# WRITE BEHIND LOGGING

Authors : Joy Arulraj, Matthew Peron, Andrew Pavlo

(Computer Science @ CMU)

Presenter: Devesh Kumar Singh

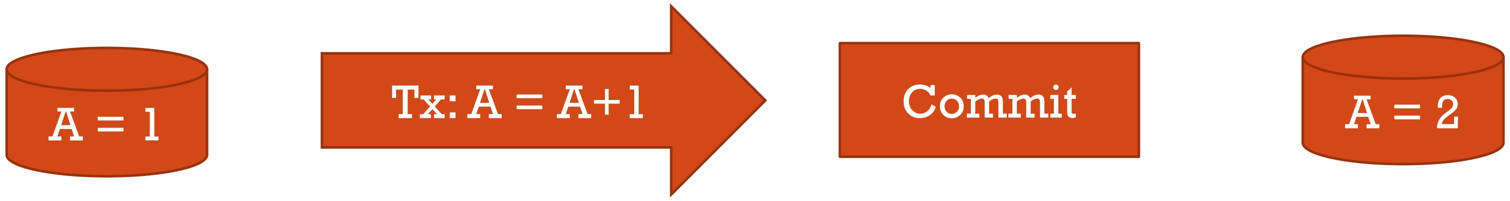
# OUTLINE

- Background
- Storage Devices
- Write Ahead Protocol
- Write Behind Protocol
- Evaluation

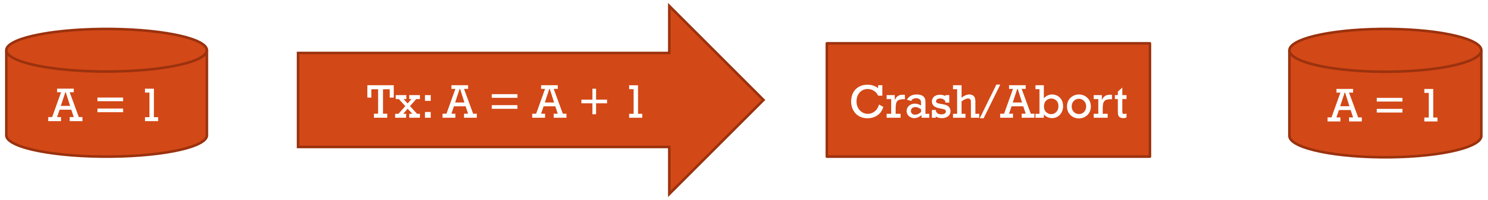
# BACKGROUND

# 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

| | No Steal | Steal |
|---|---|---|
| **No Force** | | **Desired** |
| **Force** | **Trivial** | |

# 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

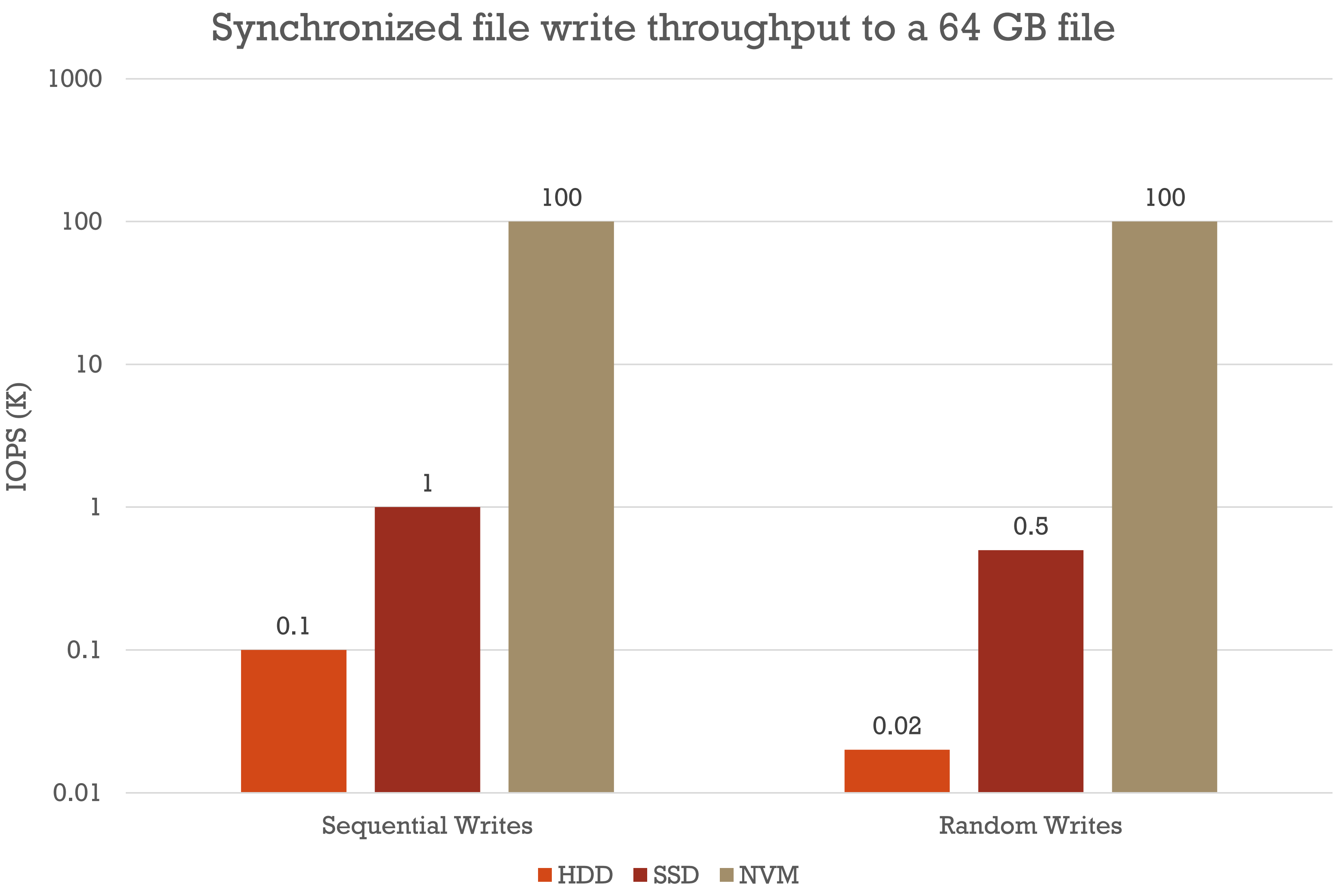
- Magnetic storage platters based
- High data density/ Low storage price per capacity
- Random access slower than sequential access
- Slowest speeds due to mechanical design choices

# SDD: FASTER BUT NOT BETTER

- NAND-based flash memory based
- Read/Write 100-1000x faster then HDD
- Storage cell durable for fixed # of writes
- 3-10x expensive then 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
IOPS (K)
1000
100
10
1
0.1
0.01
0.1
1
100
0.02
0.5
100
Sequential Writes
Random Writes
HDD
SSD
NVM

# WRITE AHEAD LOGGING

# DATA STRUCTURES

WAL Record

| LSN | Log Rec Type | Transaction Commit Timestamp | Table ID | Insert Location | Delete Location | Before/After Images |
|---|---|---|---|---|---|---|

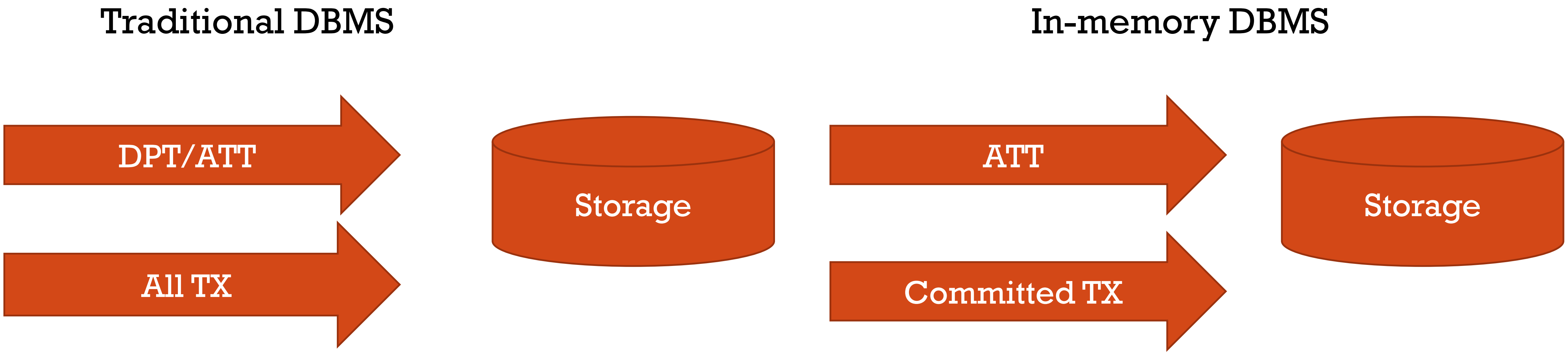
Dirty Page Table

| TxId | lastLSN | status |
|---|---|---|

Active Transaction Table

| activeTxId | latestLSN |
|---|---|

# RUNTIME OPERATIONS



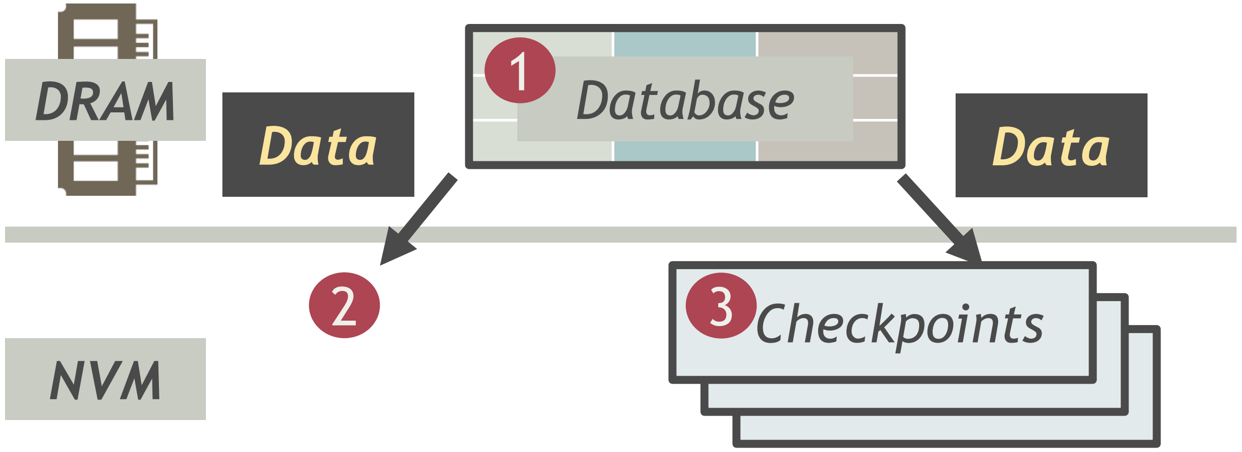
# COMMIT PROTOCOL

During Transaction

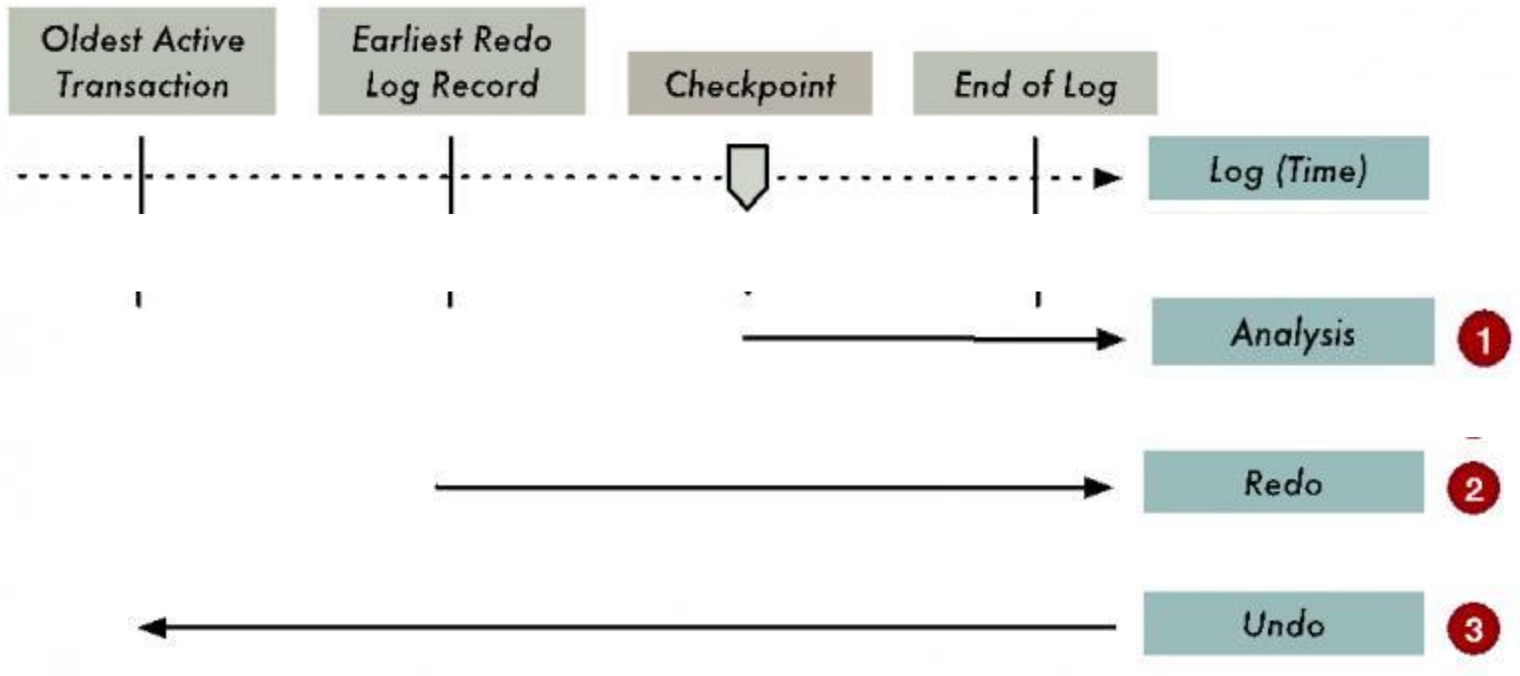
| txId | lastLSN | status |
| --- | --- | --- |
| 1 | - | Active |

rec1,rec2
,rec3

| txId | lastLSN | status |
| --- | --- | --- |
| 1 | 28 | Commit |



# RECOVERY PROTOCOL




In memory DBMS skips Undo phase

# SAMPLE RECOVERY RUN

| LSN | WRITE AHEAD LOG |
|---|---|
| 1 | BEGIN CHECKPOINT |
| 2 | END CHECKPOINT (EMPTY ATT) |
| 3 | TXN 1: INSERT TUPLE 100 (NEW: X) |
| 4 | TXN 2: UPDATE TUPLE 2 (NEW: Y') |
| ... | ... |
| 22 | TXN 20: DELETE TUPLE 20 |
| 23 | TXN 1, 3,..., 20: COMMIT |
| 24 | TXN 2: UPDATE TUPLE 100 (NEW: X') |
| 25 | TXN 21: UPDATE TUPLE 21 (NEW: Z') |
| ... | ... |
| 84 | TXN 80: DELETE TUPLE 80 |
| 85 | TXN 2, 21,..., 79: COMMIT |
| 86 | TXN 81: UPDATE TUPLE 100 (NEW: X'') |
| | SYSTEM FAILURE |

# WRITE BEHIND LOGGING

# DATA STRUCTURES

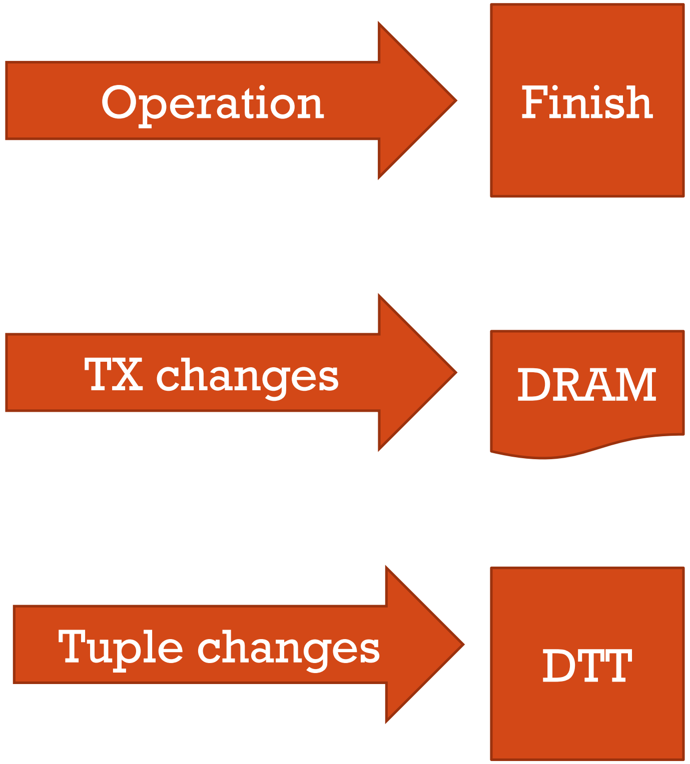
WBL record

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

Dirty Tuple table

| TX id | Table id | Tuple location |
|---|---|---|

# RUNTIME OPERATION

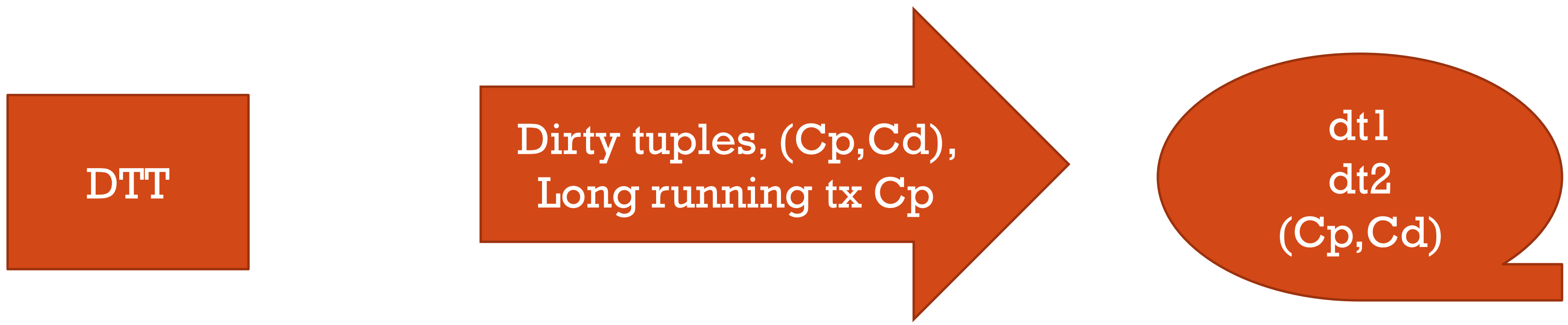


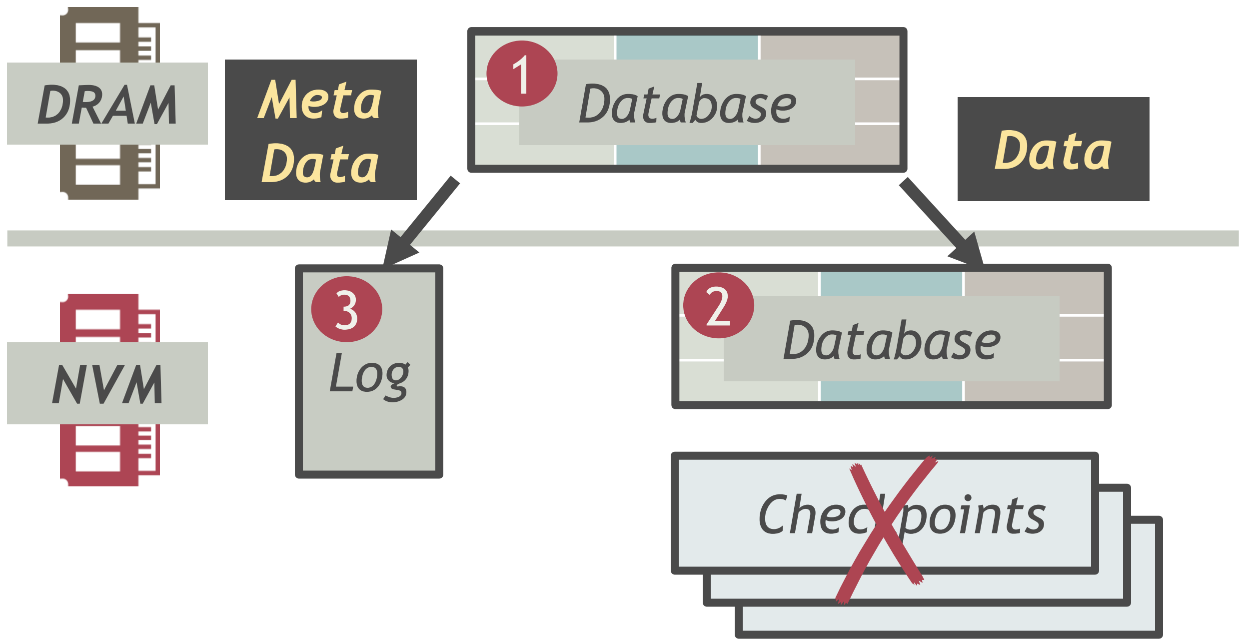

Cp: Commit timestamp of latest committed transaction

Cd: Commit timestamp not assigned to any transaction before the next group commit finishes

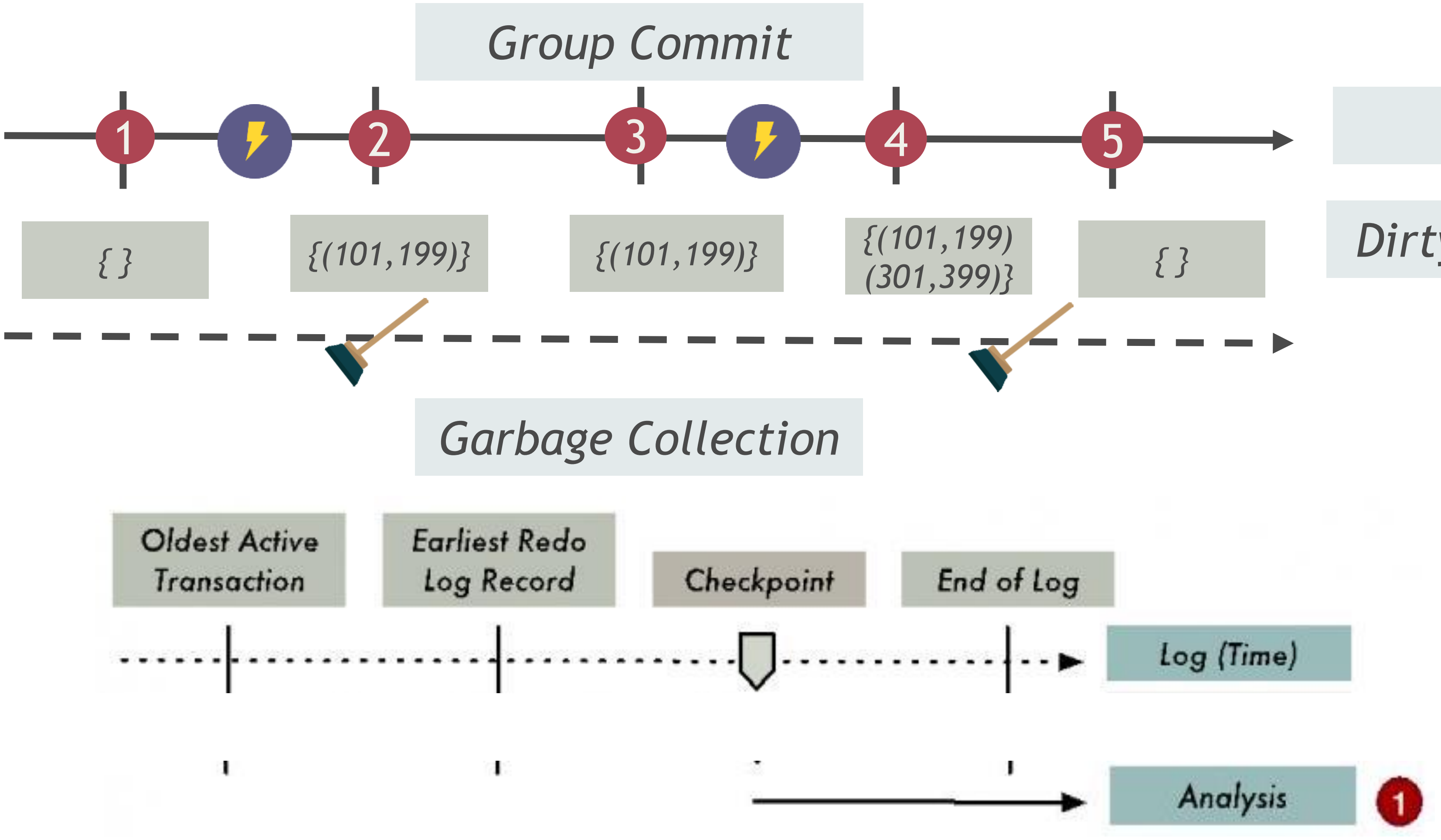
Group Commit: Flushes a batch a log records in a single write to durable storage

# COMMIT OPERATION



# RECOVERY OPERATION

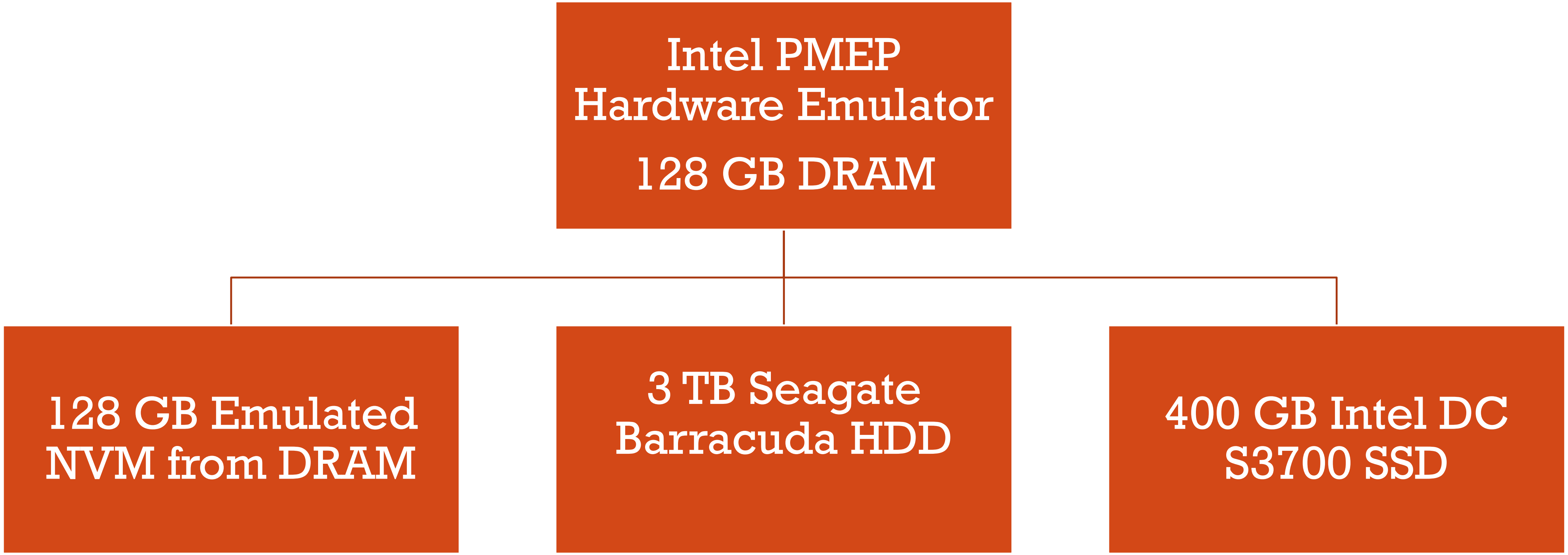


# SAMPLE RUN

| LSN | WRITE BEHIND LOG |
|---|---|
| 1 | BEGIN CHECKPOINT |
| 2 | END CHECKPOINT (EMPTY CTG) |
| 3 | { (1, 100) } |
| 4 | { 2, (21, 120) } |
| 5 | { 80, (81, 180) } |
| | SYSTEM FAILURE |

# EVALUATION

# PLATFORM


Intel PMEP
Hardware Emulator
128 GB DRAM
128 GB Emulated NVM from DRAM
3 TB Seagate Barracuda HDD
400 GB Intel DC S3700 SSD

# BENCHMARK

**Yahoo's YCSB**

1 table with 2 mil tuples (2 GB)

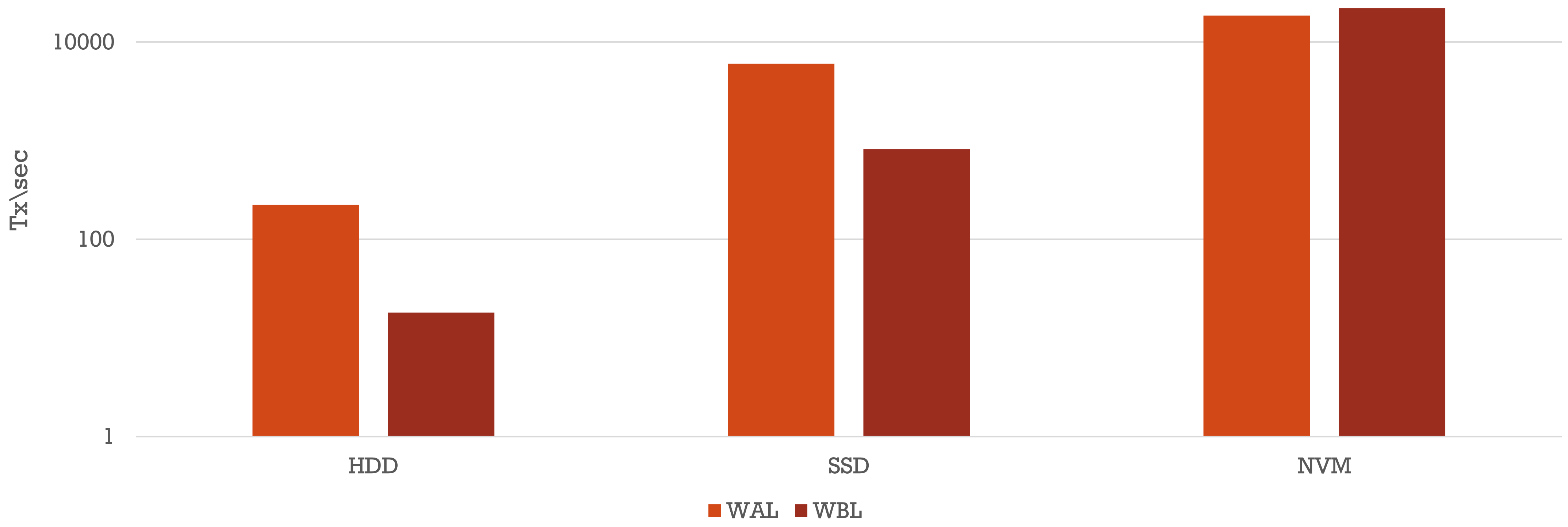
- **Read-heavy,**
  90% reads, 10% updates
- **Balanced**
  50% reads, 50% updates
- **Write-heavy**
  10% reads, 90% updates

**TPC-C**

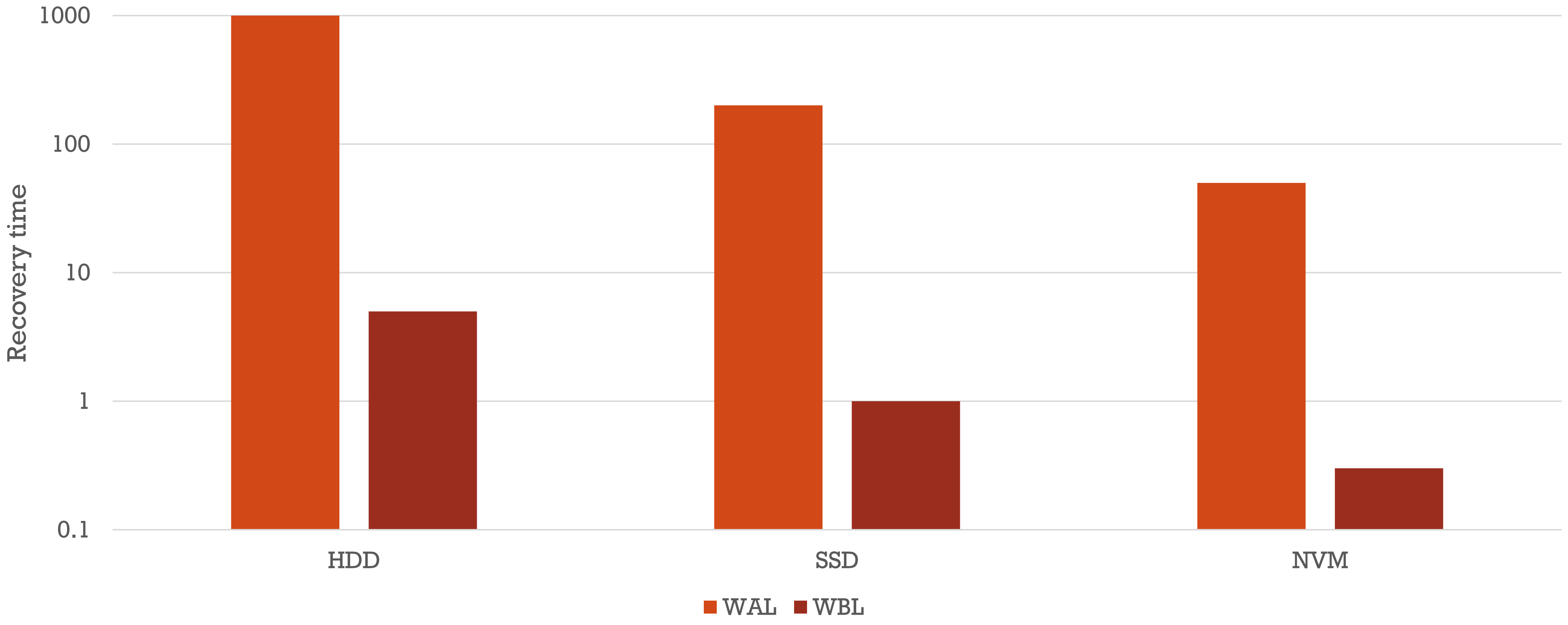
5 Tx types, 88% reads, 12% updates,

100k tuples(1 GB)

# THROUGHPUT



# RECOVERY TIME



# THANK YOU