Write-Behind Logging
Motivation

● Non-Volatile Memory (NVM) introduced fast and byte-addressable read/writes.
  ○ Almost as fast as DRAM
● Strategies in Write-Ahead Logging (WAL) doesn’t utilize the capabilities of NVM.
● WAL may also strain the NVM unnecessarily, reducing its lifetime.
Application Domain

- Maximize the utilization of NVM capabilities
  - Faster recovery after failure
  - Less duplicated data
  - Fewer writes

- Support Multi-Version Concurrency Control (MVCC)
  - A tuple can have multiple versions that is visible to different transactions

- Support replicated databaseservers
Write-Behind Logging

- Changes to the database is flushed to the NVM during regular transaction processing
  - Logs only track *what* has changed rather than *how* it has changed
  - The log is always slightly behind the contents of the database
  - Reduces data duplication

- Very fast failure recovery
  - No need for REDO
  - UNDO involves simply ignoring the uncommitted transactions
    - Garbage collection removes them in the background
Life of a transaction

1. Client initiates an INSERT, DELETE or UPDATE
2. DBMS writes the changes to the Table Heap and adds an entry to the Dirty Tuple Table (DTT)
3. Take the related changes from DTT and construct a “Group Commit”
4. Compute \( C_p \) and \( C_d \) (\( C_p < C_d \))
5. Write changes to durable storage
6. Sync log entry
“Commit timestamp gap”: The range \((C_p, C_d)\).

- **UNDO** - DBMS ignores all changes inside the Commit timestamp gaps
- **Garbage collection** regularly cleans up “Gaps”
- **No REDO** - All committed changes are always persisted.
Benchmarking

● NVM storage is emulated using Persistent Memory Evaluation Platform (PMEP)
  ○ Existing NVM storage solutions are too costly and limited in space
● Peloton is used as DBMS, because it supports NVM
● Two benchmark was conducted
  ○ YCSB: Representative of transactions handled by web based companies
  ○ TPC-C: Simulates an order-entry application of a wholesale supplier
● Compares Runtime Performance, Recovery time and storage footprint
Runtime Performance

**Figure 14: YCSB Throughput** – The throughput of the DBMS for the YCSB benchmark with different logging protocols and durable storage devices.

**Figure 15: YCSB Latency** – The latency of the DBMS for the YCSB benchmark with different logging protocols and durable storage devices.
Recovery Time

- Sub-second recovery time for WBL with NVM
- WBL doesn’t need to conduct REDO
Storage Footprint

- WBL on NVM has 31% smaller footprint
  - Less unnecessary data duplication

(a) YCSB Storage Footprint

(b) TPC-C Storage Footprint
Conclusion

- Significant faster recovery time
- Smaller storage footprint by 1.5x
- No significant improvements in runtime performance
- NVM is expensive