EDB
Postgres for the AI Generation

SLRU Performance Issues

How we have optimized it

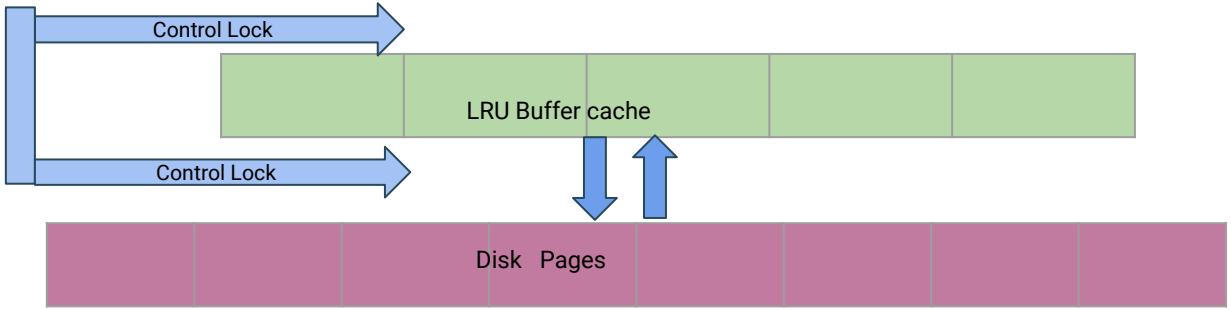
Dilip Kumar,
Principal Engineer
30-05-2024 (2024.pgconf.dev)

Agenda

- SLRU design
- SLRU design problems
- Subtransaction cache overflow problem
- Multixact contention problem
- Solutions
- Performance test

SLRU design

- Stores transaction related metadata on disk with LRU caching
- Centralized ControlLock to access buffers in the cache
- For page replacement within a slot we use slot level lock
- Finding a page in cache is done sequentially
- Victim buffer search is also a sequential operation



SLRU design problems

- #1: Frequent buffer replacement in cache
  - Problem is smaller size of SLRU caches
  - This mainly occurs if wider range of transaction data are getting accessed
- #2: High contention on centralized control lock
  - When a lot of readers are accessing the SLRU along with writers.
  - This will also be caused by frequent SLRU buffer eviction
- #3: LRU counter cause frequent CPU cache misses
  - This can happen even if only readers accessing the SLRU

Subtransaction cache overflow

Subtransaction cache overflow

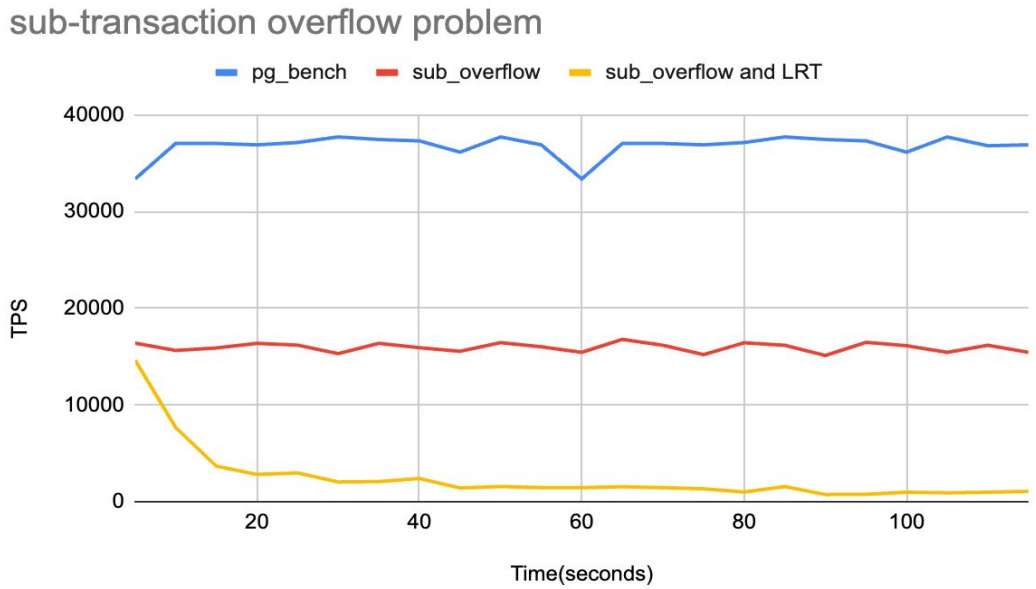
- Case1: Pgbench read-write test with scale factor 300
  - Execute the test with 128 concurrent clients

- Case2: Create a subtransaction overflow by starting a concurrent transaction with more than 64 subtransactions

- Case3: Concurrently start a long running transaction

Subtransaction cache overflow

Performance drops to half with subtransaction overflow and after starting a long running transaction it hit the bottom and Shows heavy load on SubtransSLRU and SubtransBuffers wait events



Subtransaction cache overflow

- Snapshot is a arrays of running xids and subxids bounded by xmin and xmax
- Each backend can hold 64 subxids in the cache
- If this limit crosses, snapshots can no longer get the complete information of the subxids
- Visibility checks can only rely on xid array of the snapshot
- We need to access pg_subtrans SLRU for getting the parent xid

Subtransaction cache overflow

- Only with subtransaction cache overflow, the main problem is cache line contention due to concurrent shared lock and global LRU counter update

- With long running transaction we need to lookup wider range of xids into SLRU because range of xmin and xmax is becoming larger

Multixact Contention Issue

Multixact contention issue

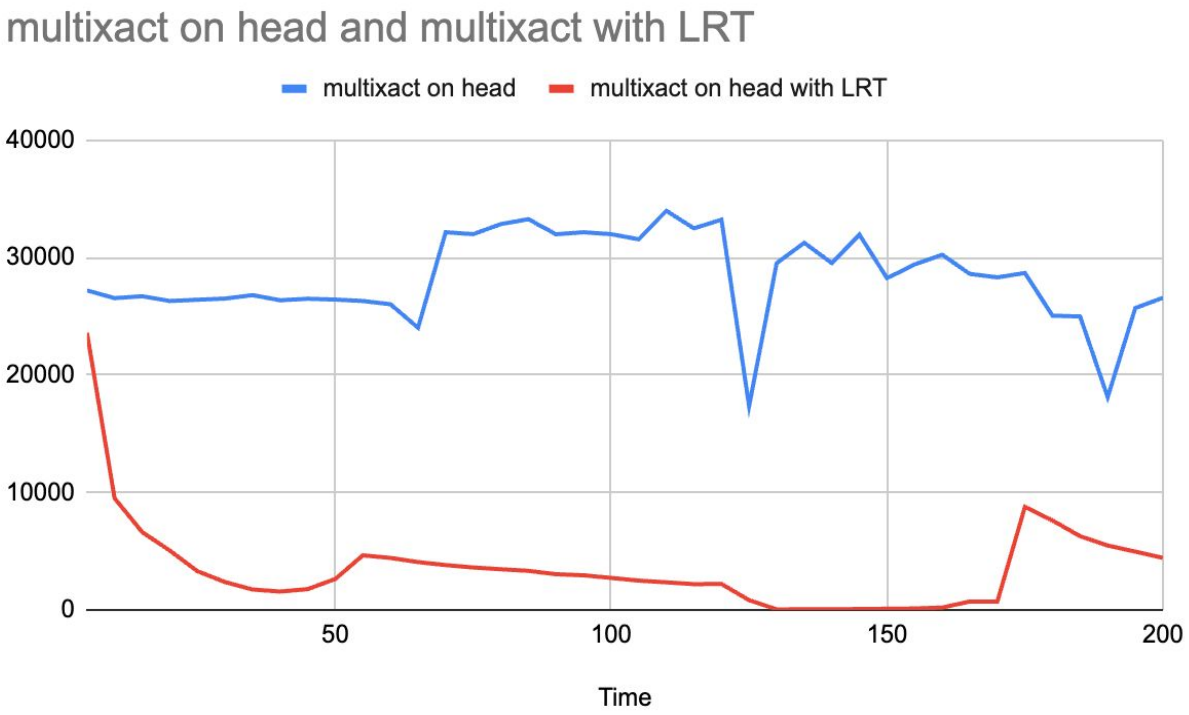
- Case1: Modify pgbench script to generate a lot of multixact
  - Execute this script with 128 concurrent sessions
- Case2: Start a concurrent long running transaction

pg_bench script to generate multixact

```
BEGIN;
SELECT FROM pgbench_accounts WHERE aid = :aid FOR UPDATE;
SAVEPOINT S1;
UPDATE pgbench_accounts SET abalance = abalance + :delta WHERE aid = :aid;
SELECT abalance FROM pgbench_accounts WHERE aid = :aid;
END;
```

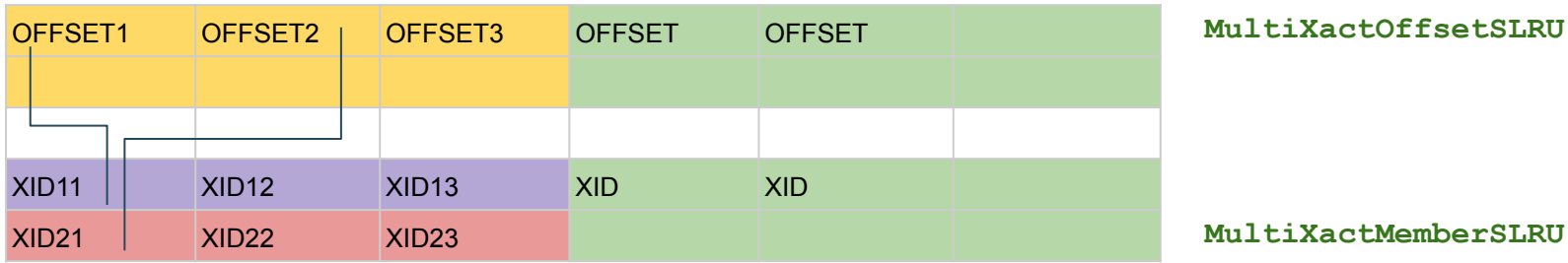
Multixact contention issue

Performance drop drastically with long running transaction and Shows heavy load on MultiXact*SLRU and MultiXact*Buffers wait events



Multixact contention issue

- For supporting row locking we need to store multiple xids on the same row
- Multixact provide a way to create a new kind of xid called multixact id
- The pg_multixact SLRU stores mapping from multixact id to member xids
- We use two SLRUs to handle the variable number of member xids



Multixact contention issue

- Every access to tuple marked with multixact-id, need to fetch xids from the SLRU
- Concurrent creation and reading of multixact create contention on ControlLock
- Long running transaction also increases the range of multixact ids to be read
- Vacuum help in cleaning up the multixact id from the tuple

Ideas for SLRU optimizations

Increase the SLRU cache size

- Pros
  - This will reduce the number of buffer eviction from cache
  - Making it configurable will help optimizing based on use case
- Cons
  - Searching buffer in SLRU is linear so doesn’t scale well with size
  - We will still have contention on a centralized control lock
  - Frequent update global LRU counter and lock variable is still there

Large cache and buffer mapping hash

- Pros
  - Searching would be fast O(1)
  - We may partition the hash and add partition level lock to reduce the contention on a centralized control lock
- Cons
  - Buffer replacement will still do linear search
  - Doesn't help with global LRU counter

Merge SLRU with main buffer pool

- Pros:
  - Do not need separate configuration of the different buffer pools
  - Maintenance of the code should be better
- Cons:
  - Hard to do some SLRU specific optimization
    - E.g. protect against evicting out the latest page
  - Might not be a best idea to compare SLRU vs Data page

Large cache size with partially associative cache

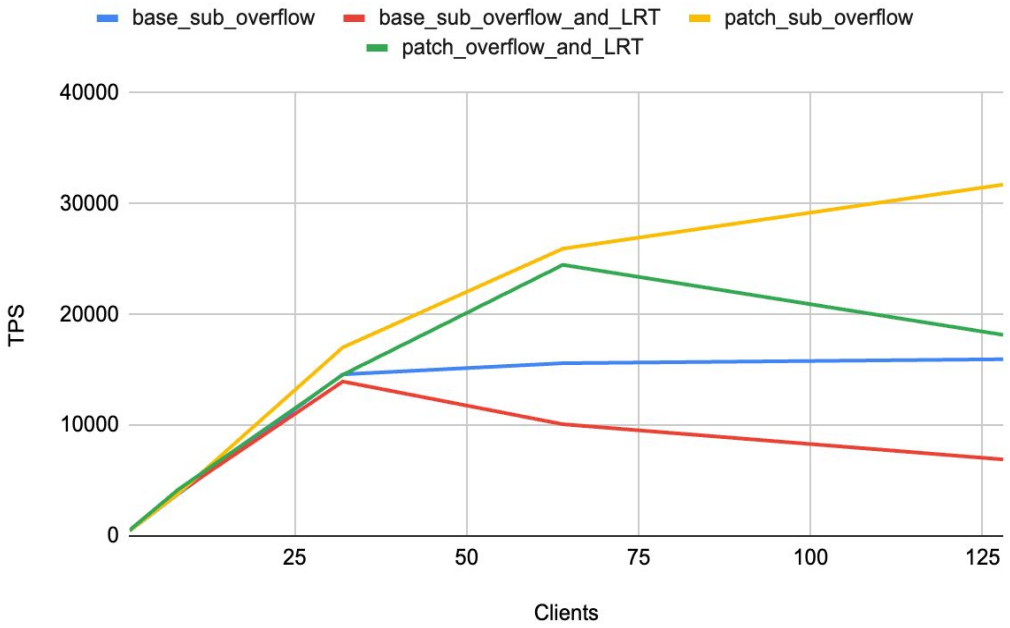
- Configurable SLRU cache size
- Divide cache into associative buffer banks
- Cache size is big but searching is limited to the associative buffer bank
- Buffer eviction is also fast as bank sizes are small
- No more Centralized control lock and LRU counter

Performance Results

Performance: Subtransaction cache overflow

pgbench with different clients in presence of sub-overflow and long running transaction

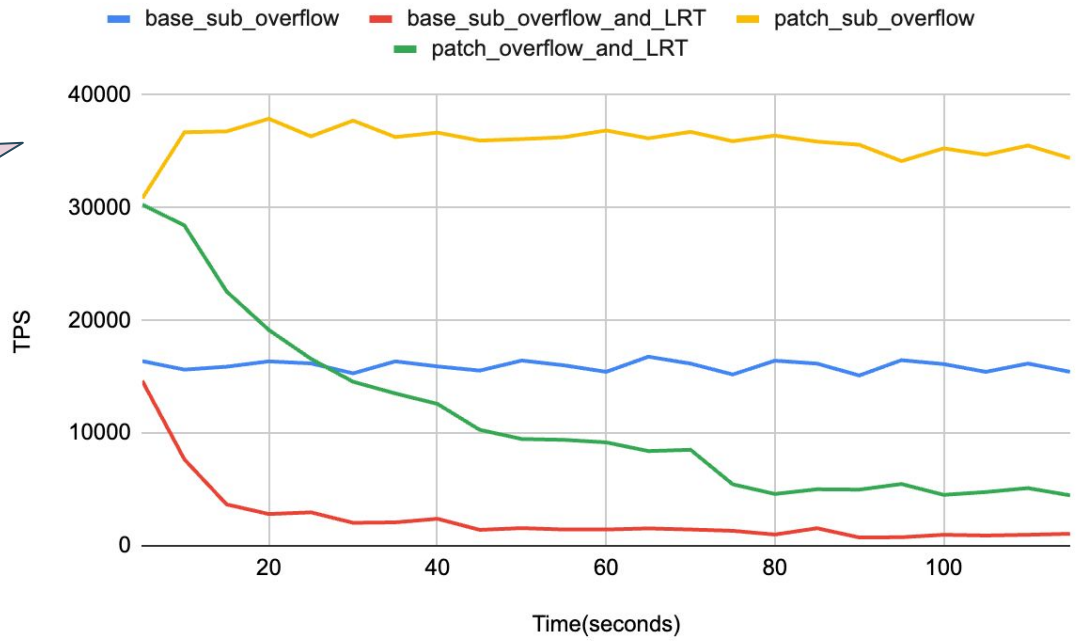
- *Test on 128 core machine*
- *Scale factor: 300*
- *Subtransaction_buffers: 1024*



Performance: Subtransaction cache overflow

Effect of patch on TPS drop at higher clients in presence long running transaction

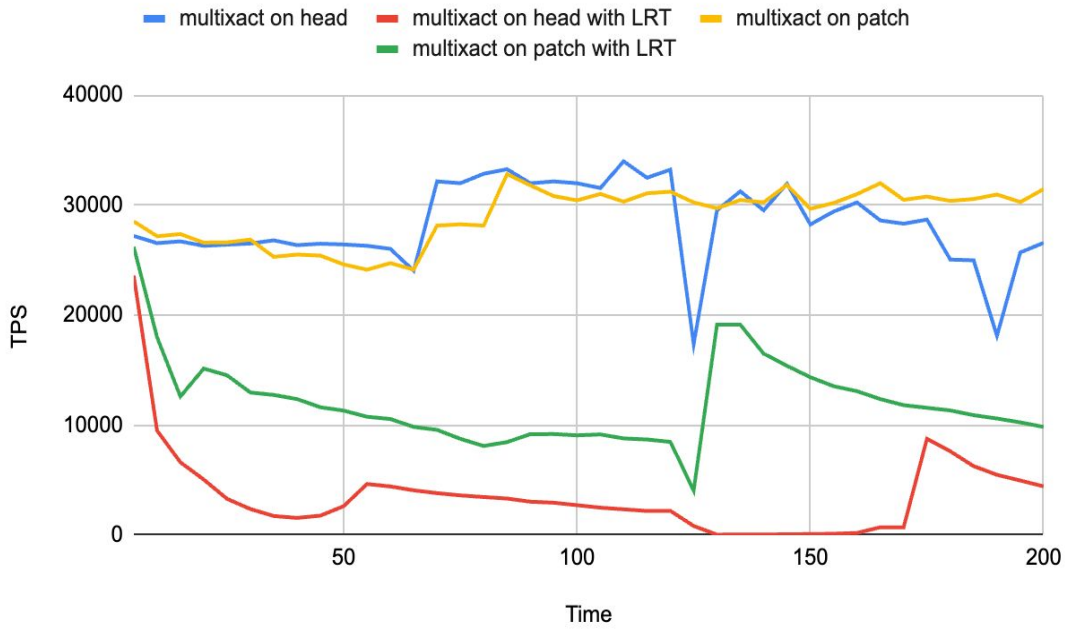
- *Test on 128 core machine*
- *Scale factor: 300*
- *Subtransaction_buffers: 1024*



Performance: Multixact contention

Effect of patch on TPS drop at higher clients in presence long running transaction

- *128 concurrent clients*
- *Scale factor: 300*
- *multixact_offsets_buffers: 256*
- *multixact_members_buffers: 512*



Summary

- SLRU design mainly has 3 main problems
  - Frequent buffer eviction due to small cache size
  - Contention on centralized control lock
  - LRU counter cause frequent CPU cache misses

- SLRU optimization
  - Make the cache size configurable
  - Divide SLRU cache into small associative banks
  - Implement bankwise lock to remove the centralized control lock contention
  - Remove centralize LRU counter
- Performance
  - 2-3 x performance gain when there is huge load on SLRU

Thank You!