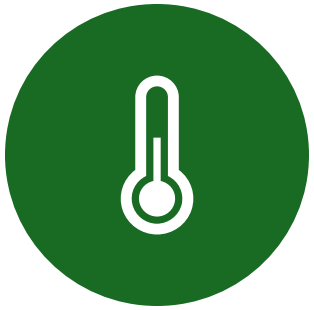
Improved Freezing In Vacuum

Melanie Plageman

Microsoft

Agenda

WHAT IS FREEZING AND WHY DO WE NEED IT



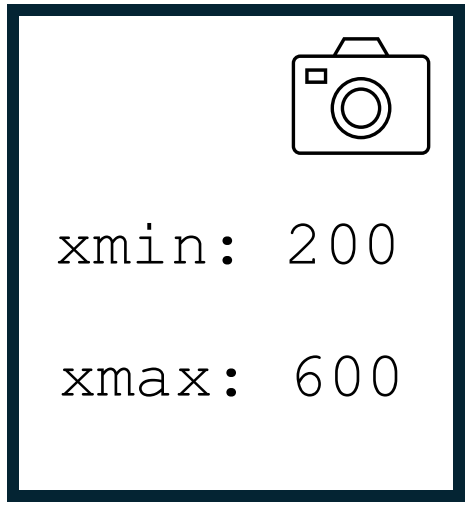
WHEN DO WE FREEZE NOW

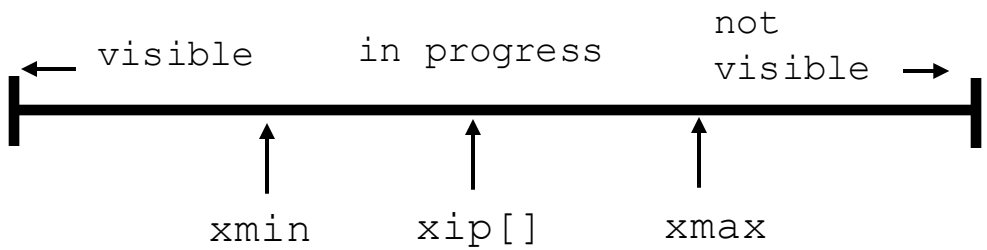
WHY IS IT HARD TO DO BETTER

WHAT DID WE DO IN 18

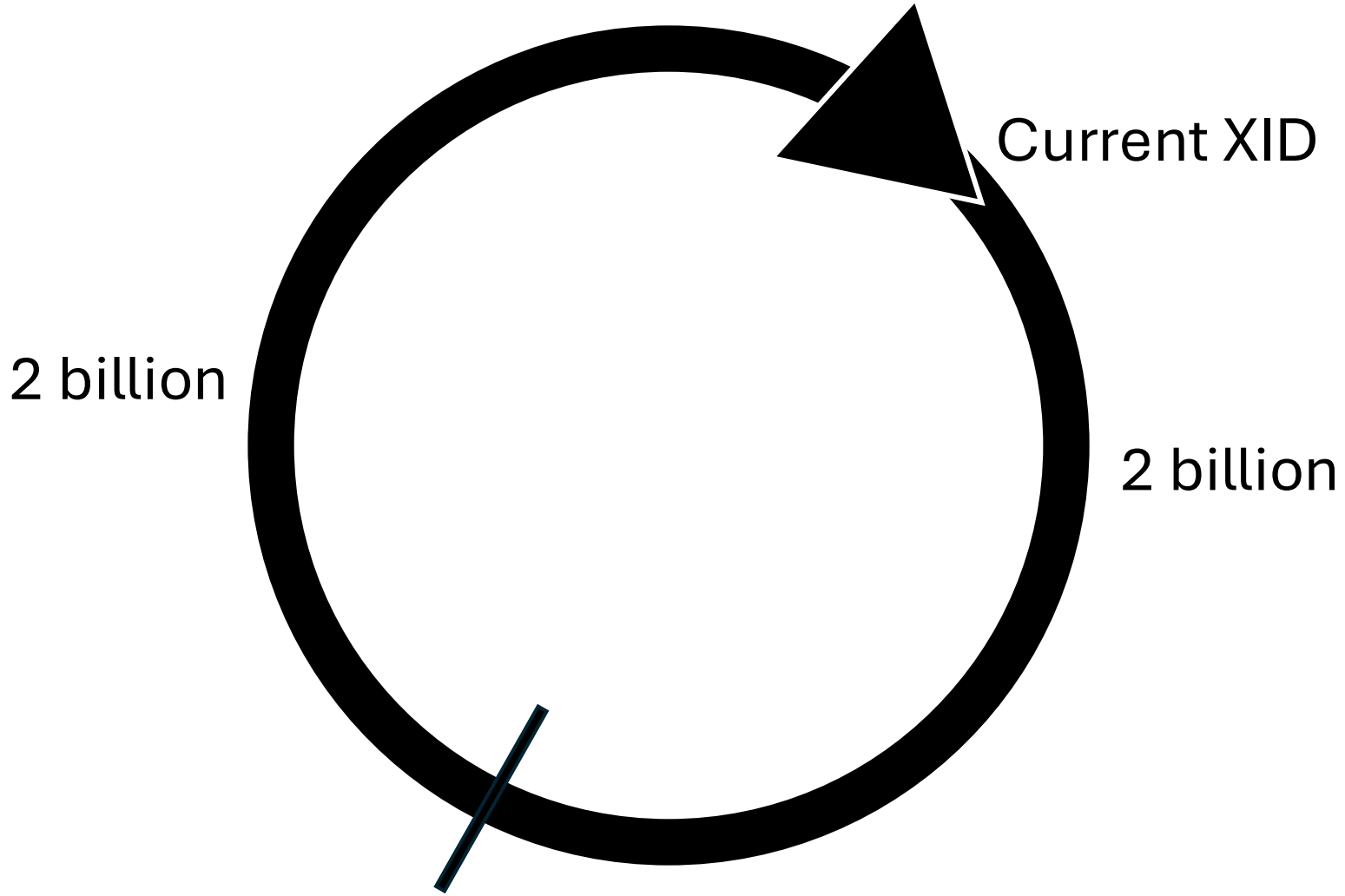
Multi-Version Concurrency Control

xmin	xmax	status	value	
20	X	R	lorem	Not visible, Live (running)
22	100	C	ipsum	Visible, Dead
50	700	C	dolor	Visible, Live; Not visible, Dead
100	X	C	sit	Visible, Live
300	X	C	amet	Check XIP

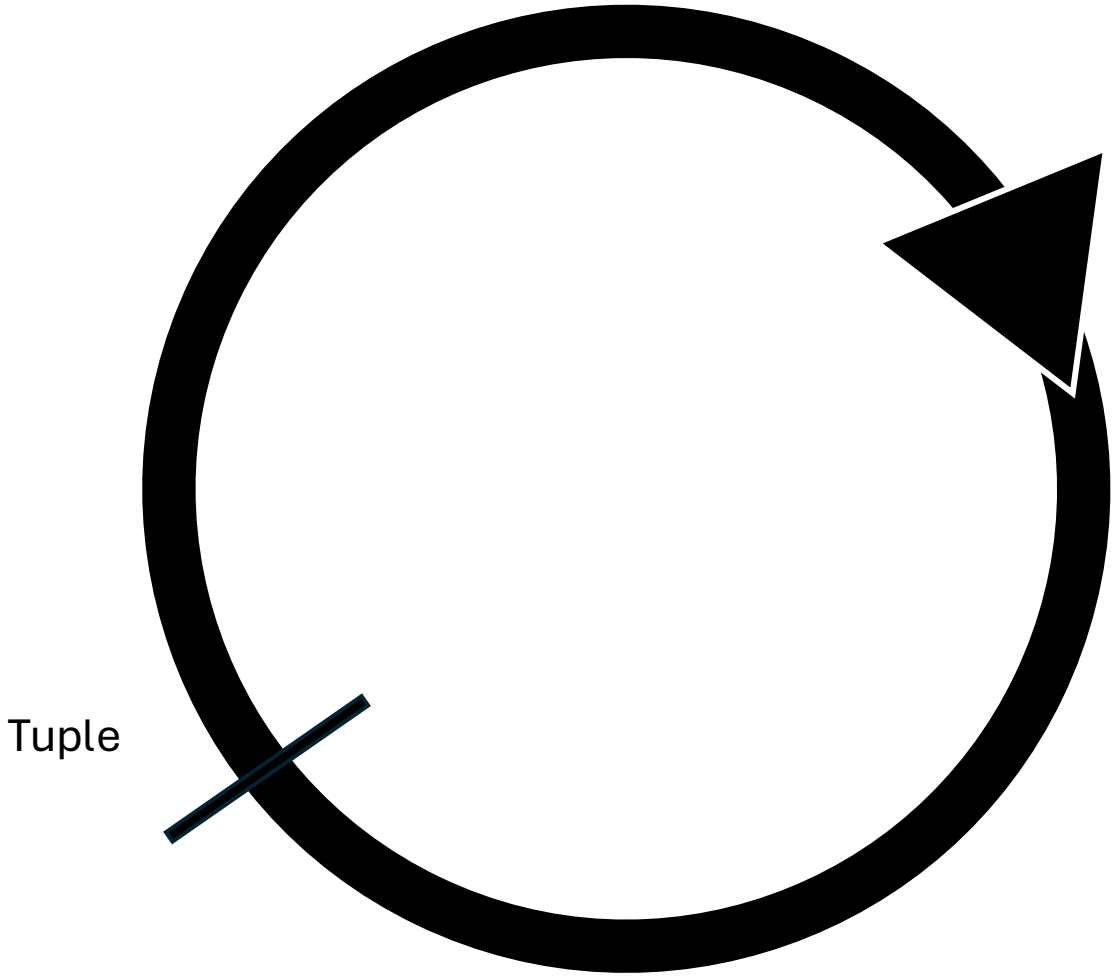




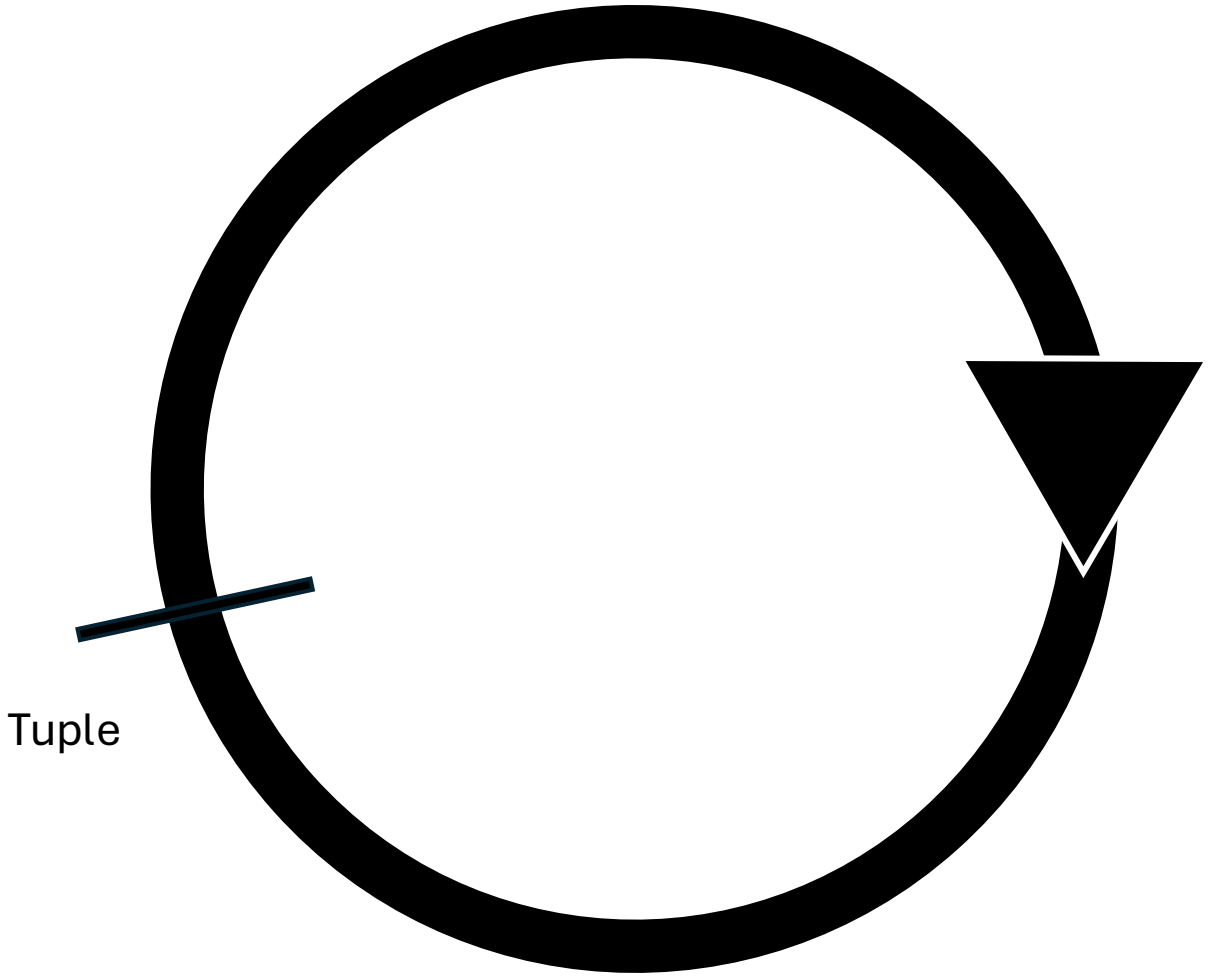
Wraparound



Wraparound



Wraparound



Wraparound  Read-only database

Freezing

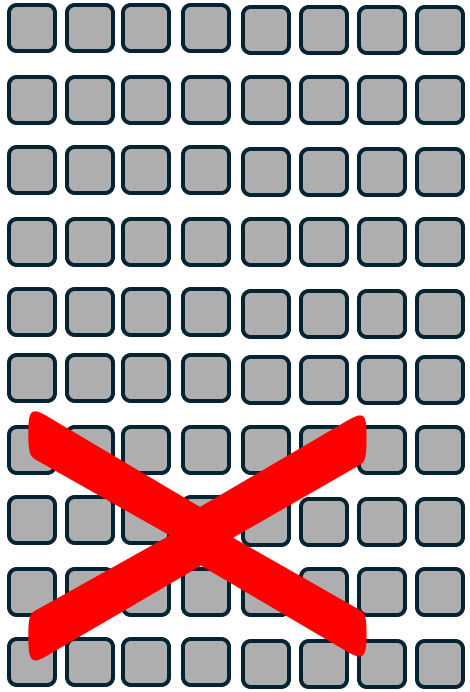
xmin	xmax	status	value	
20	X	R	lorem	Not visible, Live (running)
22	100	C	ipsum	Visible, Dead
50	700	C	dolor	Visible, Live; Not visible, Dead
❄	❄	C	sit	Visible, Live
300	X	C	amet	Check XIP

How do we know if our database is in danger?

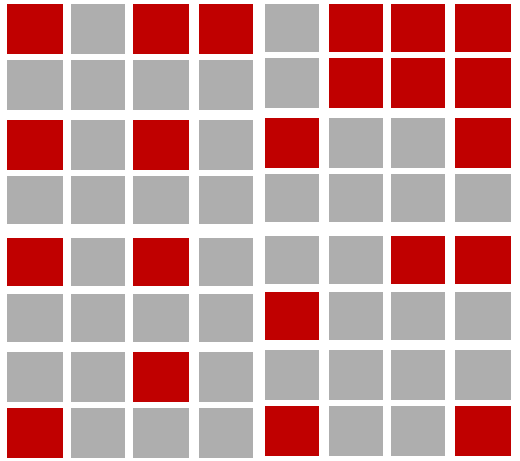
Query XID: 1,900,000,000

xmin	xmax	status	value
20	X	R	lorem
22	100	C	ipsum
50	700	C	dolor
100	X	C	sit
300	X	C	amet

relfrozenxid == 10



relfrozenxid == 1,000,000,000



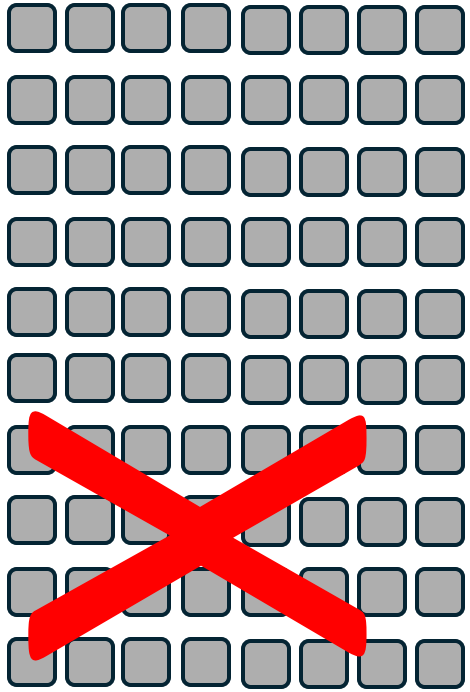
relfrozenxid == 1,200,000,000

Per table: `pg_class.relfrozenxid`

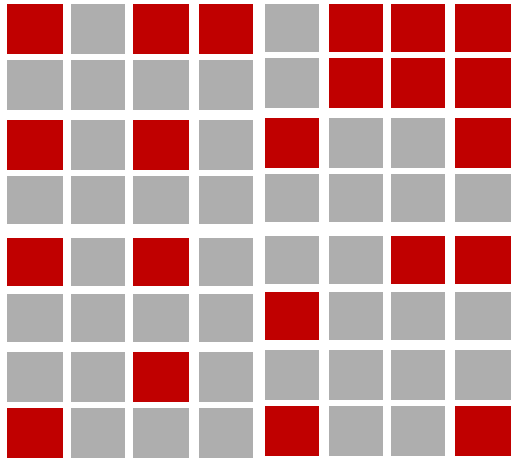
Query XID: 1,900,000,000

xmin	xmax	status	value
20	X	R	lorem
22	100	C	ipsum
50	700	C	dolor
100	X	C	sit
300	X	C	amet

relfrozenxid == 10



relfrozenxid == 1,000,000,000



relfrozenxid == 1,200,000,000

Who should freeze and when?

Emits WAL

Dirties pages

Vacuum well-positioned to freeze

Encounters tuples

Dirties pages

Emits WAL

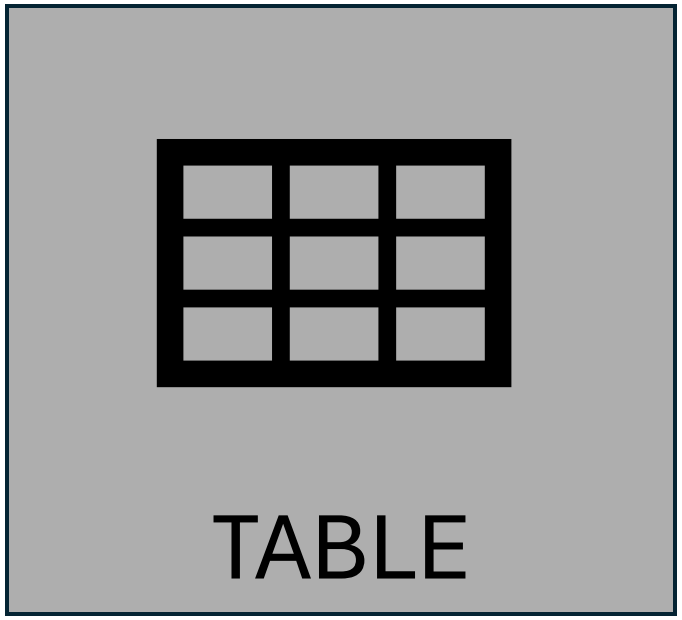
Freezing inconsistent with vacuum's mandate

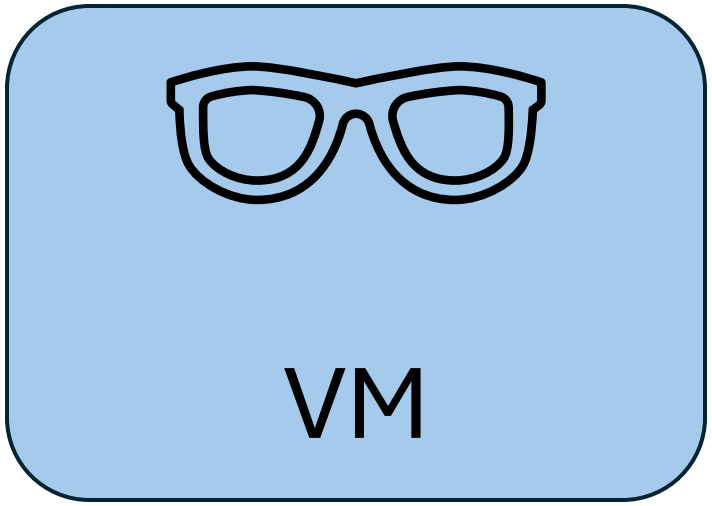
Triggered by insert and modification thresholds

Reads modified pages

Skips all-visible pages

Visibility map





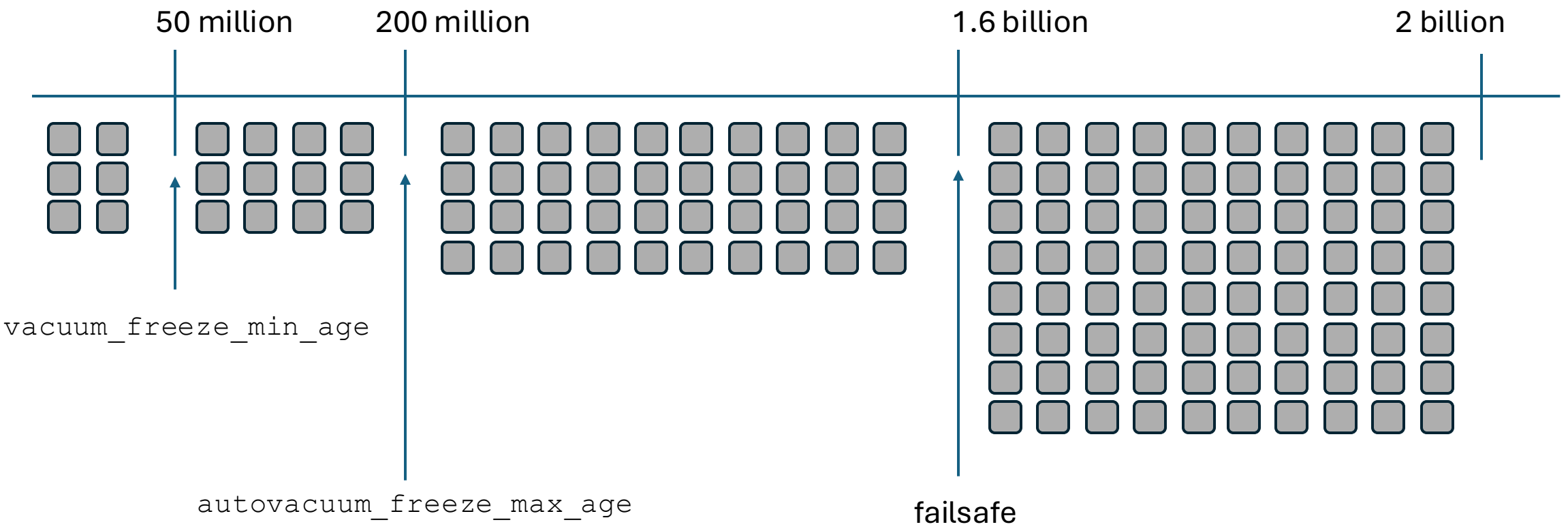
Forcing vacuum of all-visible pages

`relfrozenxid` < `autovacuum_freeze_max_age`

triggers anti-wraparound vacuum

- usually aggressive (controlled by vacuum_freeze_table_age)
- scans all all-visible pages

Freezing Timeline

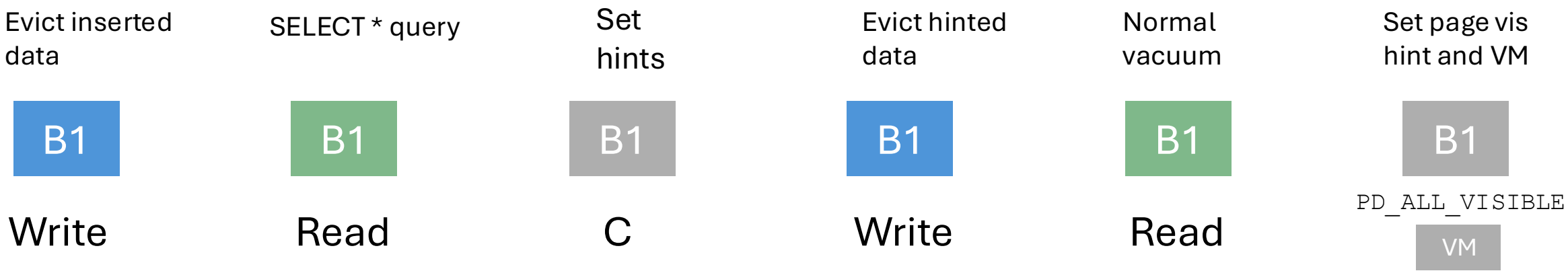


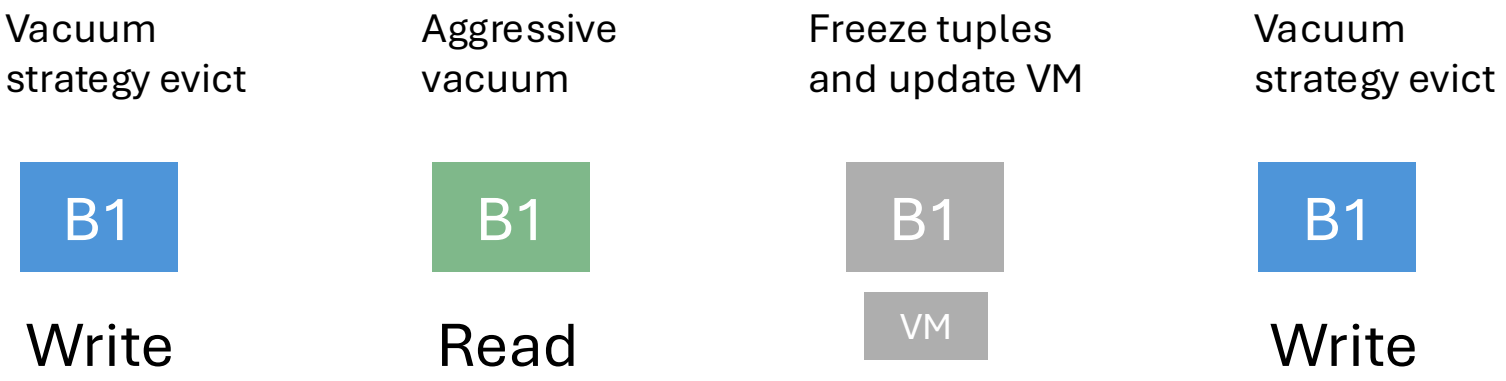
Aggressive Vacuum Overhead

I/O impact disrupts foreground workloads

Can last through failsafe

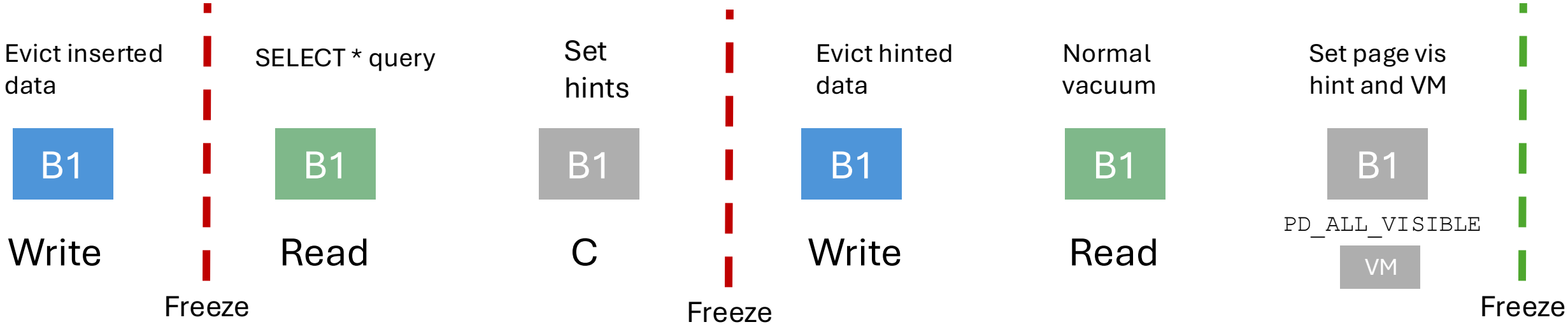
I/O Amplification Waiting to Freeze

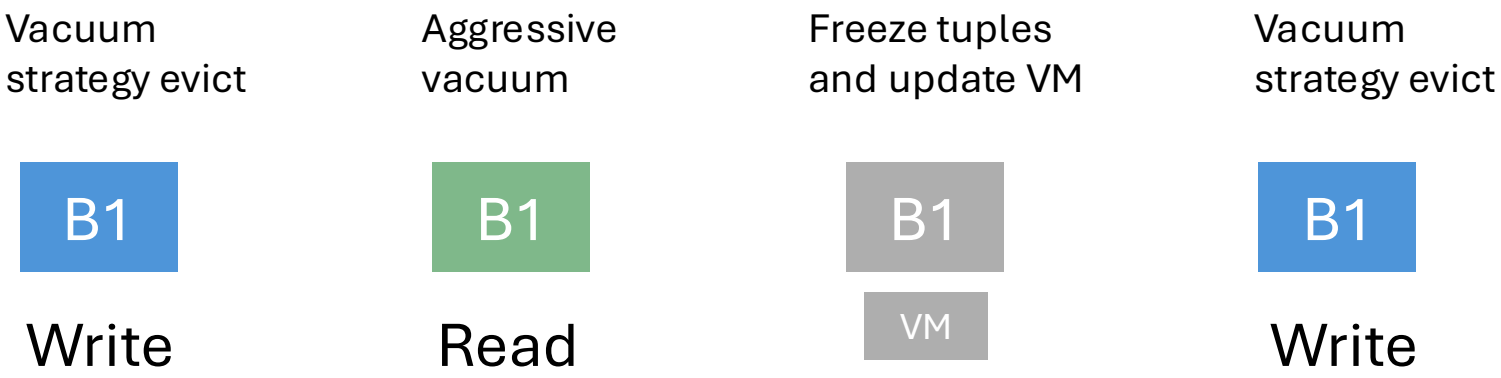




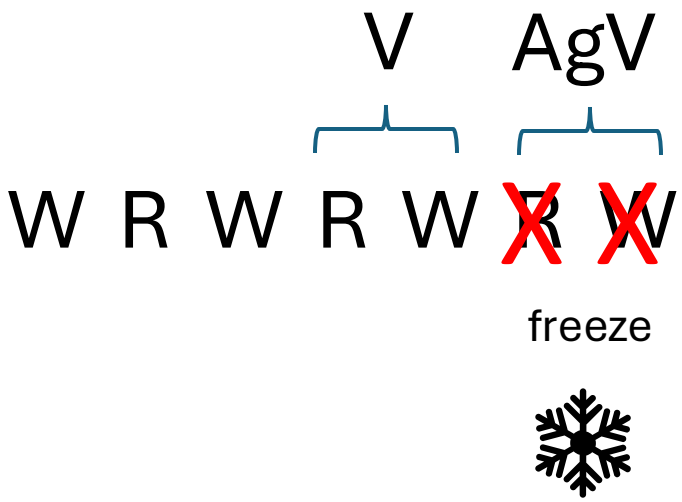
When *should* we freeze?

Ideal Time To Freeze



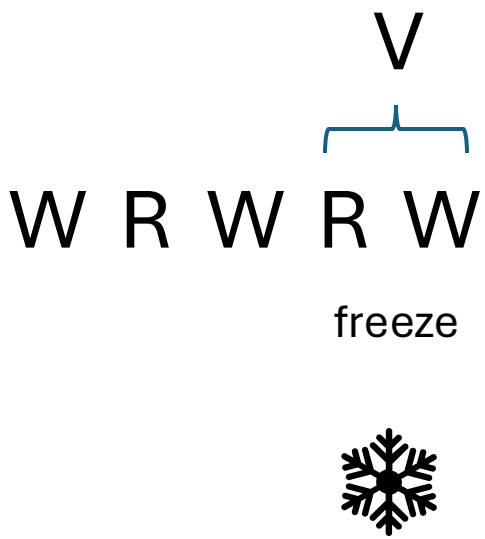


What can be eliminated

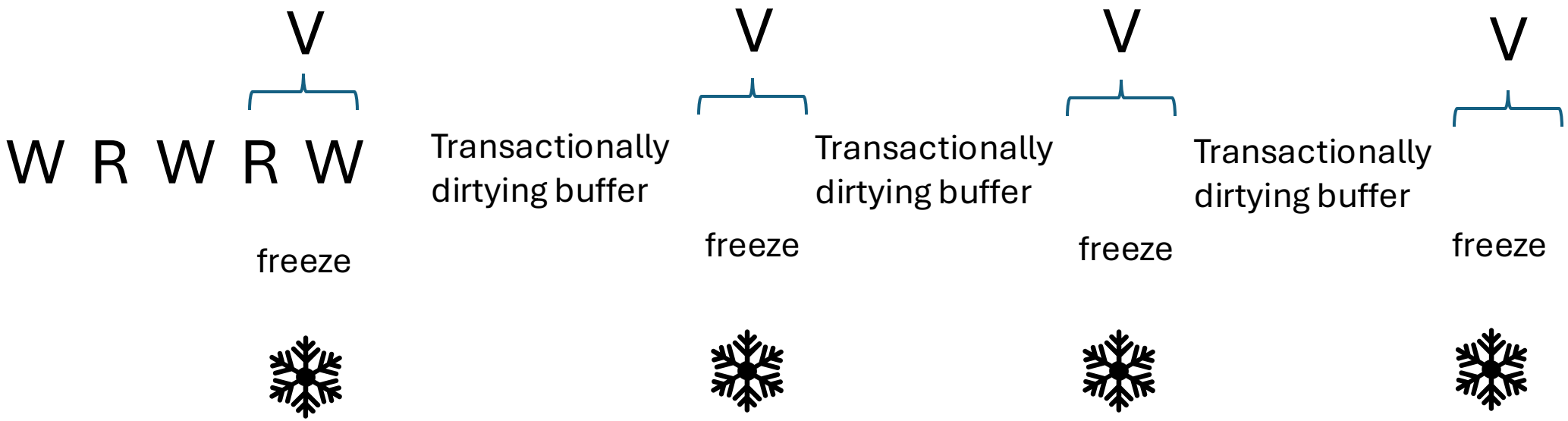


Preemptive Freeze Algorithm Possibilities

Freeze all visible tuples on every vacuum?
(Ignore vacuum_freeze_min_age)

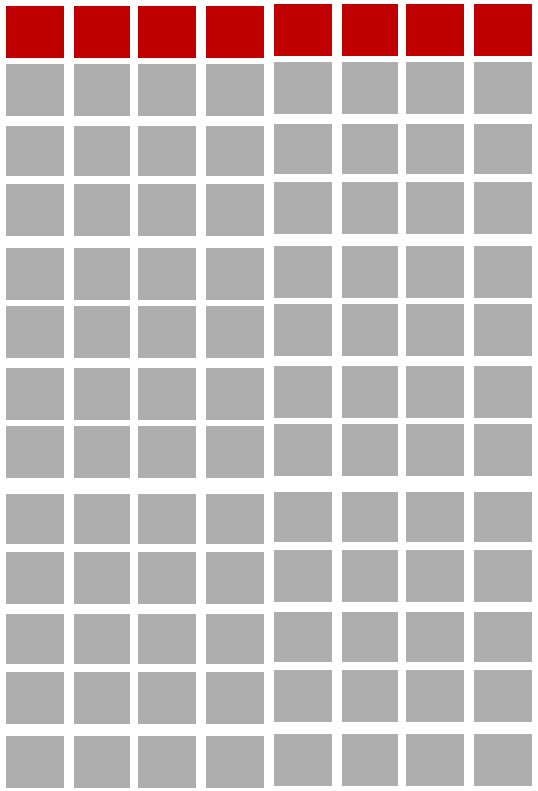


~~Freeze all visible tuples on every vacuum?~~

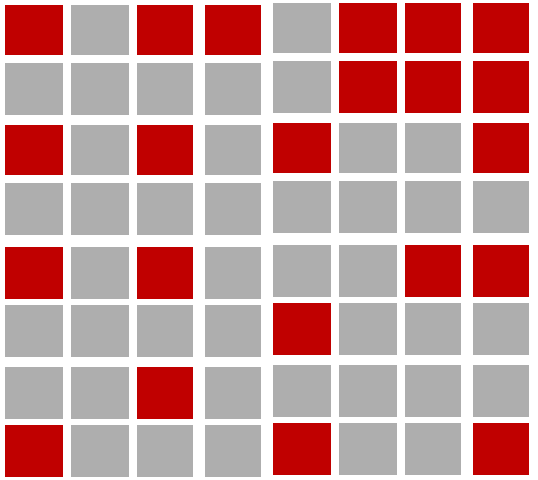


Diverse access patterns

Insert-only table



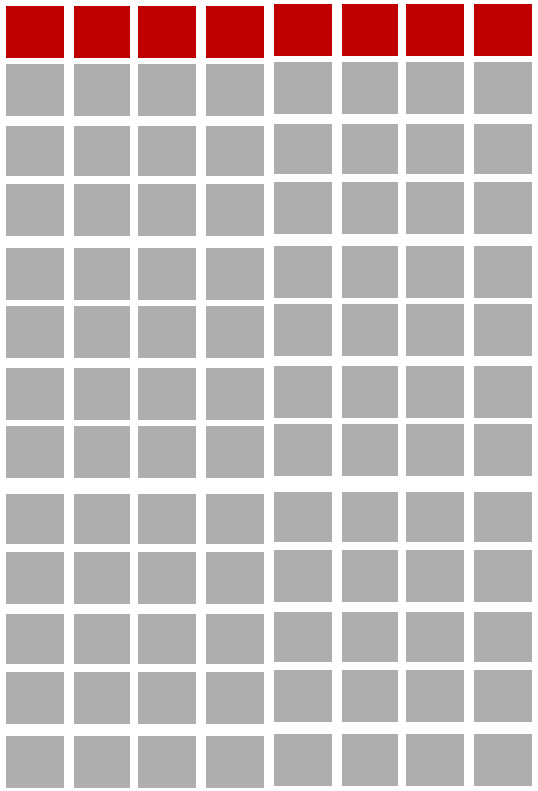
Hotly updated table



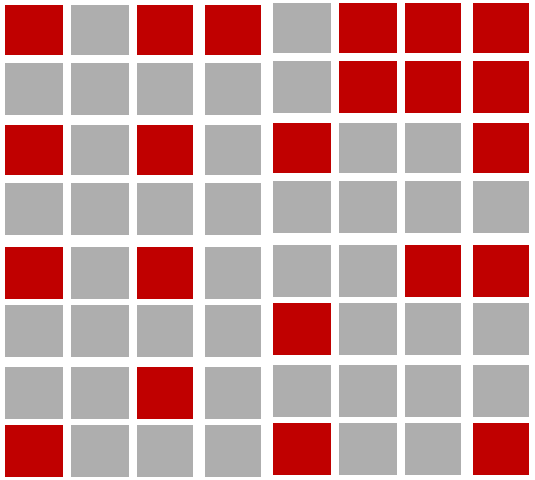
What about using # updates/deletes

`pg_stat_all_tables.n_tup_upd, n_tup_del`

Insert-only table



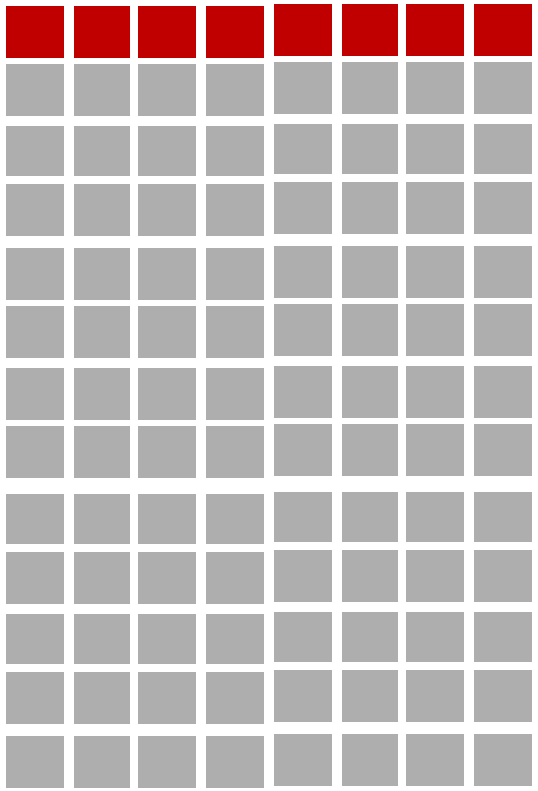
Hotly updated table



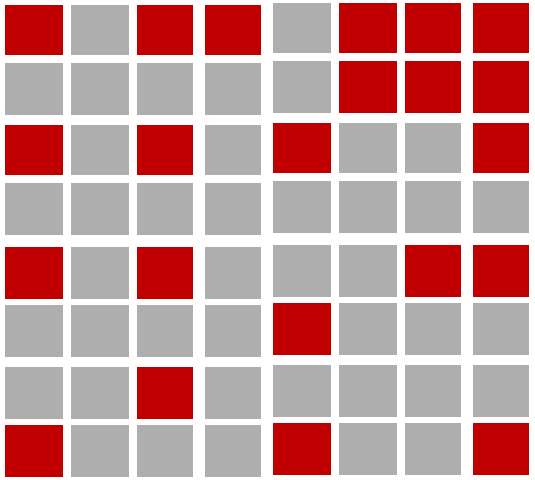
~~What about using # updates/deletes~~

`pg_stat_all_tables.n_tup_upd, n_tup_del`

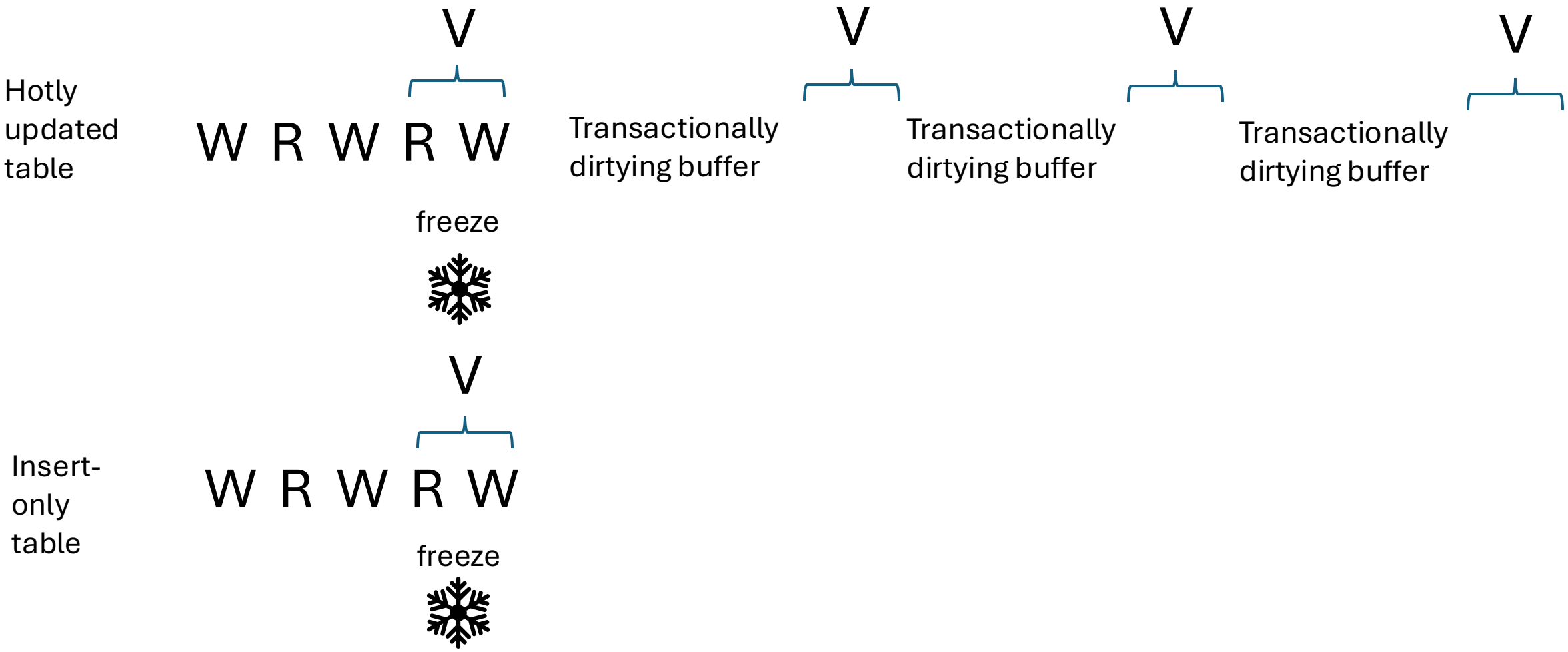
Insert-only table



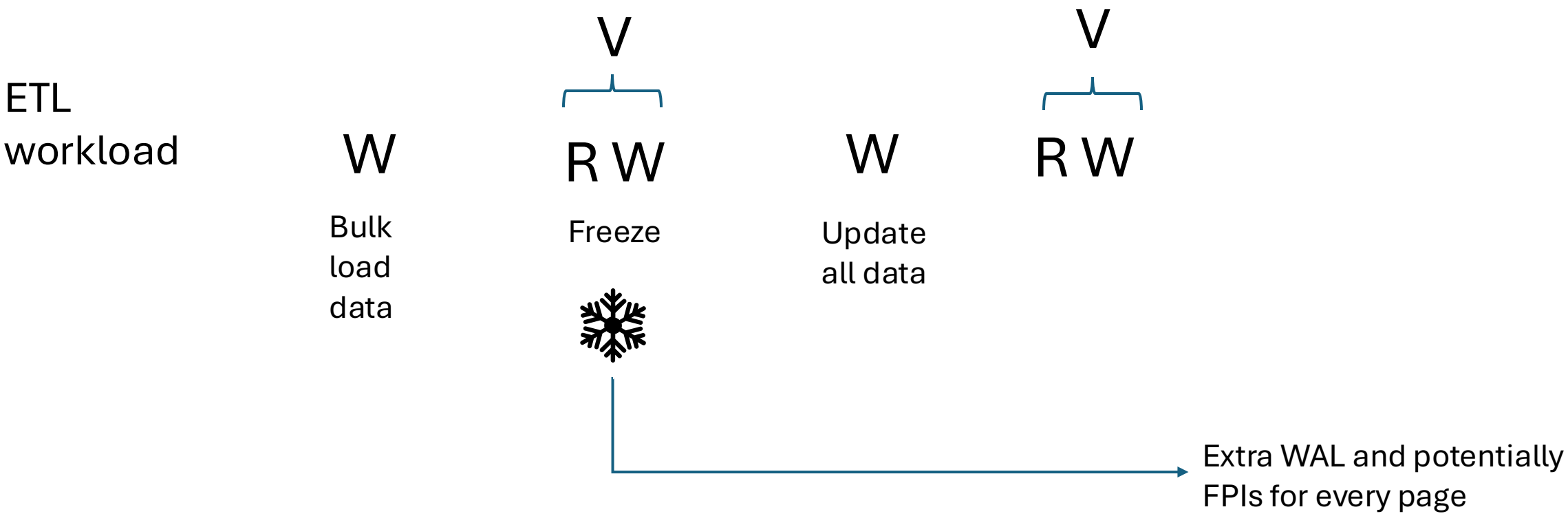
Hotly updated table



What about freezing every page once

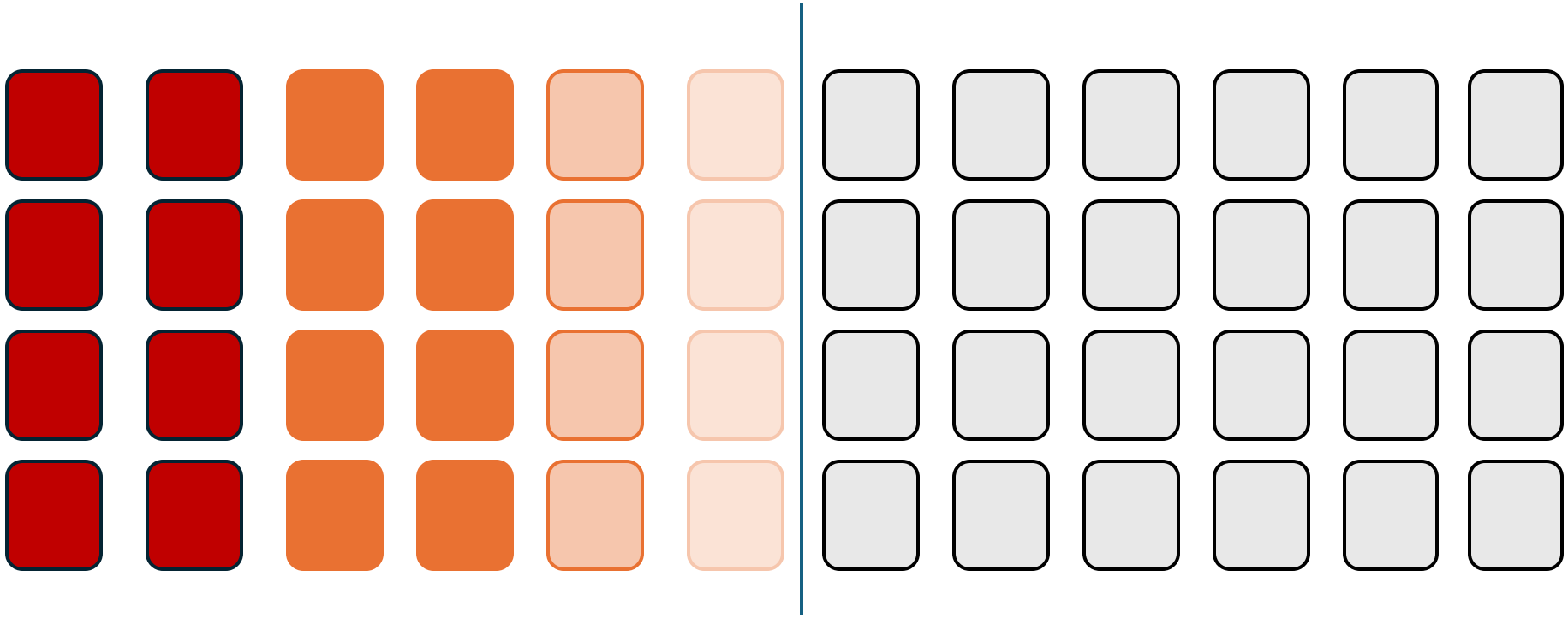


~~What about freezing every page once~~

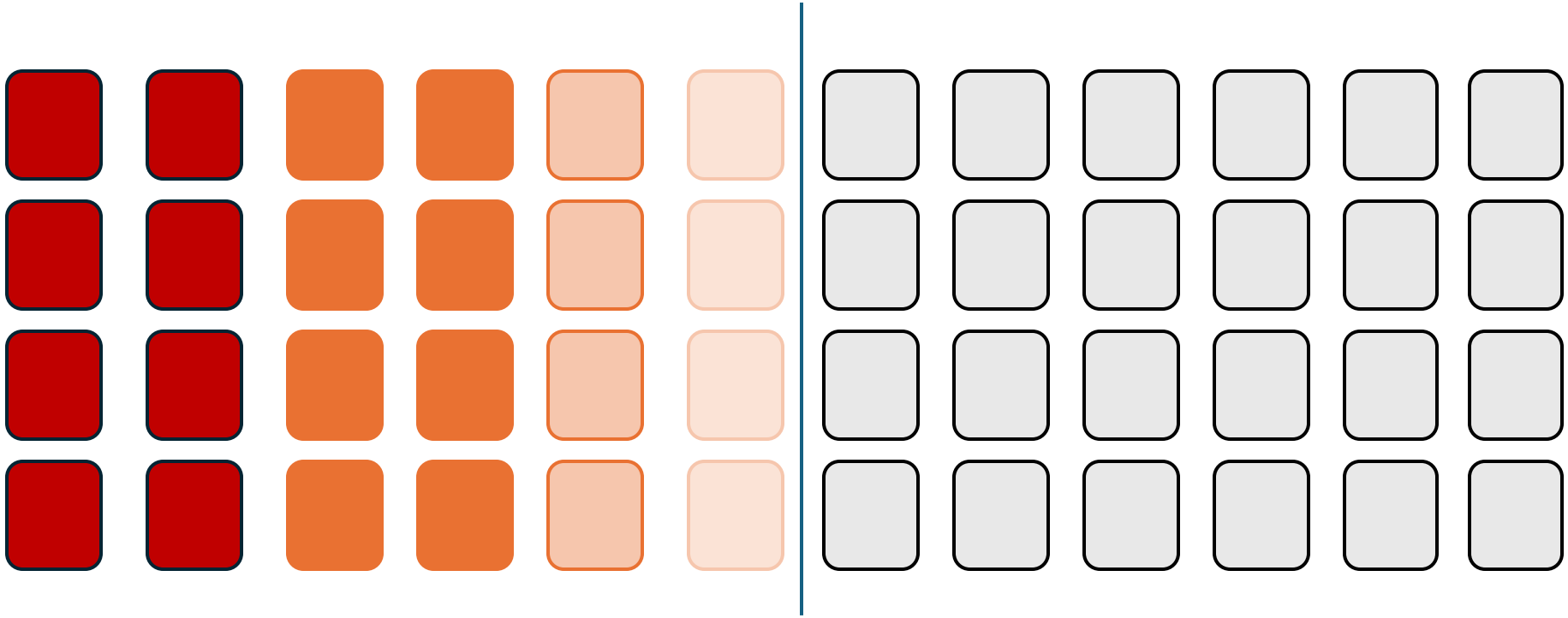


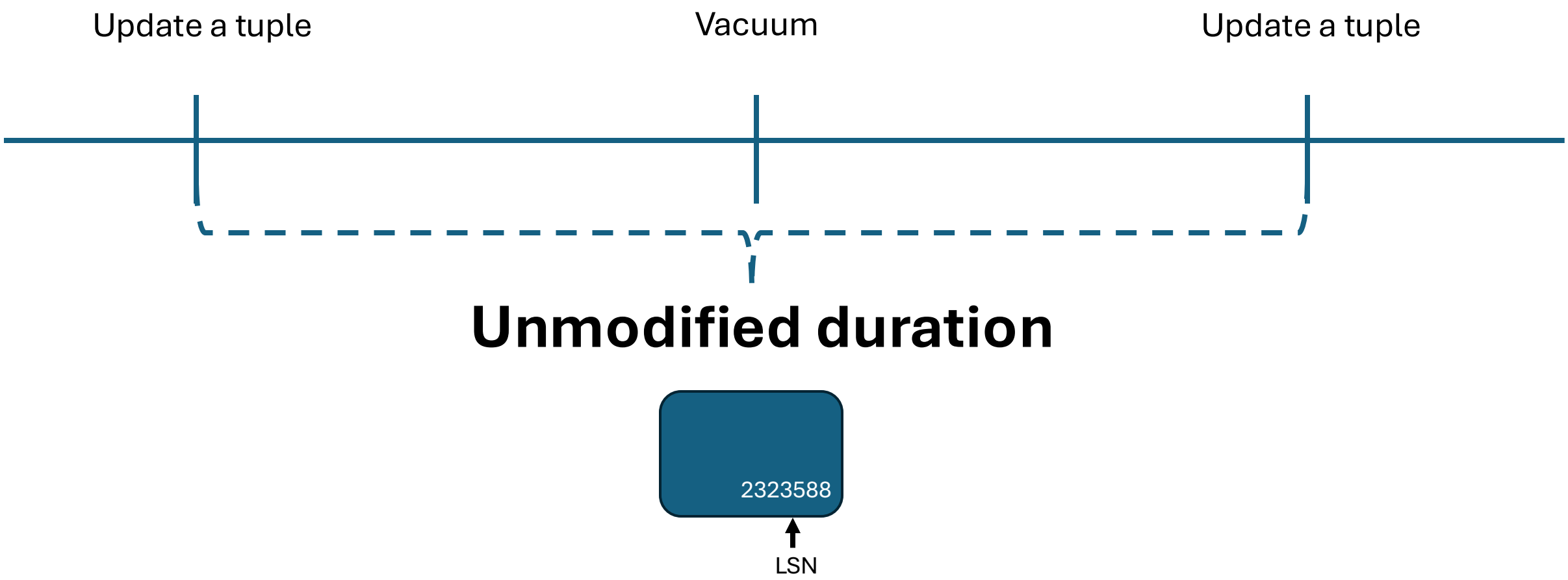
Need a model that includes time

Quiescence theory

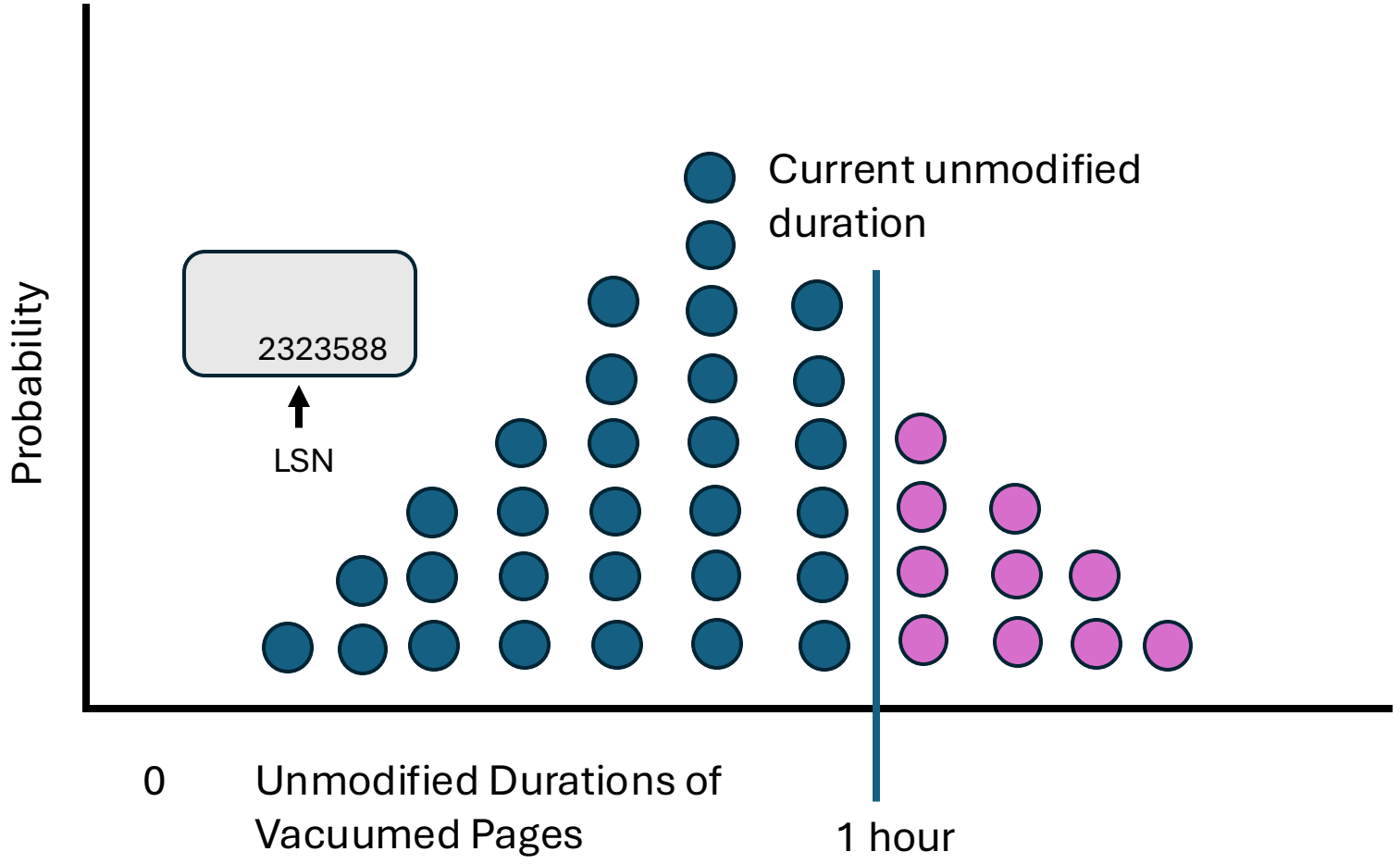


Will the page be modified again?

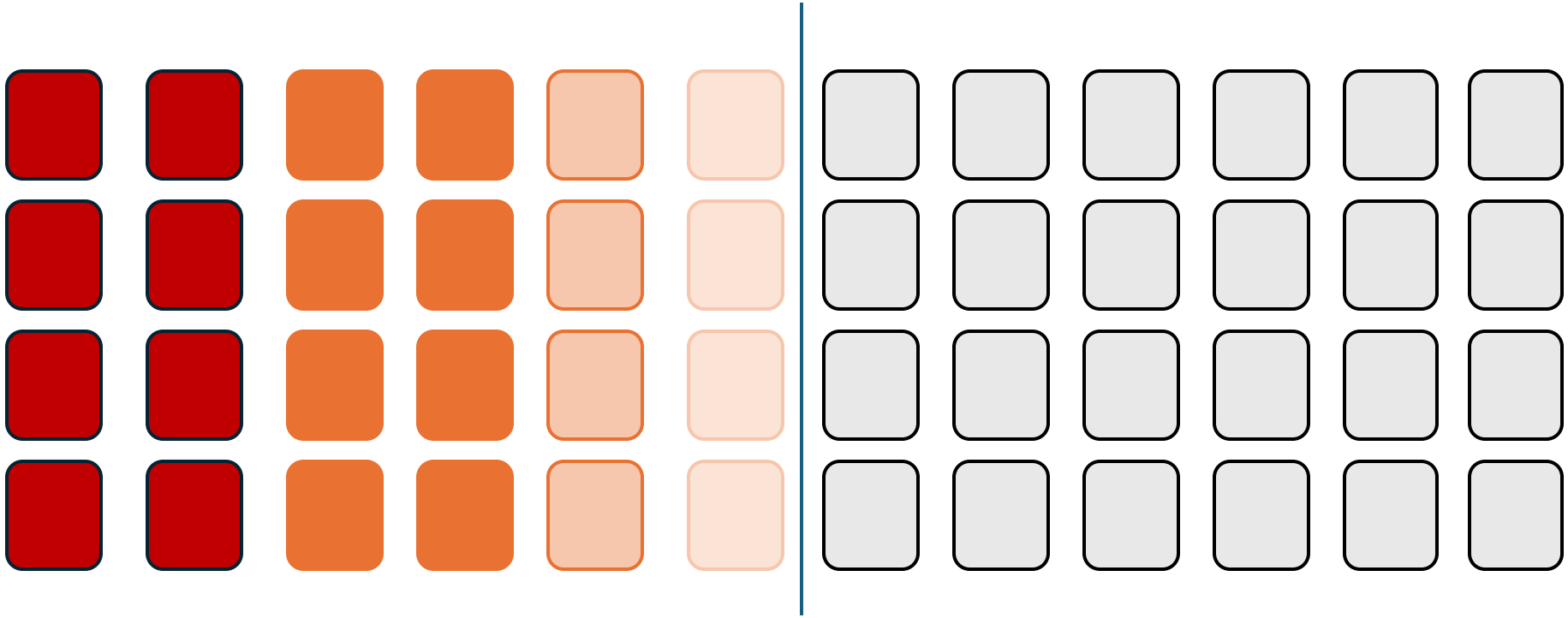



Update a tuple
Vacuum
Update a tuple
Unmodified duration
2323588
LSN

How likely is a page to stay unmodified?

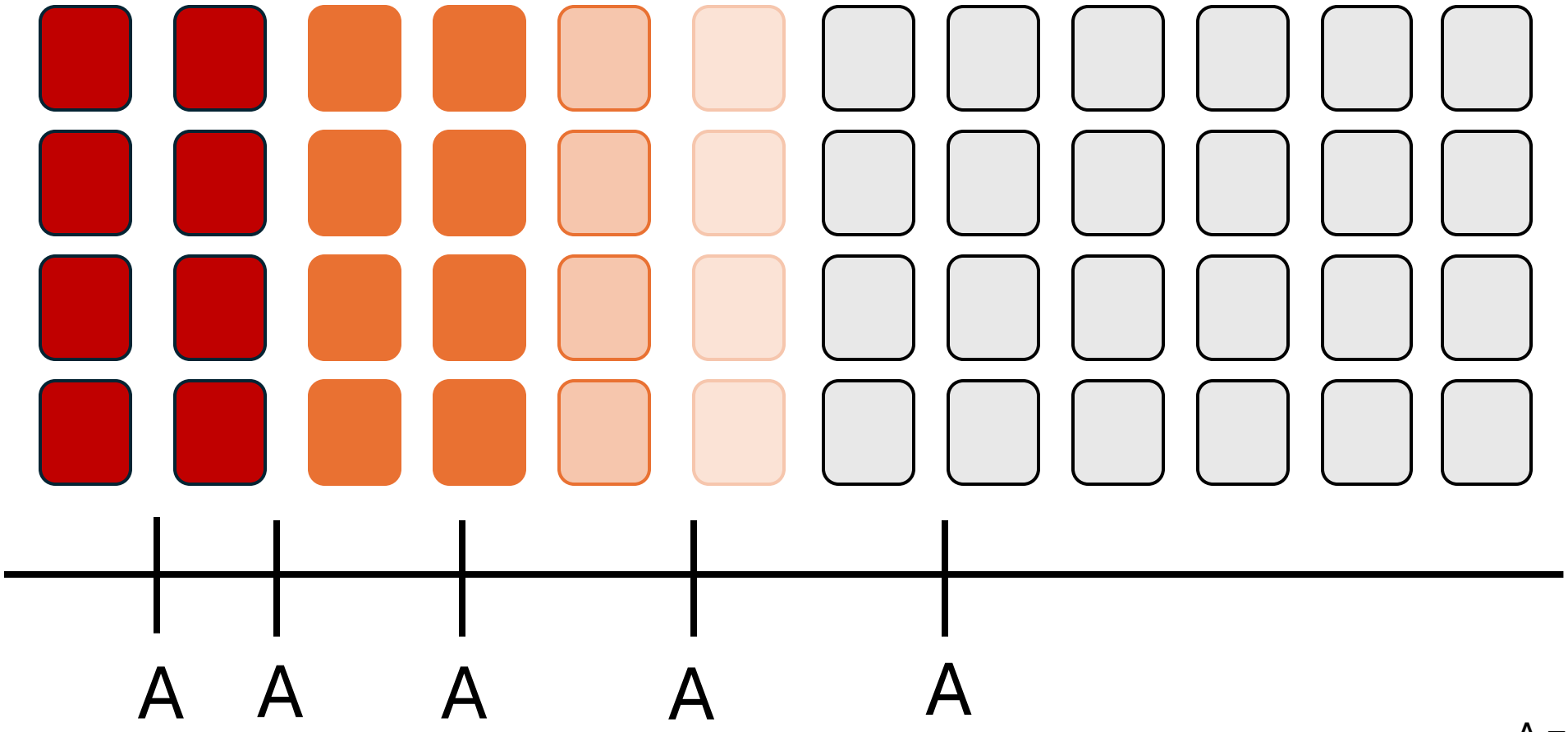


~~Will the page be modified again?~~



Evicted

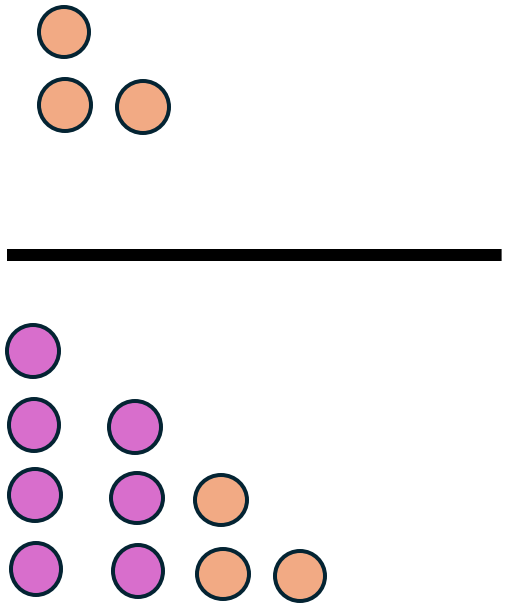
How often can you tolerate useless freezing?

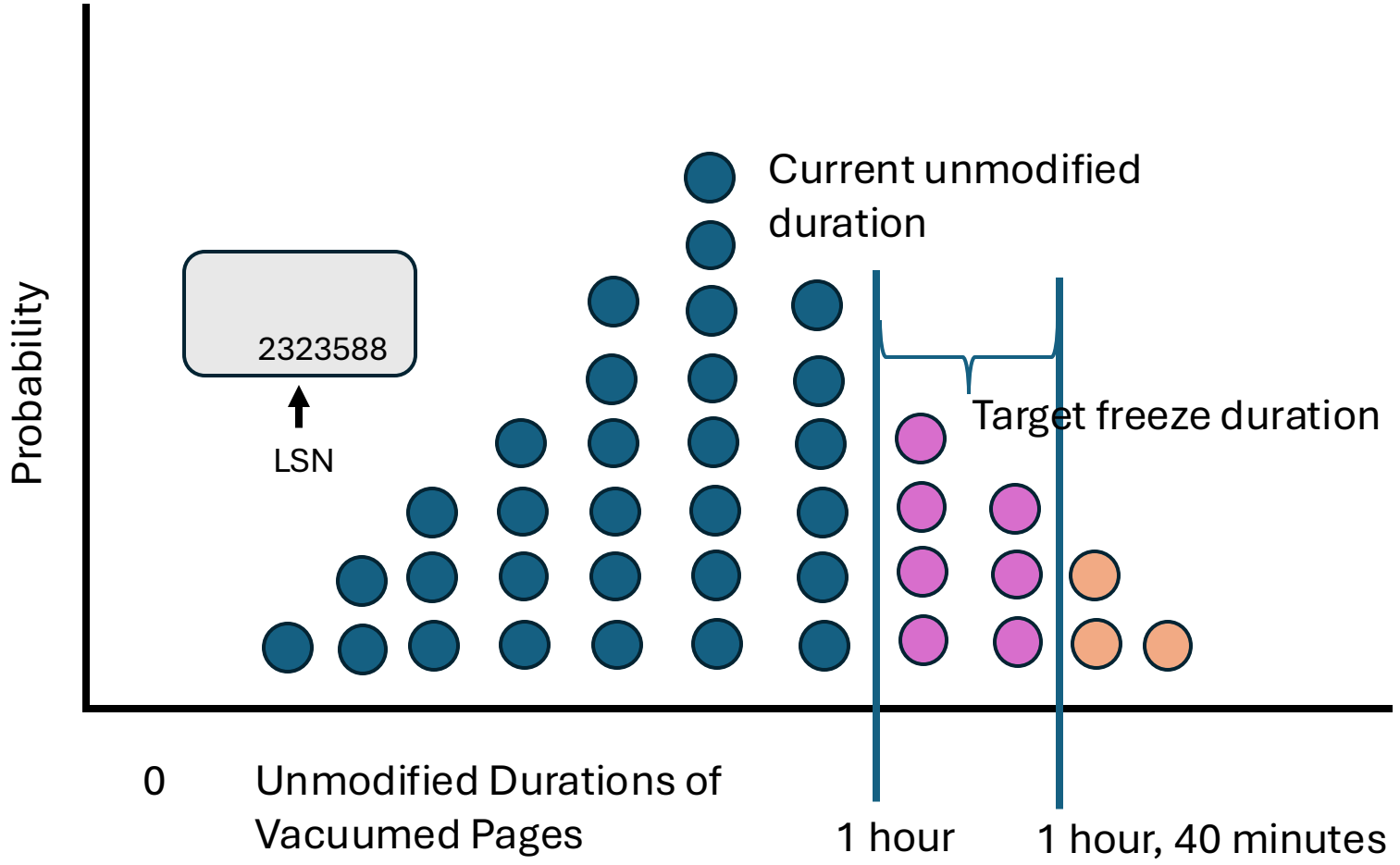


A = Autovacuum

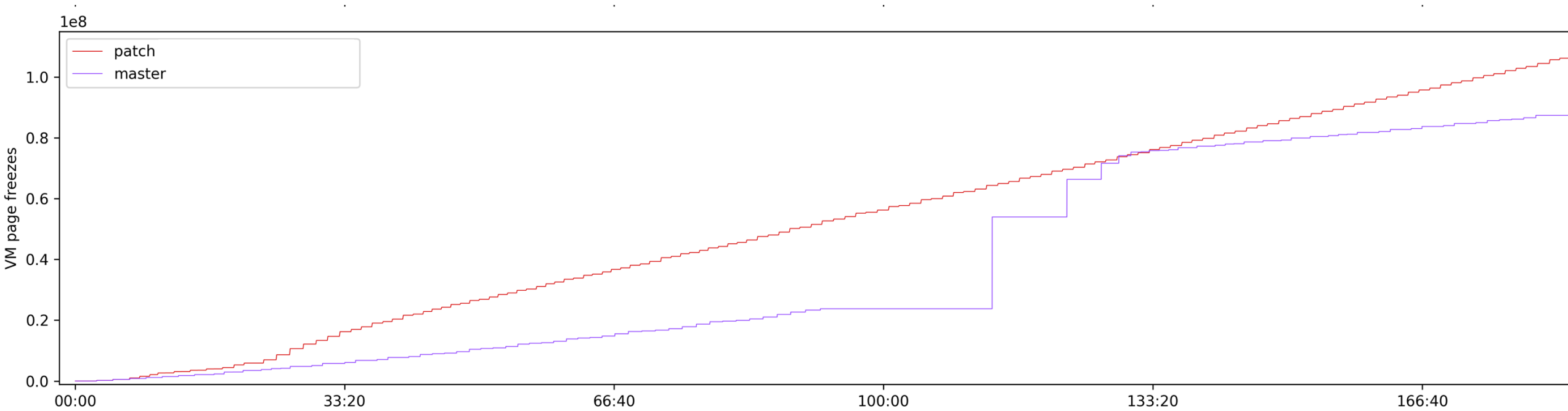
Is the page young enough to stay unmodified for target freeze duration?

How likely is our specific page to stay unmodified for target freeze duration?



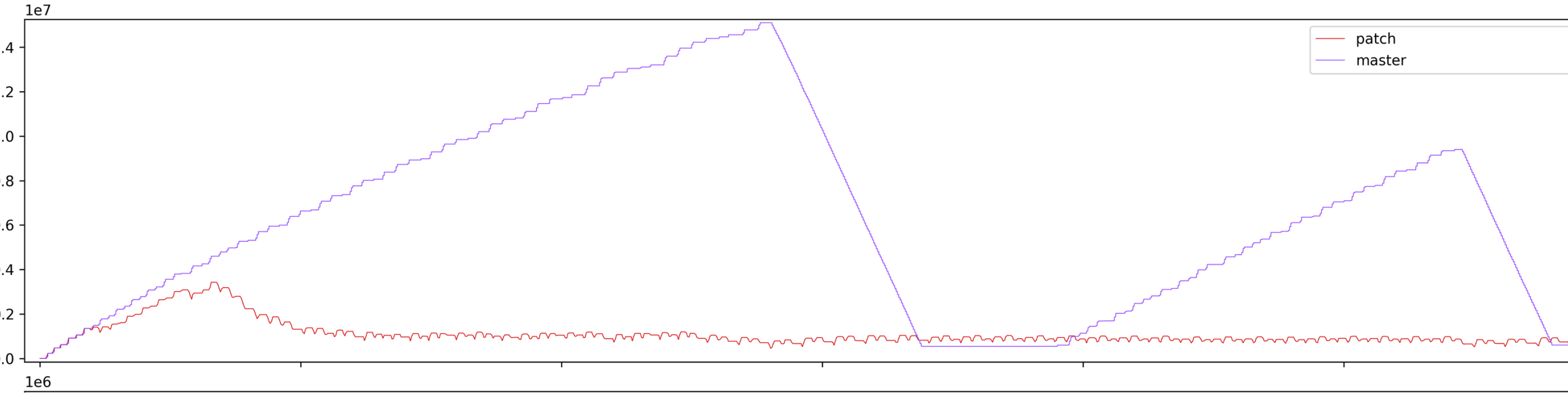


Results: Consistent freezing



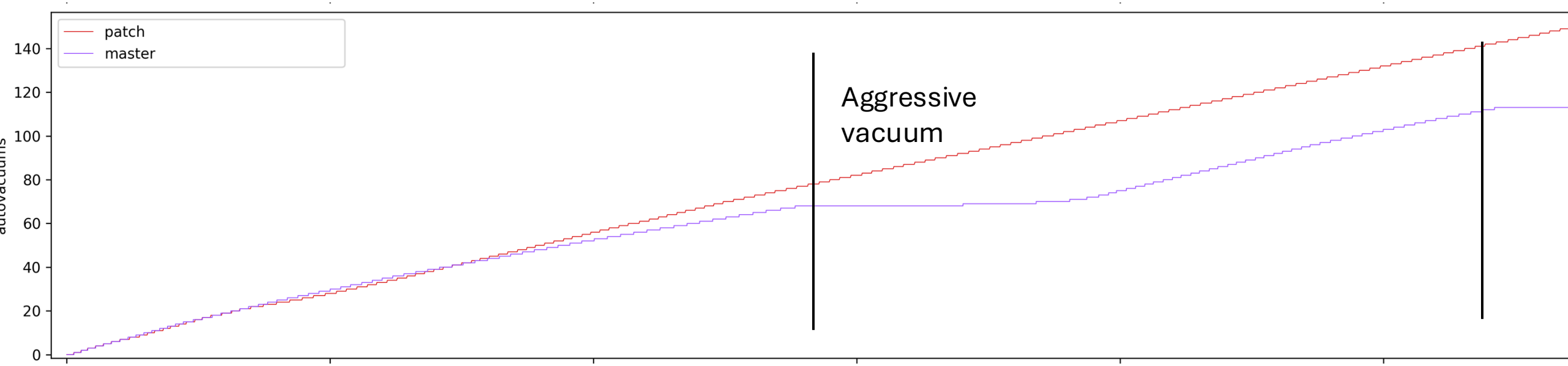
https://github.com/melanieplageman/postgres/tree/adaptive_freeze_for_presentation

Results: All-visible Debt Low and Stable

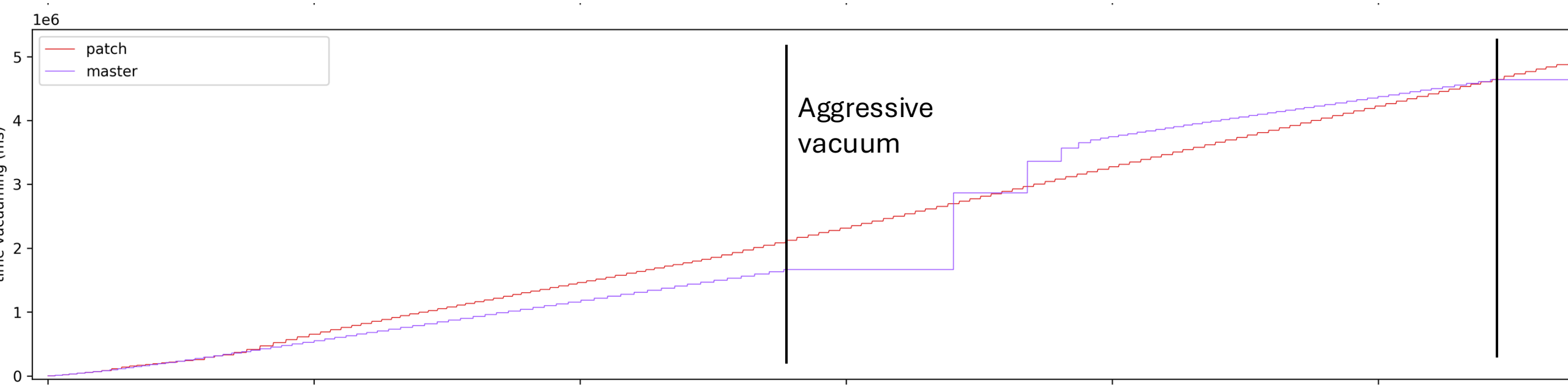


https://github.com/melanieplageman/postgres/tree/adaptive_freeze_for_presentation

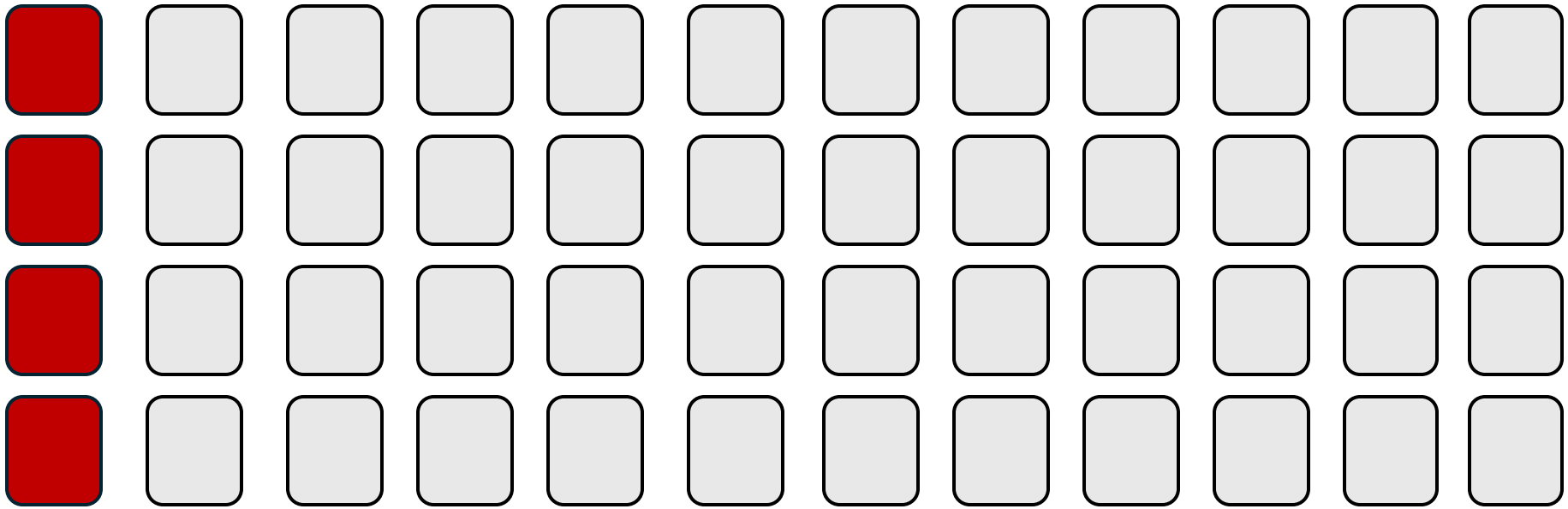
Many short autovacuums



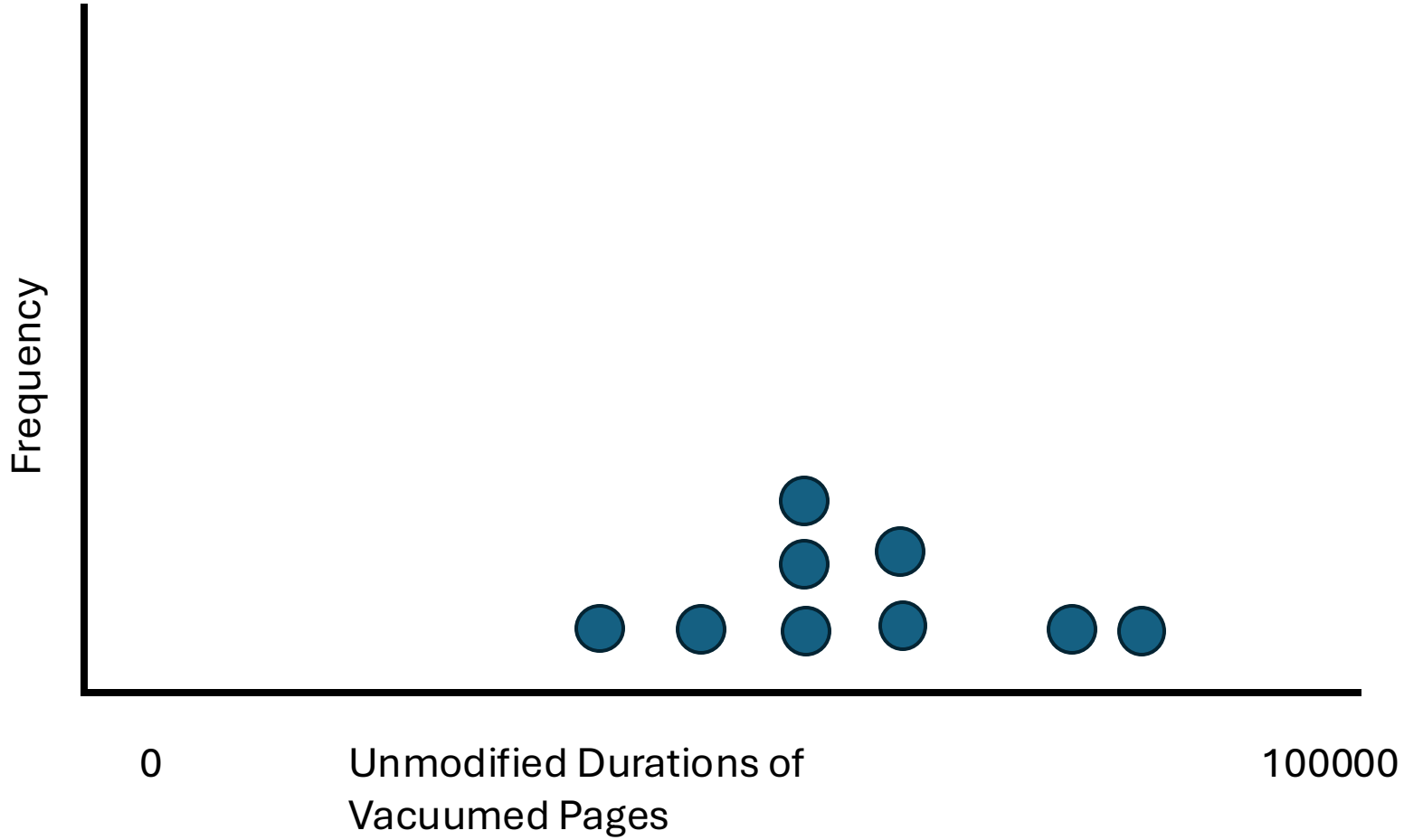
Lower Total Time Spent Vacuuming



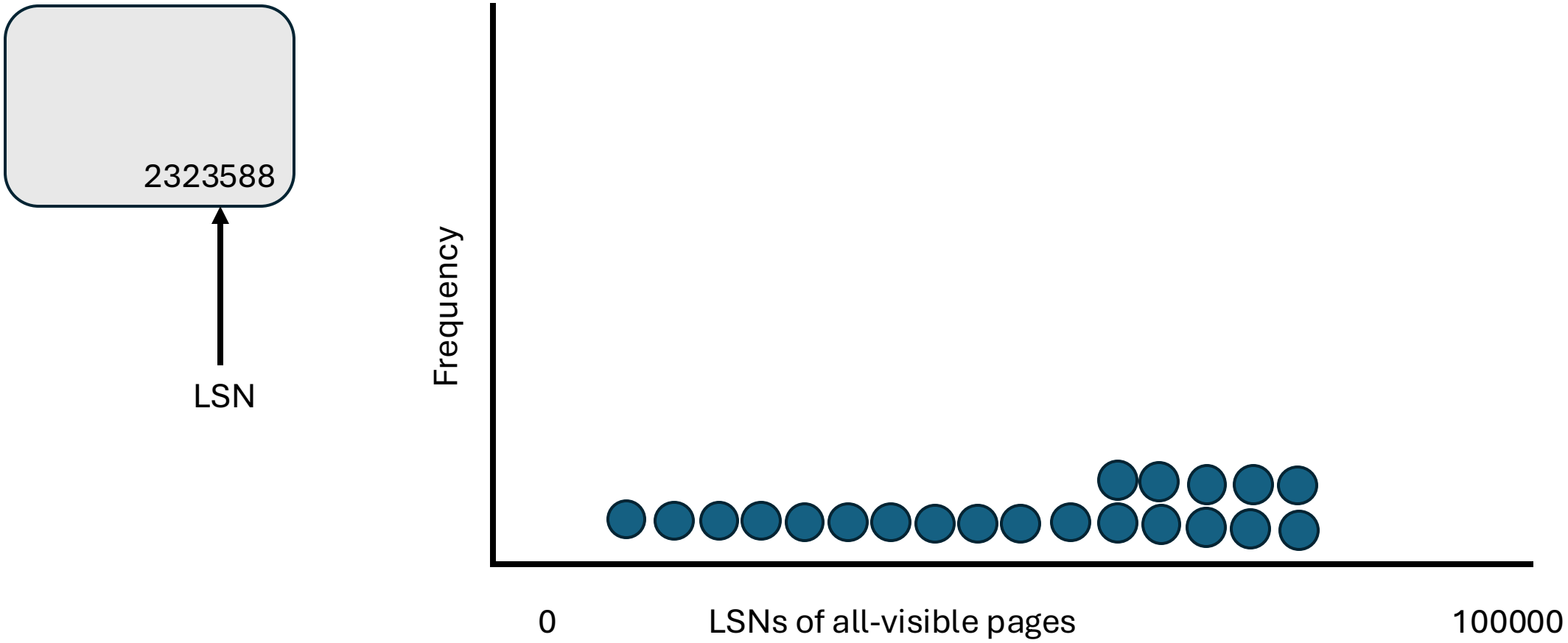
Insert-only workloads, data not modified



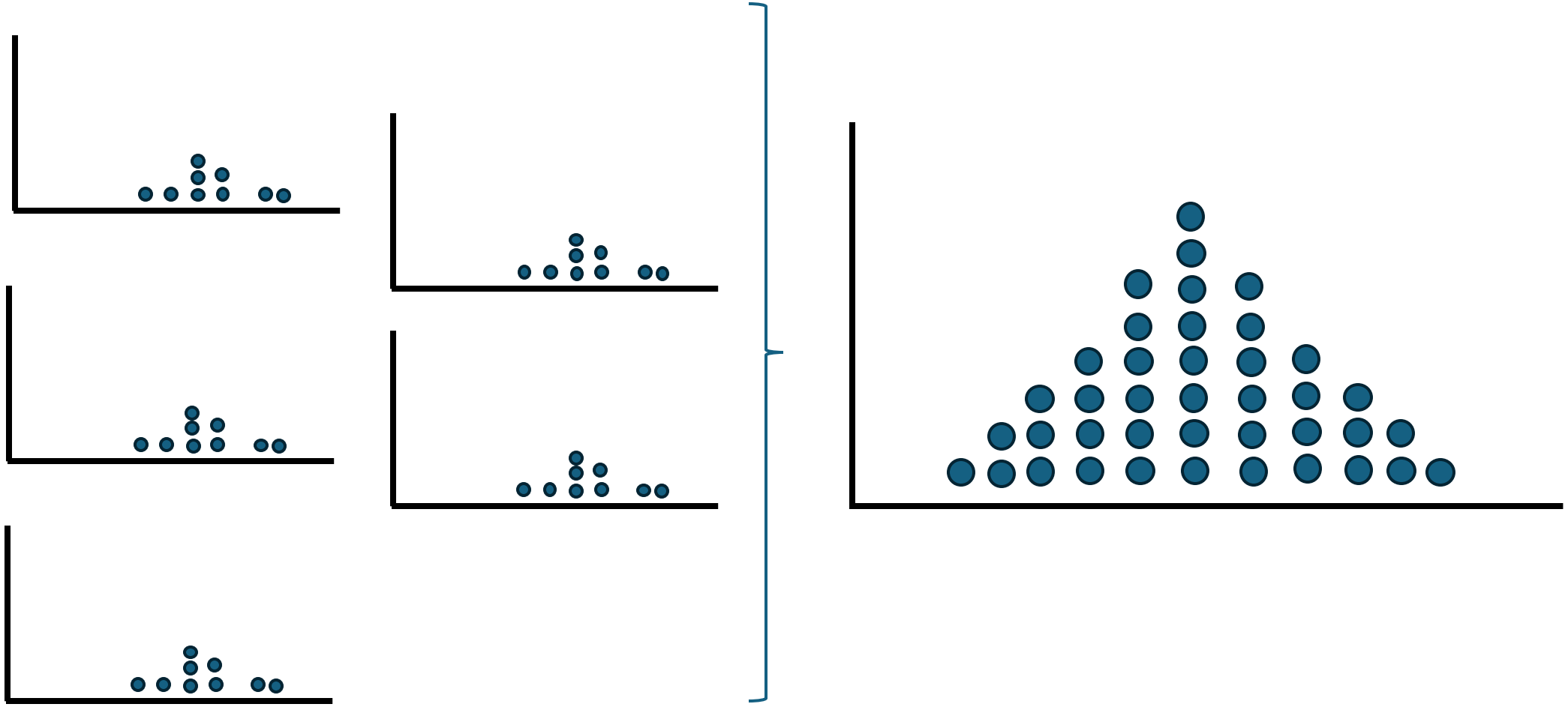
Missing data for pages not modified again



Need to add LSNs of all-visible pages



Building a distribution was complicated



High code complexity

Lots of code

- Low reusability

Major existing infrastructure issues

- Didn’t work with failover or crash

Lessons Learned

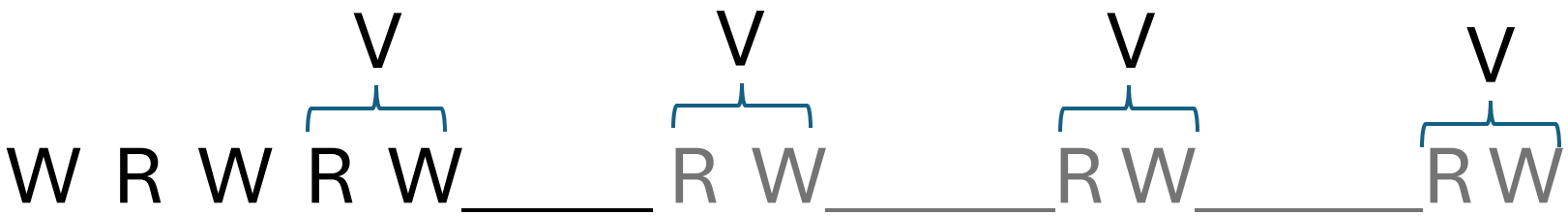
- Sometimes attempts to simplify fail
- Define the problem better sooner

New Direction

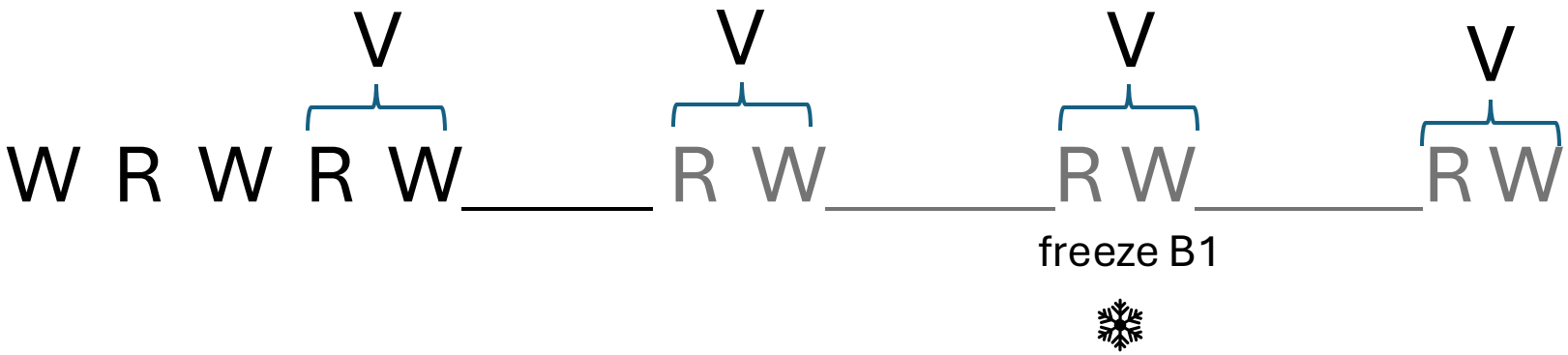
What Went into Postgres 18

Long time between vacuuming and aggressive vacuuming B1

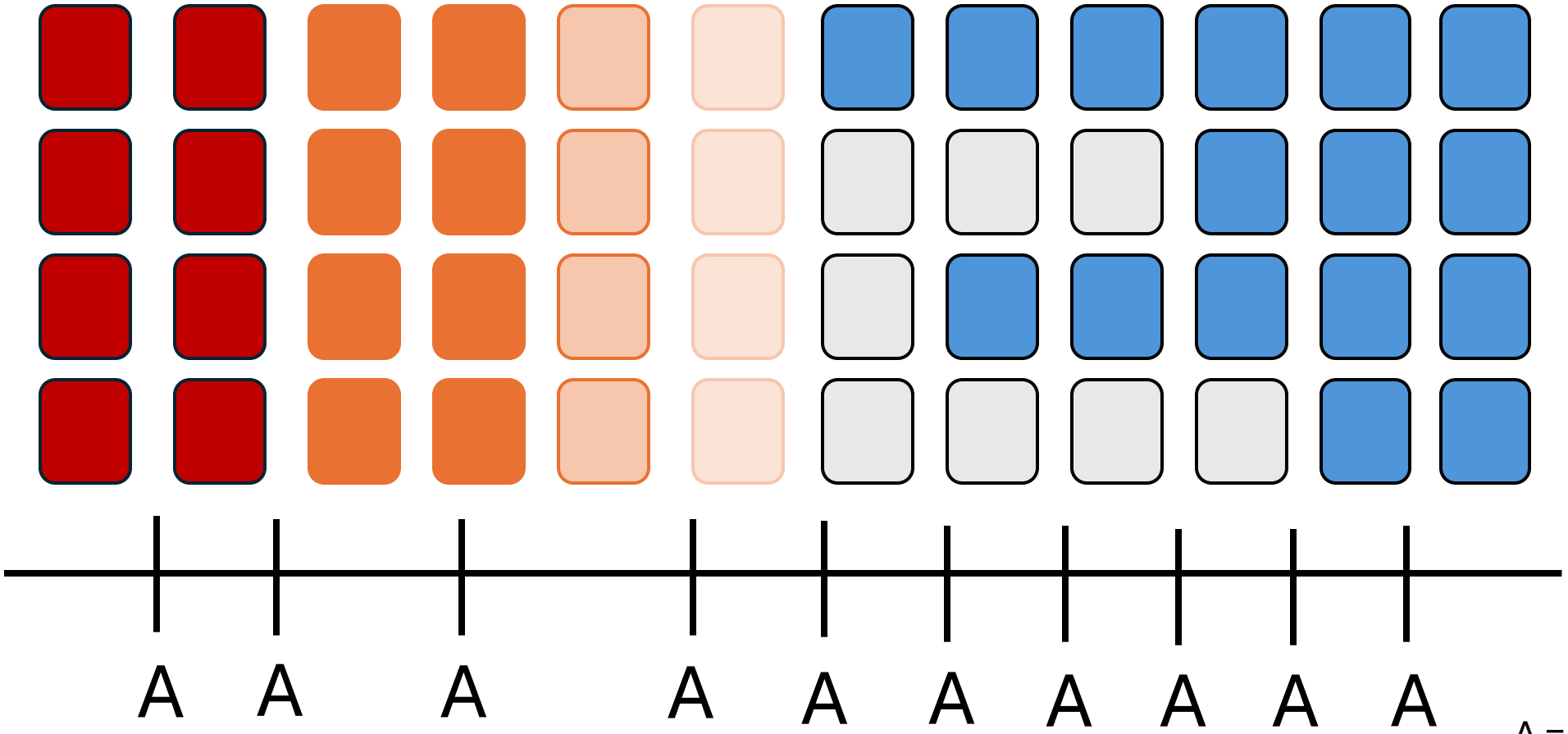
Other inserts triggered normal vacuums of new pages



What if we scan and freeze B1 sooner?



Switch framing to all-visible pages scanning

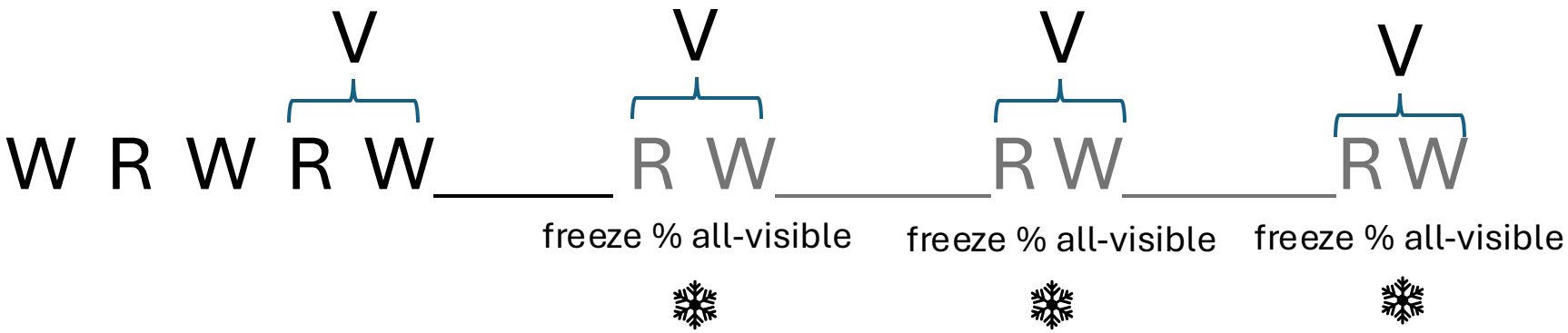


A = Autovacuum

Adaptively eager scan all-visible pages

- All-visible pages more likely to need freezing
- Stop freezing if it isn’t working
  - Only requires tracking information in one vacuum

Amortize the aggressive vacuum



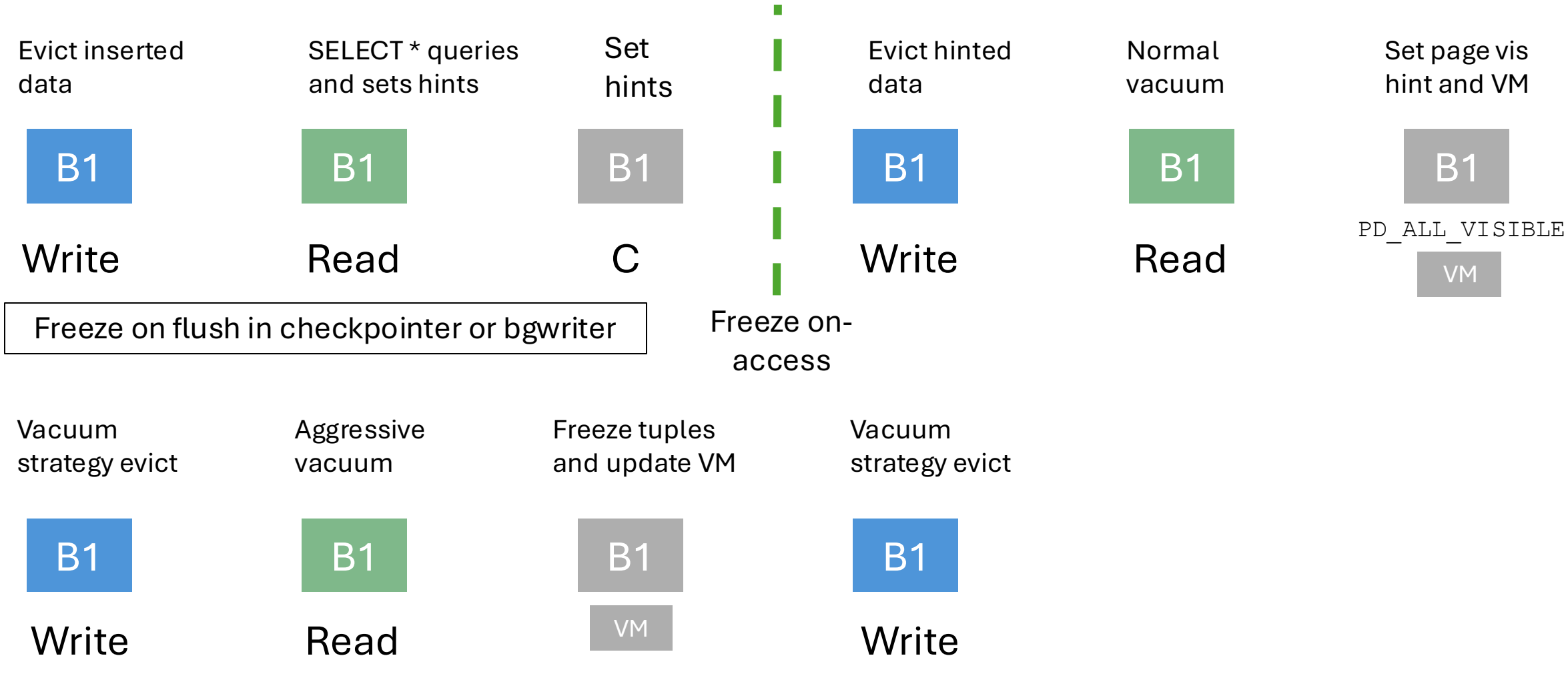
Adaptively eager scan all-visible pages

- Cap eager scanning at a percentage of all-visible pages
- Suspend eager scanning if not successfully freezing pages

GUC and table storage option

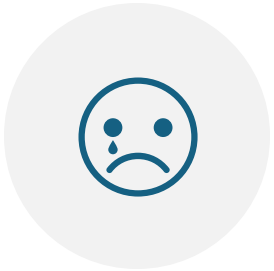
- Eagerly scan and freeze up to 20% of the table
  - No more eager scanning for remainder of vacuum
- `vacuum_max_eager_freeze_failure_rate`
  - Eagerly scan and fail to freeze 3% of 4096 block size region (32 MB)
  - Suspend eager scanning until next region

Future Directions



Conclusion

ADAPTIVE ALGORITHMS ARE POSSIBLE IN POSTGRES



BUT ARE VERY HARD

BUT WE SHOULD THINK MORE ABOUT THEM

THANKS TO ANDRES FREUND AND ROBERT HAAS

Microsoft

Got 3 minutes? We'd love your input on some of our Postgres work

Get your FREE socks @ Microsoft booth