WRITE BEHIND LOGGING

Authors: Joy Arulraj, Matthew Peron, Andrew Pavlo
        (Computer Science @ CMU)
Presenter: Devesh Kumar Singh
OUTLINE

- Background
- Storage Devices
- Write Ahead Protocol
- Write Behind Protocol
- Evaluation
BACKGROUND
DATABASE TRANSACTION PROPERTIES

Durability of updates: Persist committed transactions

Failure Atomicity: Dispose aborted transactions
DBMS FAILURE SCENARIOS

Transaction failure: Aborted by DBMS/application

System failure: Hardware failure, bugs in DBM/OS

Media failure: Data loss, storage corruption
DATA MANAGEMENT POLICY

- Steal
  - Grab buffer-pool frames from uncommitted transactions
  - Can lose dirty writes, but better performance

- No Force
  - Don’t force transaction updates to disk before committing
  - Difficult to guarantee durability, but better performance

<table>
<thead>
<tr>
<th>No Force</th>
<th>Desired</th>
</tr>
</thead>
<tbody>
<tr>
<td>Steal</td>
<td>Trivial</td>
</tr>
</tbody>
</table>
DATA LOGGING POLICIES

Changes added to a log on durable storage, then send to durable storage

- Redo log
  - Reapply updates of committed transactions

- Undo log:
  - Reverses updates by failed transactions
STORAGE DEVICES
HDD: OLD BUT NOT GOLD

- Magnetic storage platters based
- High data density/ Low storage price per capacity
- Random access slower than sequential access
- Slowest speeds due to mechanical design choices
**SDD: FASTER BUT NOT BETTER**

- NAND-based flash memory based
- Read/Write 100-1000x faster than HDD
- Storage cell durable for fixed # of writes
- 3-10x expensive than HDD
NVM: BEST OF BOTH WORLDS

- Low latency, byte sized reads/writes of DRAM
- Persistent writes, large storage capacity of HDD/SDDs
- Cache line granularity, High bandwidth, Low latency to CPU’s
Synchronized file write throughput to a 64 GB file

<table>
<thead>
<tr>
<th></th>
<th>HDD</th>
<th>SSD</th>
<th>NVM</th>
</tr>
</thead>
<tbody>
<tr>
<td>Sequential Writes</td>
<td>0.1</td>
<td>1</td>
<td>100</td>
</tr>
<tr>
<td>Random Writes</td>
<td>0.02</td>
<td>0.5</td>
<td>100</td>
</tr>
</tbody>
</table>
WRITE AHEAD LOGGING
# DATA STRUCTURES

## WAL Record

<table>
<thead>
<tr>
<th>LSN</th>
<th>Log Rec Type</th>
<th>Transaction Commit Timestamp</th>
<th>Table ID</th>
<th>Insert Location</th>
<th>Delete Location</th>
<th>Before/After Images</th>
</tr>
</thead>
</table>

## Dirty Page Table

<table>
<thead>
<tr>
<th>TxId</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
</table>

## Active Transaction Table

| activeTxId | latestLSN |
COMMIT PROTOCOL

During Transaction

<table>
<thead>
<tr>
<th>txId</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>-</td>
<td>Active</td>
</tr>
</tbody>
</table>

- rec1, rec2, rec3

<table>
<thead>
<tr>
<th>txId</th>
<th>lastLSN</th>
<th>status</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>28</td>
<td>Commit</td>
</tr>
</tbody>
</table>

Diagram:
- DRAM
- NVM
- Database
- Checkpoints

Steps:
1. Database
2. Data
3. Checkpoints
In memory DBMS skips Undo phase
**SAMPLE RECOVERY RUN**

<table>
<thead>
<tr>
<th>LSN</th>
<th>WRITE AHEAD LOG</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BEGIN CHECKPOINT</td>
</tr>
<tr>
<td>2</td>
<td>END CHECKPOINT (EMPTY ATT)</td>
</tr>
<tr>
<td>3</td>
<td>TXN 1: INSERT TUPLE 100 (NEW: X)</td>
</tr>
<tr>
<td>4</td>
<td>TXN 2: UPDATE TUPLE 2 (NEW: Y')</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>22</td>
<td>TXN 20: DELETE TUPLE 20</td>
</tr>
<tr>
<td>23</td>
<td>TXN 1, 3,…, 20: COMMIT</td>
</tr>
<tr>
<td>24</td>
<td>TXN 2: UPDATE TUPLE 100 (NEW: X')</td>
</tr>
<tr>
<td>25</td>
<td>TXN 21: UPDATE TUPLE 21 (NEW: Z')</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
<tr>
<td>84</td>
<td>TXN 80: DELETE TUPLE 80</td>
</tr>
<tr>
<td>85</td>
<td>TXN 2, 21,…, 79: COMMIT</td>
</tr>
<tr>
<td>86</td>
<td>TXN 81: UPDATE TUPLE 100 (NEW: X'')</td>
</tr>
<tr>
<td></td>
<td>SYSTEM FAILURE</td>
</tr>
</tbody>
</table>
WRITE BEHIND LOGGING
# Data Structures

**WBL record**

- LSN
- Log Record Type
- Persisted commit Timestamp
- Dirty Commit Timestamp

**Dirty Tuple table**

- TX id
- Table id
- Tuple location
**RUNTIME OPERATION**

- **Operation** → **Finish**
- **TX changes** → **DRAM**
- **Tuple changes** → **DTT**

- Cp: Commit timestamp of latest committed transaction
- Cd: Commit timestamp not assigned to any transaction before the next group commit finishes
- Group Commit: Flashes a batch a log records in a single write to durable storage
Dirty tuples, \( (C_p, C_d) \), 
Long running tx \( C_p \)

dt1
dt2
\( (C_p, C_d) \)
# Sample Run

<table>
<thead>
<tr>
<th>LSN</th>
<th>Write Behind Log</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>BEGIN CHECKPOINT</td>
</tr>
<tr>
<td>2</td>
<td>END CHECKPOINT (EMPTY CTG)</td>
</tr>
<tr>
<td>3</td>
<td>{ (1, 100) }</td>
</tr>
<tr>
<td>4</td>
<td>{ 2, (21, 120) }</td>
</tr>
<tr>
<td>5</td>
<td>{ 80, (81, 180) }</td>
</tr>
<tr>
<td></td>
<td>SYSTEM FAILURE</td>
</tr>
</tbody>
</table>
EVALUATION
Yahoo’s YCSB
1 table with 2 mil tuples (2 GB)

TPC-C
5 Tx types, 88% reads, 12% updates, 100k tuples (1 GB)

Read-heavy,
90% reads, 10% updates

Balanced
50% reads, 50% updates

Write-heavy
10% reads, 90% updates
THROUGHPUT

![Throughput Chart]

- HDD
- SSD
- NVM

Tx/sec

- WAL
- WBL
RECOVERY TIME

![Bar chart showing recovery time for HDD, SSD, and NVM with WAL and WBL](chart.jpg)
THANK YOU