk_read_time	double precision
blk_write_time	double precision



pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
blks_read	**bigint**
blks_hit	**bigint**
temp_files	bigint
temp_bytes	bigint
blk_read_time	double precision
blk_write_time	double precision

$$rate_x = \frac{\Delta x}{\Delta T}$$

$$cache_{hit} = \frac{\Delta blks\_hit}{\Delta blk\_hit + \Delta blks\_read}$$

$$cache_{miss} = \frac{\Delta blks\_read}{\Delta blk\_hit + \Delta blks\_read}$$

- **Usually $cache_{hit} < 0.7$ is correlated with performance issues.**
- **Low $cache_{hit}$ ratio is expected when workload consists in accessing uniform distributed tuples over large tables.**

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
blks_read	bigint
blks_hit	bigint
temp_files	**bigint**
temp_bytes	**bigint**
blk_read_time	double precision
blk_write_time	double precision

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \; \Delta T \in (5, 15, 30, 60\ mins, 1\ day)$$

$$avg_{filesize} = \frac{\Delta temp\_bytes}{\Delta temp\_files}$$ <- not that usefull

More information in log files (pid<->temp_files)

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
blks_read	bigint
blks_hit	bigint
temp_files	bigint
temp_bytes	bigint
blk_read_time	**double precision**
blk_write_time	**double precision**

$rate_x = \frac{\Delta x}{\Delta T}$; for which ΔT?

To be compared with what ?
Useful or useless ?

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
deadlocks	**bigint**
conflicts	bigint
checksum_failures	bigint
checksum_last_failure	timestamp with time zone

$rate_x = \frac{\Delta x}{\Delta T}$; $\Delta T \in (60\ mins, 1\ day)$?

Catches bugs not performance issues
Use log content!

Alert on detection!

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
deadlocks	bigint
conflicts	**bigint**
checksum_failures	bigint
checksum_last_failure	timestamp with time zone

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (30s, 1 - 60\, mins, 1\, day)?$$

Number of cancelled queries due to conflicts with recovery (standby only)

DataBene

pg_catalog.pg_stat_database_conflicts

Column	Type
datid	oid
datname	name
confl_tablespace	**bigint**
confl_lock	**bigint**
confl_snapshot	**bigint**
confl_bufferpin	**bigint**
confl_deadlock	**bigint**
confl_active_logicalslot	**bigint**

Only on Standby

$rate_x = \frac{\Delta x}{\Delta T}$; $\Delta T \in (30s, 1 - 60\ mins, 1\ day)$?

Causes of cancelled queries due to conflicts with recovery

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
deadlocks	bigint
conflicts	bigint
checksum_failures	**bigint**
checksum_last_failure	**timestamp with time zone**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1 - 60\, mins)$$

ALERT!!!
Be proactive on this, check regularly.

DataBene

pg_catalog.pg_stat_database

Column	Type
datid	oid
datname	name
session_time	**double precision**
active_time	**double precision**
idle_in_transaction_time	**double precision**
sessions	**bigint**
sessions_abandoned	**bigint**
sessions_fatal	**bigint**
sessions_killed	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (5 - 60\, mins, 1\, day)$$

Updated on « **pg_stat_activity.state_change** »

$$ratio_{active} = \frac{\Delta active\_time}{\Delta session\_time}$$

$$rate_{idle\_trans} = \frac{\Delta idle\_in\_transaction\_time}{\Delta session\_time}$$

DataBene

Cluster Level Statistics

Standalone

pg_catalog.pg_stat_io

Column	Type
backend_type	text
object	text
context	text
reads	bigint
read_time	double precision
writes	bigint
write_time	double precision
writebacks	bigint
writeback_time	double precision
extends	bigint
extend_time	double precision
op_bytes	bigint
hits	bigint
evictions	bigint
reuses	bigint
fsyncs	bigint
fsync_time	double precision
stats_reset	timestamp with time zone

pg_catalog.pg_stat_io

Column	Type
backend_type	**text**
stats_reset	**timestamp with time zone**

- autovacuum launcher,
- autovacuum worker,
- logical replication launcher,
- logical replication worker,
- client backend,
- parallel worker,
- background writer,
- checkpointer,
- archiver,
- standalone backend,
- startup,
- walreceiver,
- walsender,
- walwriter
- walsummarizer

pg_catalog.pg_stat_io

Column	Type
backend_type	text
object	**text**
context	**text**

- relation
- temp relation

- normal
- vacuum
- bulkread
- bulkwrite
- temporary relation

pg_catalog.pg_stat_io

Column	Type
backend_type	text
object	text
context	text
op_bytes	**bigint**

IO Operation Size (BLOCKSIZE = 8192)

pg_catalog.pg_stat_io

Column	Type
backend_type	text
object	text
context	text
reads	**bigint**
writes	**bigint**
writebacks	**bigint**
writeback_time	**double precision**
extends	**bigint**
extend_time	**double precision**
hits	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1s - 15\,mins)$$

$$\text{cache}_{\text{hit\_ratio}} = \frac{\Delta hits}{\Delta hits + \Delta reads}$$

$$\text{cache}_{\text{pending\_writes}} = \Delta\text{writes} - \Delta writebacks$$

$$\text{avg}_{\text{writeback\_time}} = \frac{\Delta writeback\_time}{\Delta writebacks}$$

$$\text{avg}_{\text{extend\_time}} = \frac{\Delta extend\_time}{\Delta extends}$$

DataBene

pg_catalog.pg_stat_io

Column	Type
backend_type	text
object	text
context	text
evictions	**bigint**
reuses	**bigint**
fsyncs	**bigint**
fsync_time	**double precision**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \in (1s - 15\,mins)$$

reuses is related to BAS Ring Buffers

$$\text{avg}_{\text{fsync\_time}} = \frac{\Delta fsync\_time}{\Delta fsyncs} \leq \sim 100ms$$

DataBene

pg_catalog.pg_stat_wal

Column	Type
wal_records	bigint
wal_fpi	bigint
wal_bytes	numeric
wal_buffers_full	bigint
wal_write	bigint
wal_sync	bigint
wal_write_time	double precision
wal_sync_time	double precision
stats_reset	timestamp with time zone

pg_catalog.pg_stat_wal

Column	Type
wal_records	bigint
wal_fpi	bigint
wal_bytes	numeric
wal_buffers_full	bigint
wal_write	bigint
wal_write_time	double precision
wal_sync	bigint
wal_sync_time	double precision
stats_reset	timestamp with time zone

$$rate_x = \frac{\Delta x}{\Delta T}\ ; \Delta T\ \in (\mathbf{1s} - 5\ mins)$$

$$\Delta \text{Volume}_{\text{fpi}} = \Delta \text{wal\_fpi} * wal\_block\_size$$

$$\Delta \text{Volume}_{\text{rec}} = \Delta \text{wal\_bytes} - \Delta Volumefpi$$

$$\text{avg}_{\text{wal\_write}} = \frac{\Delta wal\_write\_time}{\Delta wal\_write}$$

$$\text{avg}_{\text{wal\_sync}} = \frac{\Delta wal\_sync\_time}{\Delta wal\_sync} \leq 100\mu s \text{ (pg\_test\_sync)}$$

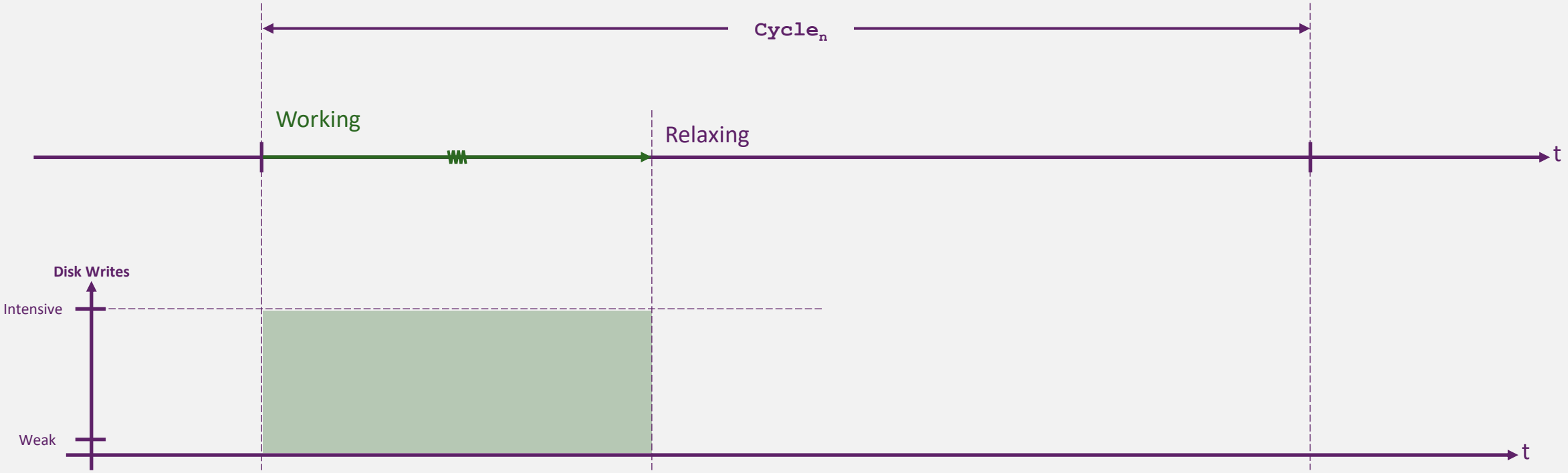
$$\text{ratio}_{\text{fullwrite}} = \frac{\Delta wal\_buffer\_full * wal\_buffers * wal\_block\_size}{\Delta wal\_bytes}$$

DataBene

Background Writer

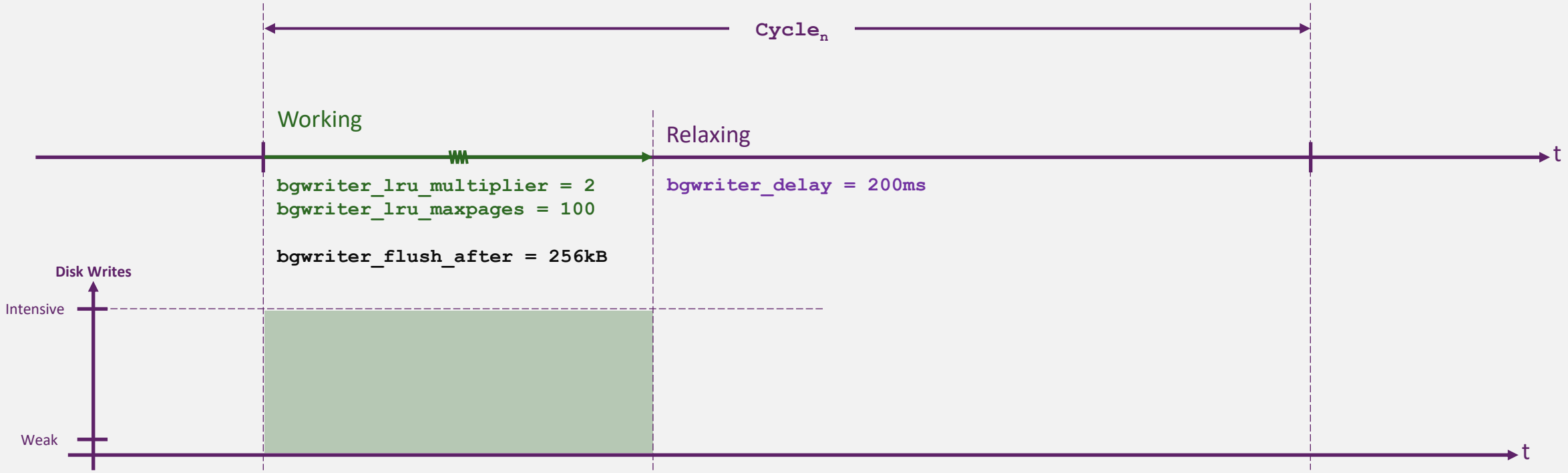
Structure – Usage – Processing

ACTIVITY and IO PRESSURE



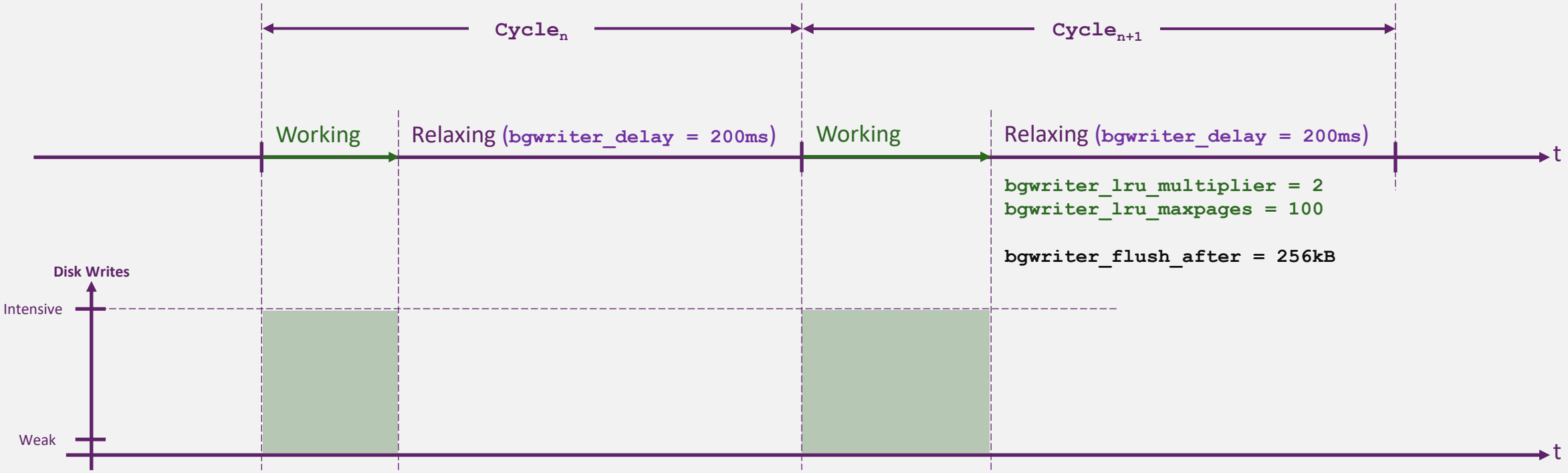
DataBene

ACTIVITY and IO PRESSURE



DataBene

ACTIVITY and IO PRESSURE



DataBene

pg_catalog.pg_stat_bgwriter

Column (**pg16**)	Type
checkpoints_timed	bigint
checkpoints_req	bigint
checkpoint_write_time	double precision
checkpoint_sync_time	double precision
buffers_checkpoint	bigint
buffers_clean	bigint
maxwritten_clean	bigint
buffers_backend	**bigint**
buffers_backend_fsync	**bigint**
buffers_alloc	**bigint**
stats_reset	timestamp with time zone

Column (**pg17**)	Type
buffers_clean	**bigint**
maxwritten_clean	**bigint**
buffers_alloc	**bigint**
stats_reset	**timestamp with time zone**

Column (**pg17**)	Type
num_timed	**bigint**
num_requested	**bigint**
restartpoints_timed	**bigint**
restartpoints_req	**bigint**
restartpoints_done	**bigint**
write_time	**double precision**
sync_time	**double precision**
buffers_written	**bigint**
stats_reset	**timestamp with time zone**

DataBene

pg_catalog.pg_stat_bgwriter

Column	Type
buffers_clean	**bigint**
maxwritten_clean	**bigint**
buffers_alloc	bigint
stats_reset	timestamp with time zone

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \; \in (\mathbf{1s} - 15\, mins)$$

- $maxwritten\_clean \times \boldsymbol{bgwriter\_lru\_maxpages} \leq buffers\_clean$
- pg16
  - Minimize pg_stat_bgwriter.backend_buffers
- Pg17
  - Minimize pg_stat_io.evictions for « client backend »

Checkpoint(er)

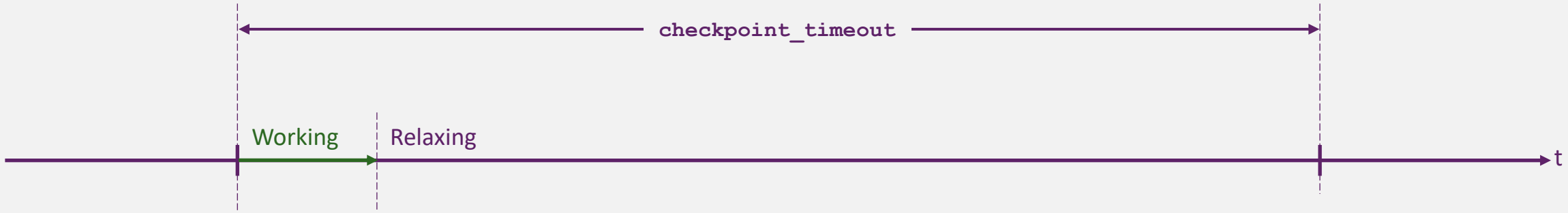
Understanding Performance Key Points

CHECKPOINT KEY CONCEPTS

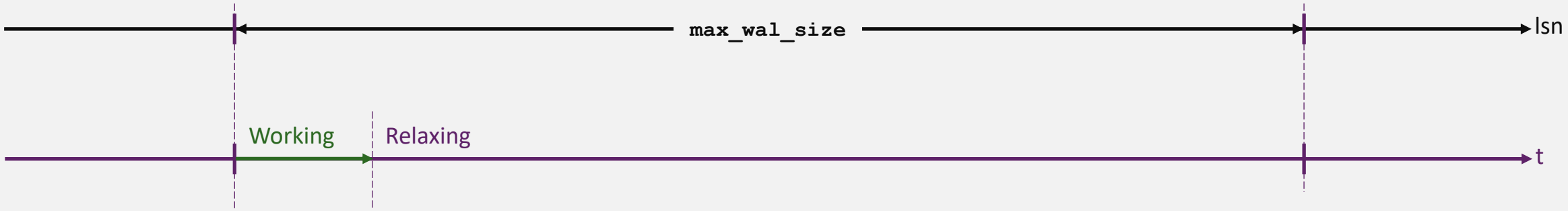
checkpoint_timeout - max_wal_size - checkpoint_completion_target
checkpoint_warning - min_wal_size - checkpoint_flush_after

CHECKPOINT TRIGGERING

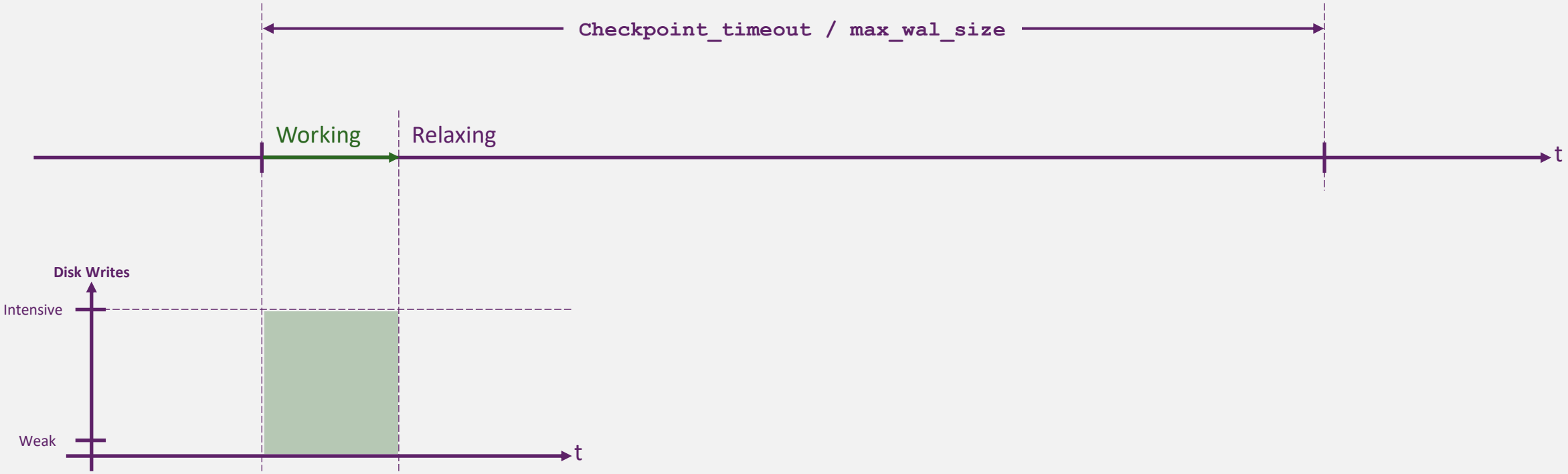
TEMPORAL TRIGGERING



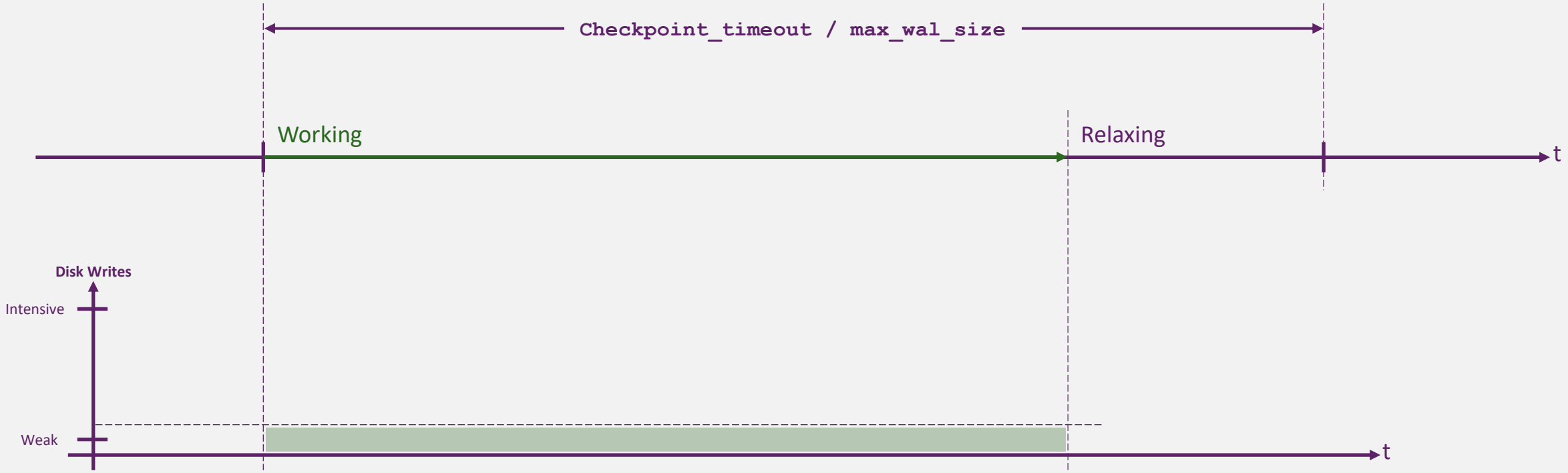
SPATIAL TRIGGERING



IO PRESSURE (intensive)



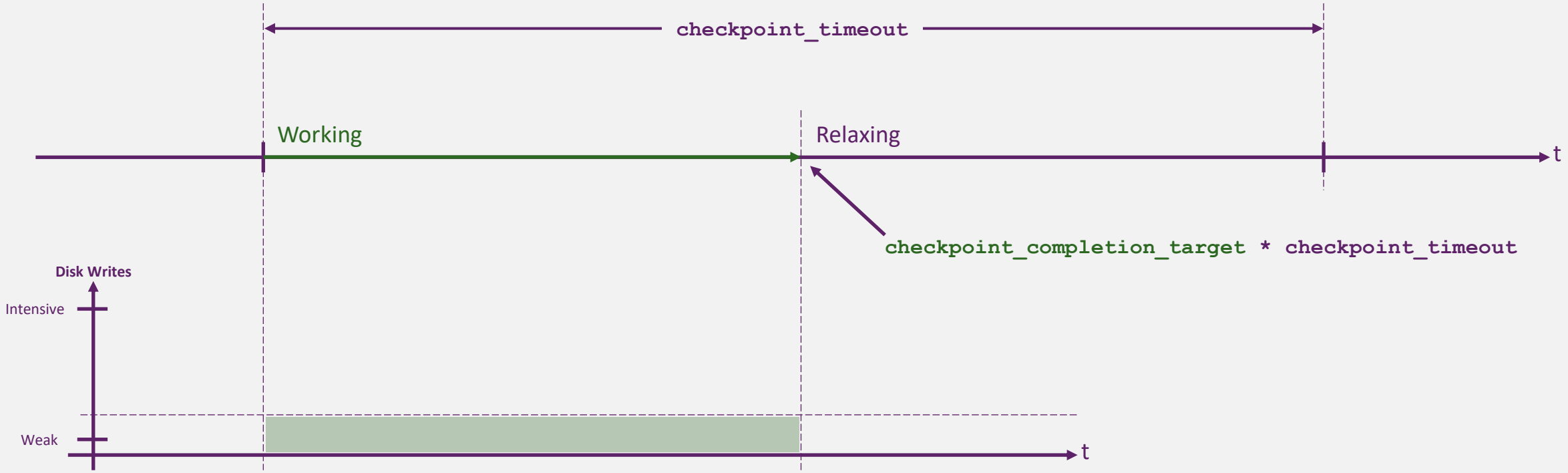
IO PRESSURE (spread)



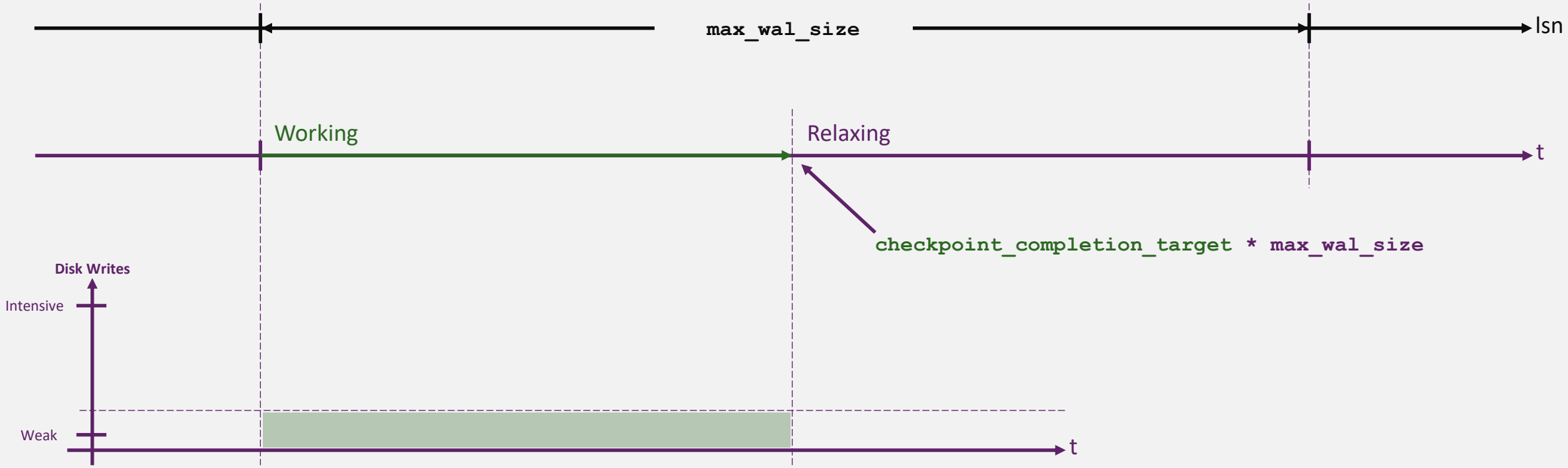
DataBene

IO PRESSURE SPREADING FEEDBACK LOOP

TEMPORAL SPREADING FEEDBACK LOOP



SPATIAL SPREADING FEEDBACK LOOP

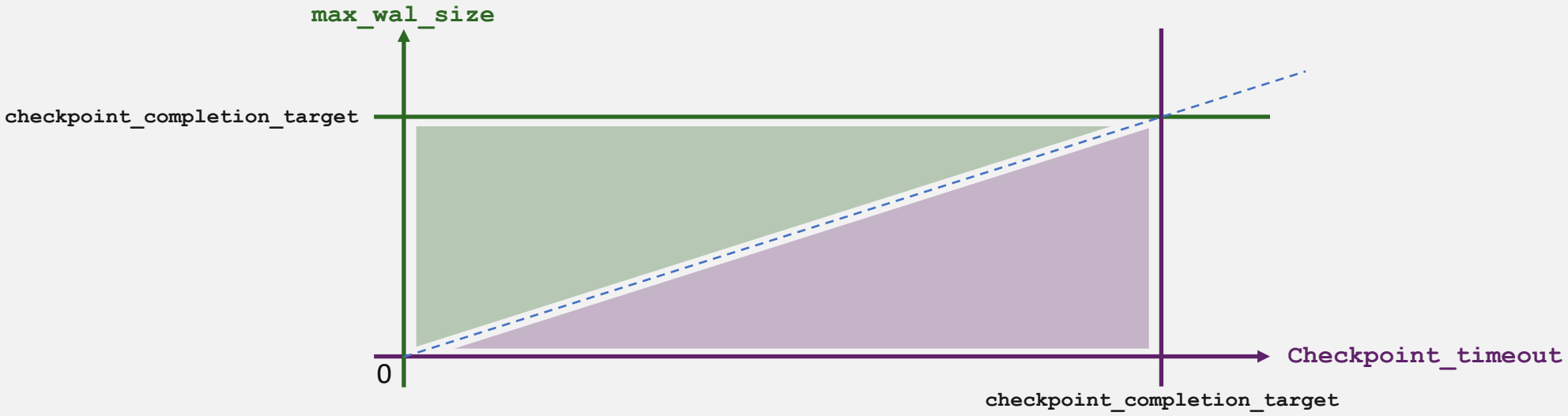


DataBene

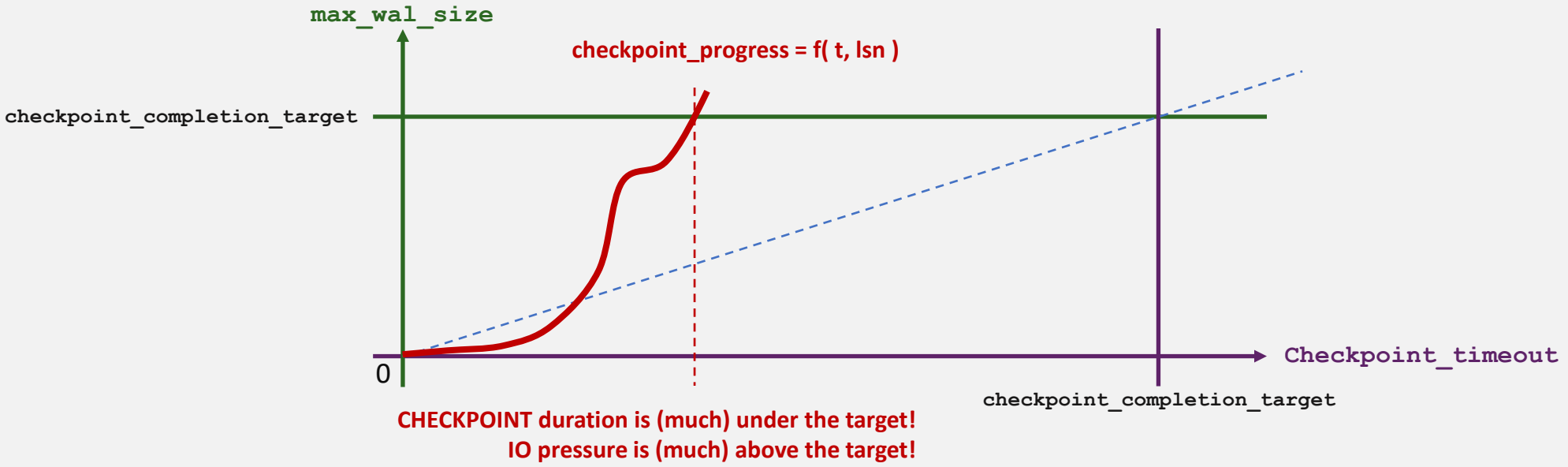
FEEDBACK LOOP

```
while num_processed < num_to_scan

   WriteDirtyPageToDisk()
   num_processed++

   checkpoint_progress = num_processed / num_to_scan
   checkpoint_progress = checkpoint_progress * checkpoint_completion_target

   wal_progress = (current LSN - start LSN) / max_wal_size
   if checkpoint_progress < wal_progress then continue

   time_progress = (now - start_time) / checkpoint_timeout
   if checkpoint_progress < time_progress then continue

   sleep( CheckpointWriteDelay = 100ms )
```

BIDIMENSIONNAL FEEDBACK LOOP DIAGRAM



BIDIMENSIONNAL FEEDBACK LOOP – EXAMPLE



pg_catalog.pg_stat_checkpointer

Column	Type
num_timed	bigint
num_requested	bigint
restartpoints_timed	bigint
restartpoints_req	bigint
restartpoints_done	bigint
write_time	**double precision**
sync_time	**double precision**
buffers_written	**bigint**

$$rate_x = \frac{\Delta x}{\Delta T} \; ; \Delta T \; \in (1\ mins)$$

- sync_time must be kept low

$$\Delta sync\_time \leq \sim 100ms$$

- Check IO spreading

$$\text{avg}_{\text{iops}} = \frac{\Delta buffe\_written}{\Delta writte\_time} \leq IOPressure$$

- Check completion_target is respected

$$\frac{avg_w(write\_time)}{checkpoint\_timeout} = checkpoint\_completion\_target$$

DataBene

Logging

- Do not forgot to logs some events
  - logging_collector = on
  - log_destination = ‘stderr,csv’ (csv can be injected to a database and then queried)
  - log_temp_files = 1
  - log_lock_waits = on
  - log_line_prefix = ‘%m [%p] %a %u@%d %a’ as a minimum
  - log_checkpoints = on
  - log_autovacuum_min_duration = 60000 (ms)
  - log_min_duration_statement = the value which make sense for you
- Mine your log every day
  - pgbadger can help with that

Hey, I did not cover the following yet:

- pg_stat_slru
- pg_stat_replication
  - SELECT pg_wal_lsn_diff(pg_current_wal_lsn(), sent_lsn) AS sent_lag_bytes from pg_stat_replication;
- pg_class size catalog tables
  - SELECT relname,relpages,pg_size_pretty(8192::bigint*relpages) FROM pg_class WHERE relnamespace= 'pg_catalog'::regnamespace ORDER BY 2 DESC;
- pg_index is not valid (pg_index.indisvalid)
  - select * from pg_index where not indisvalid;
- pg_stat_archiver
  - SELECT now() - COALESCE(last_failed_time, '1970-01-01'::date) <= interval '5 mins' AS recent_failure FROM pg_stat_archiver ;
- pg_stat_activity (wait_event_type, wait_event, state, long running query/snapshot) with pg_locks (granted ?)

Linux

procFS, cgroups, perf

ProcFS - CPU

```
$ cat /proc/stat
#    user     nice    system  idle       iowait irq softirq steal guest guest_nice
cpu  31597599 2158402 9278540 1083188543 1623186 0  303365  0     0     0
cpu0  6700402  488305 2278642  272134488  392242 0   19813  0     0     0
cpu1  7706591  563056 2072923  271348166  413324 0   18152  0     0     0
cpu2  8151274  502605 2405923  270298148  422380 0  173093  0     0     0
cpu3  9039330  604434 2521050  269407739  395239 0   92305  0     0     0
```

$$CPU\ Utilization = \frac{user\ +\ nice\ +\ system\ +\ stealth\ +\ guest}{user\ +\ nice\ +\ system\ +\ stealth\ +\ guest\ +\ idle\ [+\ iowait]} < 0.85$$

DataBene

ProcFS - CPU

```
$ cat /proc/stat
…
intr 697368832 48 0 0 0 0 0 0 1 0 0 0 0 0 0 0 0 0 0 6 0 0 0 25 23437850 15669310 2 11670157 0 0 0 … 0 0 0
ctxt 426424223
btime 1726754569
processes 1486375
procs_running 1
procs_blocked 0
#                    TIMER      NET_TX    NET_RX    BLOCK       TASKLET SCHED      HRTIMER RCU
softirq 837985299 6 108387462 15434898 35102963 11475396 0 4390    475399864 48060   192132260
```

DataBene

ProcFS - Network

```
$ cat /proc/net/dev
Inter-|   Receive
 face |          bytes  packets errs drop fifo frame compressed multicast
    lo:           19449      119    0    0    0     0          0         0
  eno0:    10296287653 26877099    0    0    0     0          0   1416671
docker0:     684107064 12947192    0    0    0     0          0         0
veth303a4b4: 250762391  3804884    0    0    0     0          0         0
veth37d40dd: 251701833  3817537    0    0    0     0          0         0
veth4c6976c:  33518004   489729    0    0    0     0          0         0
veth75f0dc9: 252377309  3828110    0    0    0     0          0         0
veth49d99b2:  24114346   312563    0    0    0     0          0         0
veth4fd0c00:  23808783   309302    0    0    0     0          0         0
veth2a4269b:  23354884   302527    0    0    0     0          0         0
```

ProcFS - Disk

```
$ cat /proc/diskstats
8 0 sda 1274558 151495 85887753 4817963 9019869 7340199 238013064 28267395 0 24084484 48700750 0 0 0 0 1615784 15615391
8 1 sda1 236 0 11618 368 0 0 0 0 0 660 368 0 0 0 0 0 0
8 2 sda2 437 211 40076 1732 257 114 85912 2763 0 3452 4495 0 0 0 0 0 0
```

ProcFS - Diskstat

```
==  ====================================
 1  major number
 2  minor number
 3  device name
 4  reads completed successfully
 5  reads merged
 6  sectors read
 7  time spent reading (ms)
 8  writes completed
 9  writes merged
10  sectors written
11  time spent writing (ms)
12  I/Os currently in progress
13  time spent doing I/Os (ms)
14  weighted time spent doing I/Os (ms)
==  ====================================
```

Kernel 4.18+ appends four more fields for discard

tracking putting the total at 18:

```
==  ====================================
15  discards completed successfully
16  discards merged
17  sectors discarded
18  time spent discarding
==  ====================================
```

Kernel 5.5+ appends two more fields for flush requests:

```
==  ====================================
19  flush requests completed successfully
20  time spent flushing
==  ====================================
```

DataBene

ProcFS – Diskstat – Service Time

$$\frac{\delta[io_{ticks}]}{(\delta[read_{IOs} + read_{merges} + write_{IOs} + write_{merges}])}$$

ServiceTime is given in milliseconds.

Rewritten with field numbers:

$$\frac{\delta[f13]}{(\delta[f4 + f5 + f8 + f9])}$$

ProcFS – Diskstat – Queue Time

$$\frac{\delta[time_{in\_queue}]}{(\delta[read_{IOs} + read_{merges} + write_{IOs} + write_{merges}] + \delta[in\_flight])} - servicetime$$

QueueTime is given in milliseconds.

Rewritten with field numbers:

$$\frac{\delta[f14]}{(\delta[f4 + f5 + f8 + f9] + \delta[f12])} - \frac{\delta[f13]}{(\delta[f4 + f5 + f8 + f9])}$$

cgroups

- cgroups are used everywhere and automatically
- `/sys/fs/cgroup`
  - `blkio`
  - `cpu | cpu,cpuacct | cpuacct | cpuset`
  - `devices`
  - `freezer`
  - `hugetlb | memory`
  - `misc`
  - `net_cls | net_cls,net_prio | net_prio`
  - `perf_event`
  - `pids`
  - `rdma`
  - `systemd`
  - `unified`

CPU consumption of my PostgreSQL service

```
$ cat /sys/fs/cgroup/cpuacct/system.slice/ \
      system-postgresql.slice/cpuacct.usage_all
cpu user system
0 1841620000000 38442320000000
1 700600000000 13517880000000
2 1627472000000 33766160000000
3 1489560000000 30669040000000
```

Unit: 10ms (1/100th s) => direct percent of CPU consumption.

Compute rate as usual $rate_{cpu\%} = \frac{\Delta user + \Delta system}{\Delta T}$

DataBene

IODisk comsumption for my docker

```
cat /sys/fs/cgroup/blkio/docker/0f040c…8c7bc0e3\
    /blkio.throttle.io_service_bytes
8:0 Read 3897999360
8:0 Write 2056773632
8:0 Sync 4718170112
8:0 Async 1236602880
8:0 Discard 0
8:0 Total 5954772992
Total 5954772992
```

Perf

- Very interesting tool you have to try at least once.
- Only for troubleshooting because it is kind of manual.
- Usually perf.data file has to be processed on the recorded server.
  - Processing data file can be consuming

Perf – System Probe

```
sudo perf list
```

Gives a lot of available hardware and system metrics!

Perf - Example

```
sudo \
perf stat -e
block:block_rq_*,syscalls:sys_enter_write,syscalls:sys_enter_fsync \
-a -r 5 -- psql -q -p 5434 -U postgres postgres \
-c "drop table if exists x; create table x as select a FROM
generate_series(1,1000000) a;";
```

Perf – Completely Fair Scheduler

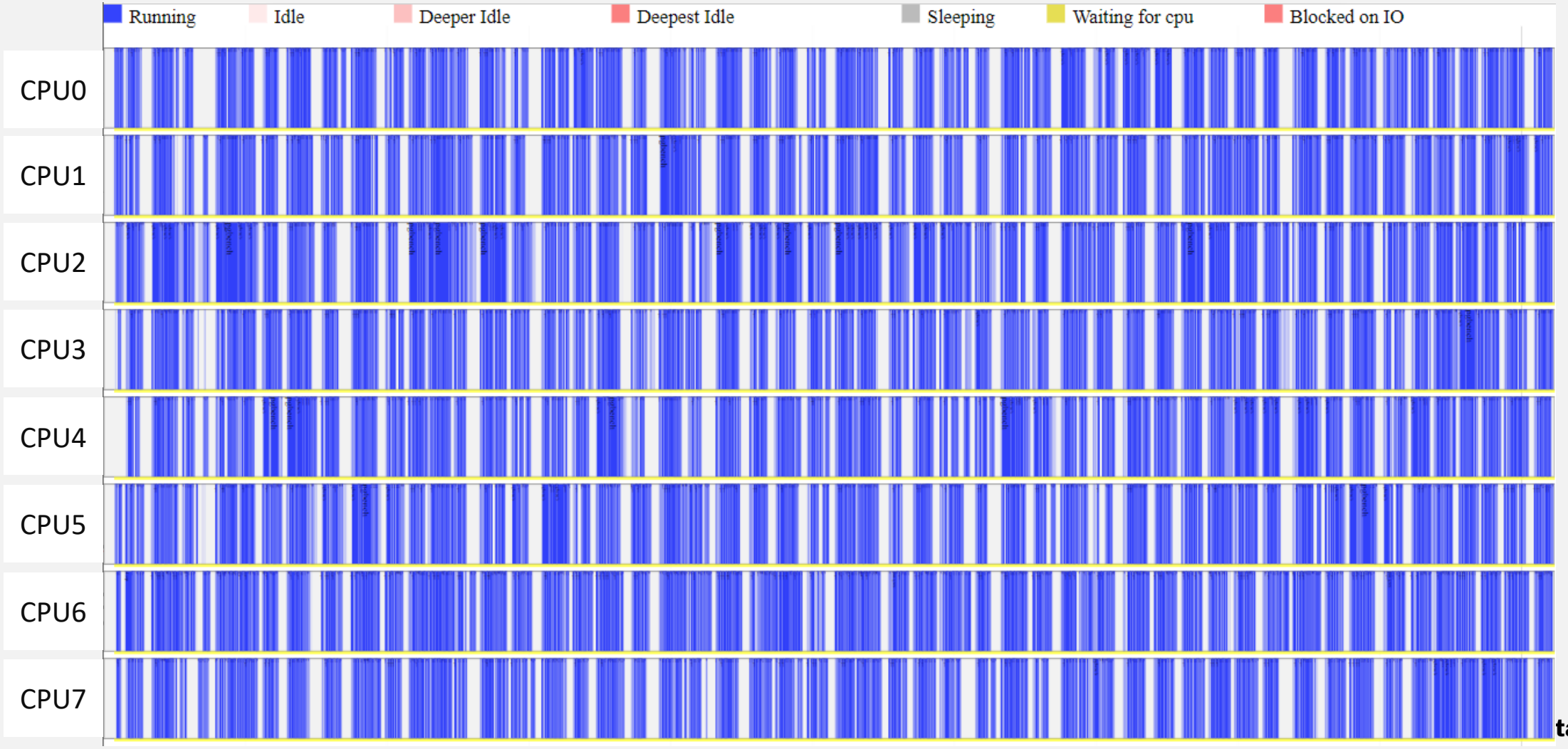
To investigate

```
perf sched record sleep 0.1
perf sched latency -s max
perf sched map
perf timechart
```

[Announce] 'perf sched': Utility to capture, measure and analyze scheduler latencies and behavior

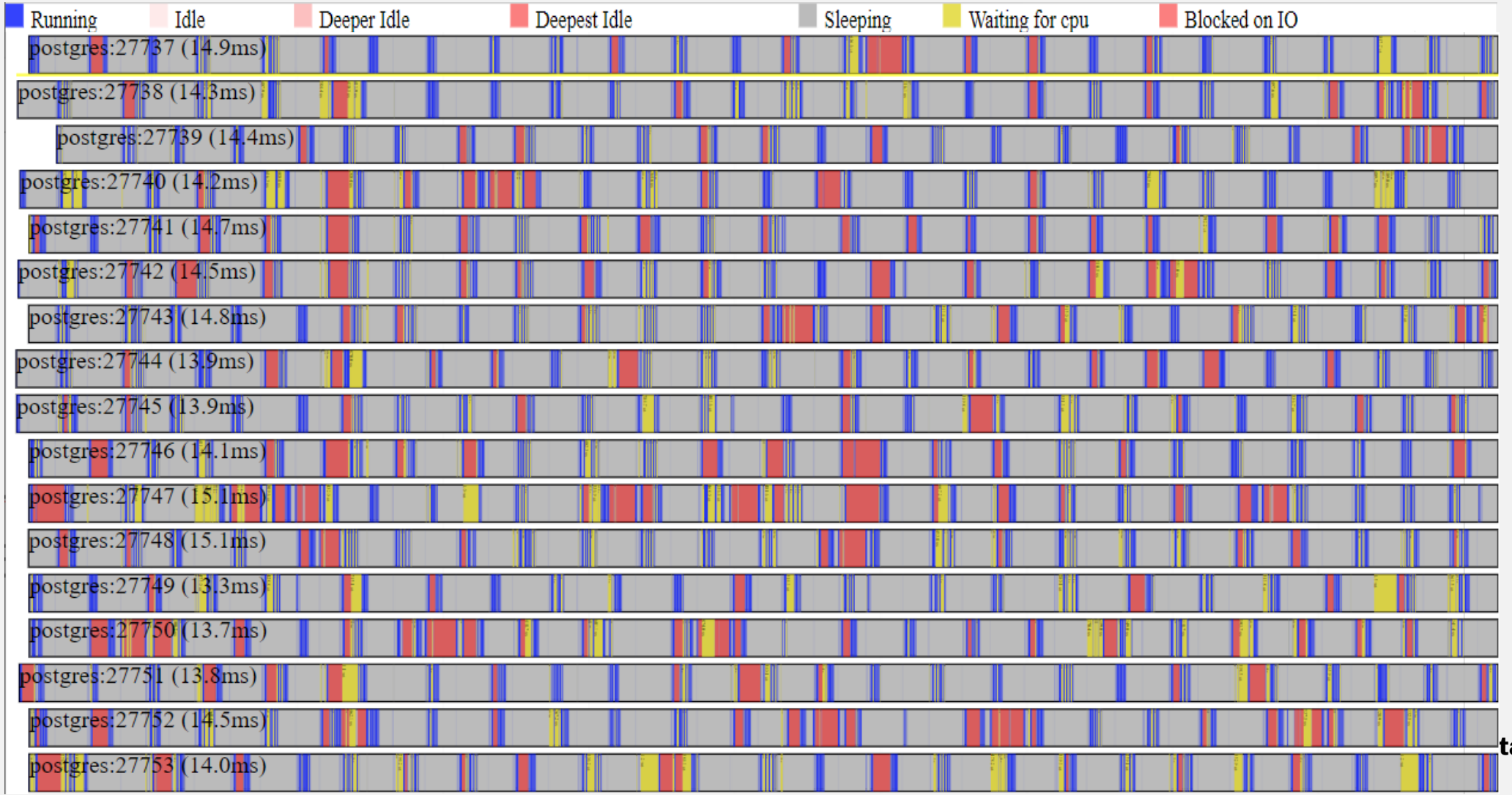
https://lwn.net/Articles/353295/

Linux – Completely Fair Scheduler

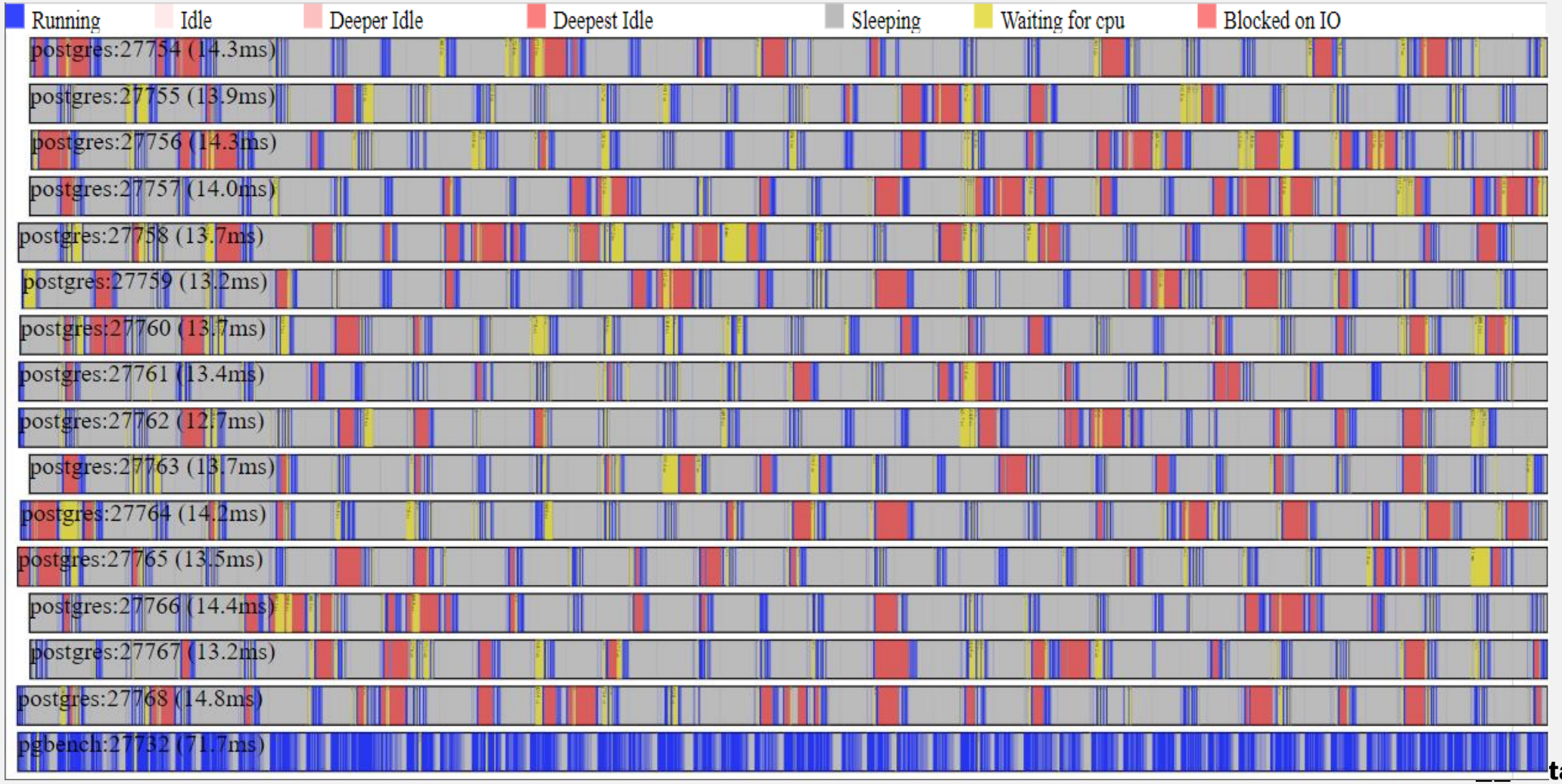


taBene

Linux – Completely Fair Scheduler



Linux – Completely Fair Scheduler

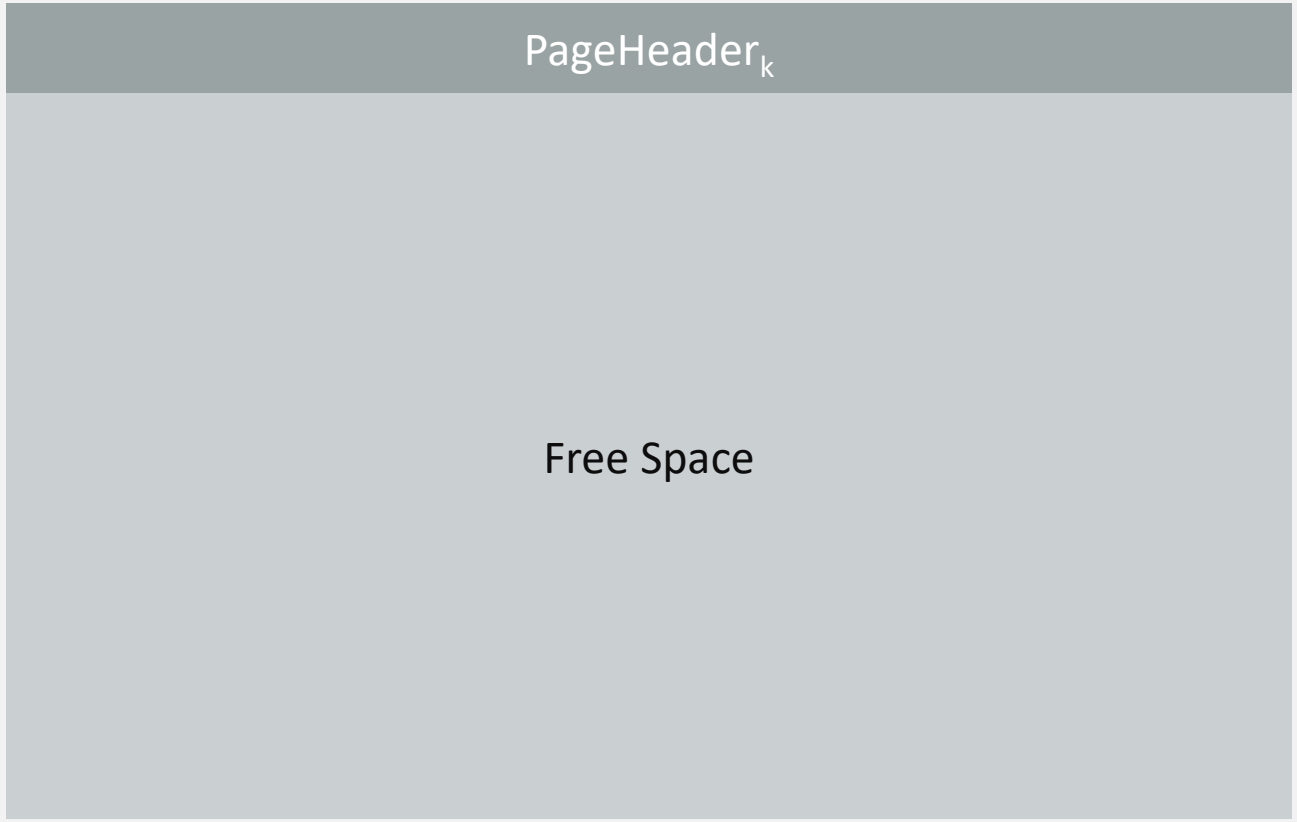


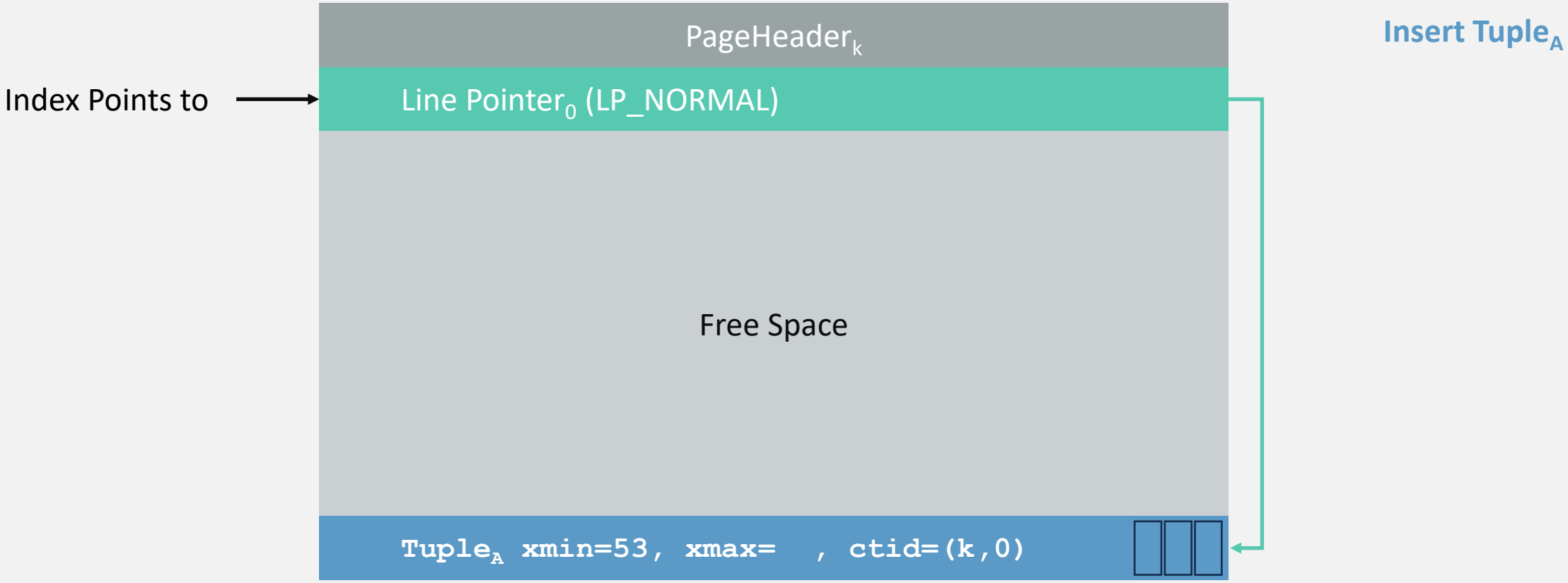
taBene

Thank You

Without Heap Only Tuples

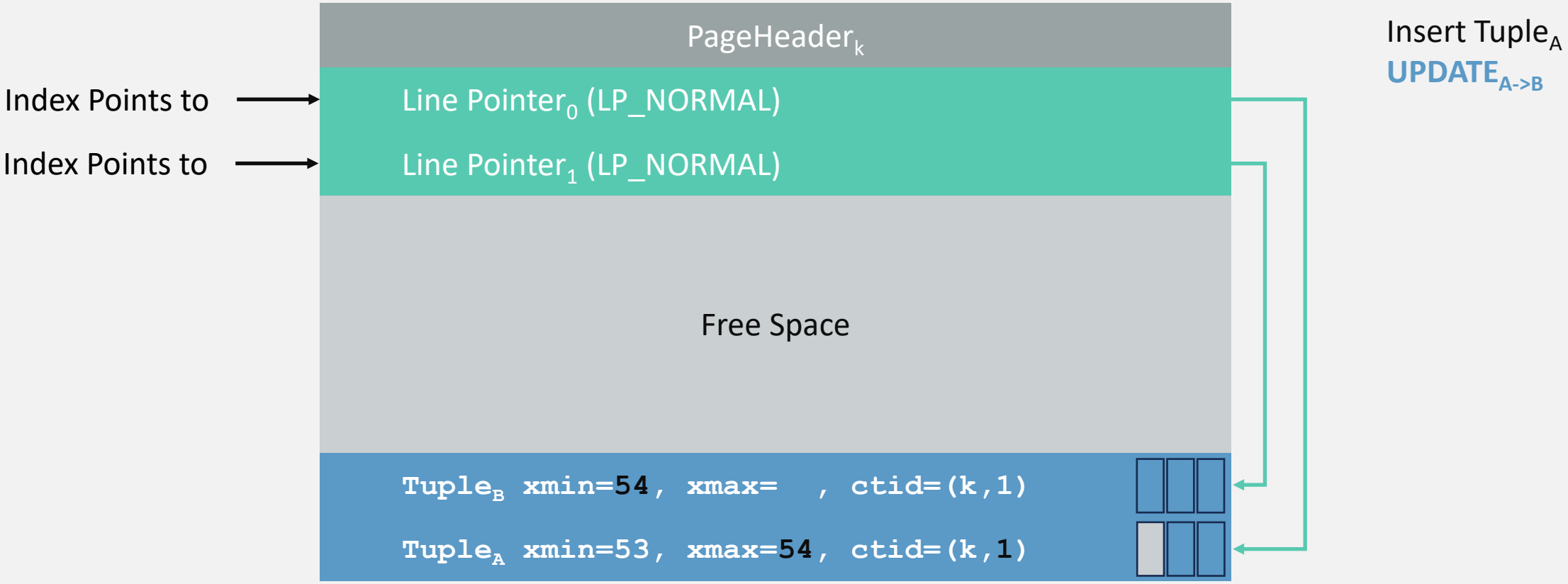
Structure – Usage – Processing


PageHeader$_k$
Free Space



LP_NORMAL | LP_REDIRECT | LP_DEAD | LP_UNUSED | Unvisible

DataBene



LP_NORMAL | LP_REDIRECT | LP_DEAD | LP_UNUSED | Unvisible

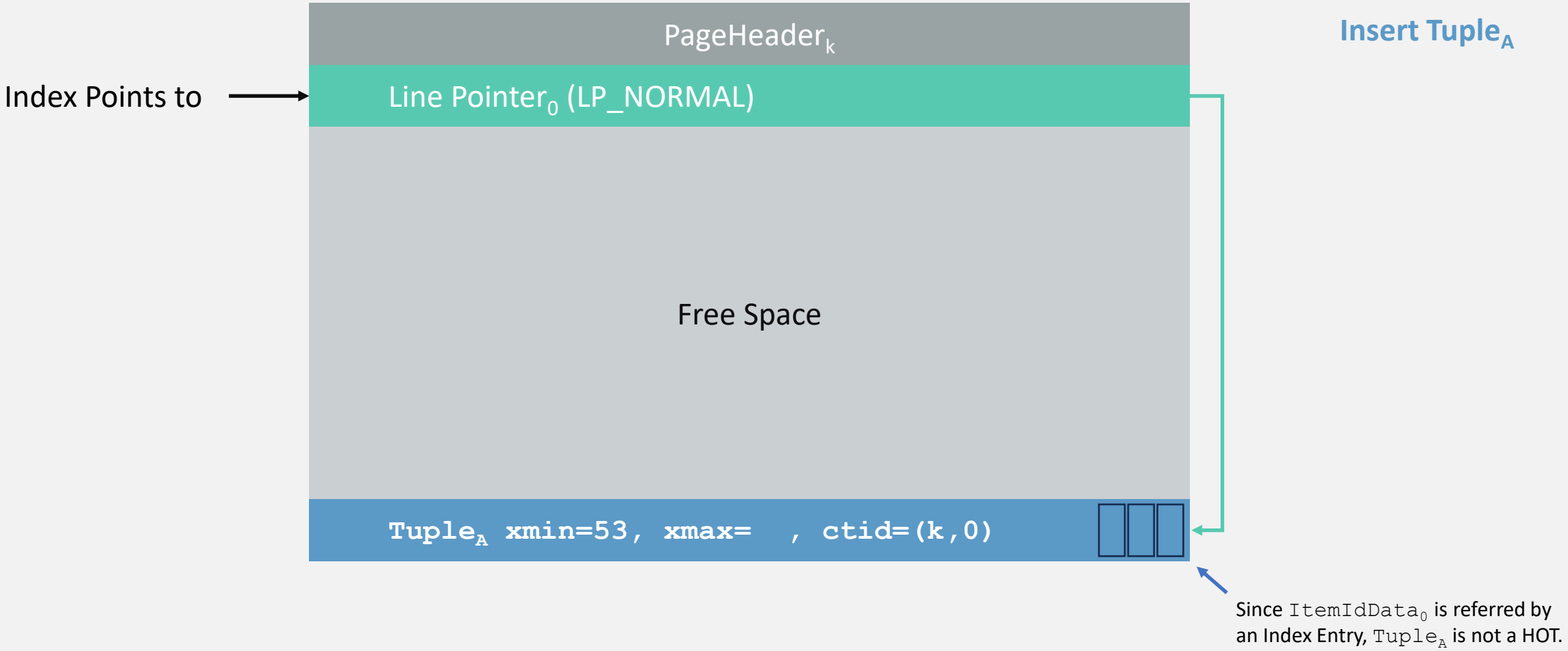
DataBene

Heap Only Tuples

Structure – Usage – Processing

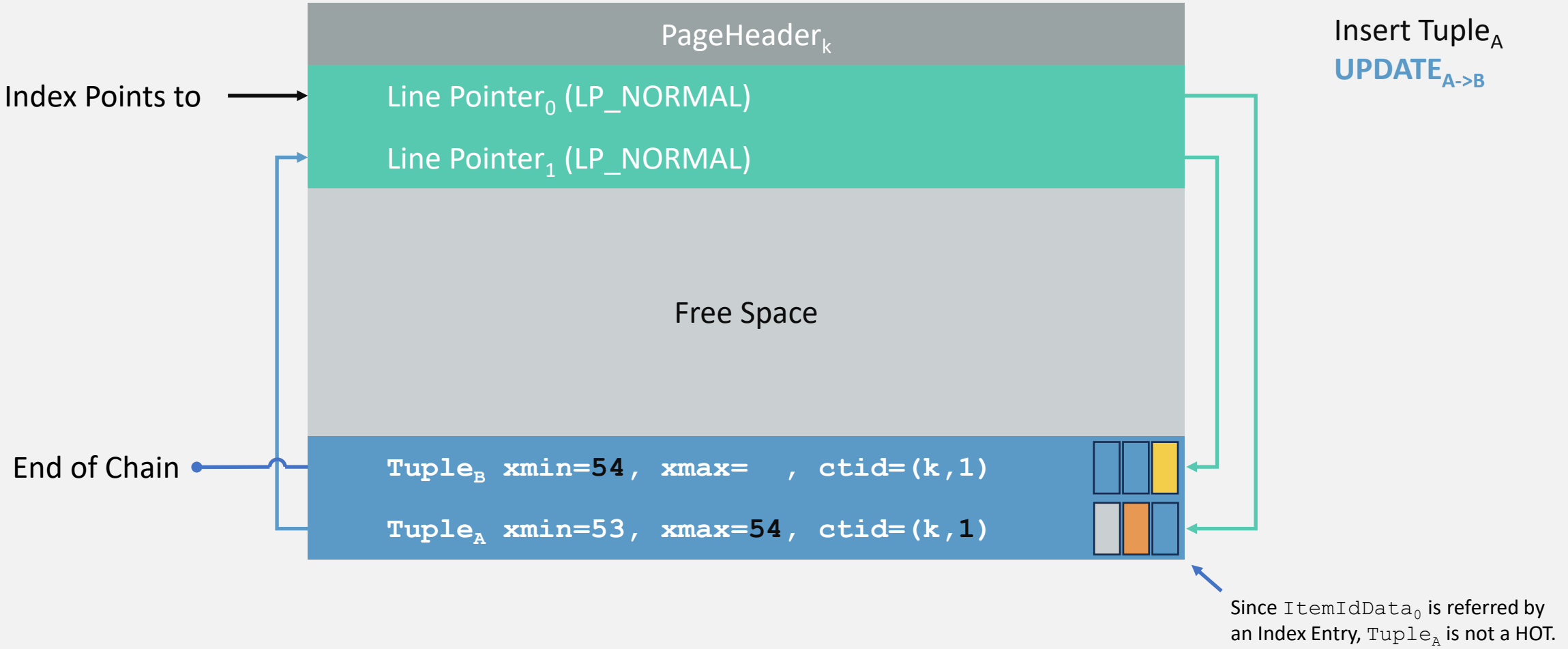
LP_NORMAL | LP_REDIRECT | LP_DEAD | LP_UNUSED | Unvisible | HEAP_ONLY_UPDATE | HEAP_ONLY_TUPLE

DataBene



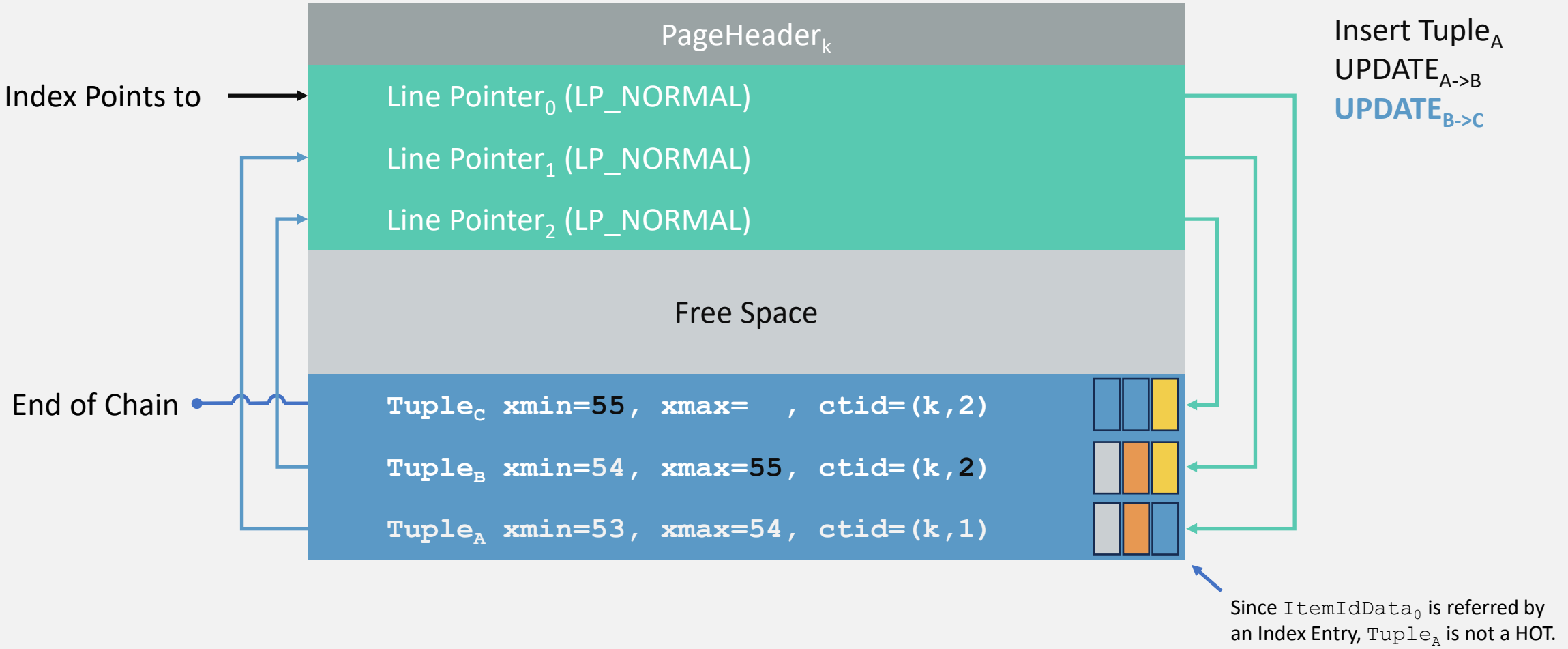
LP_NORMAL | LP_REDIRECT | LP_DEAD | LP_UNUSED | Unvisible | HEAP_ONLY_UPDATE | HEAP_ONLY_TUPLE

DataBene



LP_NORMAL | LP_REDIRECT | LP_DEAD | LP_UNUSED | Unvisible | HEAP_ONLY_UPDATE | HEAP_ONLY_TUPLE

DataBene



LP_NORMAL | LP_REDIRECT | LP_DEAD | LP_UNUSED | Unvisible | HEAP_ONLY_UPDATE | HEAP_ONLY_TUPLE

DataBene