NVRAM study

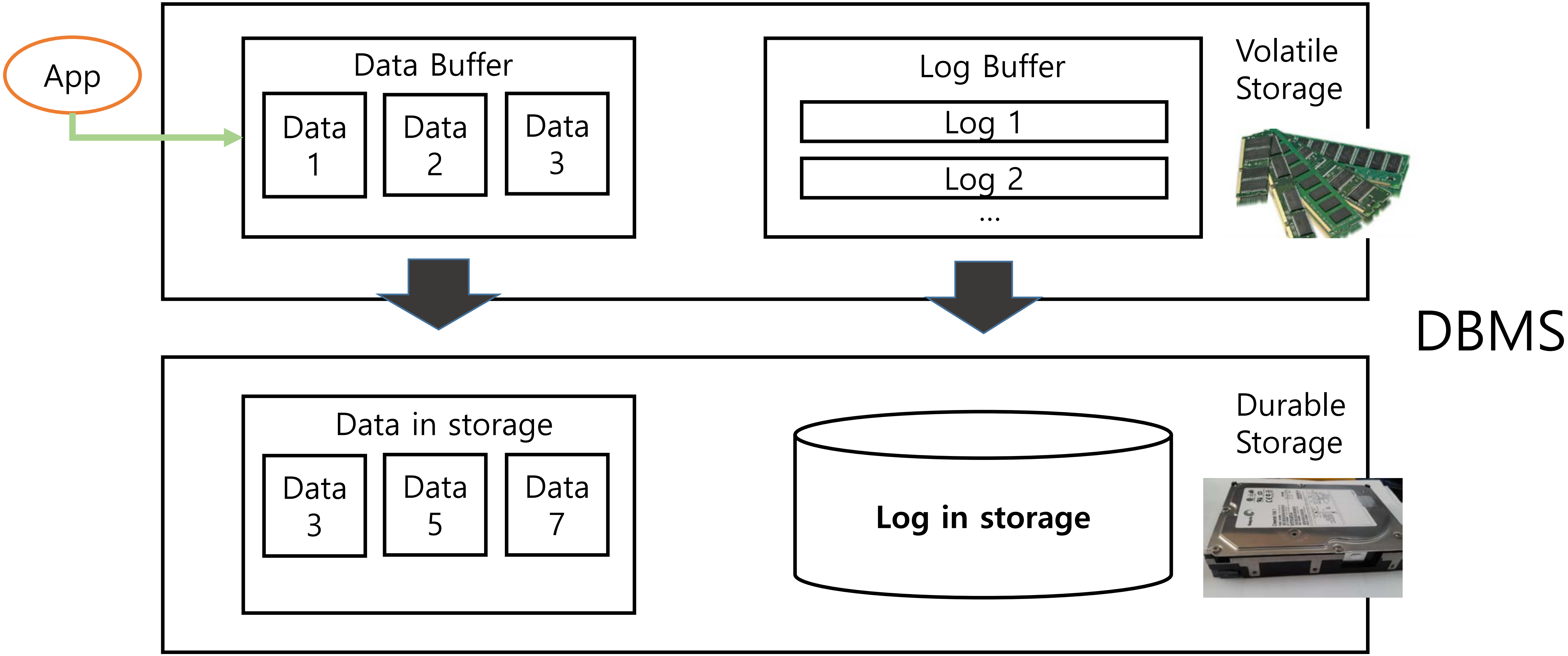
Write Behind Logging. Arulraj et al. , VLDB ‘16

박사과정 최원기

OutLine

- Background
- Write Ahead Logging(Previous mechanism)
- Write Behind Logging
- Evaluation

Background : Why we need Log?

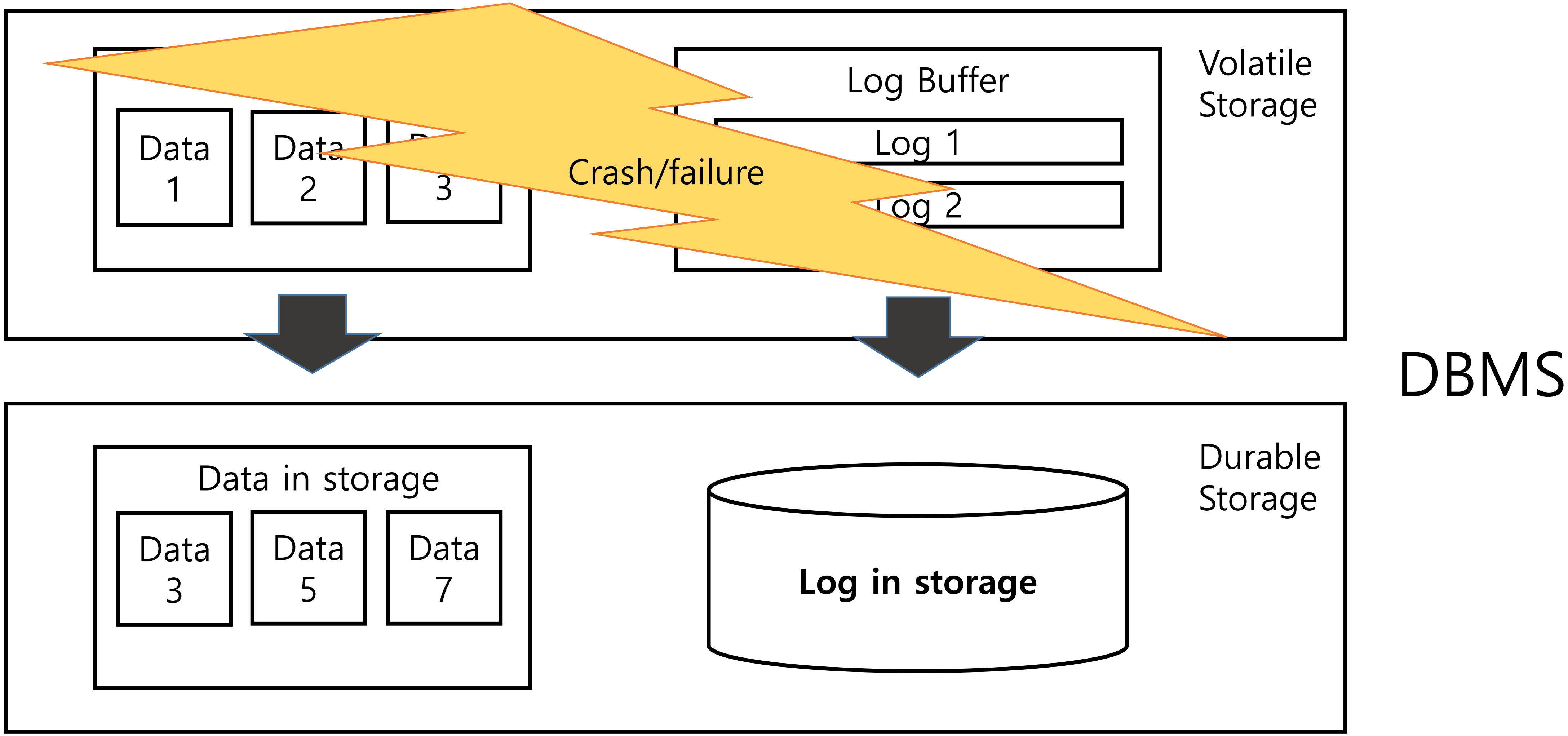


Background : Why we need Log?

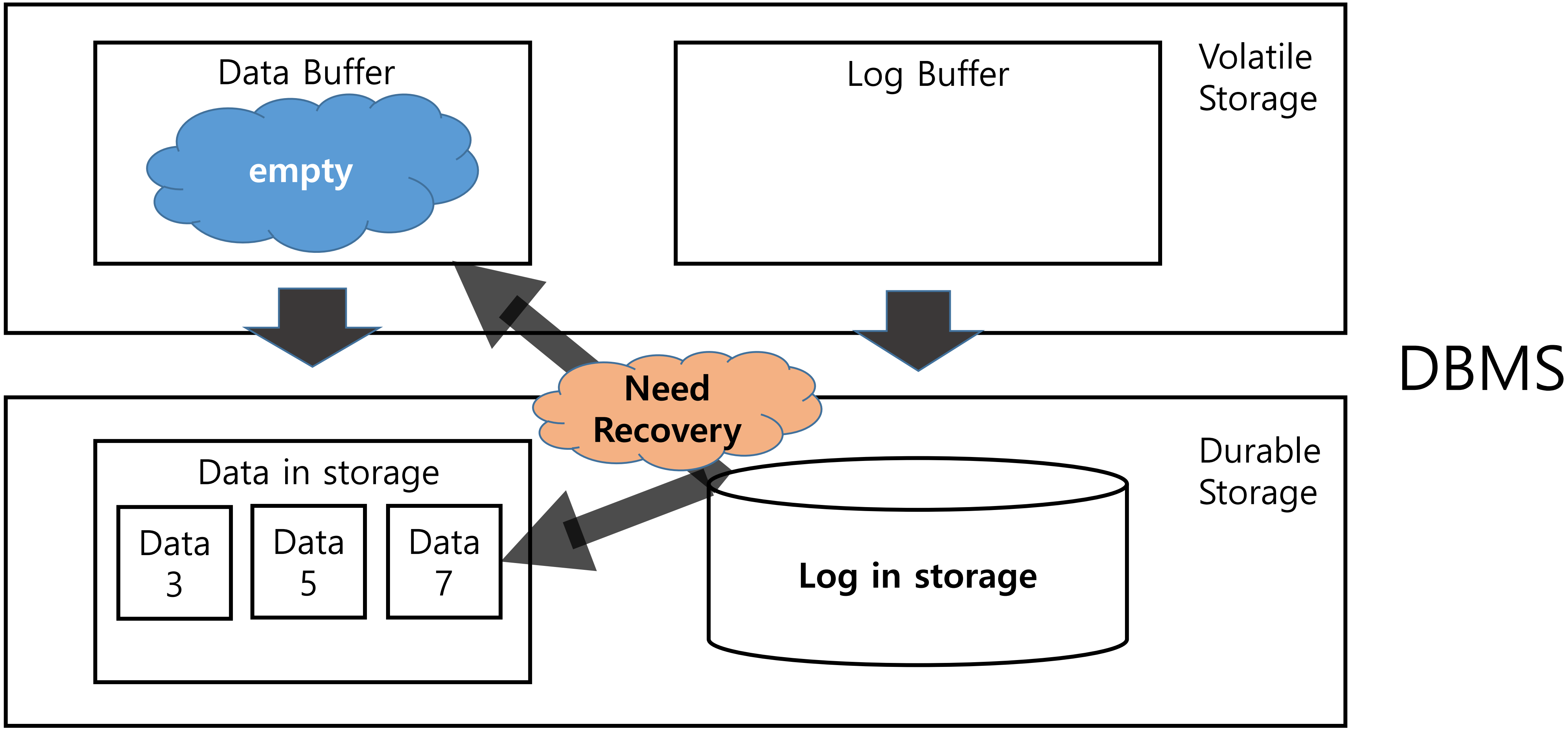
- Transactional failure
  - Aborted by DBMS
  - Aborted by application

- System failure
  - Hardware failure
  - Bugs in DBMS/OS

Background : Why we need Log?



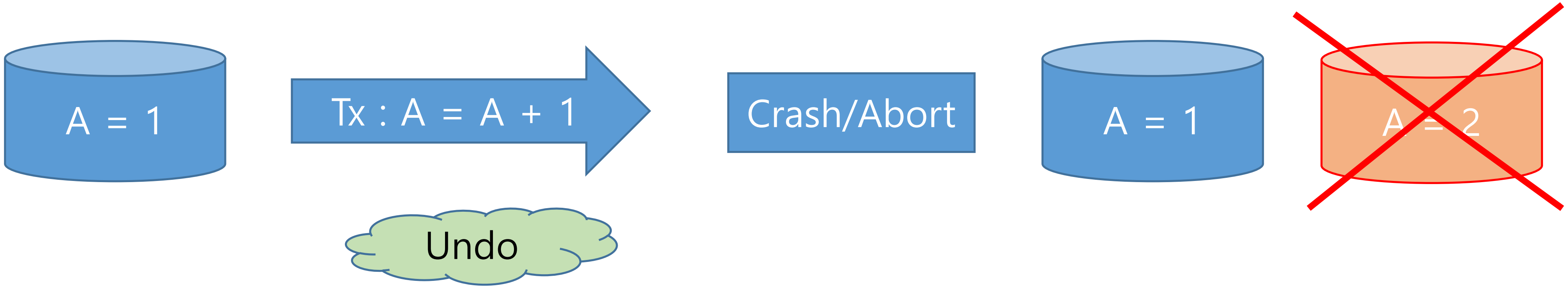
Background : Why we need Log?



Background : Why we need Log?

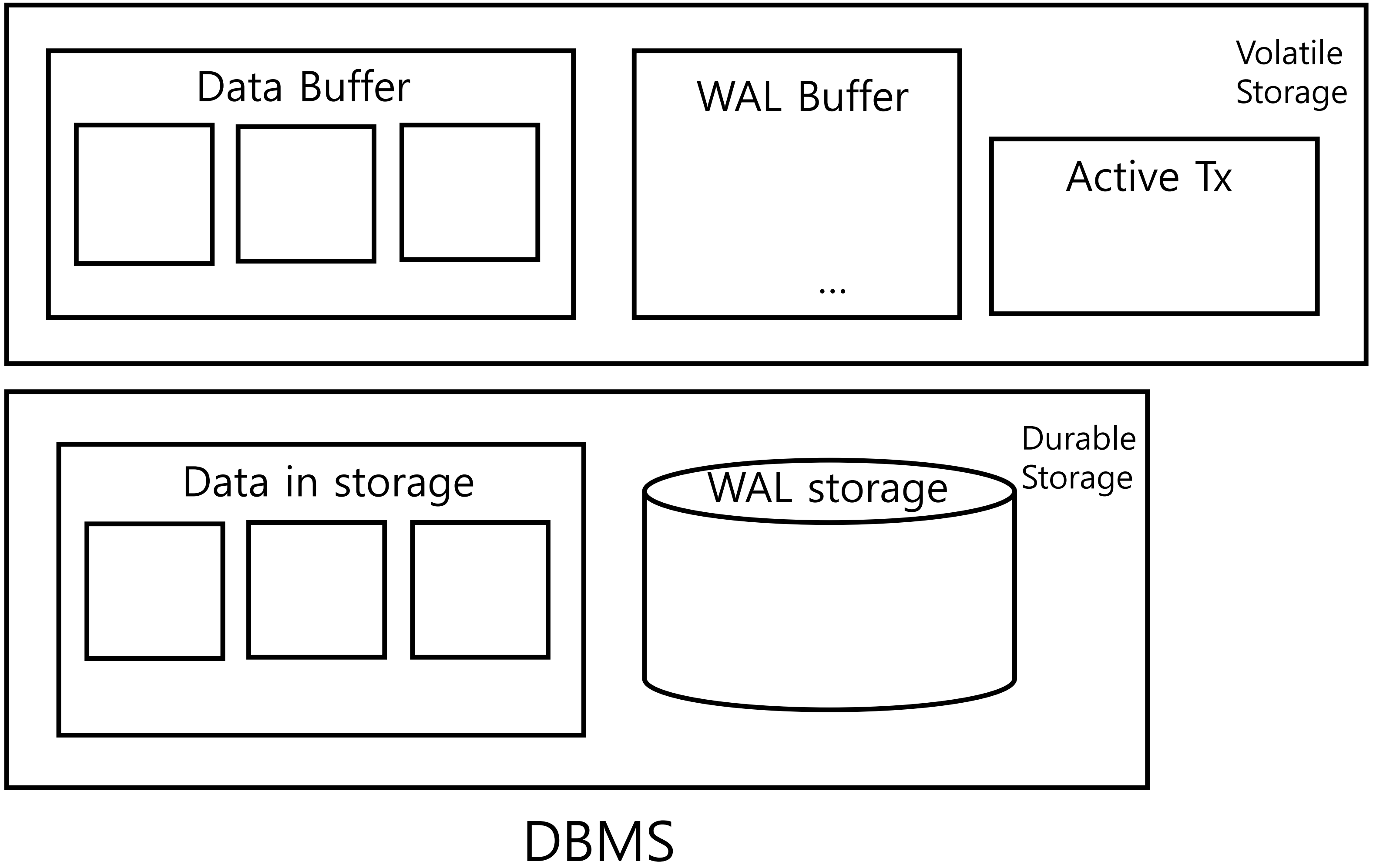
Durability of update : Persist committed transaction

Failure Atomicity : Dispose aborted transaction



Write Ahead Logging (WAL)

checksum	LSN	Type	Tx identifier	Location	after image	WAL record

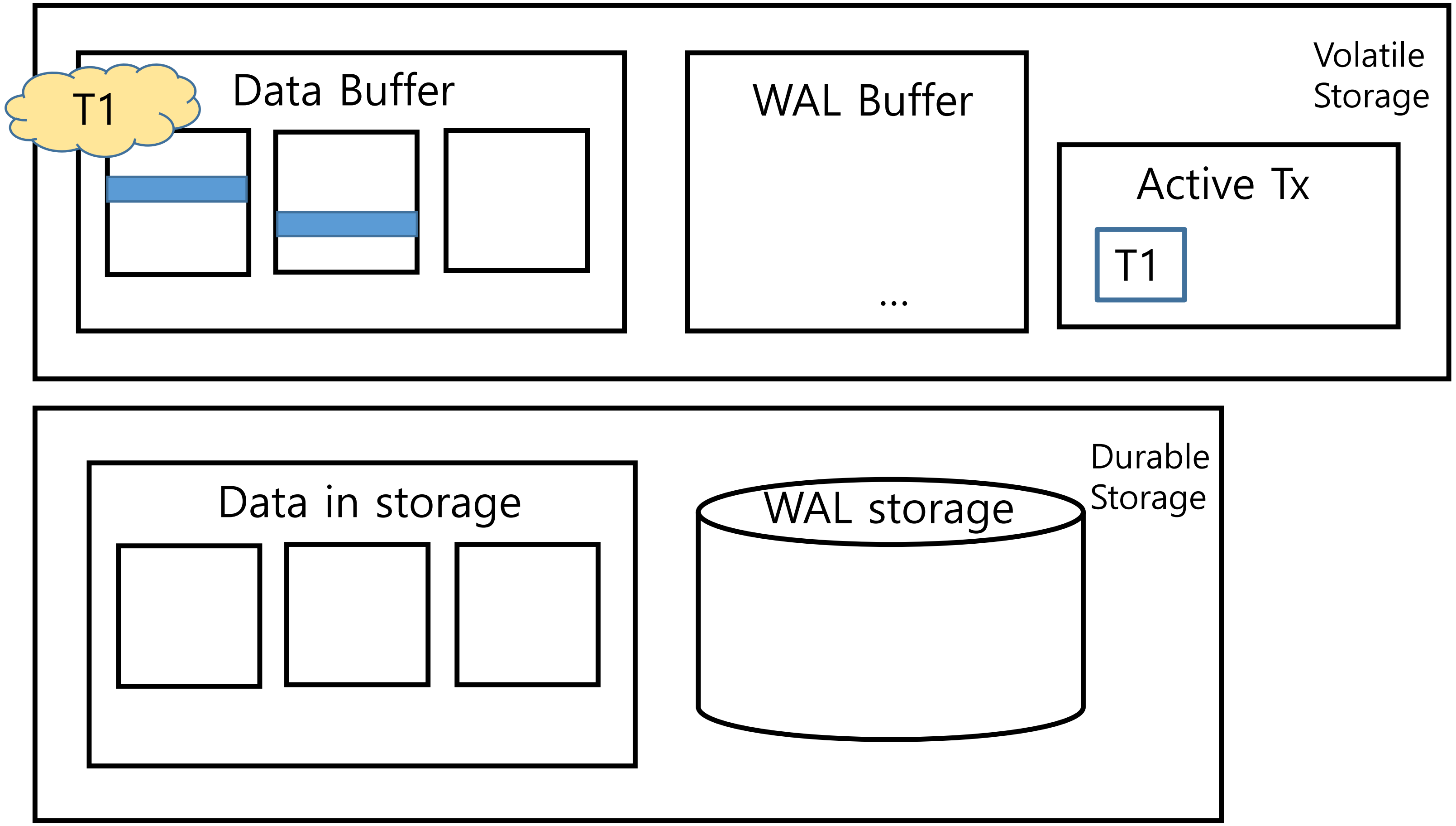


1. Write changes to data in buffer

Write Ahead Logging (WAL)

checksum	LSN	Type	Tx identifier	Location	after image

WAL record

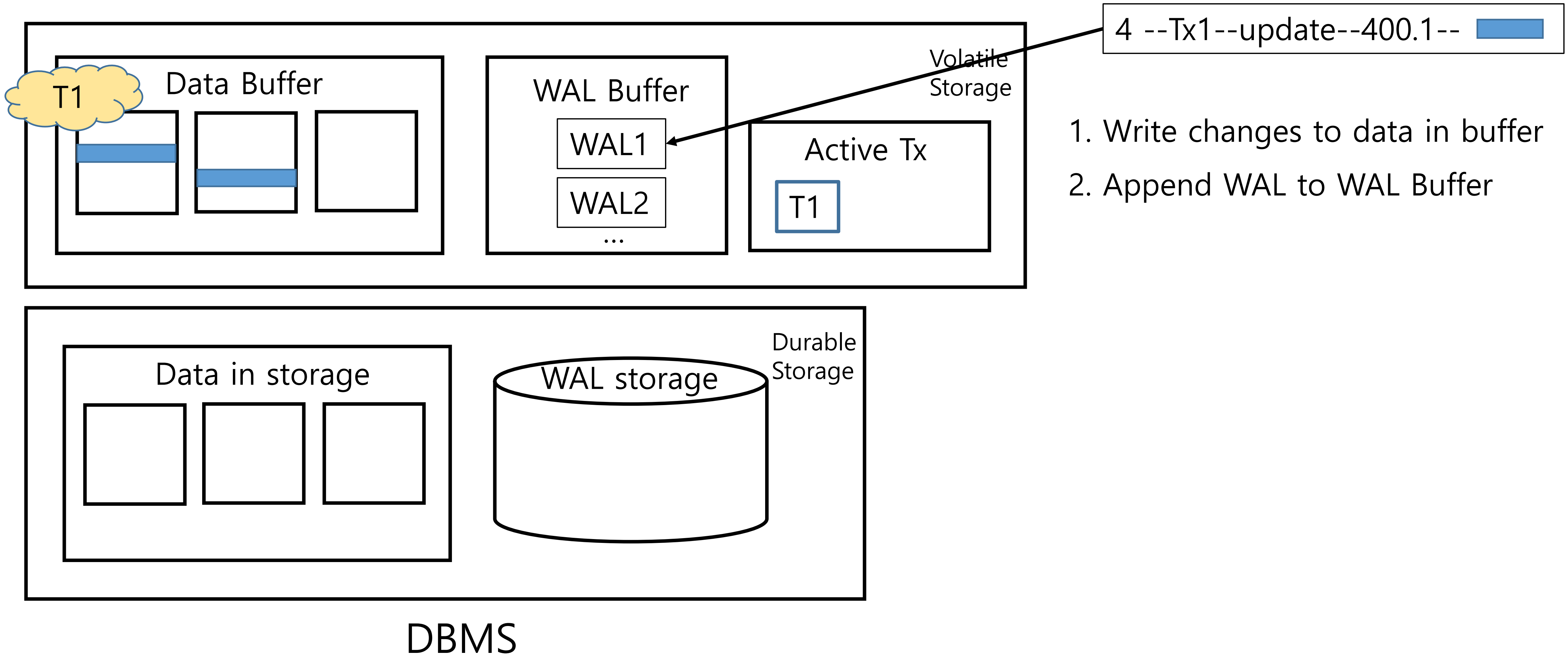


DBMS

1. Write changes to data in buffer

Write Ahead Logging (WAL)

checksum	LSN	Type	Tx identifier	Location	after image	WAL record

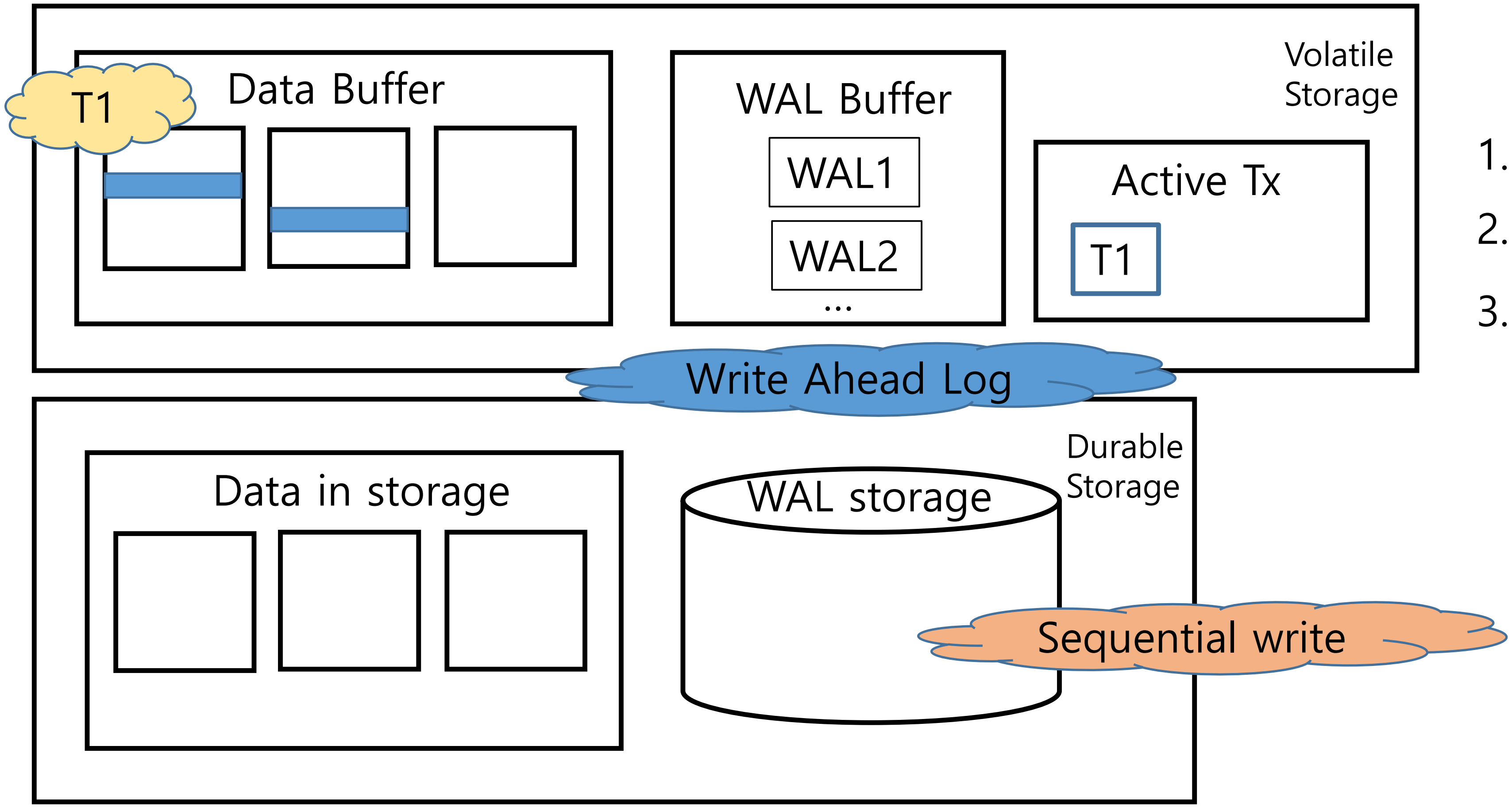


1. Write changes to data in buffer
2. Append WAL to WAL Buffer

Write Ahead Logging (WAL)

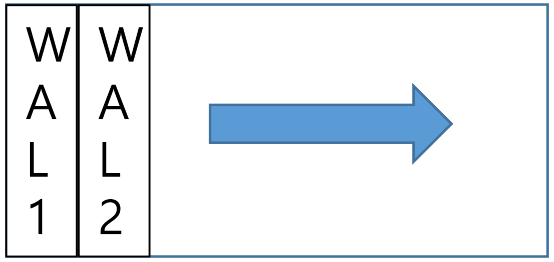
checksum	LSN	Type	Tx identifier	Location	after image

WAL record



DBMS

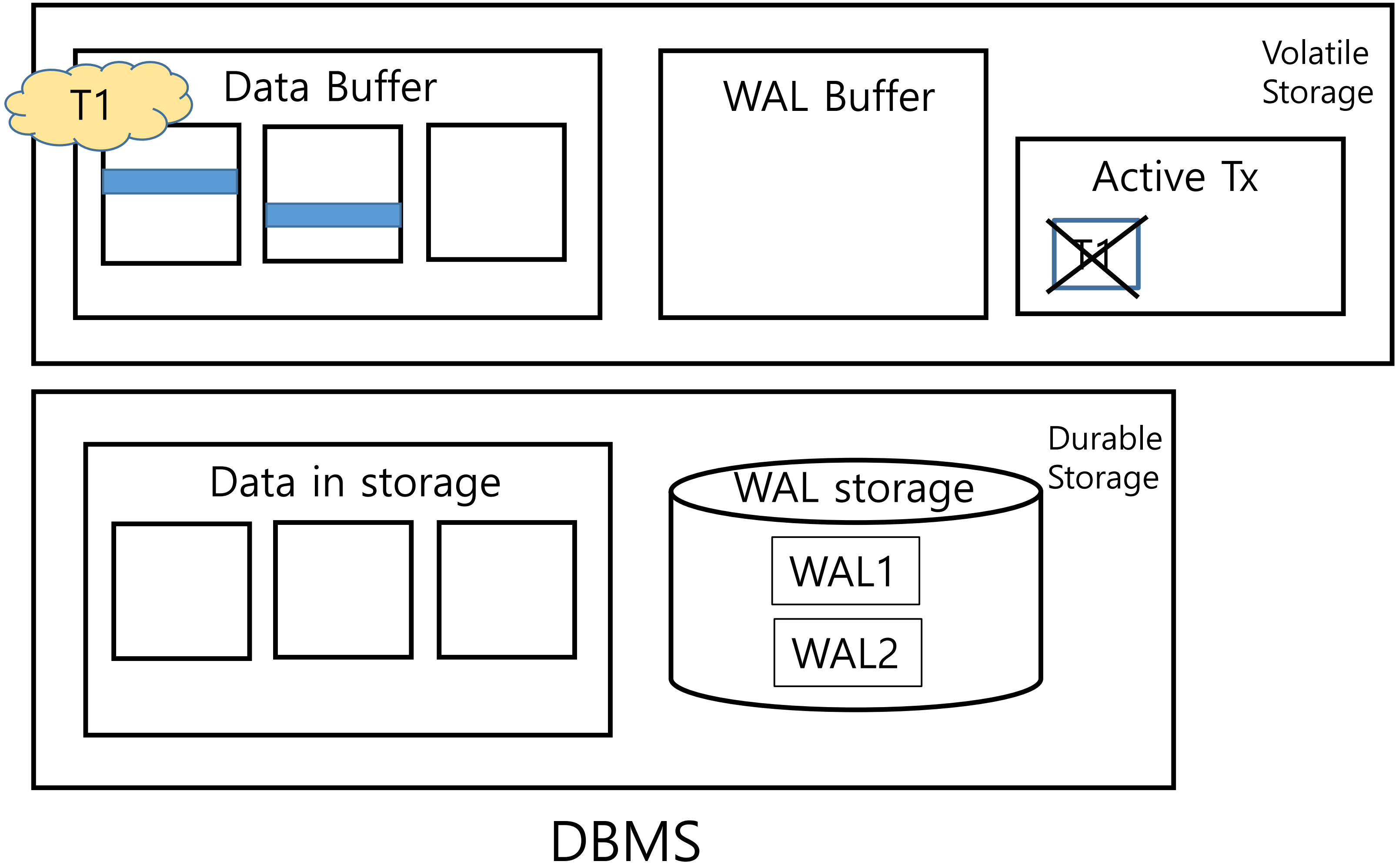
1. Write changes to data in buffer
2. Append WAL to WAL Buffer
3. Sync the WAL



Write Ahead Logging (WAL)

checksum	LSN	Type	Tx identifier	Location	after image

WAL record

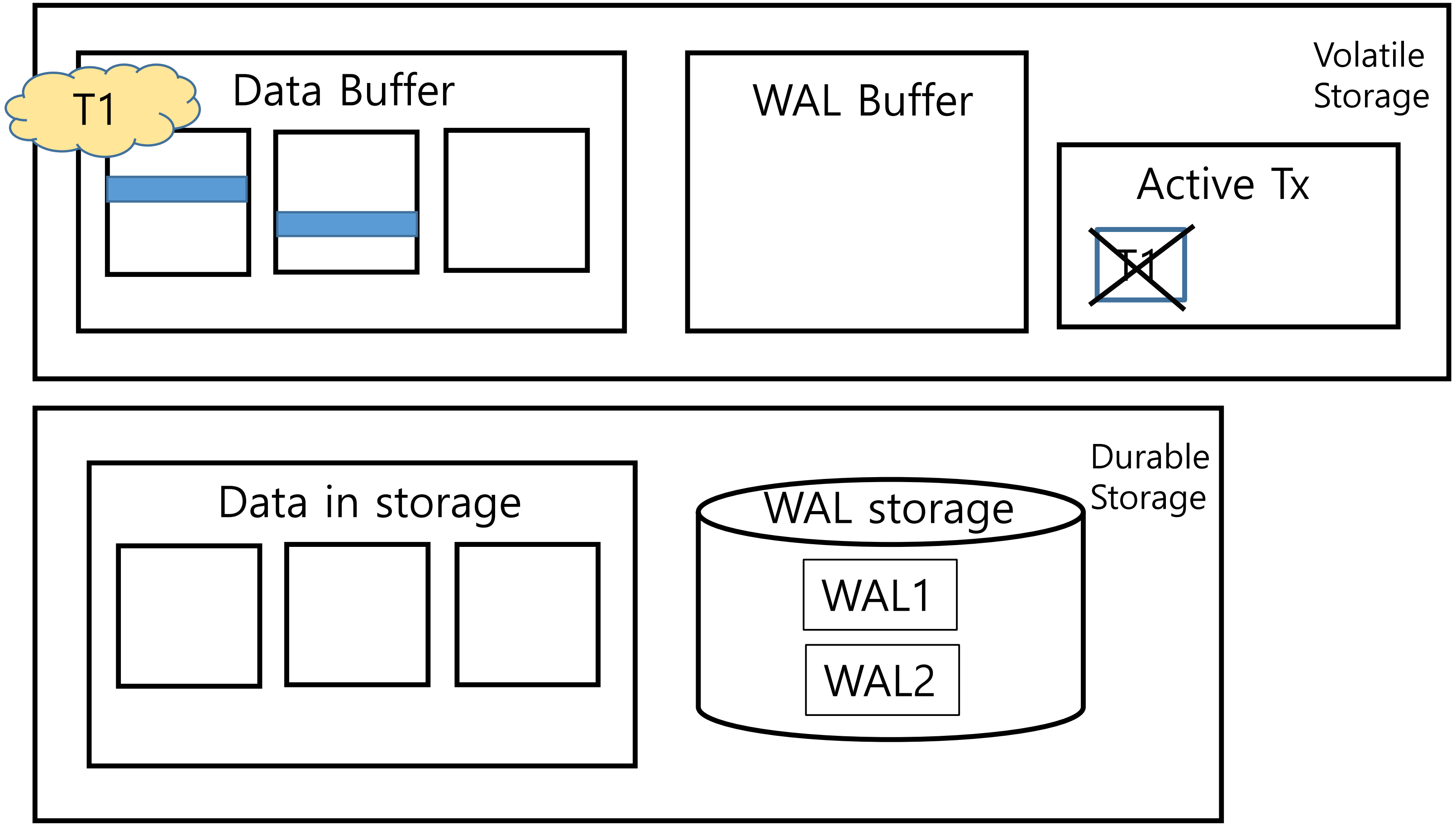


1. Write changes to data in buffer
2. Append WAL to WAL Buffer
3. Sync the WAL
4. Mark transaction commit

Write Ahead Logging (WAL)

checksum	LSN	Type	Tx identifier	Location	after image

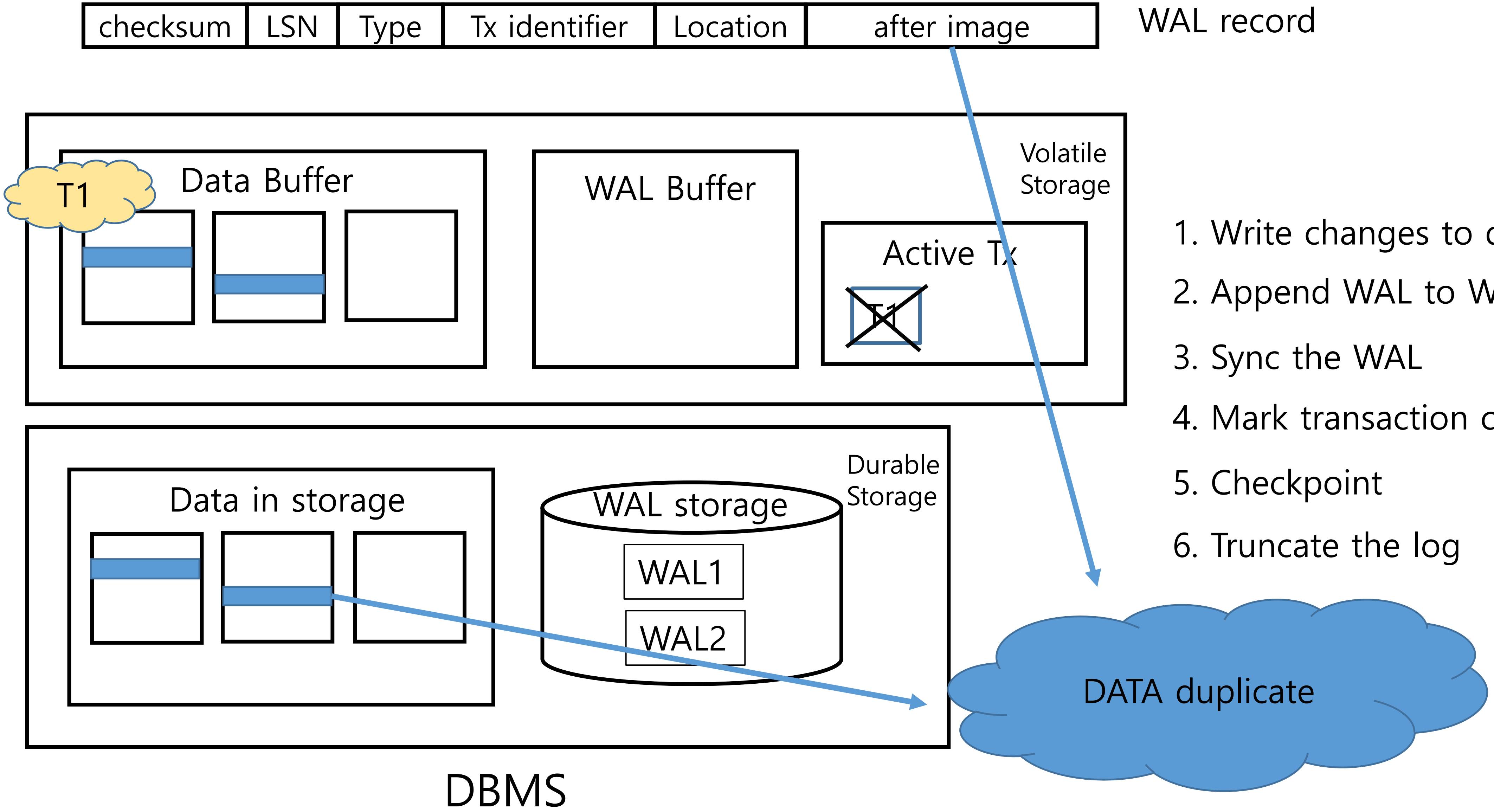
WAL record



DBMS

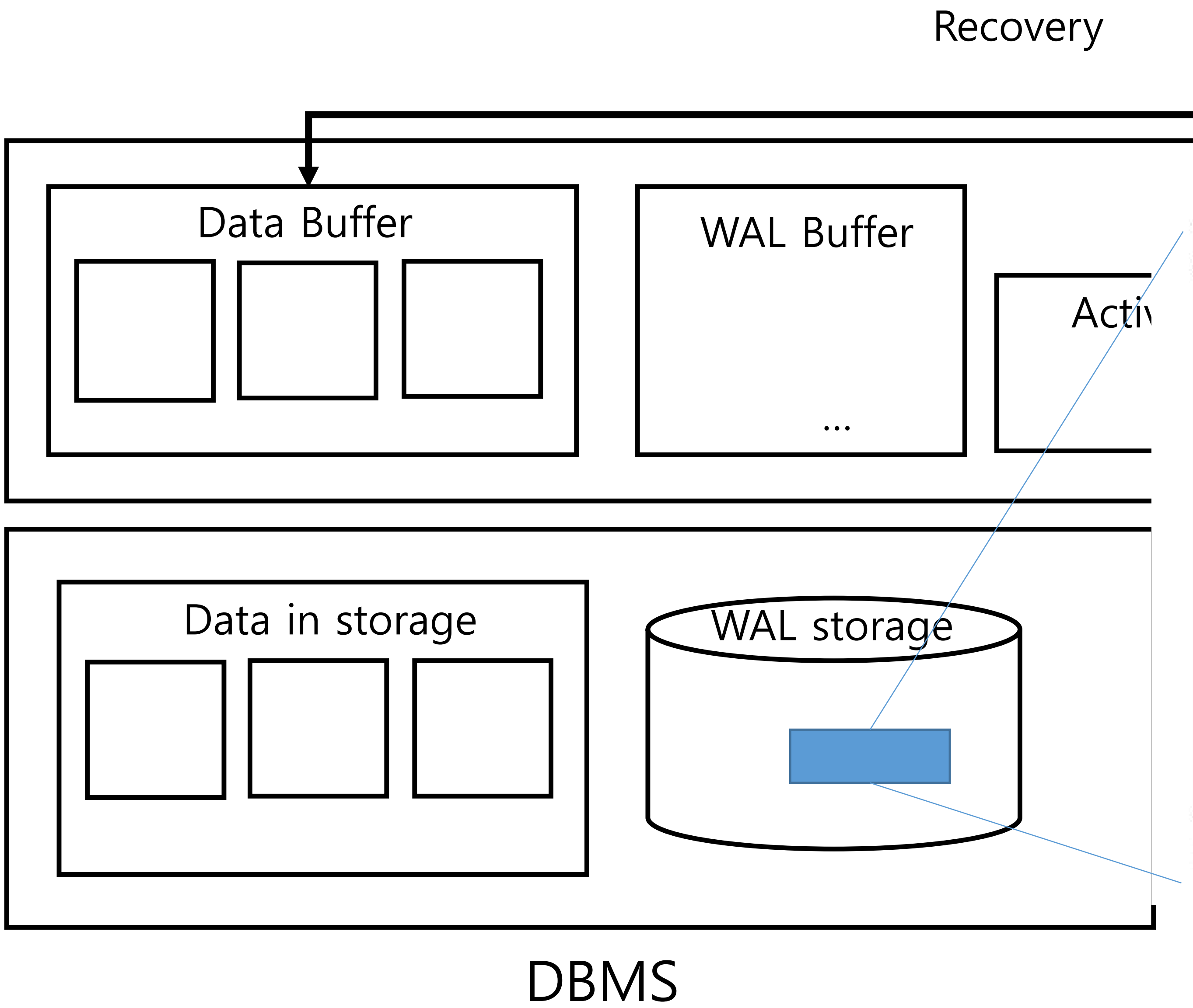
1. Write changes to data in buffer
2. Append WAL to WAL Buffer
3. Sync the WAL
4. Mark transaction commit
5. Checkpoint

Write Ahead Logging (WAL)



1. Write changes to data in buffer
2. Append WAL to WAL Buffer
3. Sync the WAL
4. Mark transaction commit
5. Checkpoint
6. Truncate the log

Write Ahead Logging (WAL) Recovery

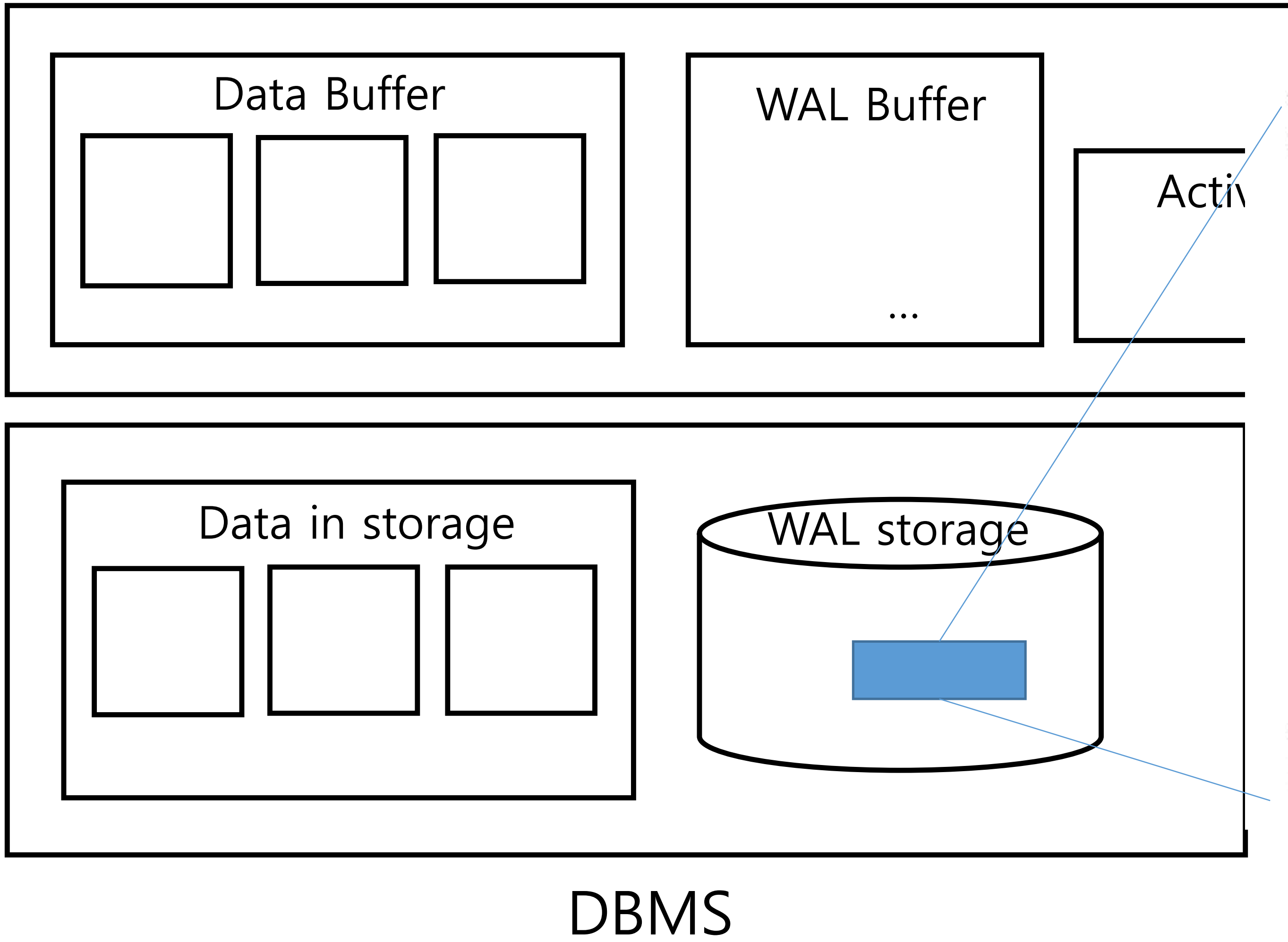


LSN	WRITE AHEAD LOG
1	BEGIN CHECKPOINT
2	END CHECKPOINT (EMPTY ATT)
3	TXN 1: INSERT TUPLE 100 (NEW: X)
4	TXN 2: UPDATE TUPLE 2 (NEW: Y′)
...	...
22	TXN 20: DELETE TUPLE 20
23	TXN 1, 3,..., 20: COMMIT
24	TXN 2: UPDATE TUPLE 100 (NEW: X′)
25	TXN 21: UPDATE TUPLE 21 (NEW: Z′)
...	...
84	TXN 80: DELETE TUPLE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

Figure 7: WAL Example – Contents of the WAL during recovery.

analysis

Write Ahead Logging (WAL) Recovery

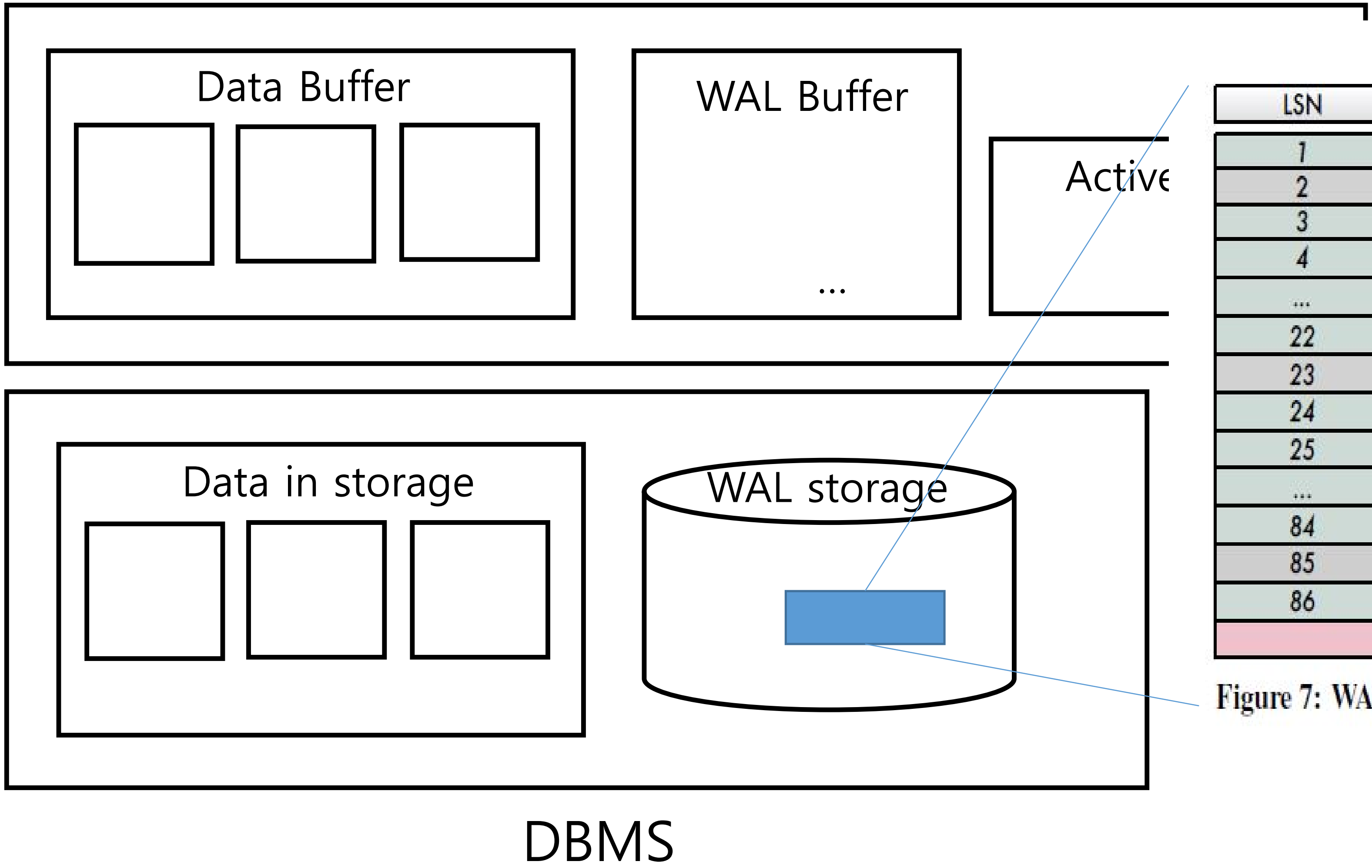


LSN	WRITE AHEAD LOG
1	BEGIN CHECKPOINT
2	END CHECKPOINT (EMPTY ATT)
3	TXN 1: INSERT TUPLE 100 (NEW: X)
4	TXN 2: UPDATE TUPLE 2 (NEW: Y')
...	...
22	TXN 20: DELETE TUPLE 20
23	TXN 1, 3,..., 20: COMMIT
24	TXN 2: UPDATE TUPLE 100 (NEW: X')
25	TXN 21: UPDATE TUPLE 21 (NEW: Z')
...	...
84	TXN 80: DELETE TUPLE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

Redo

Figure 7: WAL Example – Contents of the WAL during recovery.

Write Ahead Logging (WAL) Recovery



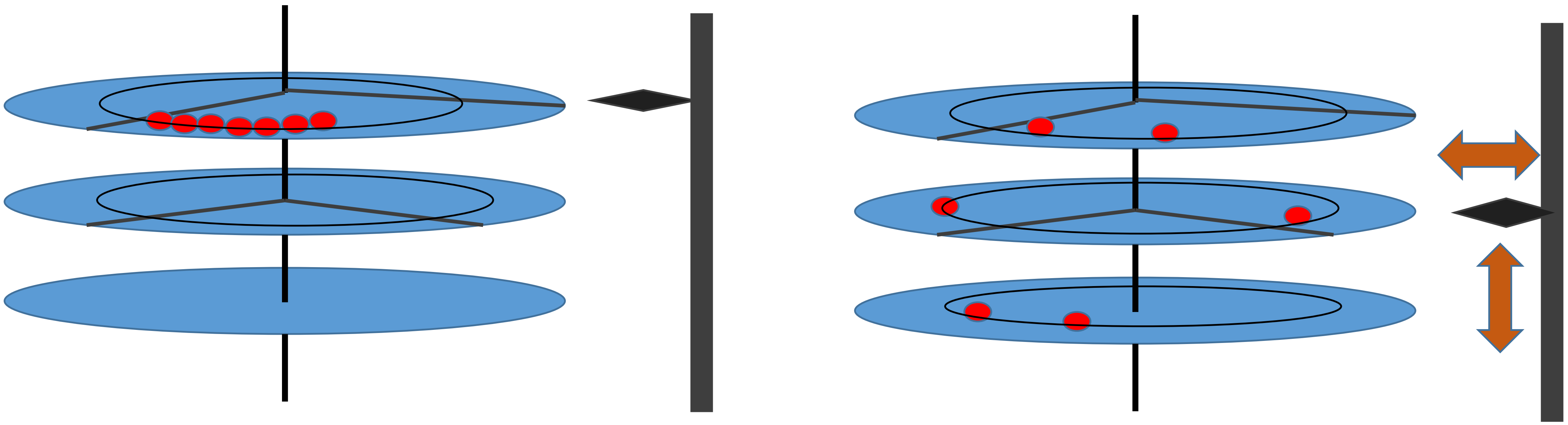
LSN	WRITE AHEAD LOG
1	BEGIN CHECKPOINT
2	END CHECKPOINT (EMPTY ATT)
3	TXN 1: INSERT TUPLE 100 (NEW: X)
4	TXN 2: UPDATE TUPLE 2 (NEW: Y')
...	...
22	TXN 20: DELETE TUPLE 20
23	TXN 1, 3,..., 20: COMMIT
24	TXN 2: UPDATE TUPLE 100 (NEW: X')
25	TXN 21: UPDATE TUPLE 21 (NEW: Z')
...	...
84	TXN 80: DELETE TUPLE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

Figure 7: WAL Example – Contents of the WAL during recovery.

Undo

Uncommitted : Tx 80, Tx 81

Sequential Write vs Random Write



Sequential pattern

Random pattern

Hard Disk Drive

In case SSD(solid state drive), because of parallelism, sequential write is faster than random write

NVM(Non-volatile memory)

- Next Generation Storage
- Fast like DRAM, Non-Volatile unlike DRAM
- Byte-Addressable
- Gap between sequential and random write performance is small

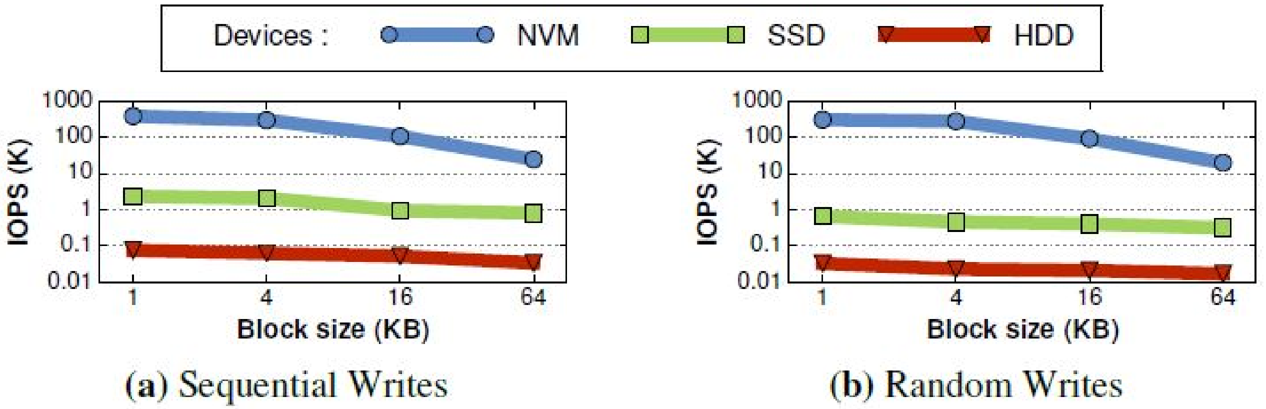
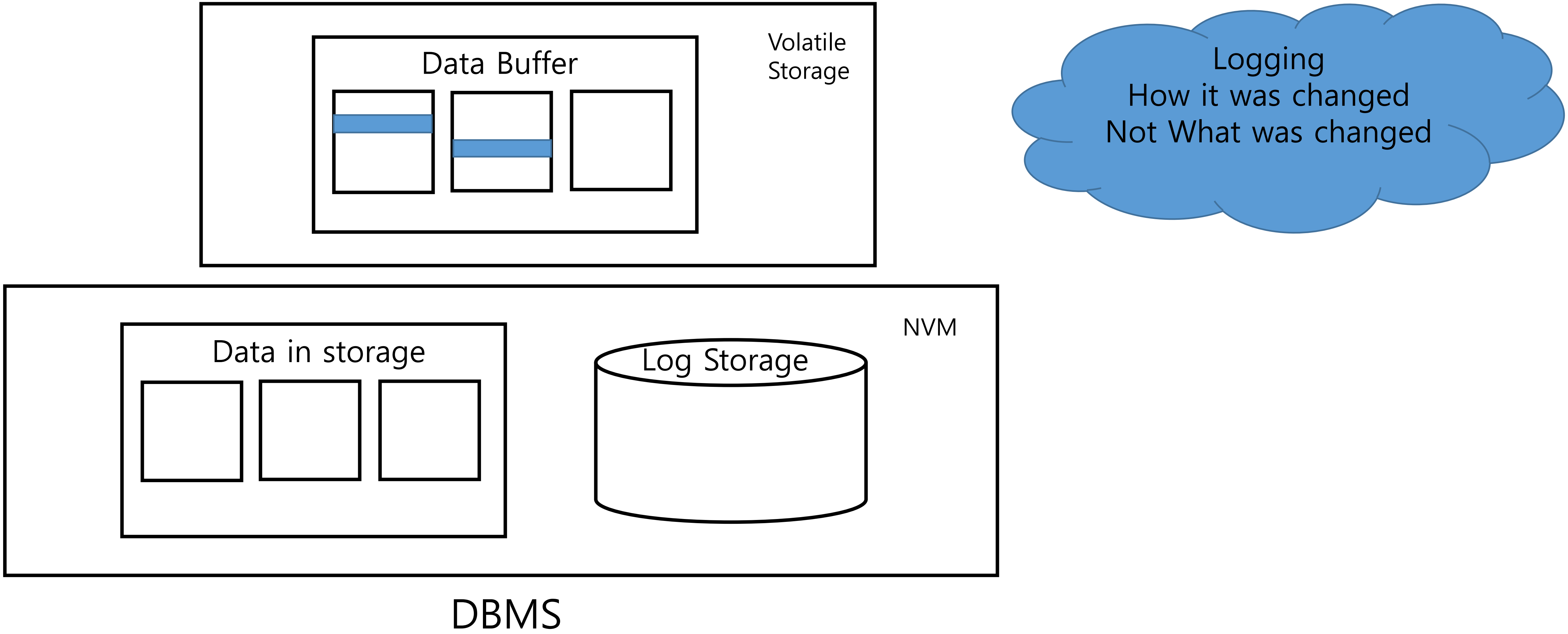


Figure 1: I/O Performance – Synchronous file write throughput obtained on different storage devices including emulated NVM, SSD, and HDD.

Write Behind Logging (Paper's proposal)

checksum	LSN	Type	Persisted Commit Timestamp	Dirty Commit Timestamp

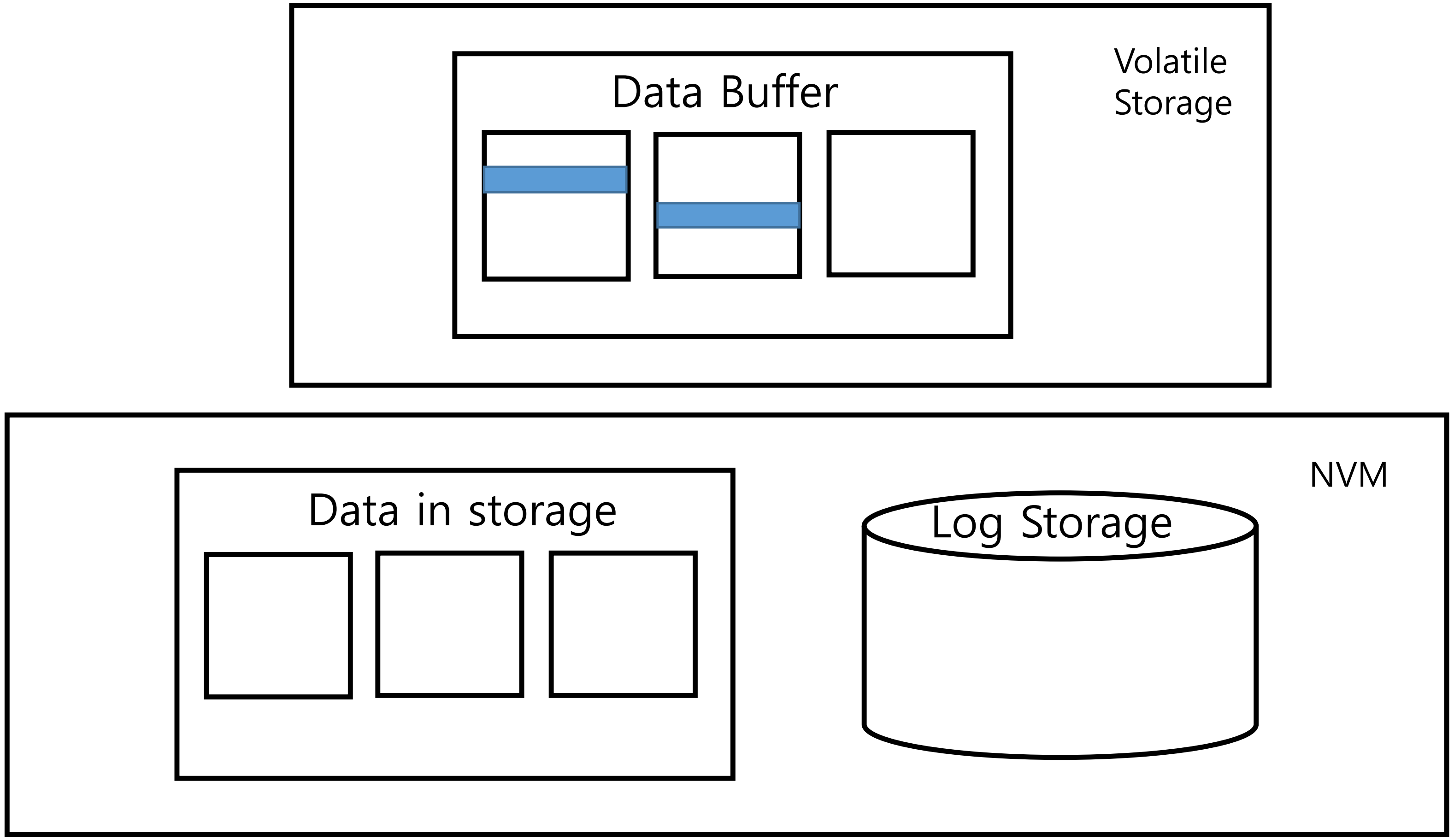
WBL record



Write Behind Logging

checksum	LSN	Type	Persisted Commit Timestamp	Dirty Commit Timestamp

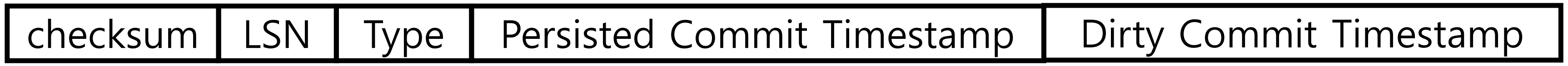
WBL record



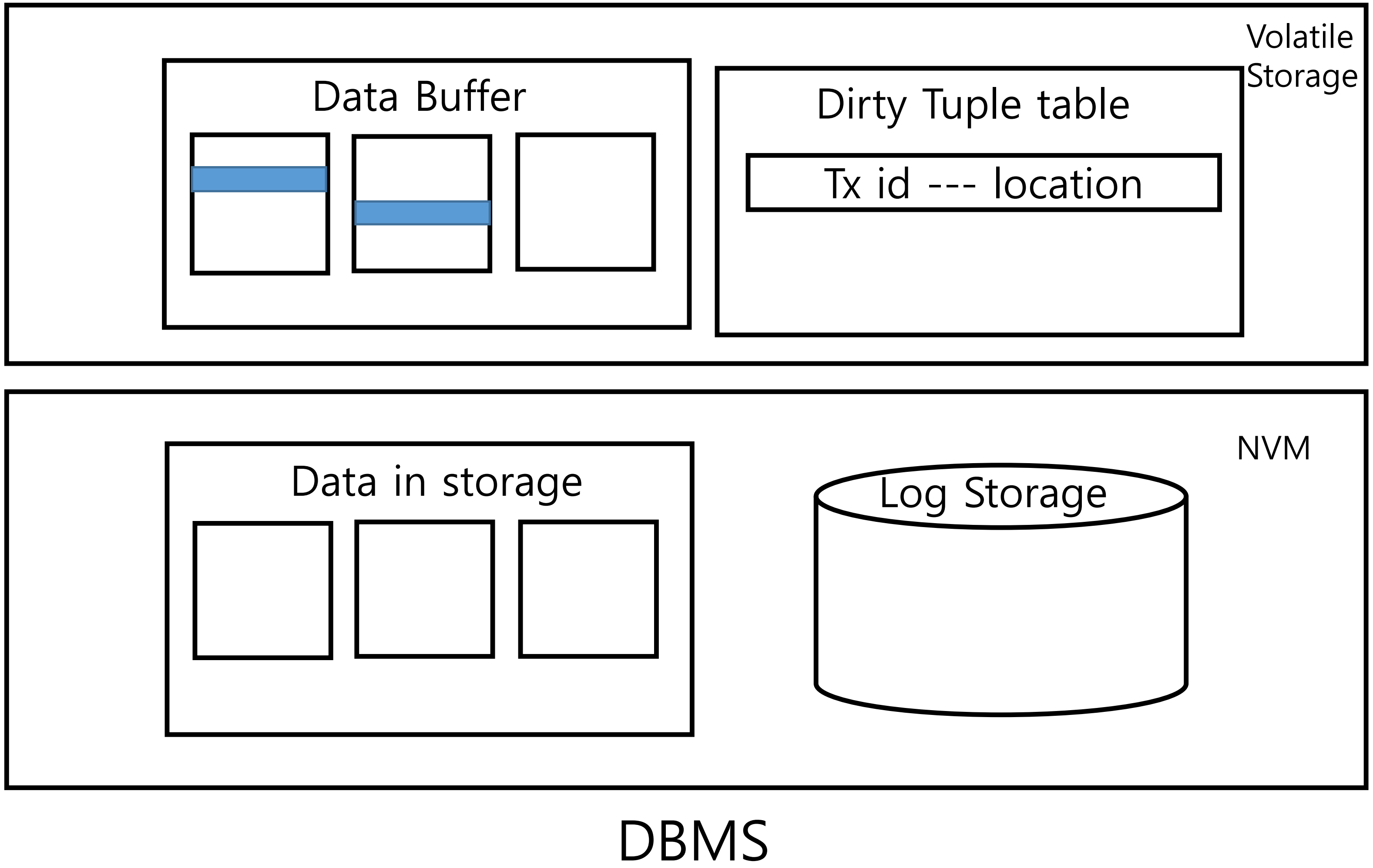
DBMS

1. Write changes to data in buffer

Write Behind Logging



WBL record



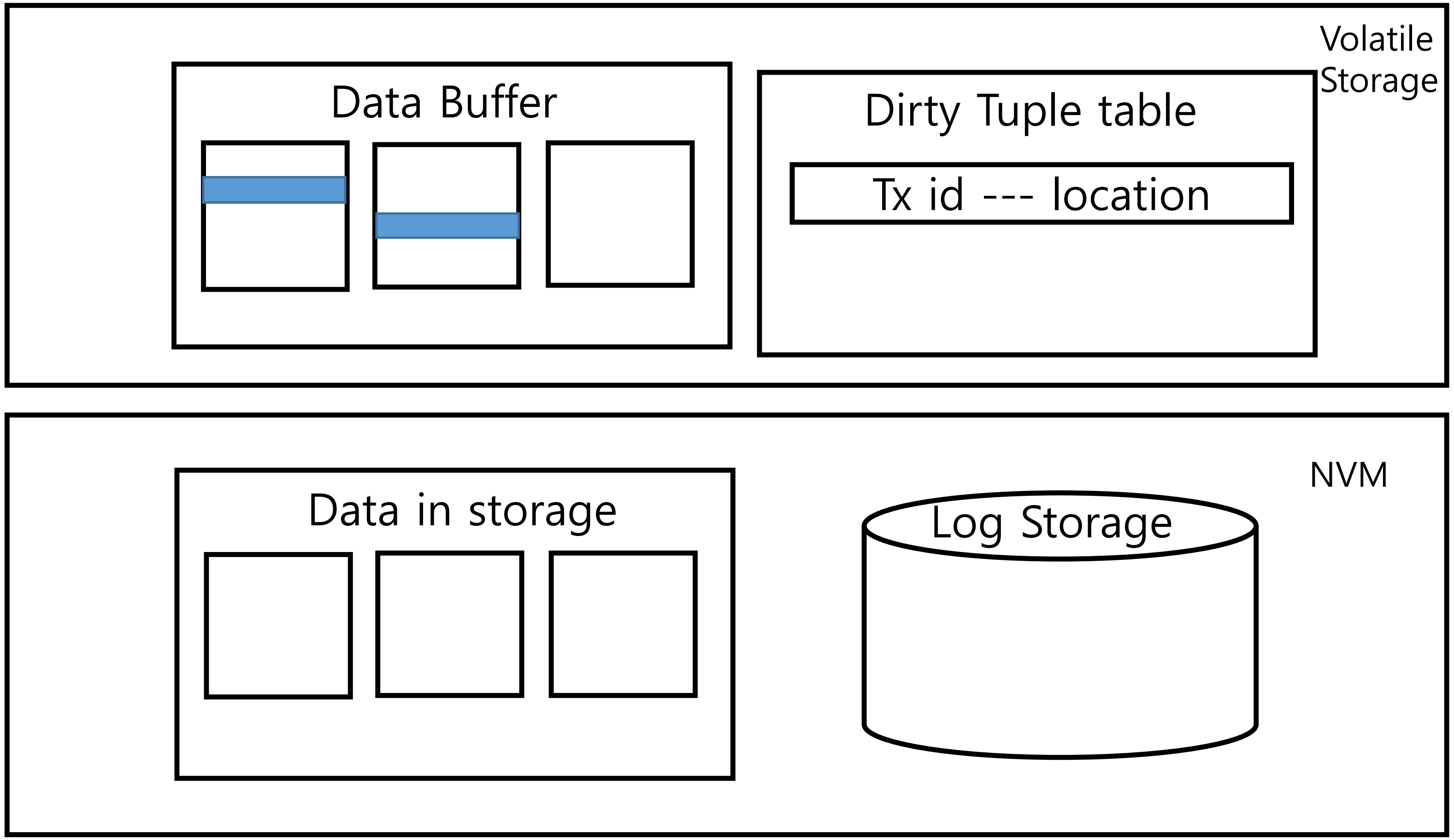
1. Write changes to data in buffer

2. Add entry to DTT

Write Behind Logging

checksum	LSN	Type	Persisted Commit Timestamp	Dirty Commit Timestamp

WBL record



DBMS

1. Write changes to data in buffer

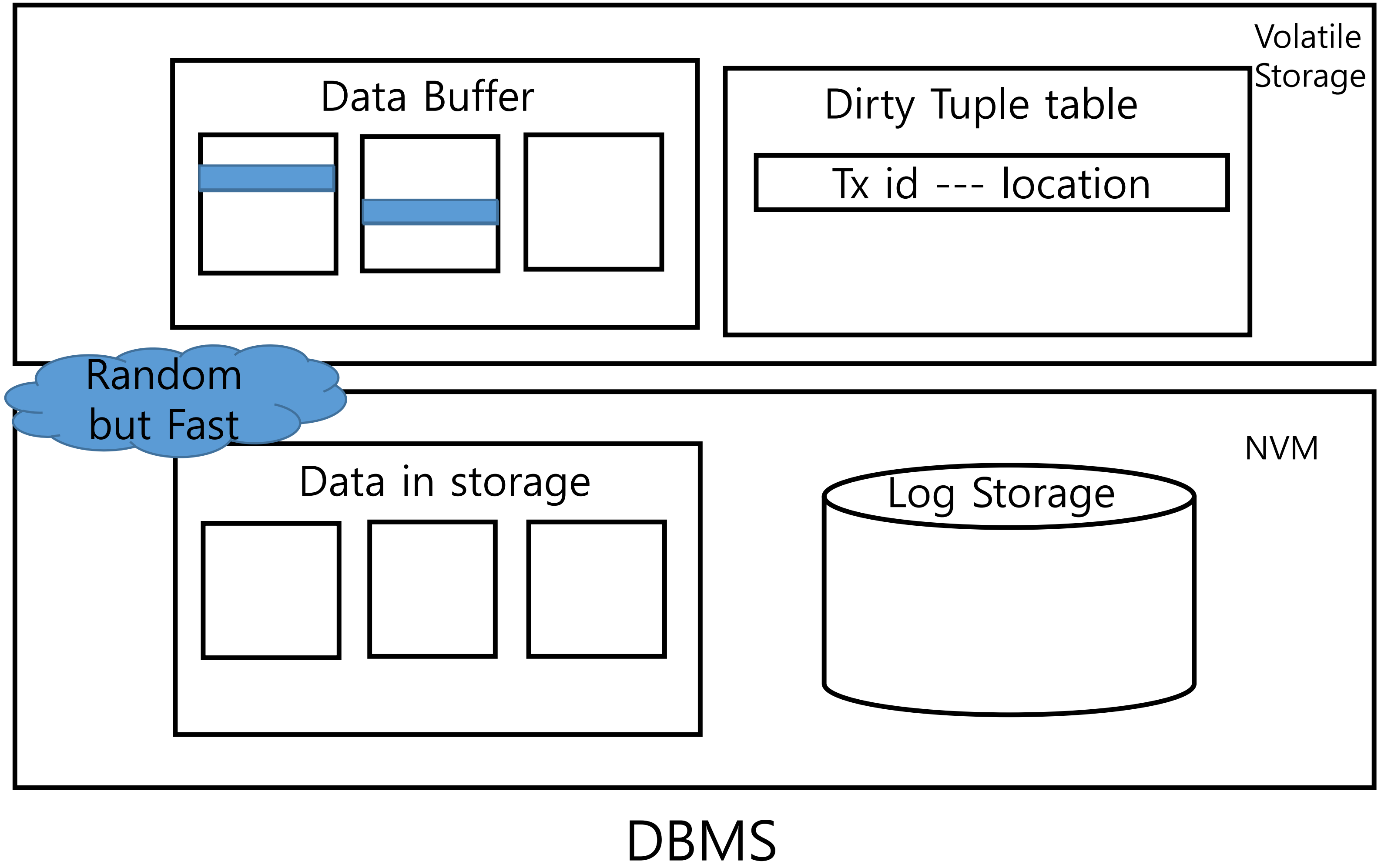
2. Add entry to DTT

3. Compute Cp (Persisted Commit Timestamp) and Cd (Dirty Commit Timestamp)

Write Behind Logging

checksum	LSN	Type	Persisted Commit Timestamp	Dirty Commit Timestamp

WBL record

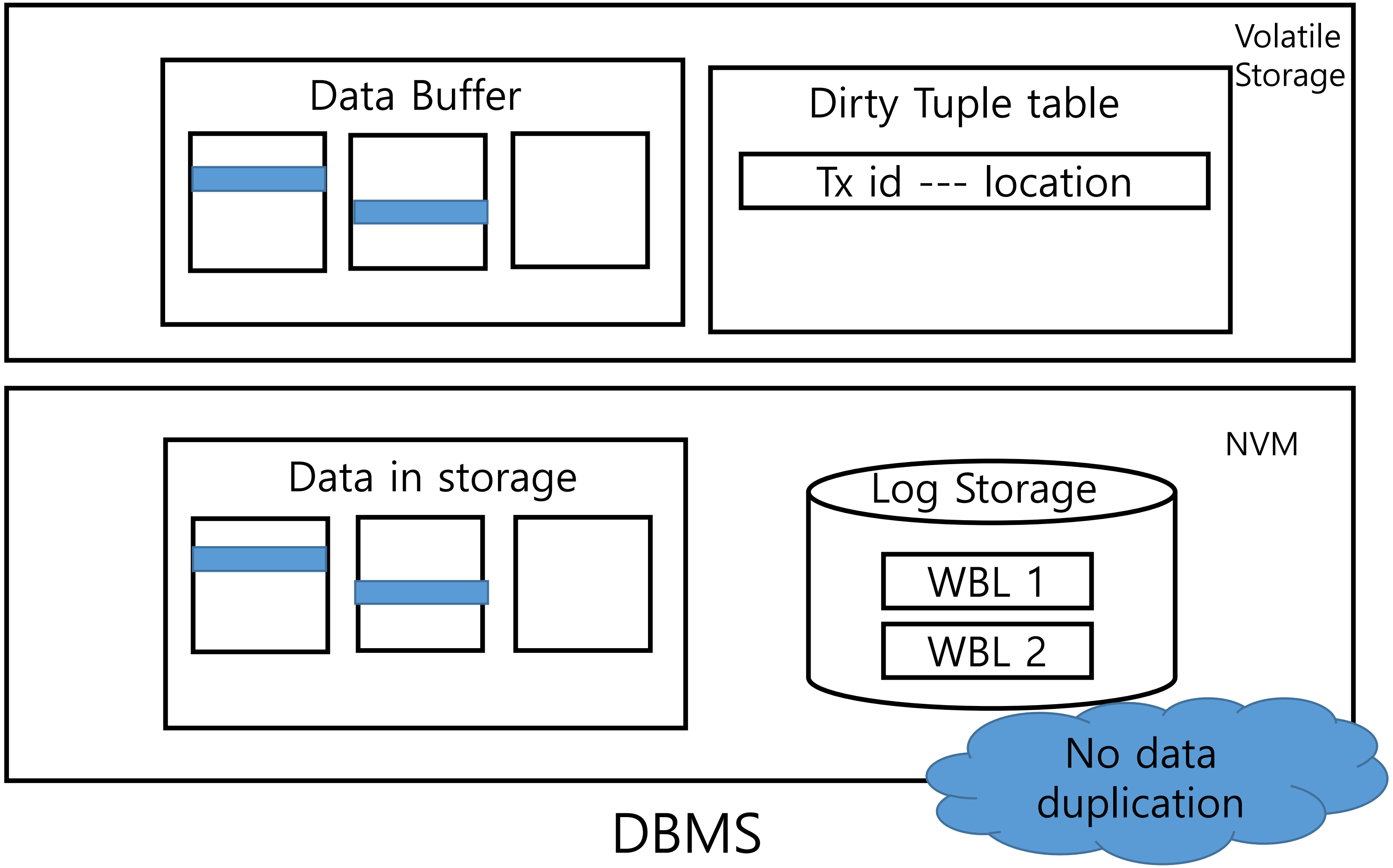


1. Write changes to data in buffer
2. Add entry to DTT
3. Compute Cp (Persisted Commit Timestamp) and Cd (Dirty Commit Timestamp)
4. Sync dirty block

Write Behind Logging

checksum	LSN	Type	Persisted Commit Timestamp	Dirty Commit Timestamp

WBL record



1. Write changes to data in buffer
2. Add entry to DTT
3. Compute Cp (Persisted Commit Timestamp) and Cd (Dirty Commit Timestamp)
4. Sync dirty block
5. Sync WBL record

Write Ahead Log vs Write Behind Log

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

Figure 7: WAL Example – Contents of the WAL during recovery.

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

Figure 12: WBL Example – Contents of the WBL during recovery.

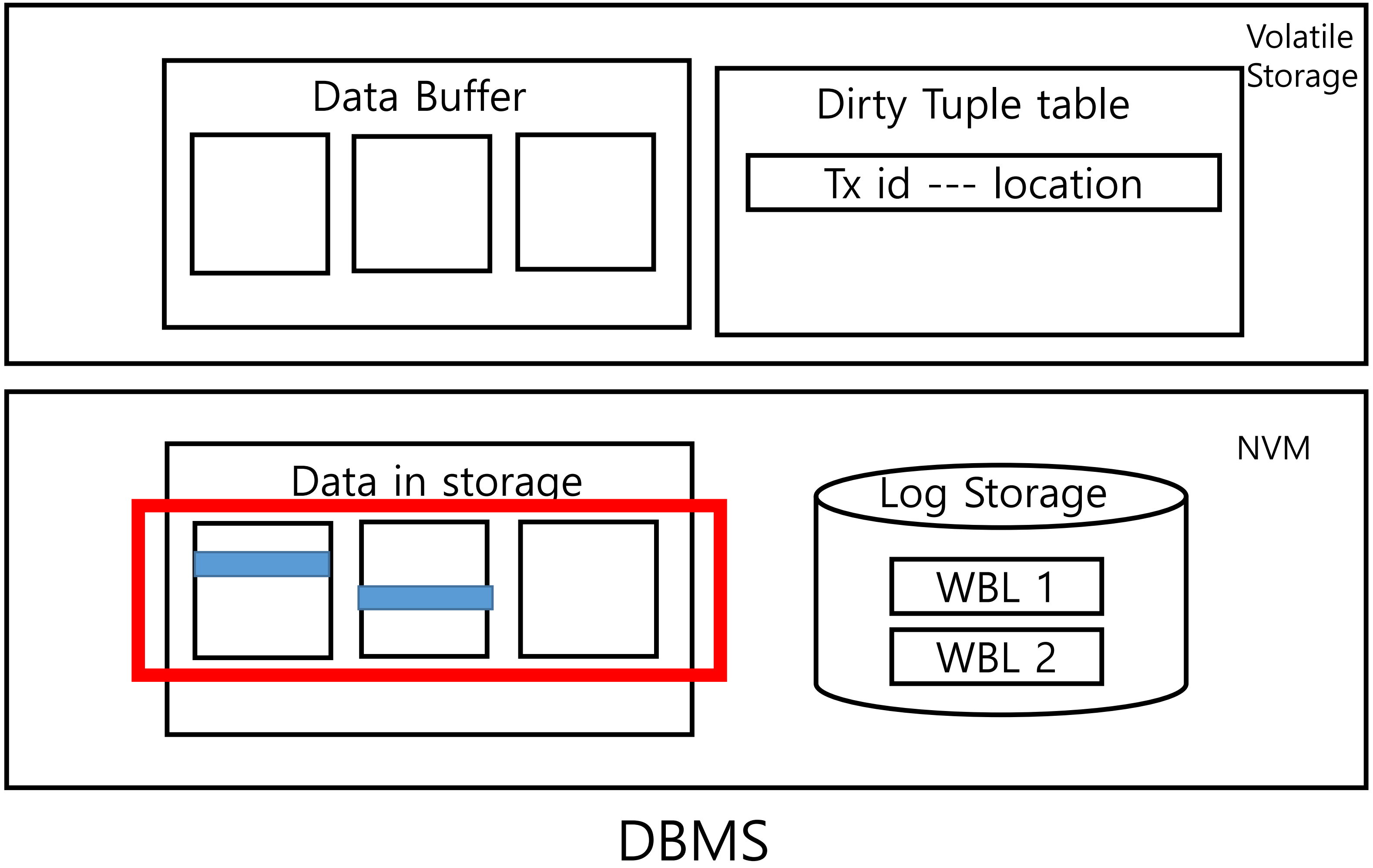
Uncommitted : Tx 80, Tx 81

Same workload,
but WBL size is smaller

Write Behind Logging : Recovery

checksum	LSN	Type	Persisted Commit Timestamp	Dirty Commit Timestamp

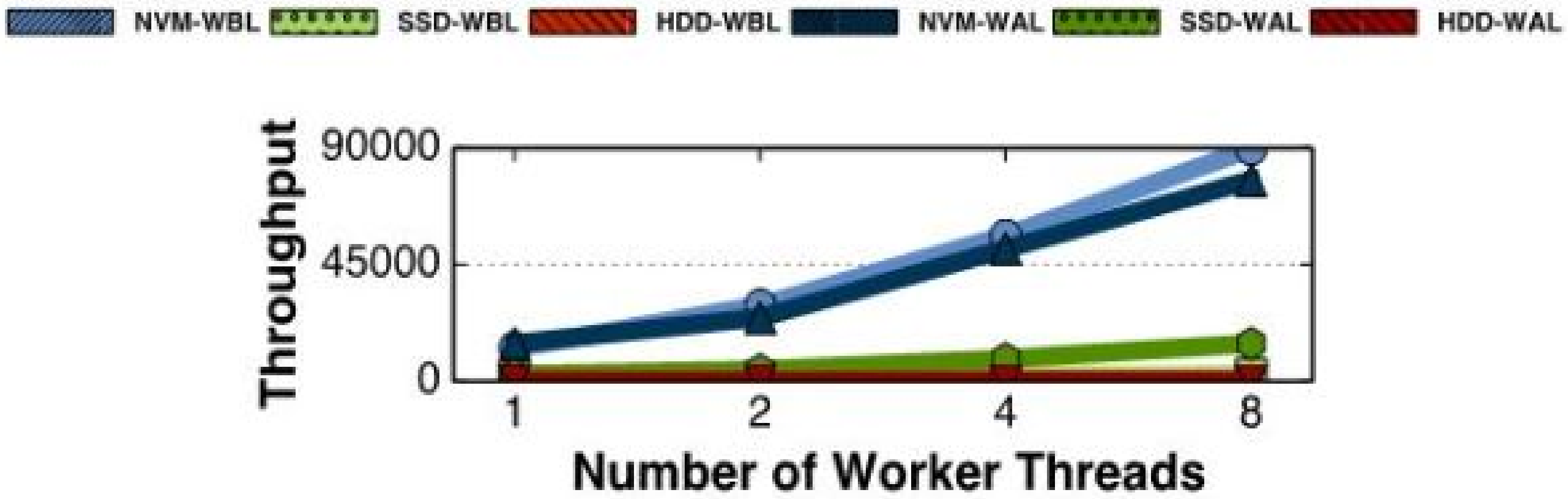
WBL record



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

Figure 12: WBL Example – Contents of the WBL during recovery.

Evaluation (1)

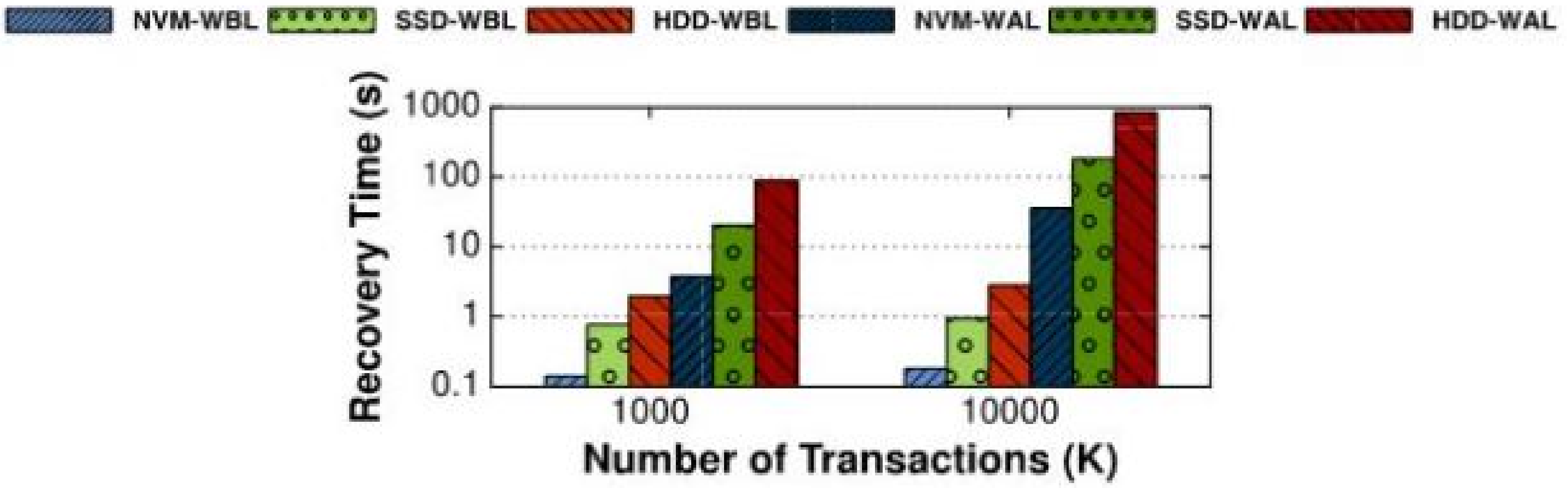


Throughput in YCSB

Result

1. NVM-WBL > NVM-WAL 1.3x
2. SSD-WAL > SSD-WBL

Evaluation (2)



Recovery performance

Result

1. WBL < WAL
2. WBL recovery performance is independent of the number of transactions

Thank You

Any question?