RECOVERY TECHNIQUES FOR IN-MEMORY DATABASE

06/29/2022

CS 848 Modern Database Systems

Presenter: Anuradha Kulkarni

UNIVERSITY OF WATERLOO | FACULTY OF MATHEMATICS

Agenda

- What is IMDB (In-memory database)?
- Architecture
- MMDB Recovery
- Transaction logging
- Consistent checkpointing
- Analysis of the recovery techniques

What is IMDB / MMDB?

- AKA Main Memory Database
- Data resides permanently in the main physical memory unlike conventional database system
- Better performance as data is accessed directly in memory.
- It is becoming feasible to store larger and larger databases in memory [2].

Architecture

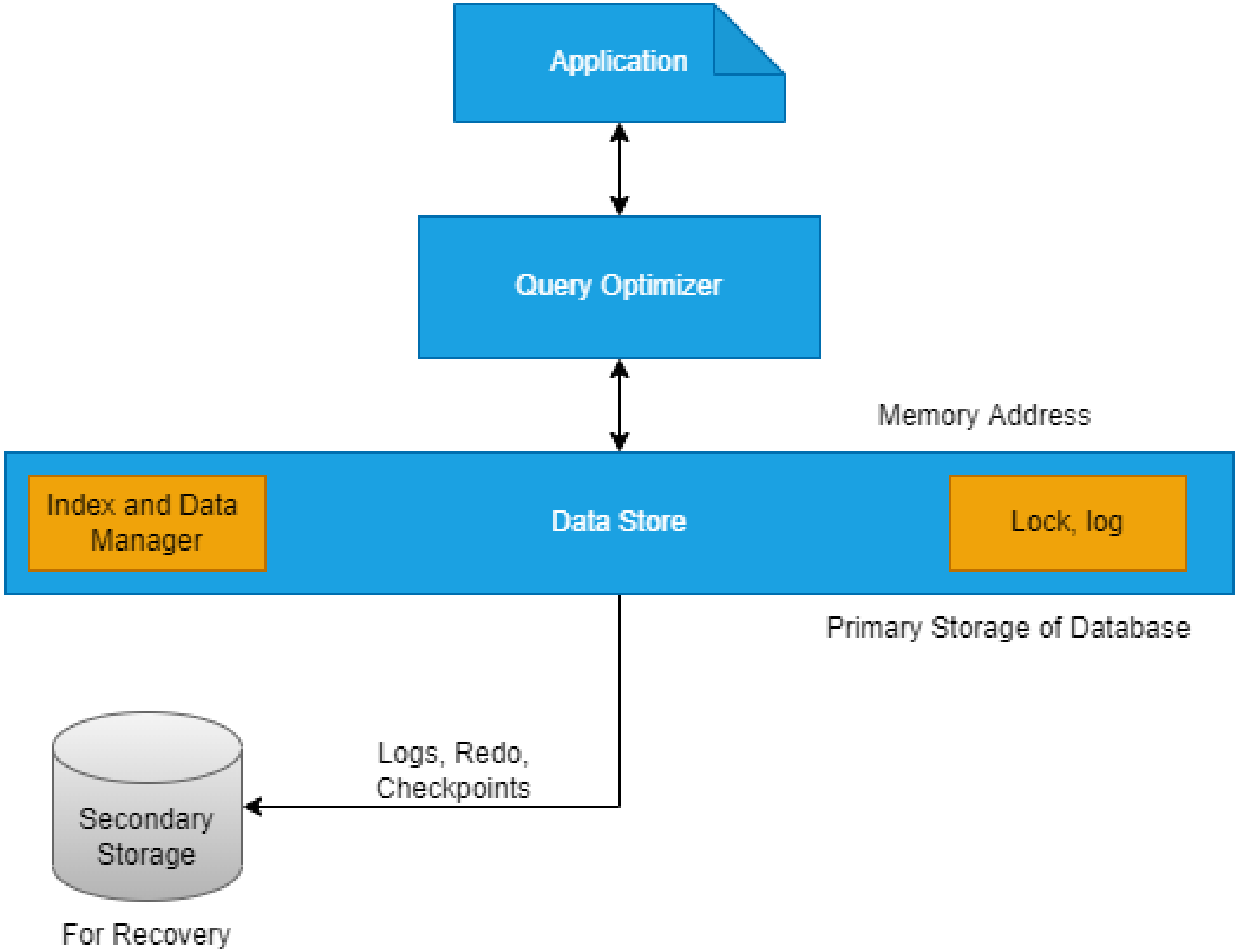


Fig. 1 MMDB Architecture

Main Memory Database Recovery

- What is effective and efficient database recovery?
- After crash, **recovery manager** must ensure that:
  - Unfinished transactions will not have their actions reflected in database (atomicity)
  - Completed transactions will have their modifications written in database, even if they have not flushed to the secondary memory (Durability)
- There are two buffer manager page replacement policies:
- **Steal approach:** Buffer manager protocol allows flushing dirty pages to secondary storage before the transaction commitment.
- **No-force approach:** Pages of committed transactions do not need to be flushed at commit time.

Transaction Logging

- AKA Command Logging
- Transaction's logic is written to the log rather than the transaction's operations
- Each transaction must be a predefined stored procedure
- The log records the stored procedure identifier of a transaction and its corresponding query parameters
- Very lightweight and needs only one record to store entire transaction. Hence, less overhead of transaction processing
- However, it can slow down recovery process because it needs to "replay" the transaction again
- It uses steal approach i.e. transaction can be logged before execution begins instead of executing the transaction and waiting for the log data to flush

Transaction Logging in Action

- **Step 1:** Log of the invocations are held in the memory
- **Step 2:** At the set interval the logs are physically written to the disks
- **Step 3:** At broader interval, the server initiates the snapshot.
- **Step 4:** Command logging process truncate the log keeping only a record of procedure invocations since the last snapshot

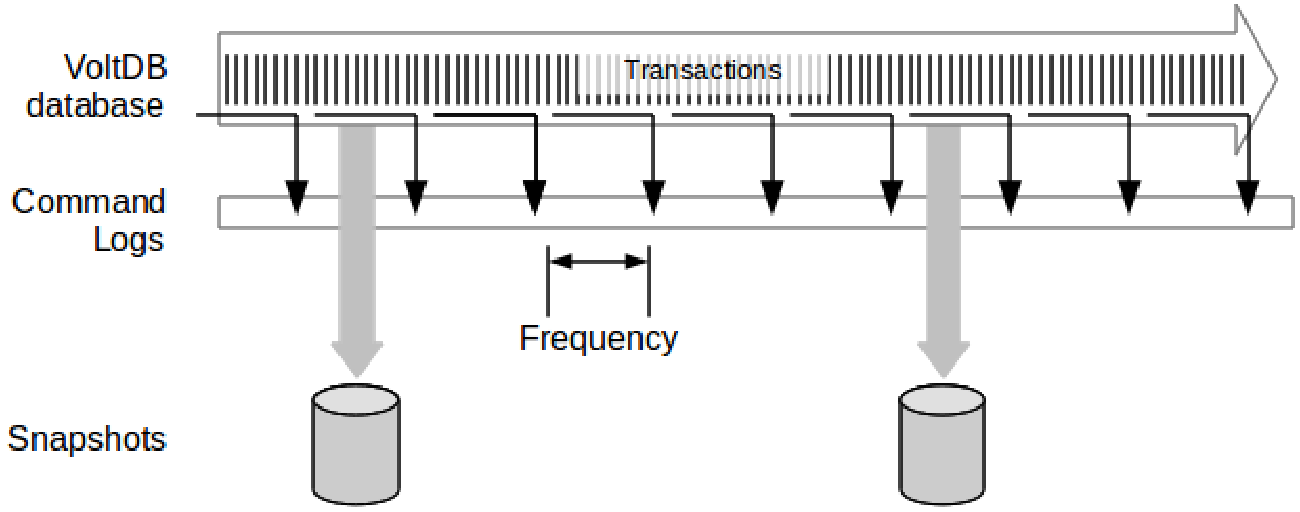


Fig. 2 Transaction logging in action

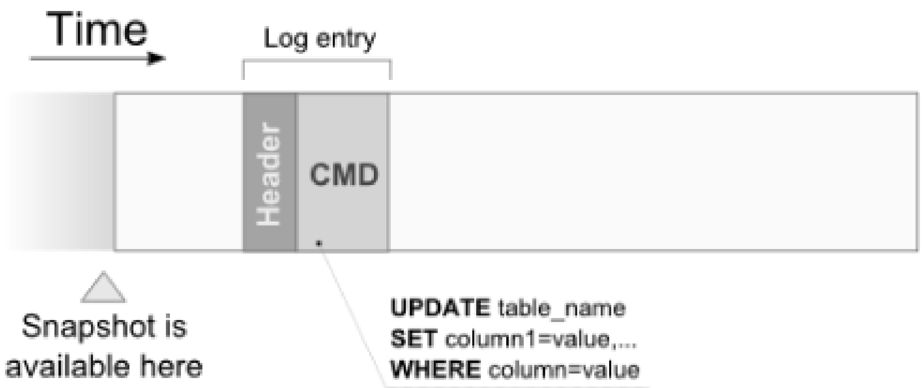


Fig. 3 Transaction log format

Transaction Logging Recovery in Action

- In reverse, when it is time to "replay" the logs, database is started and the server nodes establish a quorum.
- Servers restore the most recent snapshot. Then they replay all of the transactions in the log since that snapshot

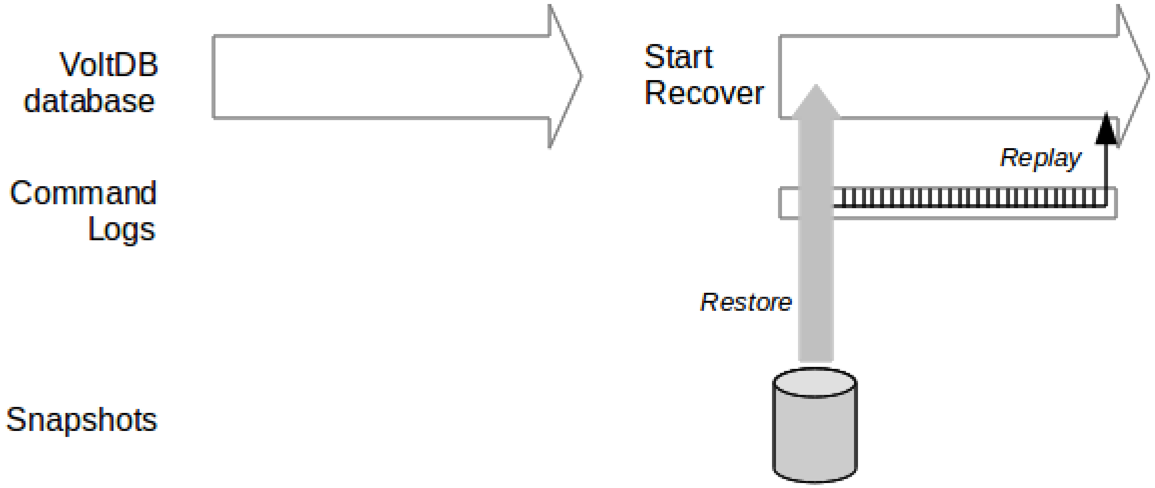


Fig. 4 Transaction logging recovery

Consistent Checkpointing

- A snapshot is a materialized database state in a specific instant of time
- Each checkpoint record is stored on the log asynchronously
- Reduces recovery time since loading data from the snapshot into memory is less costly than performing logical log operations
- **Definition (In-Memory Consistent Snapshot):** *Let D be an update intensive in-memory database. A consistent snapshot is a consistent state of D at a particular time-in-point, which should satisfy following two constraints:*
  - **Read Constraint:** Clients should be able to read the latest data items
  - **Update Constraint:** Any data item in the snapshot should not be overwritten i.e. snapshot must be read-only

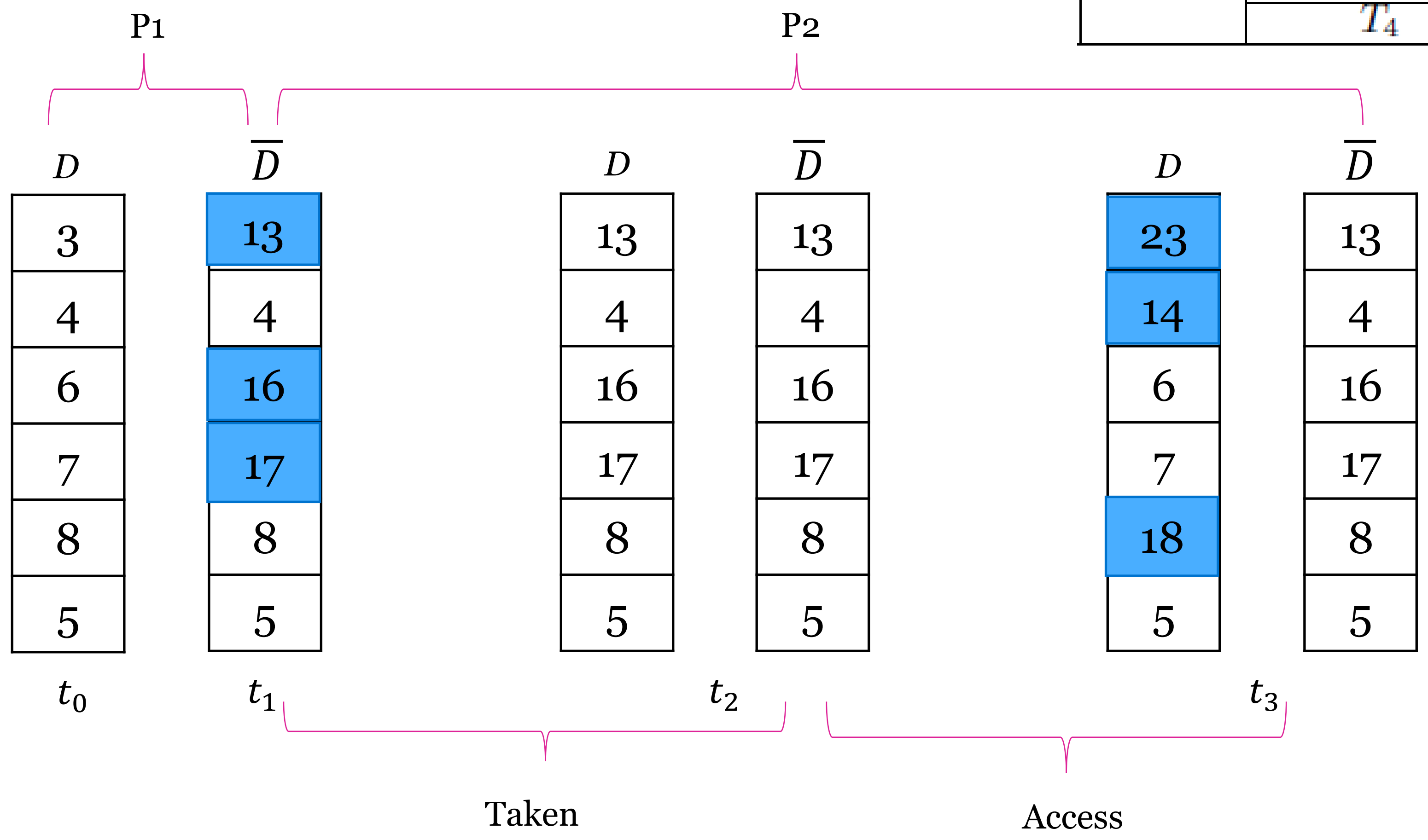
Consistent Checkpointing Algorithms

- An in-memory consistent snapshot algorithm for update-intensive applications must fulfil the following requirements:
  - **Consistent and Full Snapshots:** No dirty and incremental backups
  - **Lock-free and Copy-Optimized:** No synchronous operations
  - **Low latency and no Latency spikes**
  - **Small memory footprint**
- Snapshot Algorithm Framework:

```
Client::Read(index);
Client::Write(index,newValue);
Snapshotter::Trigger();
Snapshotter::TakeSnapshot();
Snapshotter::TraverseSnapshot();
```

Naïve Algorithm

Period	Transactions	Data to be Updated
P_1	T_1	$< 0, 13 >$
	T_2	$< 2, 16 >, < 3, 17 >$
P_2	T_3	$< 0, 23 >$
	T_4	$< 1, 14 >, < 4, 18 >$



Copy On Update Algorithm

Period	Transactions	Data to be Updated
P_1	T_1	$< 0, 13 >$
	T_2	$< 2, 16 >, < 3, 17 >$
P_2	T_3	$< 0, 23 >$
	T_4	$< 1, 14 >, < 4, 18 >$

D	$\overline{D}$	$\overline{D}_b$
3	3	0
4	4	0
6	6	0
7	7	0
8	8	0
5	5	0

t_0

D	$\overline{D}$	$\overline{D}_b$
13	3	0
4	4	0
16	6	0
17	7	0
8	8	0
5	5	0

t_1

D	$\overline{D}$	$\overline{D}_b$
23	13	1
14	4	1
6	6	0
7	7	0
18	8	1
5	5	0

t_2

Zigzag Algorithm

Period	Transactions	Data to be Updated
P_1	T_1	$< 0, 13 >$
	T_2	$< 2, 16 >, < 3, 17 >$
P_2	T_3	$< 0, 23 >$
	T_4	$< 1, 14 >, < 4, 18 >$

t_0

D	$\overline{D}$	$\overline{D}_{br}$	$\overline{D}_{bw}$
3	3	0	1
4	4	0	1
6	6	0	1
7	7	0	1
8	8	0	1
5	5	0	0

t_1

D	$\overline{D}$	$\overline{D}_{br}$	$\overline{D}_{bw}$
3	13	1	1
4	4	0	1
6	16	1	1
7	17	1	1
8	8	0	1
5	5	0	1

t_2

D	$\overline{D}$	$\overline{D}_{br}$	$\overline{D}_{bw}$
3	13	1	0
4	4	0	1
6	16	1	0
7	17	1	0
8	8	0	1
5	5	0	1

t_3

D	$\overline{D}$	$\overline{D}_{br}$	$\overline{D}_{bw}$
23	3	0	0
4	14	1	1
6	6	1	0
7	7	1	0
8	18	1	1
5	5	0	1

Ping-Pong Algorithm

D	$\overline{D}_u$		$\overline{D}_d$	
3		0	3	1
4		0	4	1
6		0	6	1
7		0	7	1
8		0	8	1
5		0	5	1

t_0

D	$\overline{D}_u$		$\overline{D}_d$	
13	13	1	3	0
4	4	0	4	0
16	16	1	6	0
17	17	1	7	0
8	8	0	8	0
5	5	0	5	0

t_1

D	$\overline{D}_d$		$\overline{D}_u$	
13	13	1		0
4	4	0		0
16	16	1		0
17	17	1		0
8	8	0		0
5	5	0		0

t_2

D	$\overline{D}_d$		$\overline{D}_u$	
23	13	0	23	1
14	4	0	14	1
16	16	0		0
17	17	0		0
18	8	0	18	1
5	5	0		0

t_3

Hourglass Algorithm

Zigzag (bit array marking)
+
Ping-Pong (Pointers swapping)

pU pD

D	$\overline{D}_{b1}$	$\overline{D}$	$\overline{D}_{b2}$	$\overline{D}_{br}$
3	0	3	1	1
4	0	4	1	1
6	0	6	1	1
7	0	7	1	1
8	0	8	1	1
5	0	5	1	1

t_0

pU pD

D	$\overline{D}_{b1}$	$\overline{D}$	$\overline{D}_{b2}$	$\overline{D}_{br}$
13	1	3	0	0
4	0	4	0	1
16	1	6	0	0
17	1	7	0	0
8	0	8	0	1
5	0	5	0	1

t_1

pD pU

D	$\overline{D}_{b1}$	$\overline{D}$	$\overline{D}_{b2}$	$\overline{D}_{br}$
13	1	3	0	0
4	0	4	0	1
16	1	6	0	0
17	1	7	0	0
8	0	8	0	1
5	0	5	0	1

t_2

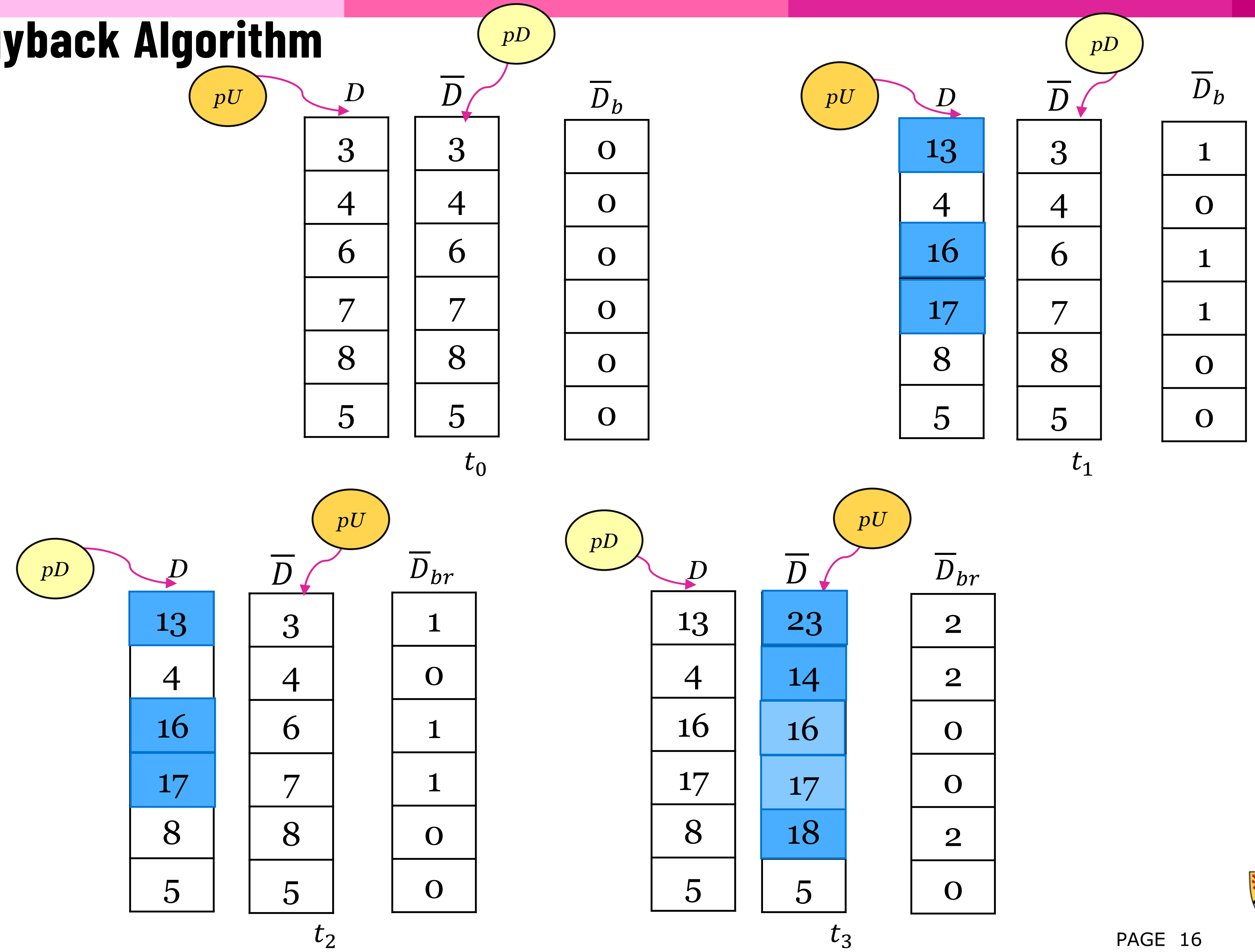
pD pU

D	$\overline{D}_{b1}$	$\overline{D}$	$\overline{D}_{b2}$	$\overline{D}_{br}$
13	0	23	1	1
4	0	14	1	1
16	0	6	1	0
17	0	7	1	0
8	0	18	1	1
5	0	5	1	1

t_3

t_3

Piggyback Algorithm



t_3

Comparison of Snapshot Algorithms

- Fork is a standard method in many industrial IMDBs
- In theory, Piggyback outperforms the rest in all metrics
- 2× memory consumptions of HG and PB are only for the abstract array model (Static memory allocation). This can be reduced further using dynamic memory allocation technique
- Comparison of algorithms in different metrics; “(*)” represents the drawback

Algorithms	Average latency	Latency Spike	Snapshot time complexity	Max throughput	Is full Snapshot	Max memory footprint
Naïve Snapshot	low	(*) high	(*) O(n)	Low	Yes	2×
Copy-On-Update	(*) high	(*) middle	(*) O(n)	Middle	Yes	2×
Zigzag	middle	(*) middle	(*) O(n)	Middle	Yes	2×
Ping-Pong	(*) high	almost none	O(1)	Low	No	(*) 3×
Hourglass	low	almost none	O(1)	High	No	2×
Piggyback	low	almost none	O(1)	High	Yes	2×

Summary

- Proposed the need for MMDB recovery
- Various MMDB recovery techniques
- An emphasis on working of transaction logging and recovery
- Analyzed, compared and evaluated consistent snapshot algorithms
- Demonstrated better tradeoffs among latency, throughput, complexity and scalability

References

[1] L. Li, G. Wang, G. Wu, Y. Yuan, L. Chen and X. Lian, "A Comparative Study of Consistent Snapshot Algorithms for Main-Memory Database Systems," in IEEE Transactions on Knowledge and Data Engineering, vol. 33, no. 2, pp. 316-330, 1 Feb. 2021, DOI: 10.1109/TKDE.2019.2930987.

[2] Faerber, Frans & Kemper, Alfons & Larson, Per-Åke & Levandoski, Justin & Neumann, Tjomas & Pavlo, Andrew. (2017). Main Memory Database Systems. Foundations and Trends in Databases. 8. 1-130. 10.1561/1900000058.

[3] Brayner, Angelo & Magalhães, Arlino & Monteiro, José. (2021). Main Memory Database Recovery: A Survey. ACM Computing Surveys. 54. 1-36. 10.1145/3442197.

THANK YOU!

UNIVERSITY OF WATERLOO

FACULTY OF MATHEMATICS

YOU+WATERLOO

Our greatest impact happens together.