Logging and Recovery
In Peloton

Gandeewan Raghuraman
Anirudh Kanjani
Aaron Tian
Motivation

To ensure durability in Peloton:

- in-memory self-driving SQL DBMS with fast transaction processing

- weakness: durability!
Not anymore.
Goals

✓ 75% - basic vanilla logging and recovery with group commit. **DONE**

✓ 100% - cache and contention friendly logging and recovery with group commit. **DONE**

✓ 125% - combine logging and checkpoint through epochs

★ Star - Improved logging, checkpointing, and recovery system
Single threaded logging
  - with maximal work distribution at the worker threads.

Commit logic moved to worker threads.

Group Commits

Delta logging for updates.
  - captured in the codegen
Recovery

✓ Two phase recovery (similar to Aries):

- **Phase 1**: Discard aborted transactions.

- **Phase 2**: Replay committed transactions.

✓ Assumptions:

- Log records for the all the committed transactions fit in memory.
Code quality

- Changes in traffic cop are production-quality
- Changes in moody-camel queue are production-quality
- Logging code is production-quality
- Recovery code works
Benchmarks
17% overhead
Unexpected issues

- Commit being done by network threads instead of worker threads
- Catalog changes not using codegen
- Abort issue
- Update issues
- Need for tuple ids
Invalid SQL statement in multi-statement transaction does not abort the transaction properly #1296

One update updates the values multiple times | Halloween Problem #1222

Updates change the order and offset of attributes #1223
Future work and improvements

- Add support for tuple ids in peloton
- Log tuple ids and incorporate it in recovery
- Integrate logging and recovery with checkpoints
- Probably switch to MVCC with delta storage.
Thank you