THE CHIPS TO SYSTEMS CONFERENCE

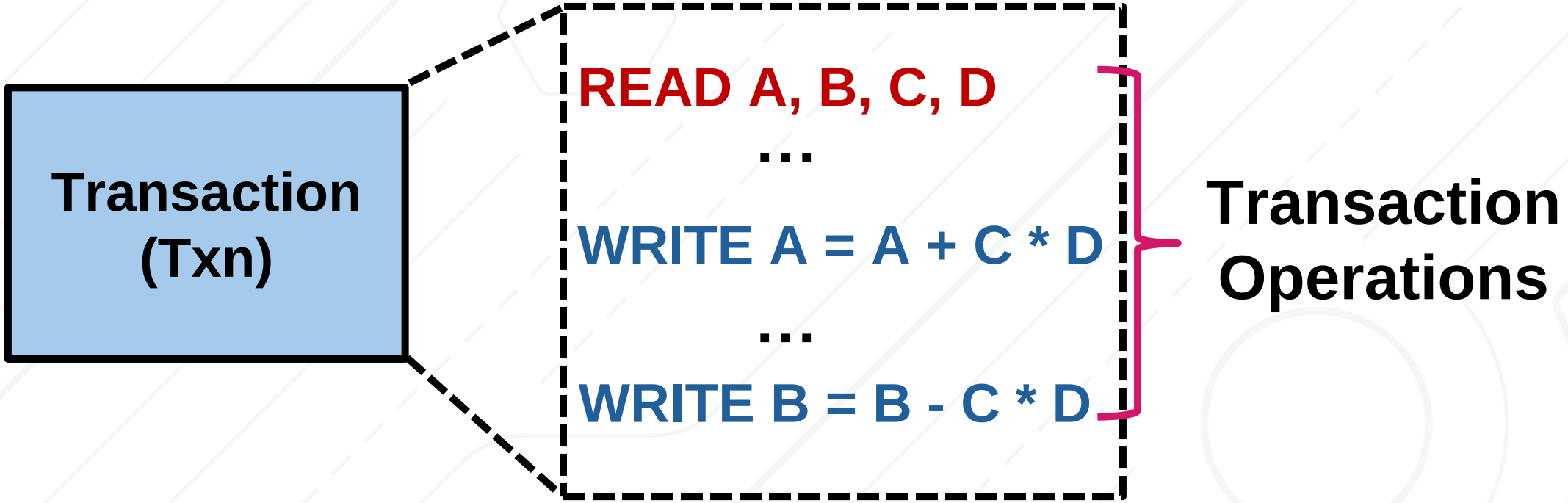
SPONSORED BY: CEDA IEEE Council on Electronic Design Automation, SIGDA ACM

Onyx: Efficient Transaction Processing with Real Processing-in-Memory Prototypes

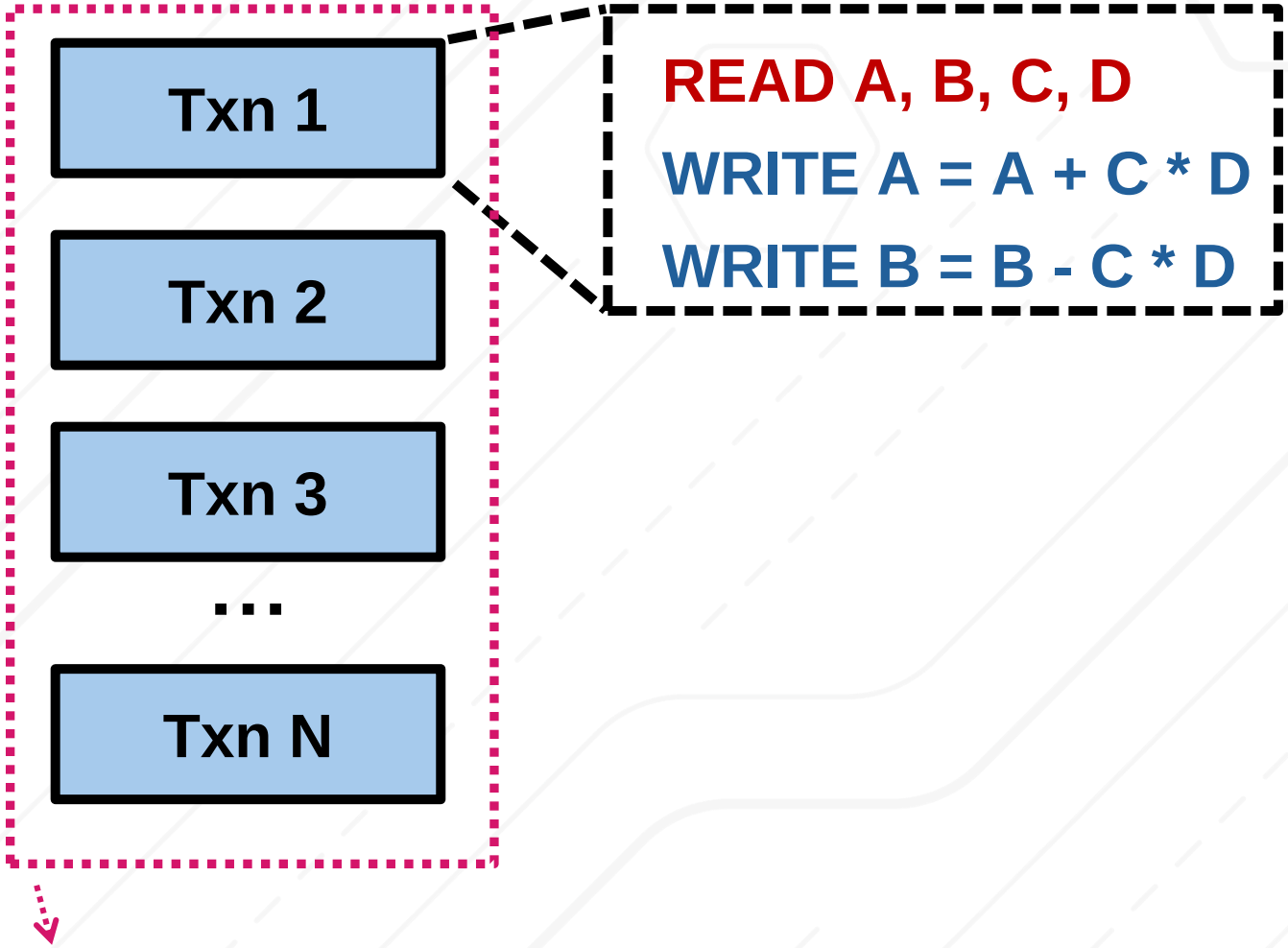
Menglei Chen, Yixiao Wang, Yu Hua

Huazhong University of Science and Technology

Online Transaction Processing

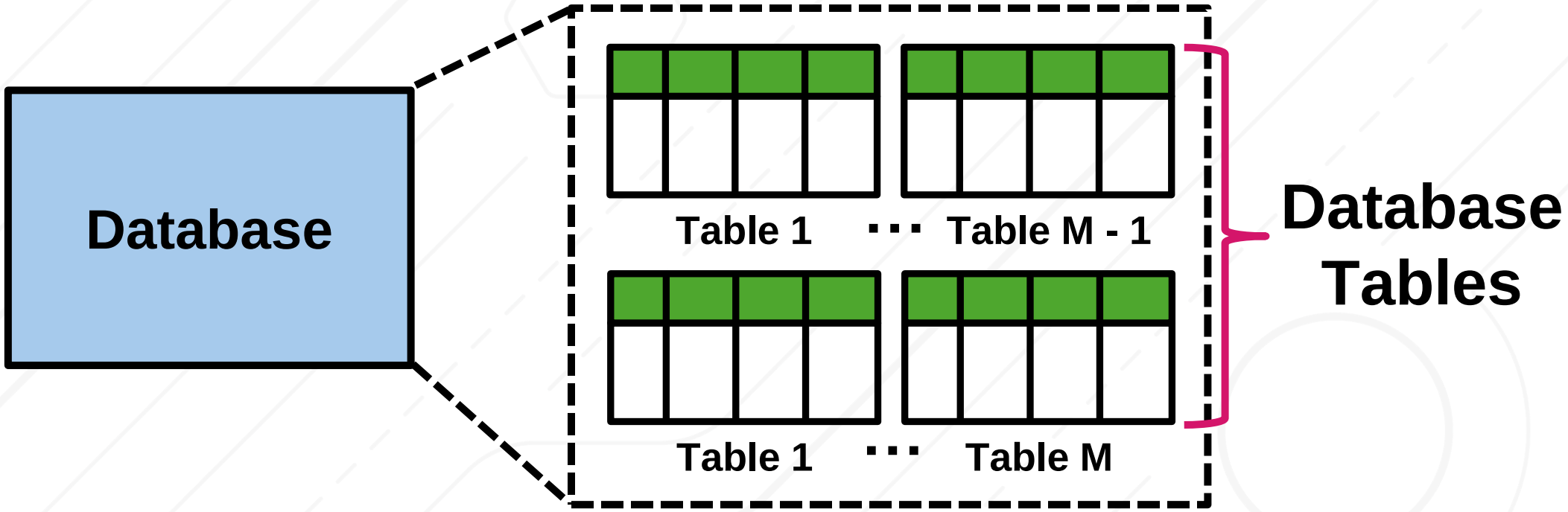


Online Transaction Processing

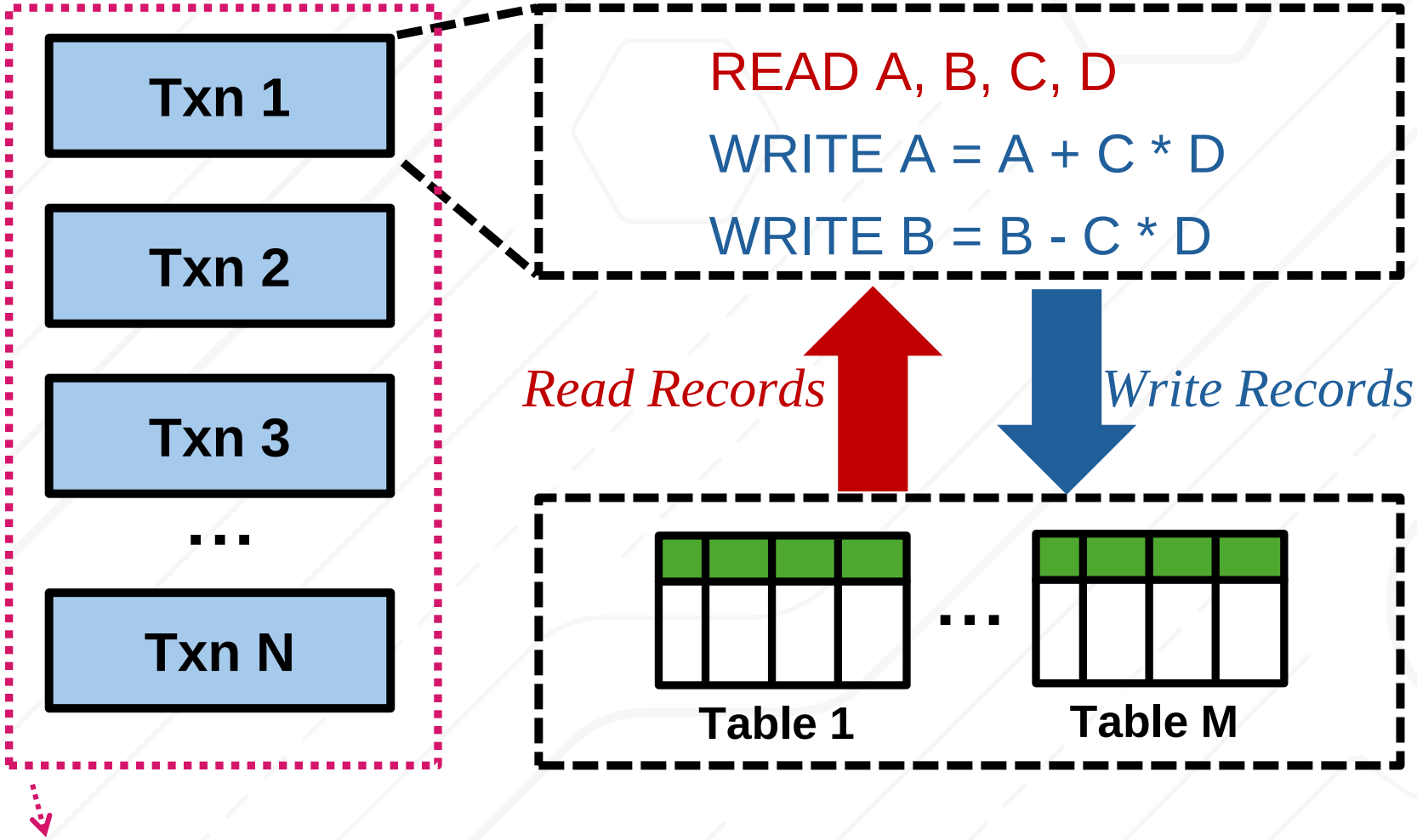


Processed in parallel with concurrency control mechanisms

Online Transaction Processing

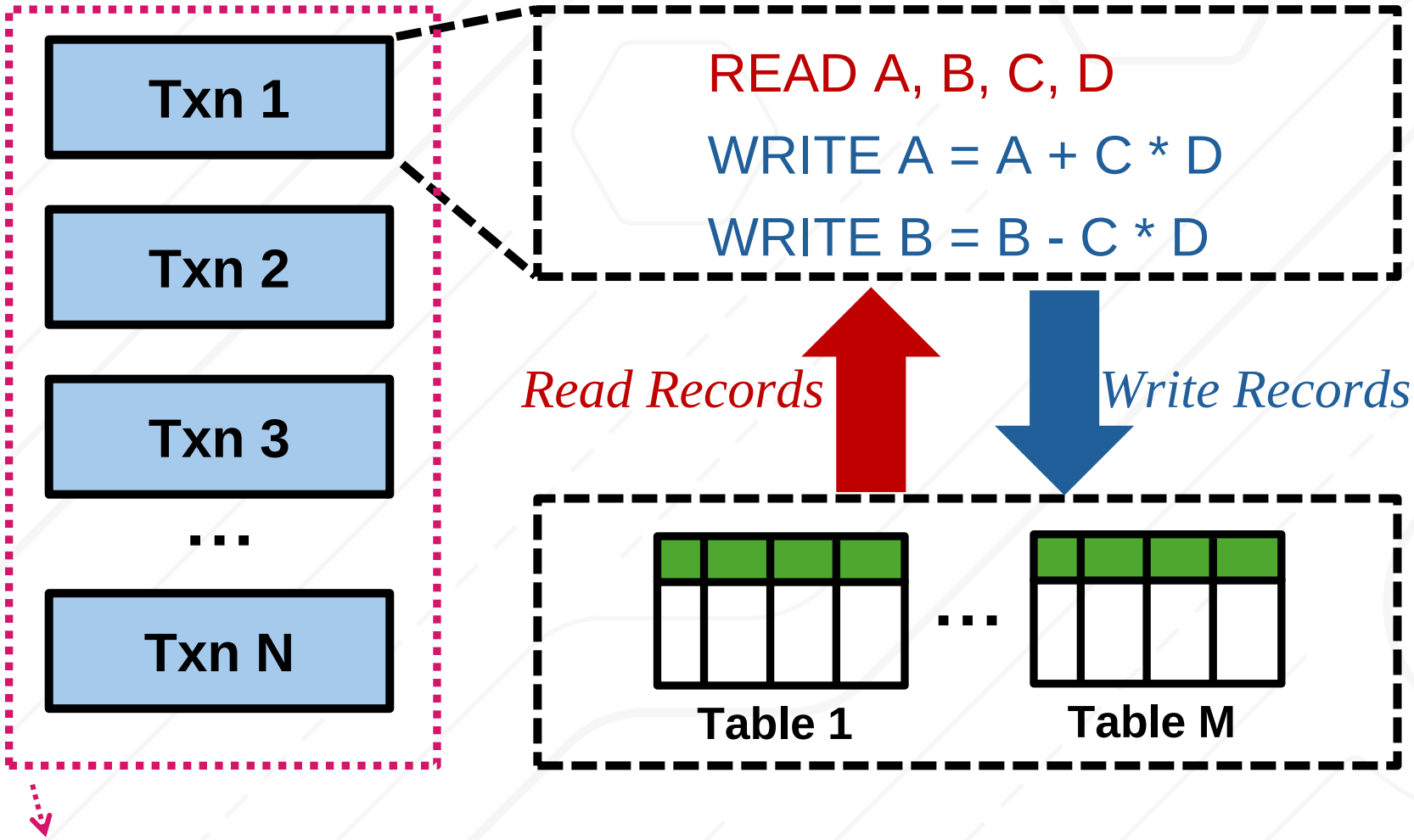


Online Transaction Processing



Processed in parallel with concurrency control mechanisms

Online Transaction Processing

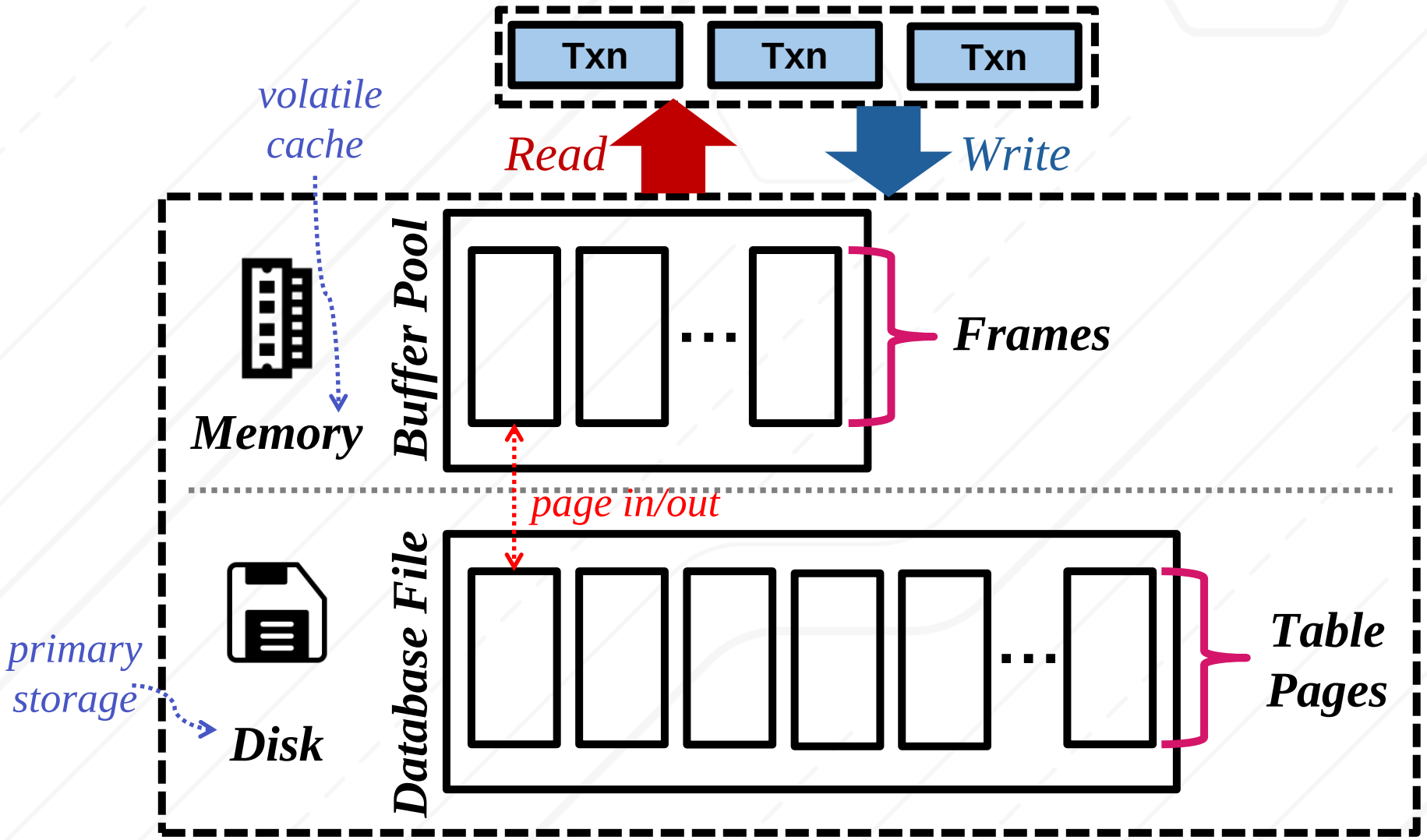


Enabling:

- *Atomicity*
- *Consistency*
- *Isolation*
- *Durability*

Processed in parallel with concurrency control mechanisms

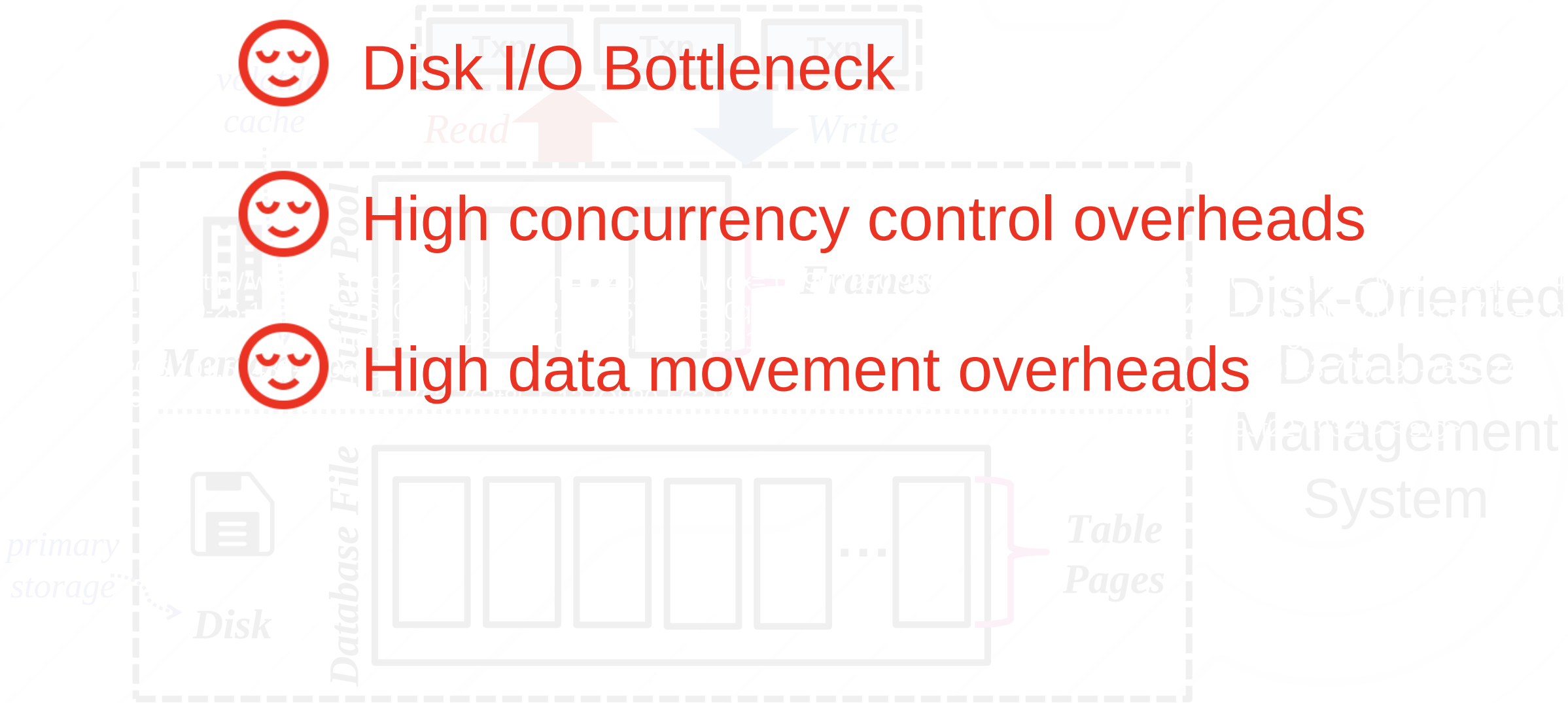
Existing OLTP Systems



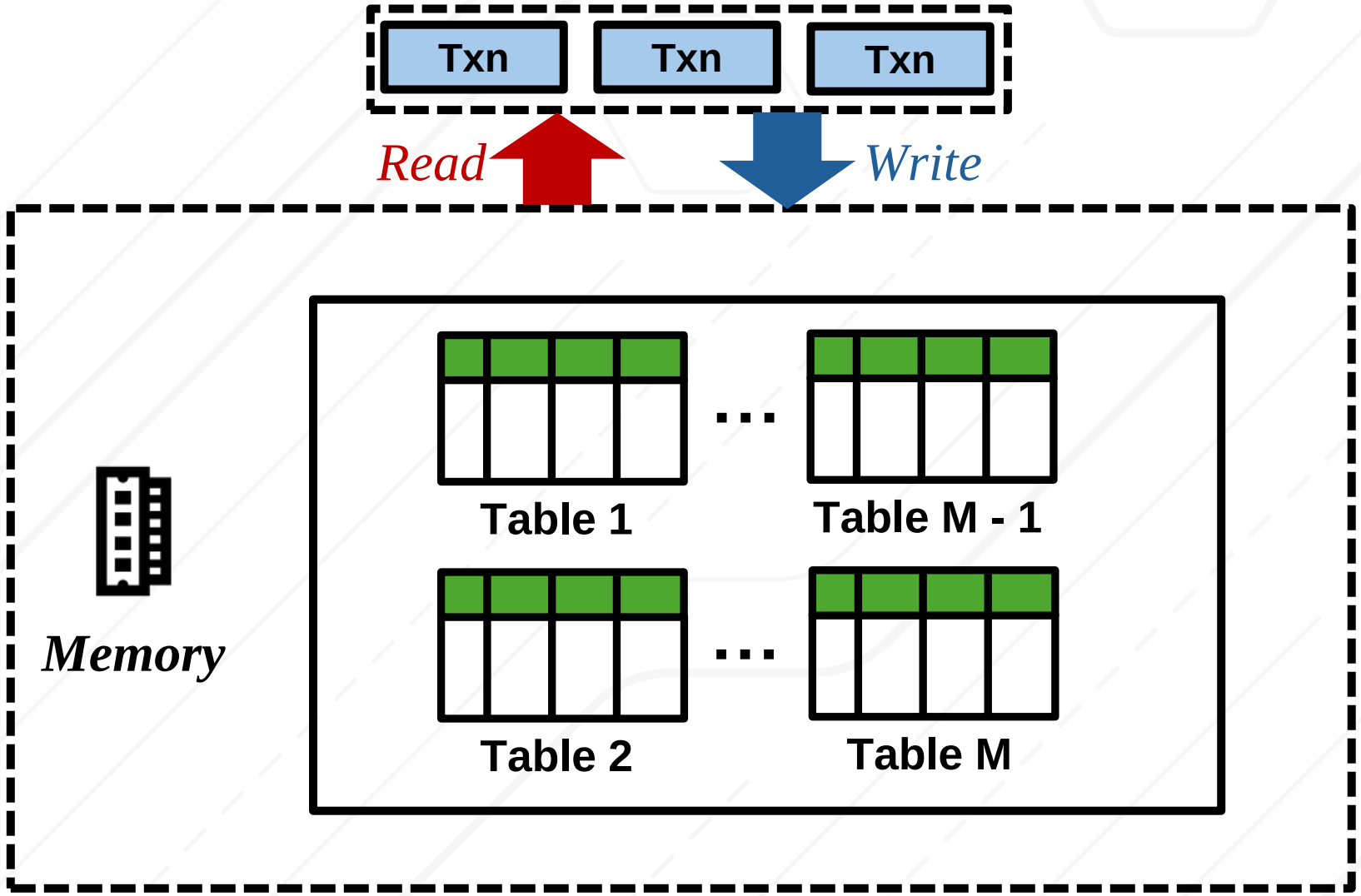
Disk-Oriented Database Management System

Existing OLTP Systems

- Disk I/O Bottleneck
- High concurrency control overheads
- High data movement overheads



Existing OLTP Systems

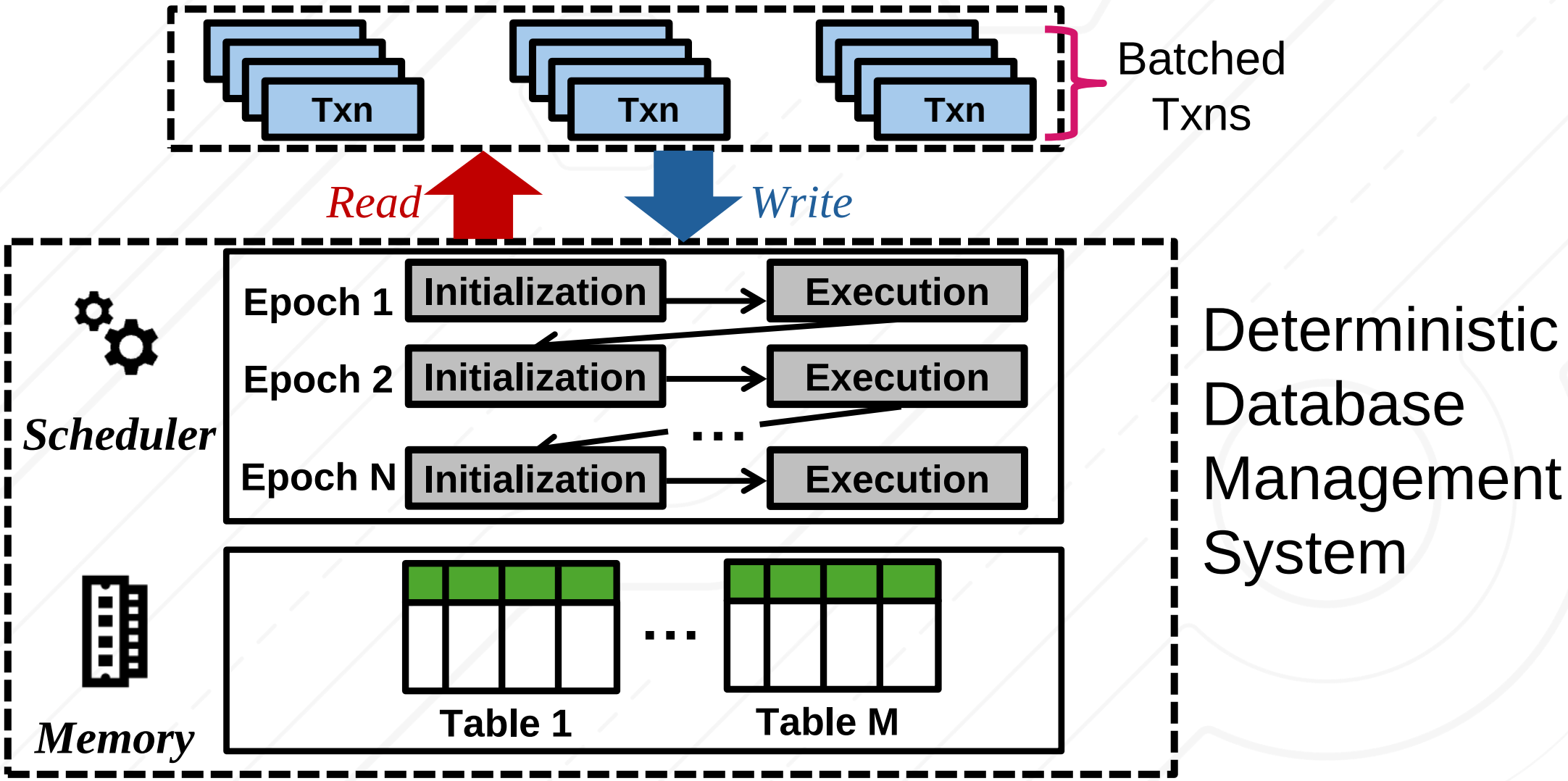


In-Memory Database Management System

Existing OLTP Systems

- Address disk I/O bottleneck
- High concurrency control overheads
- High data movement overheads

Existing OLTP Systems

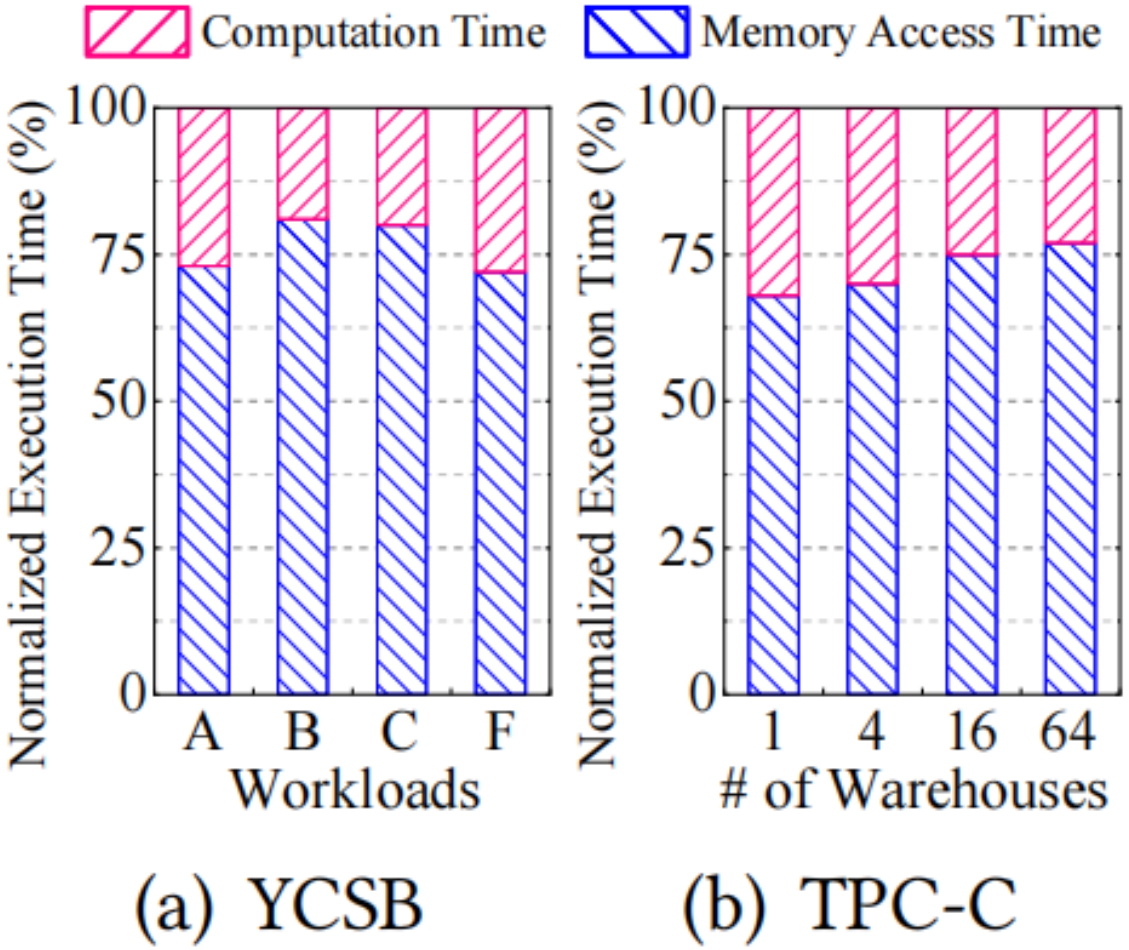


Deterministic Database Management System

Existing OLTP Systems

- Address disk I/O bottleneck
- Low concurrency control overheads
- High data movement overheads

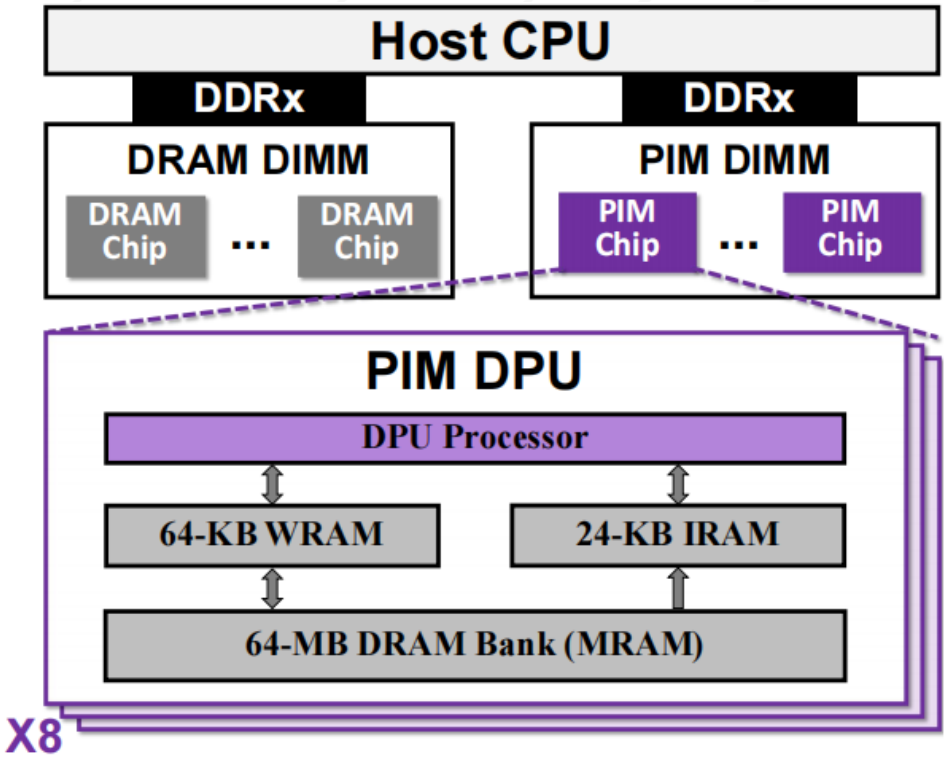
Existing OLTP Systems



68% ~ 81%

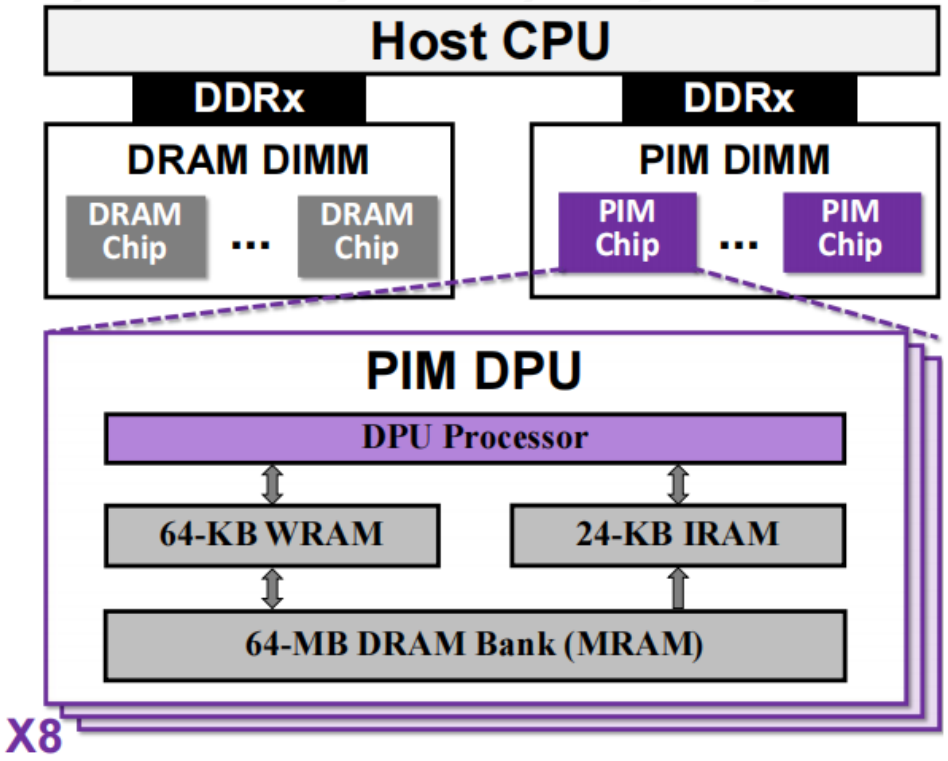
Memory bandwidth becomes the performance bottleneck!

UPMEM Processing In Memory



- 1.6 TB/s high memory bandwidth
- Offloading computing from CPU to DPUs

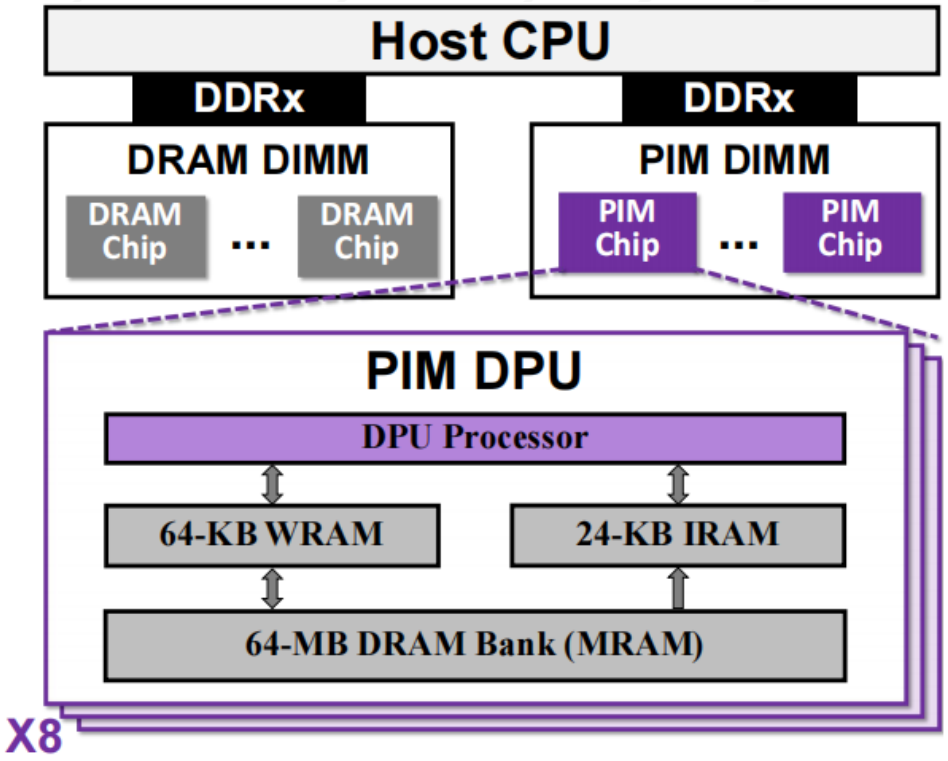
UPMEM Processing In Memory



- 1.6 TB/s high memory bandwidth
- Offloading computing from CPU to DPUs

Promising!

UPMEM Processing In Memory

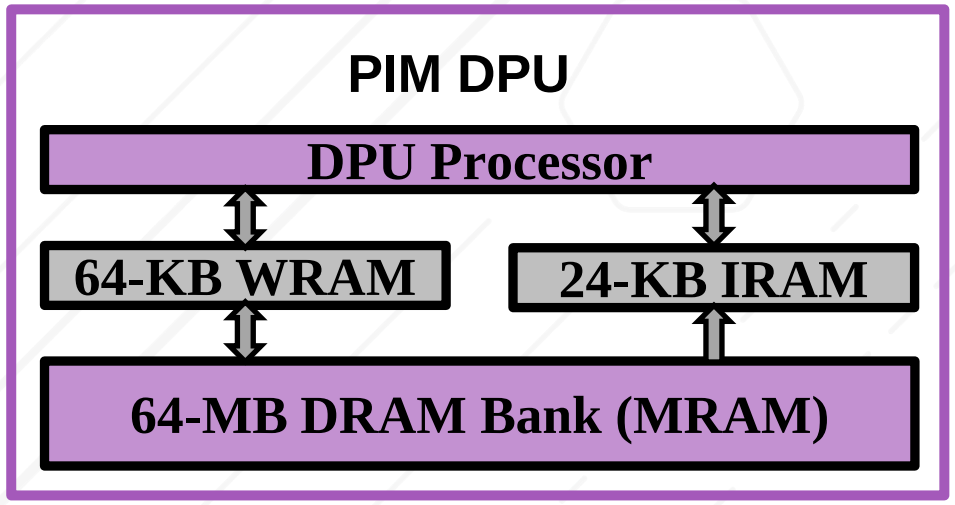


- 1.6 TB/s high memory bandwidth
- Offloading computing from CPU to DPUs

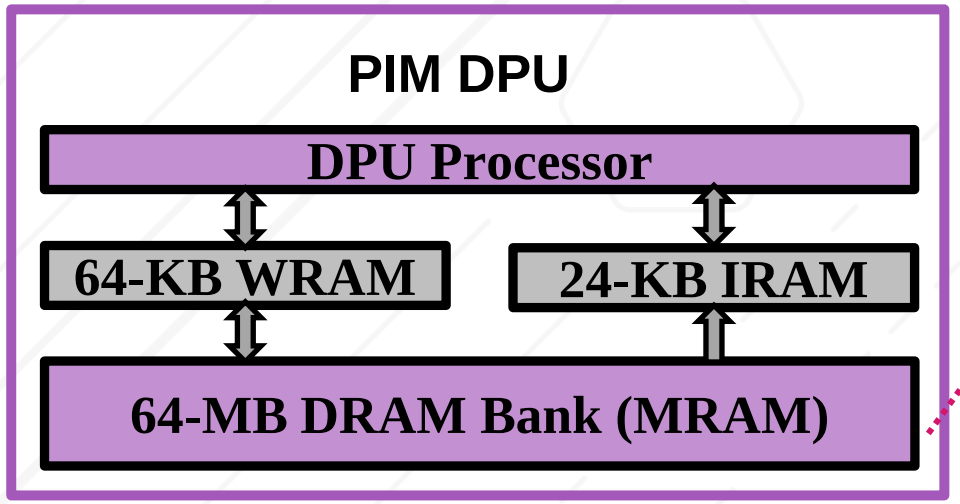
Promising!

However, it is non-trival to design a transaction processing system on PIM

Challenge 1: Execution across Partitioned DPUs

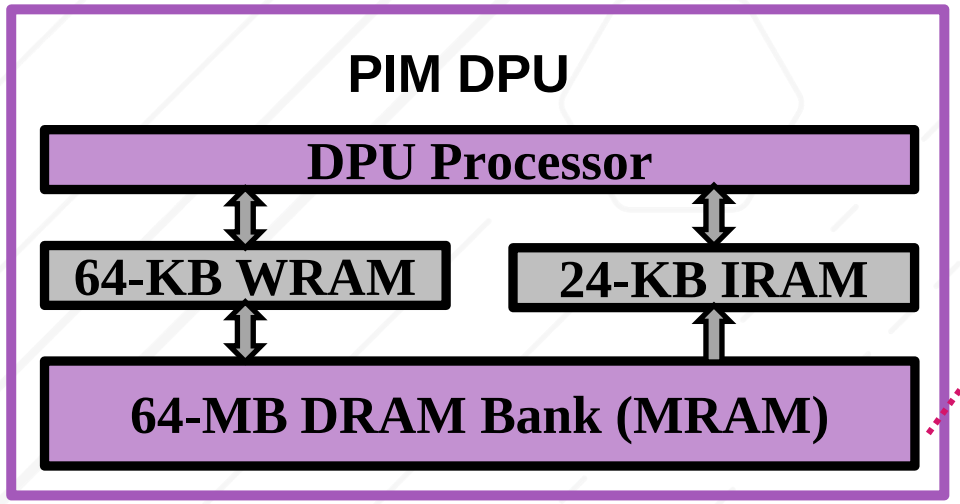


Challenge 1: Execution across Partitioned DPUs



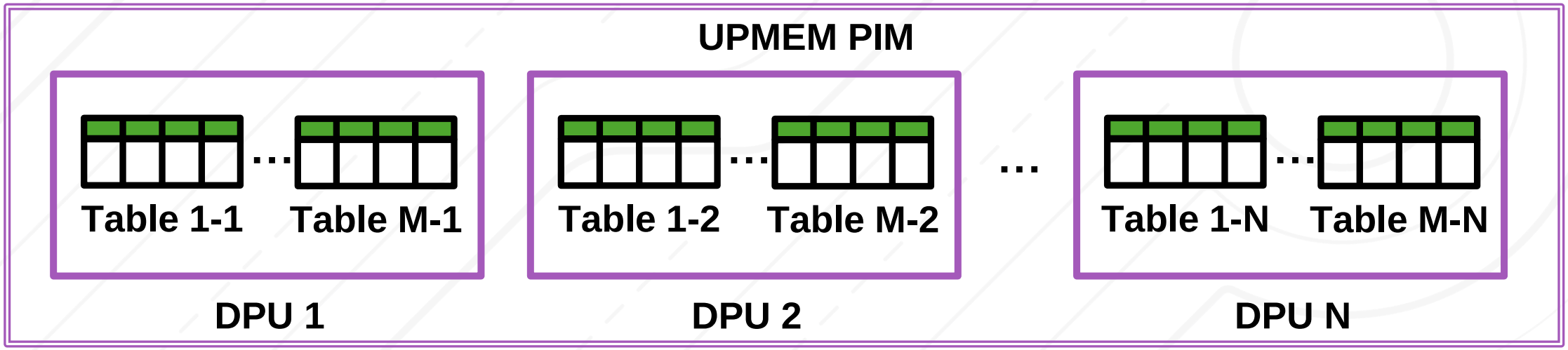
Limited memory per DPU

Challenge 1: Execution across Partitioned DPUs



Limited memory per DPU

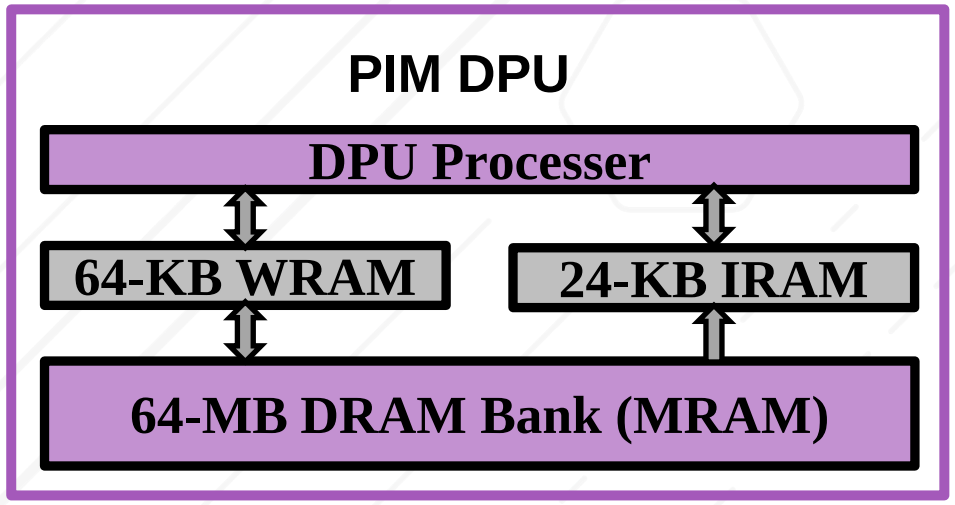
Database tables need to be horizontally partitioned

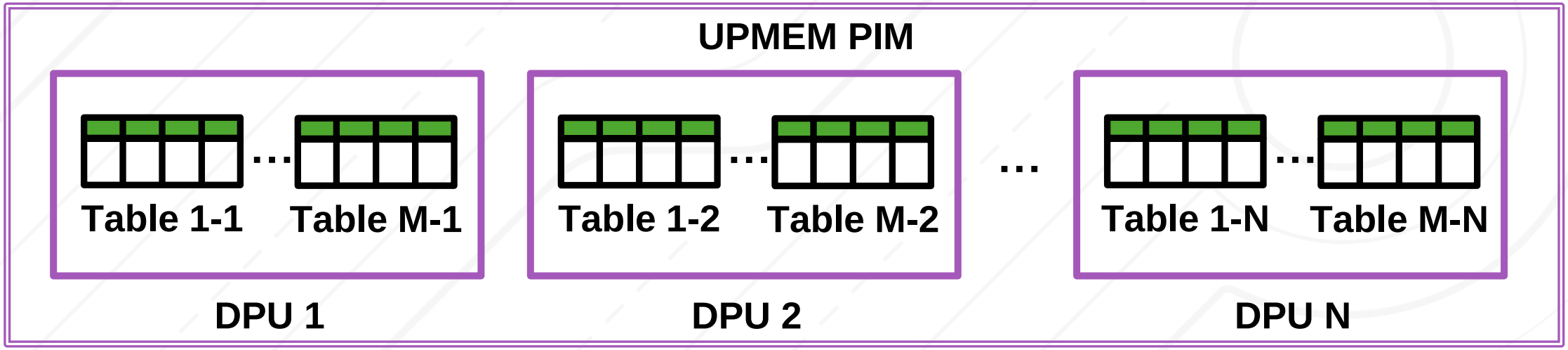


Challenge 1: Execution across Partitioned DPUs

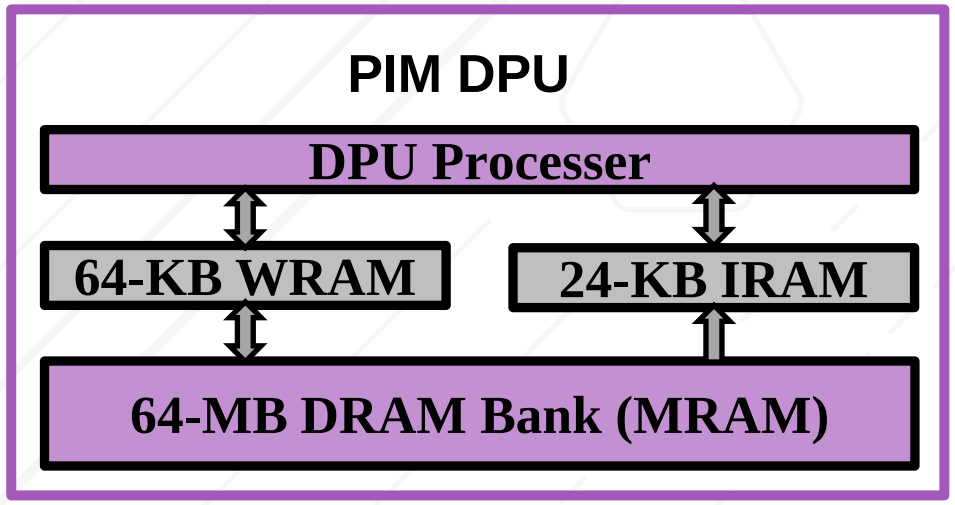
We orchestrate the transaction execution workflow to fully utilize the high parallelism and bandwidth of DPU.

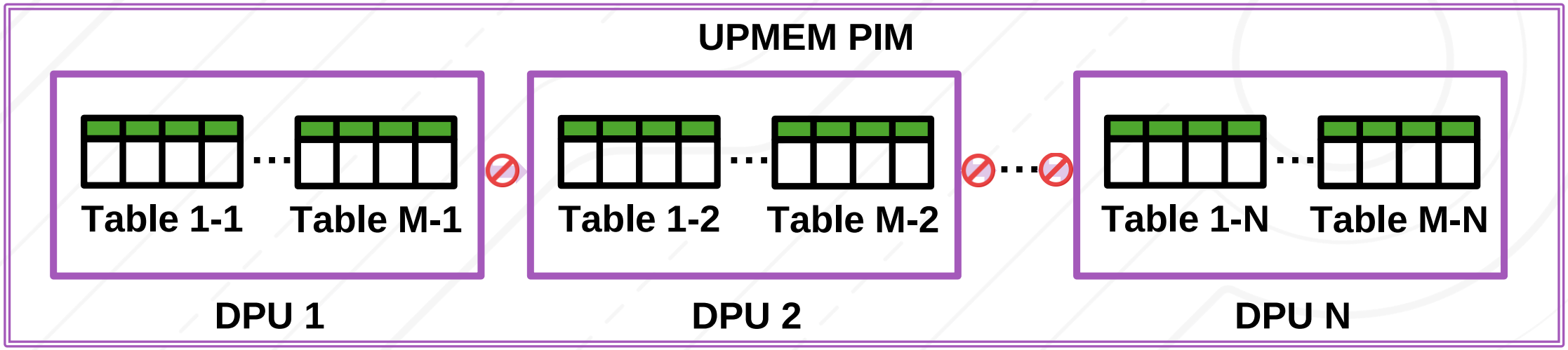
Challenge 2: Lack of Inter-DPU Communication



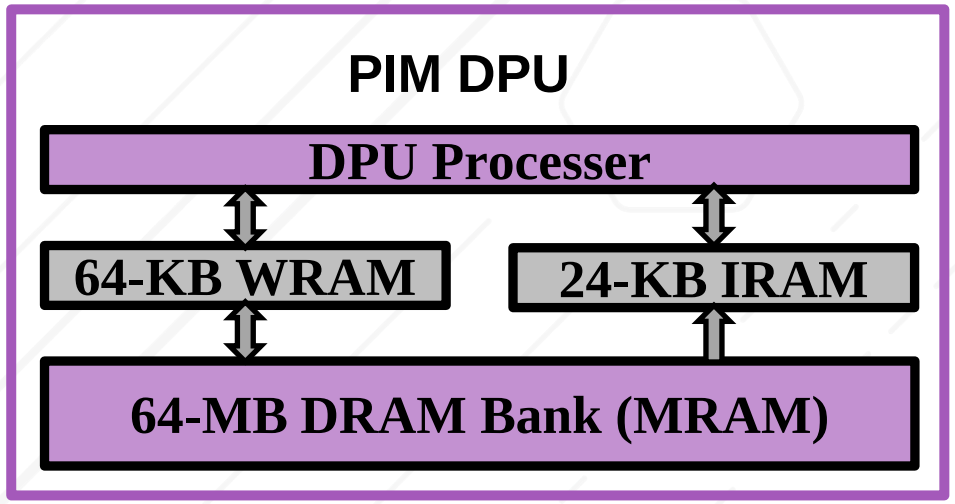


Challenge 2: Lack of Inter-DPU Communication

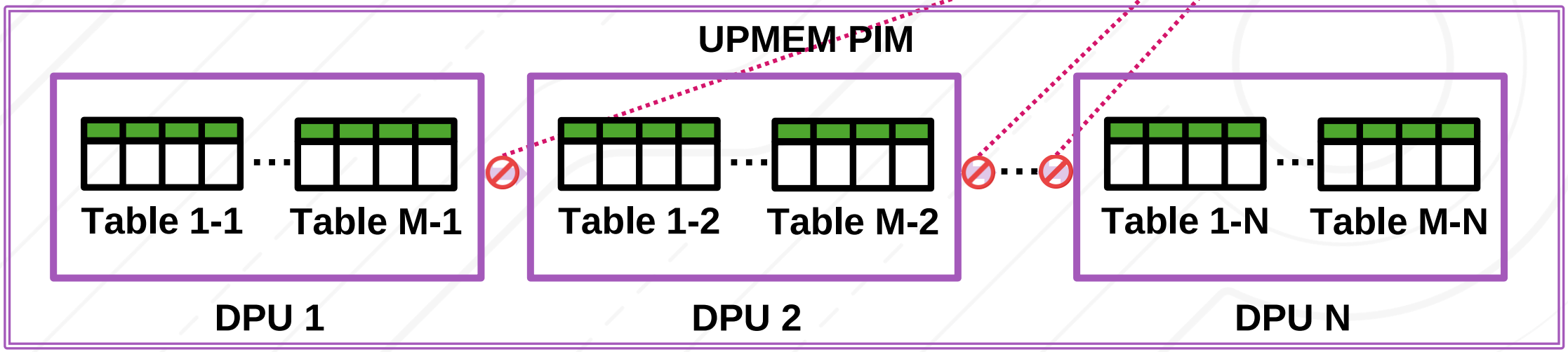




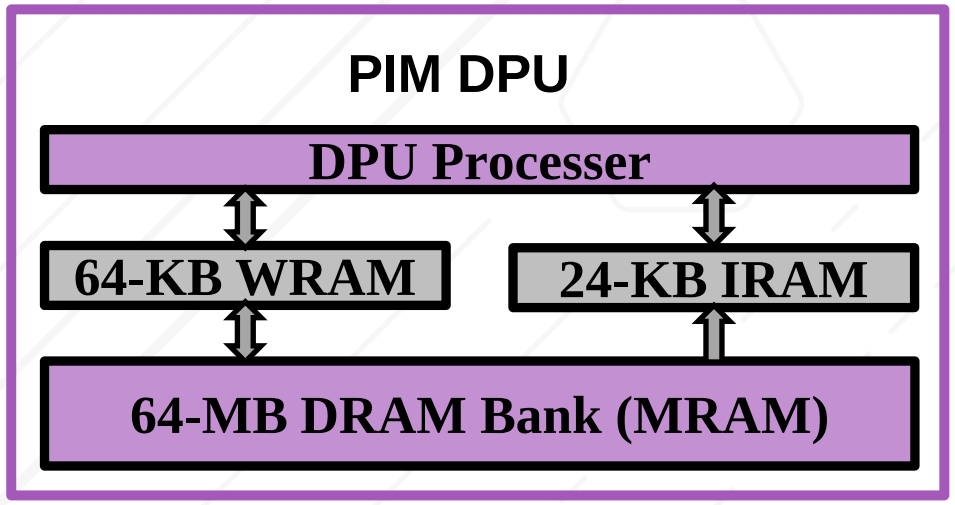
Challenge 2: Lack of Inter-DPU Communication



No direct data transfer between DPUs

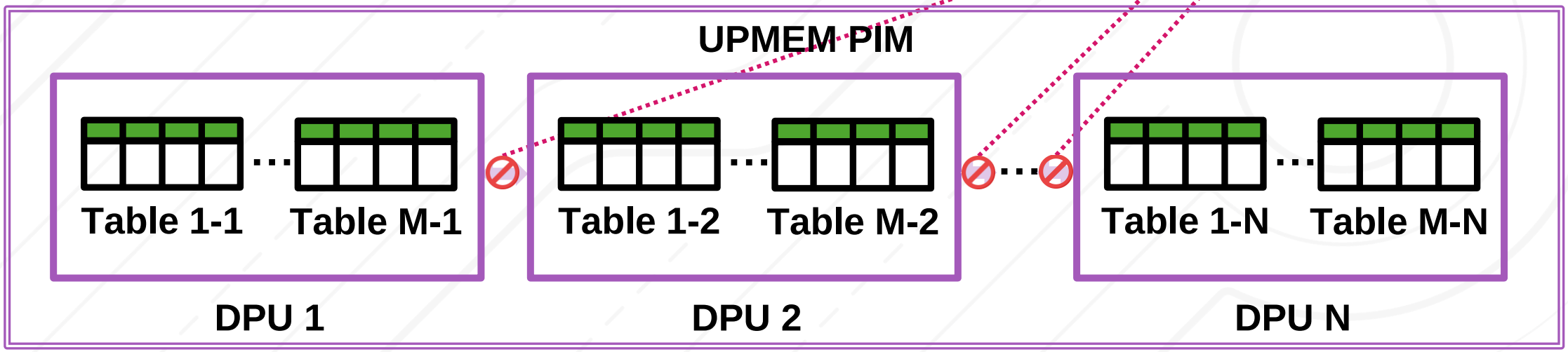


Challenge 2: Lack of Inter-DPU Communication



Data transfer overhead between CPU and DPUs is inevitable

No direct data transfer between DPUs



Challenge 2: Lack of Inter-DPU Communication

Data transfer overhead between CPU and DPUs is inevitable

No direct data transfer between DPUs

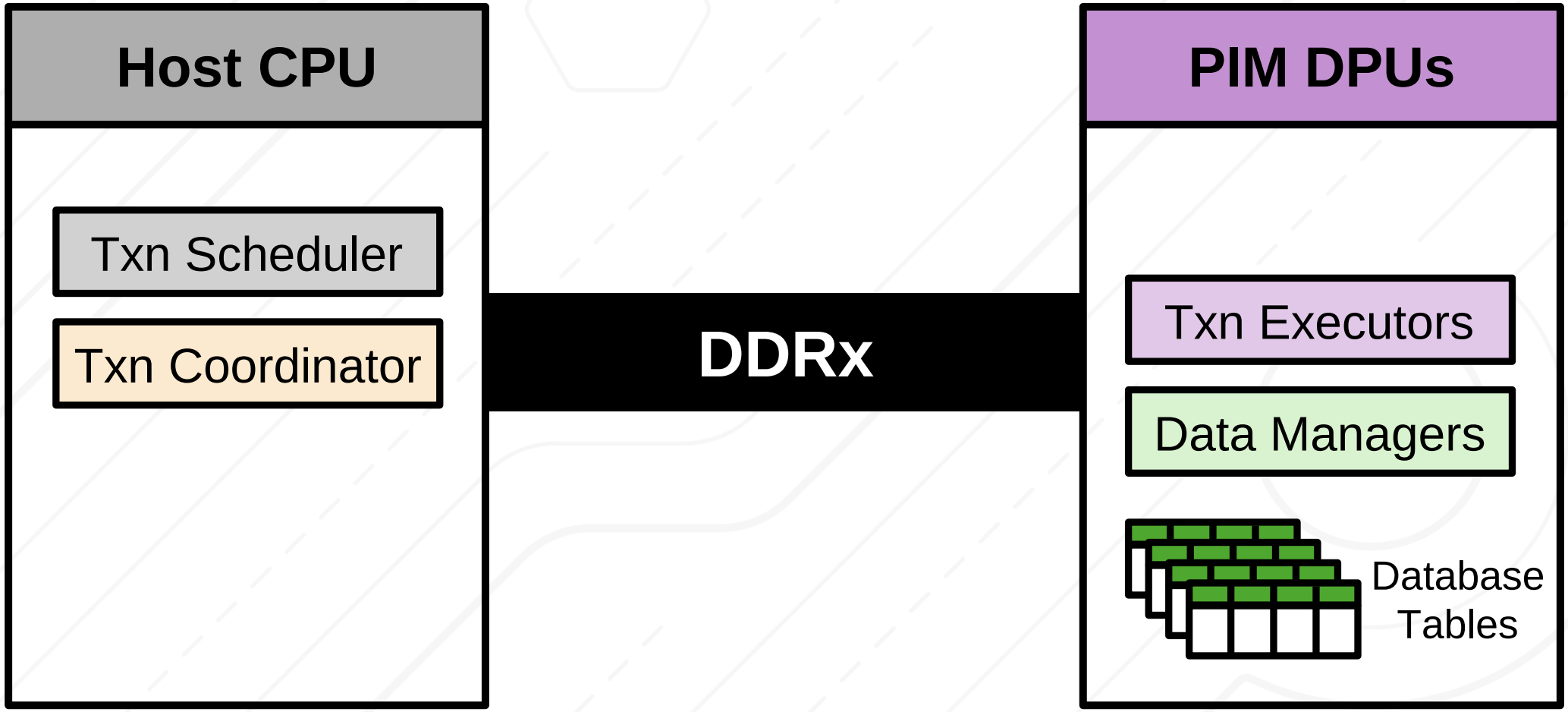
We need to minimize data transfer overhead between the host CPU and DPUs.

Our Solution: Onyx

Onyx: ***A Transaction Processing System with Real PIM Prototypes***

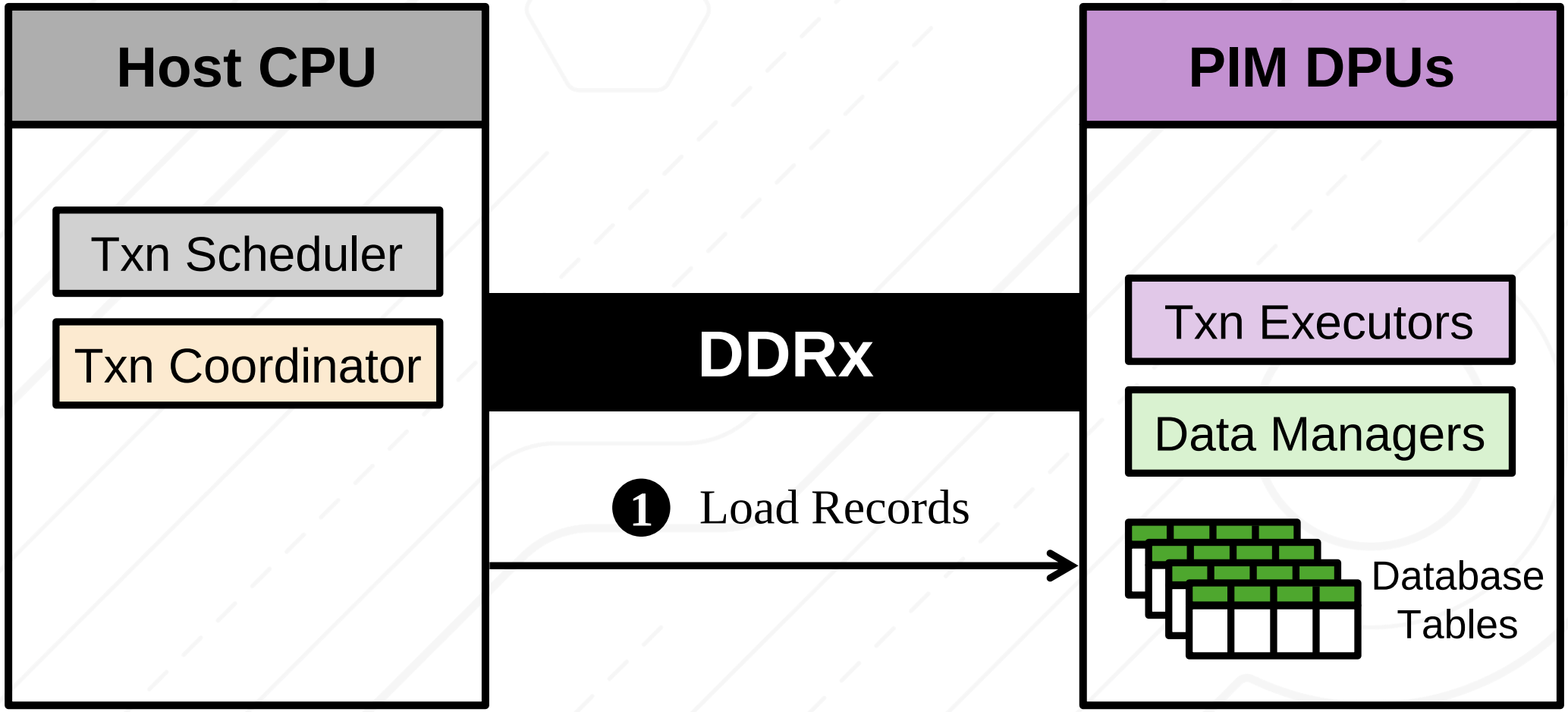
Our Solution: Onyx

Onyx: ***A Transaction Processing System with Real PIM Prototypes***



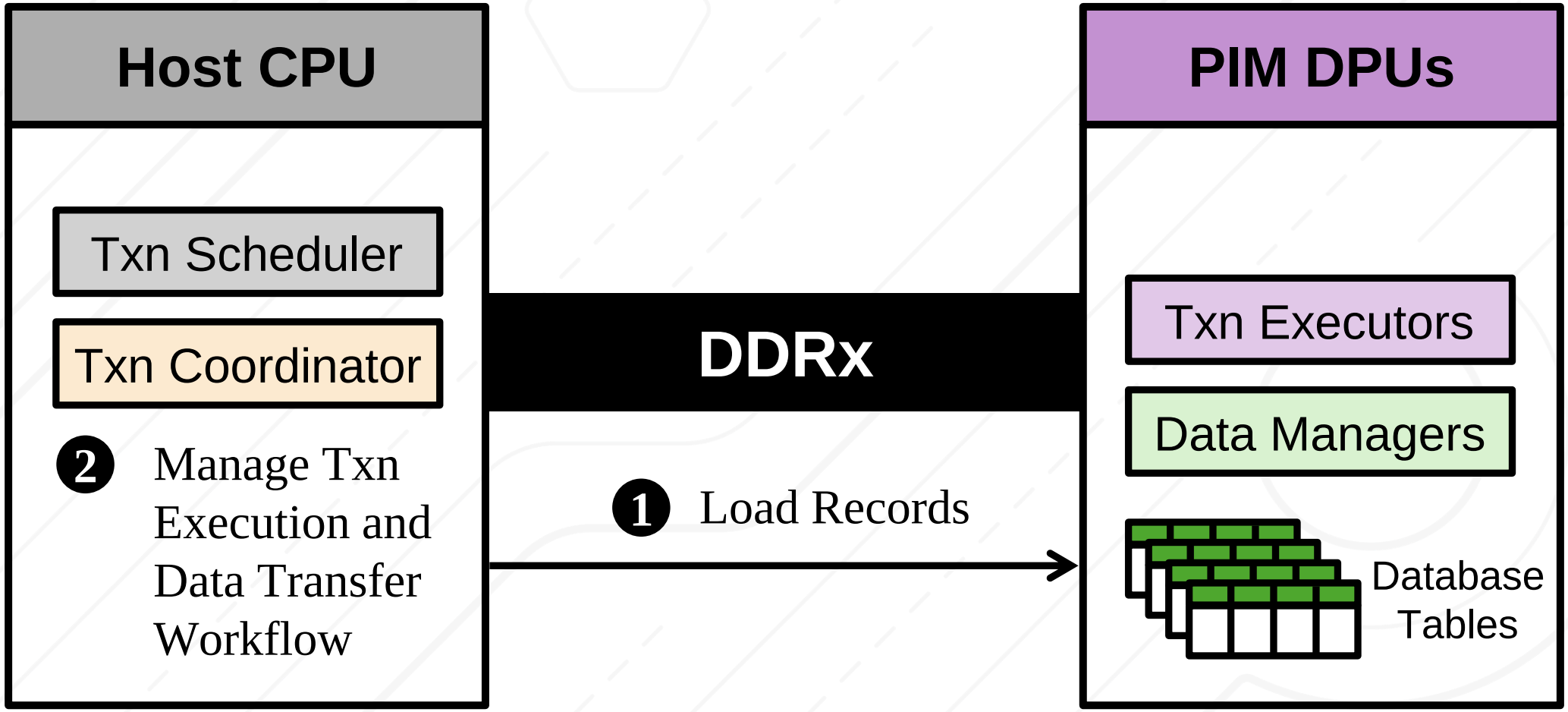
Our Solution: Onyx

Onyx: ***A Transaction Processing System with Real PIM Prototypes***



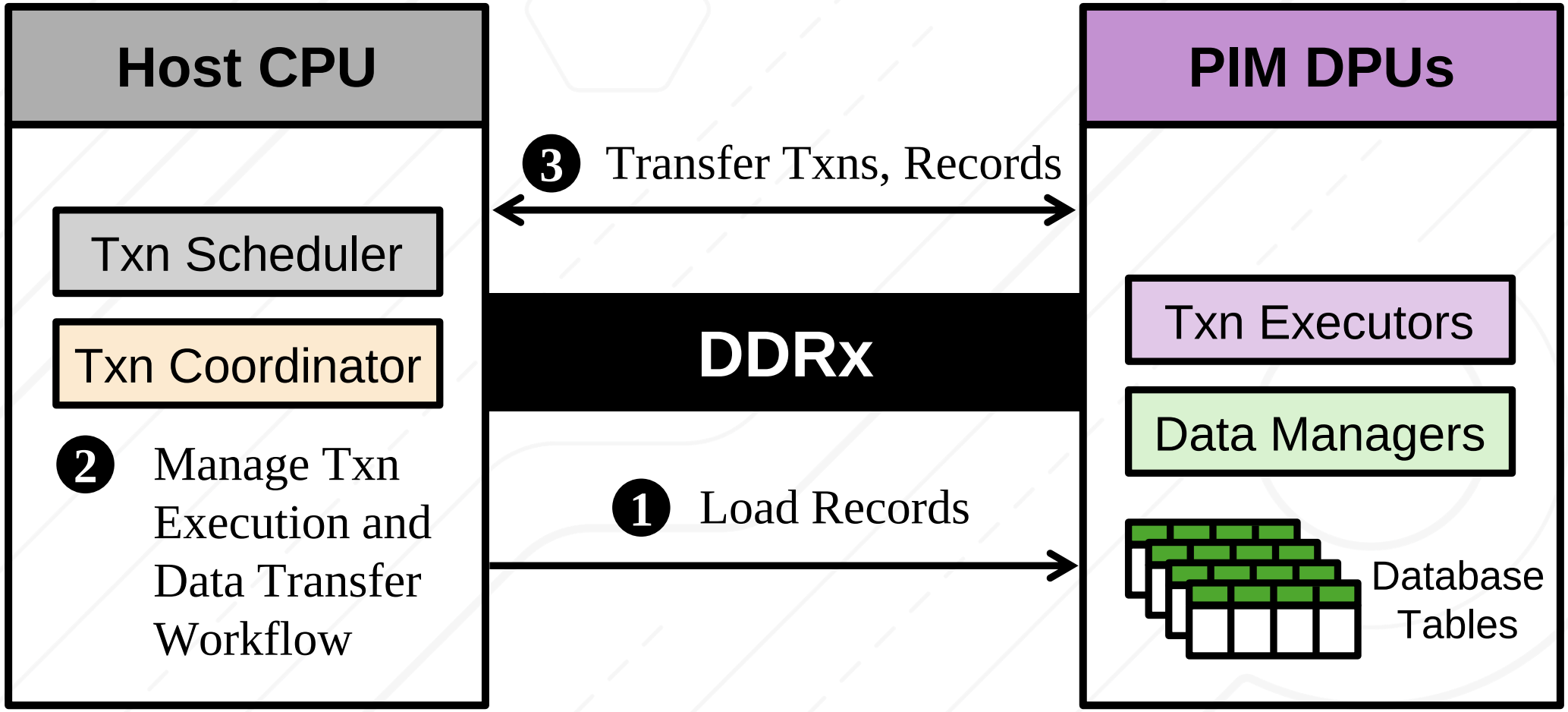
Our Solution: Onyx

Onyx: ***A Transaction Processing System with Real PIM Prototypes***



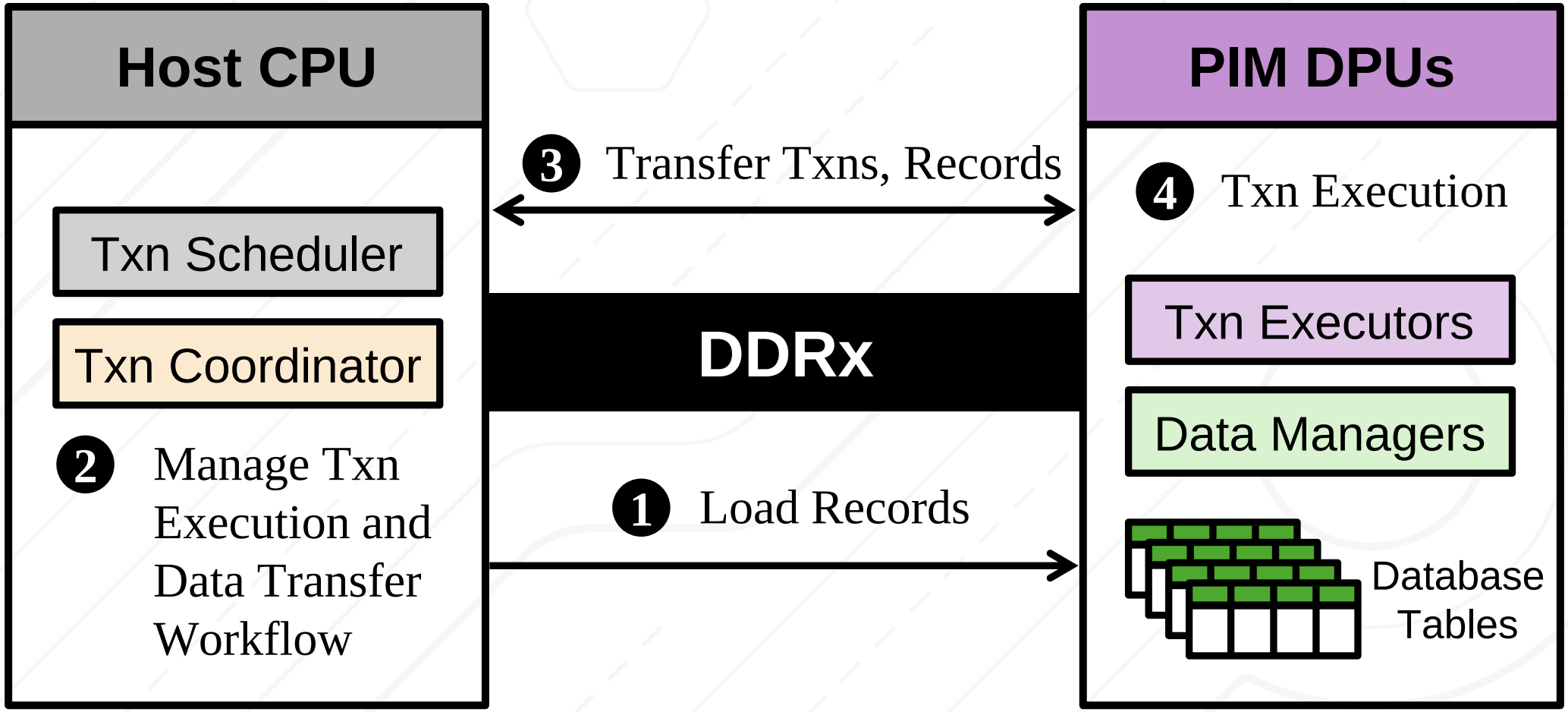
Our Solution: Onyx

Onyx: ***A Transaction Processing System with Real PIM Prototypes***

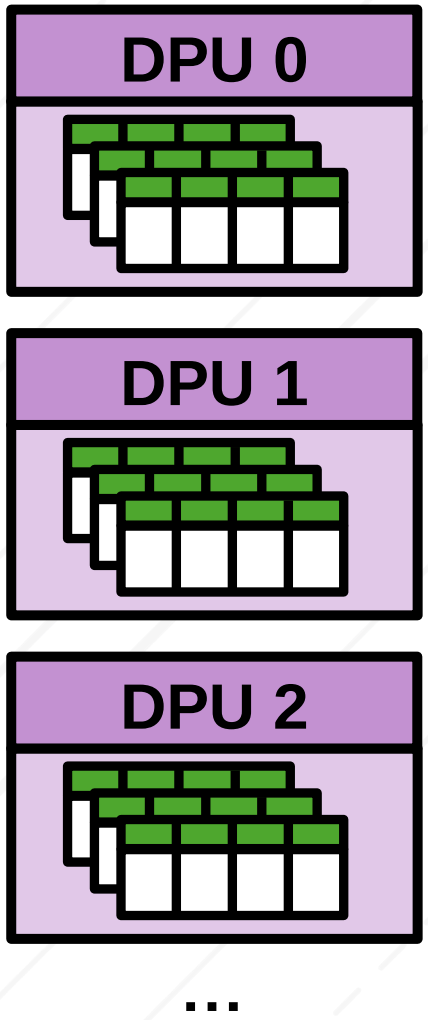


Our Solution: Onyx

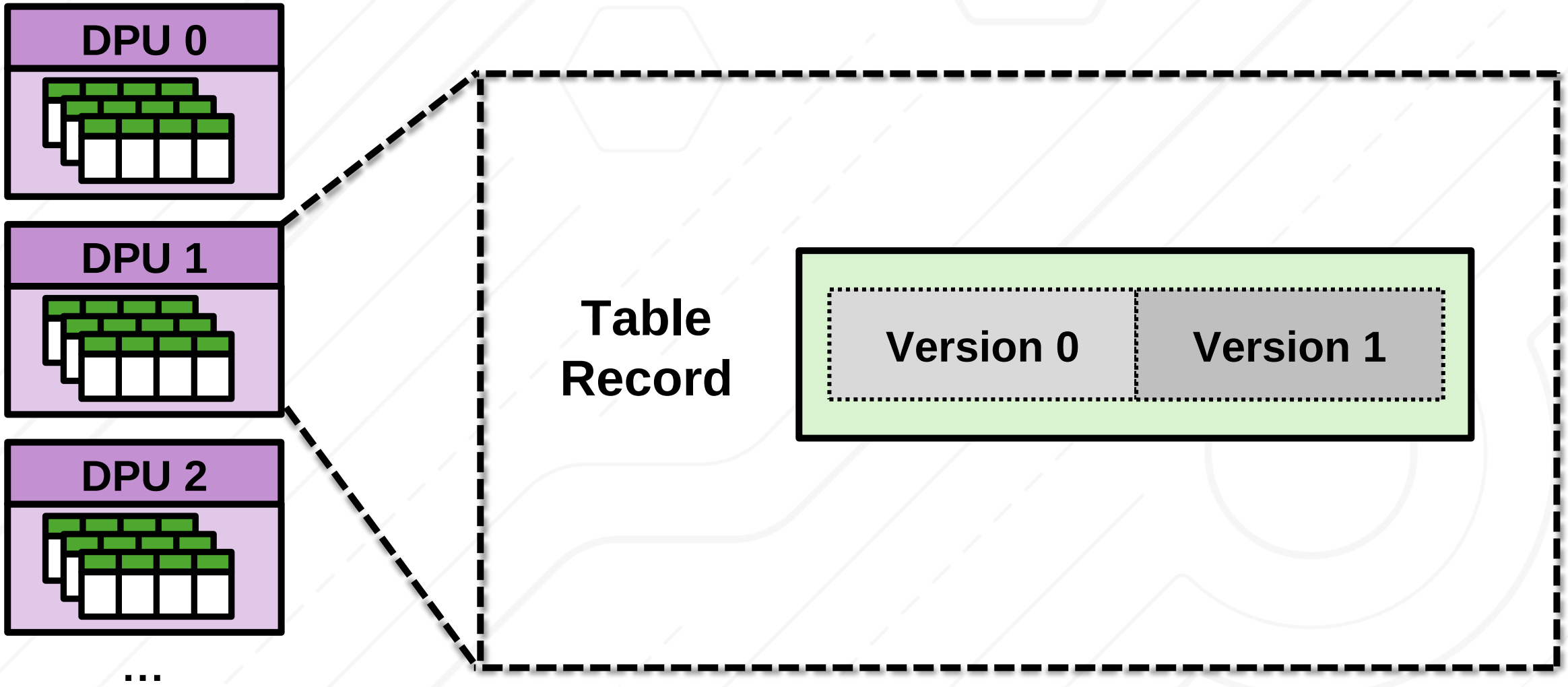
Onyx: ***A Transaction Processing System with Real PIM Prototypes***



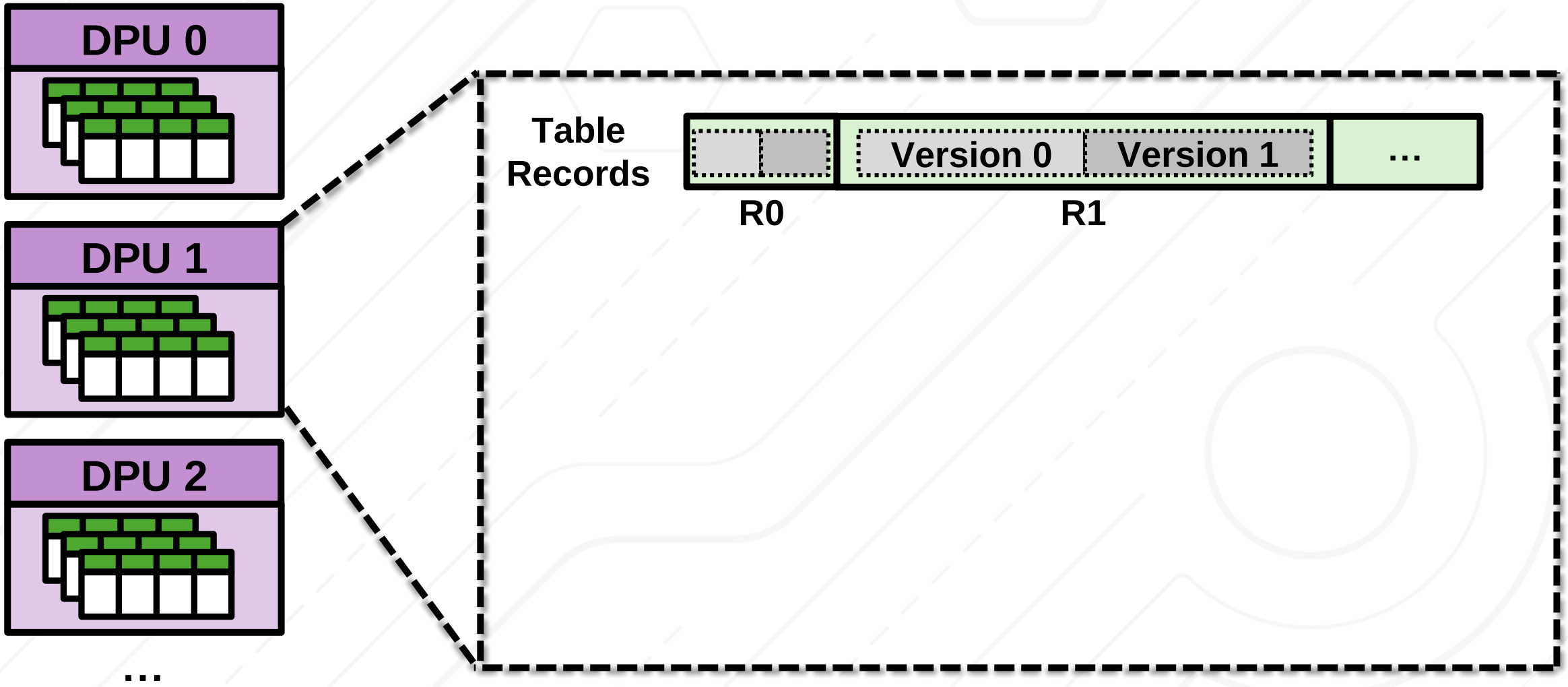
Multi-version Storage Scheme



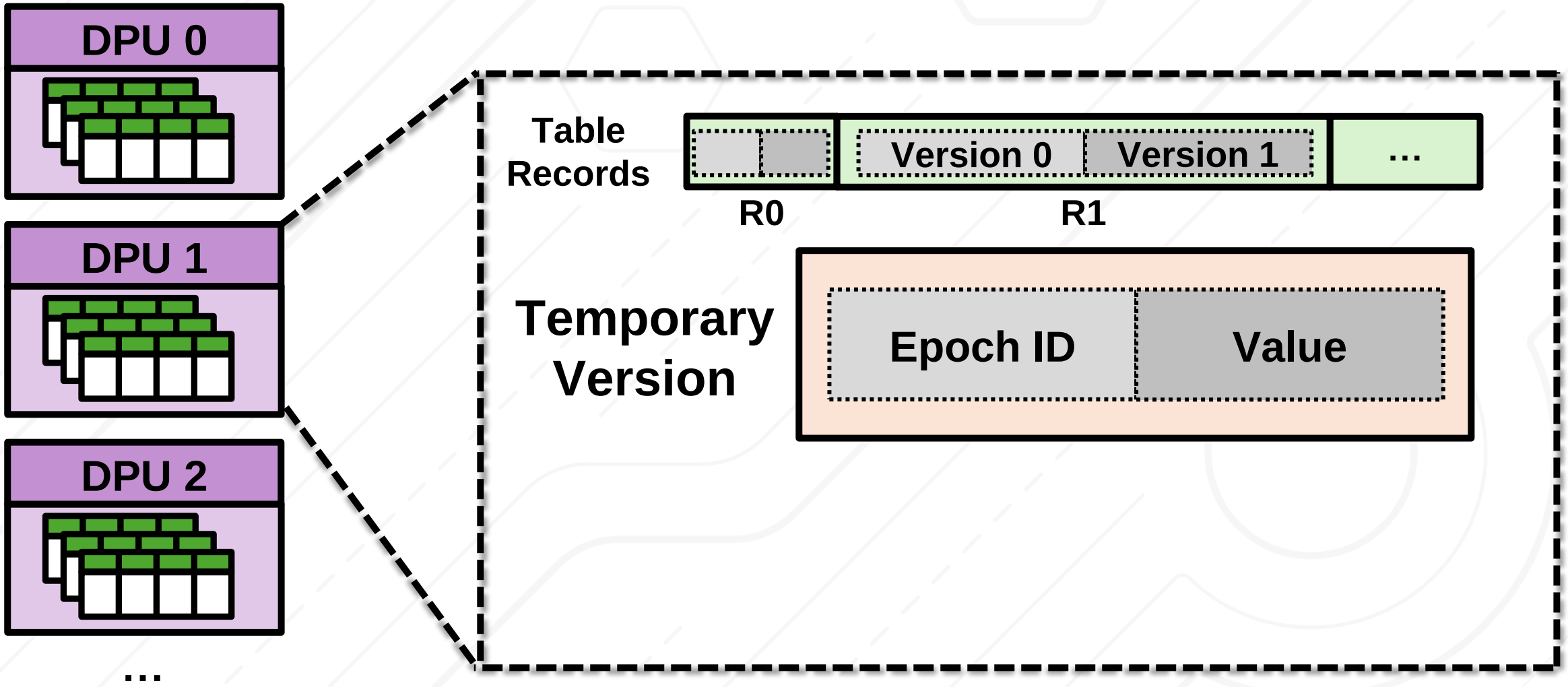
Multi-version Storage Scheme



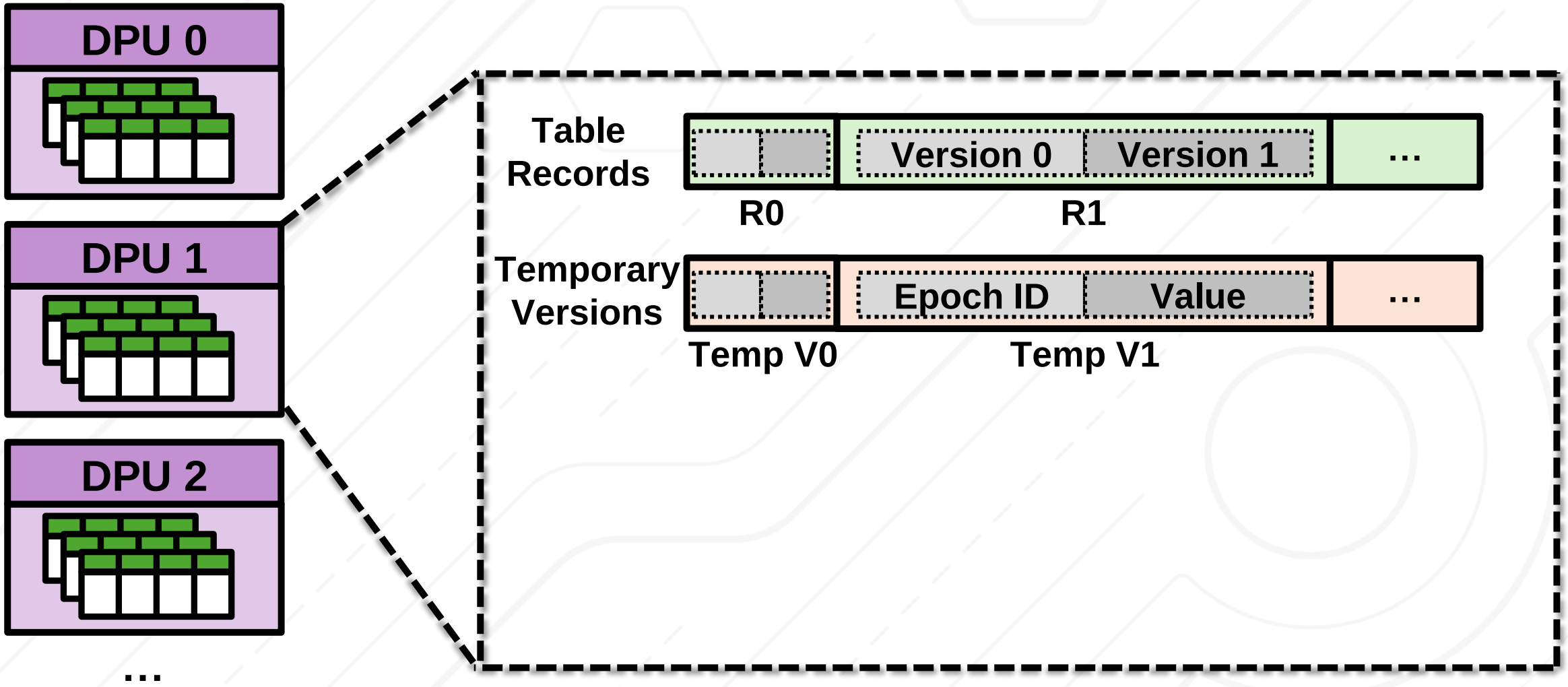
Multi-version Storage Scheme



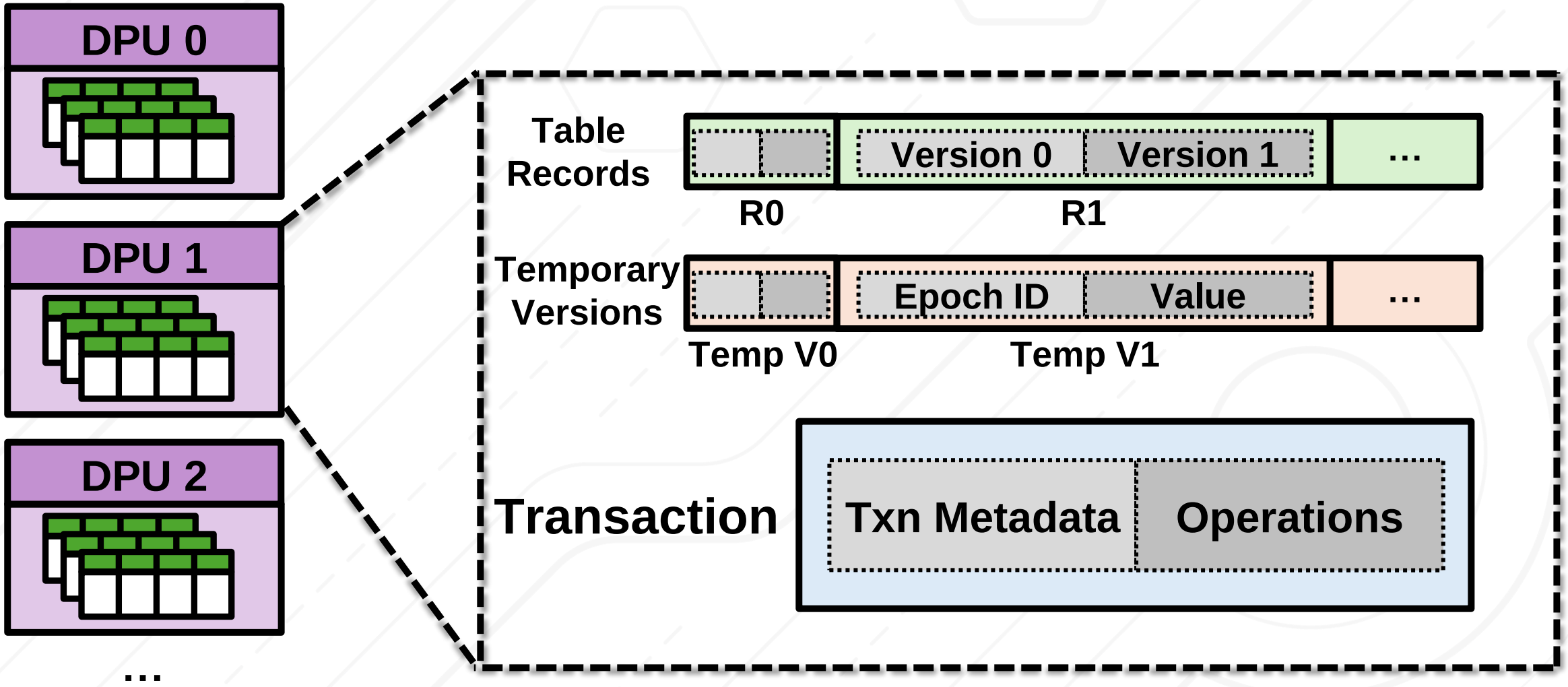
Multi-version Storage Scheme



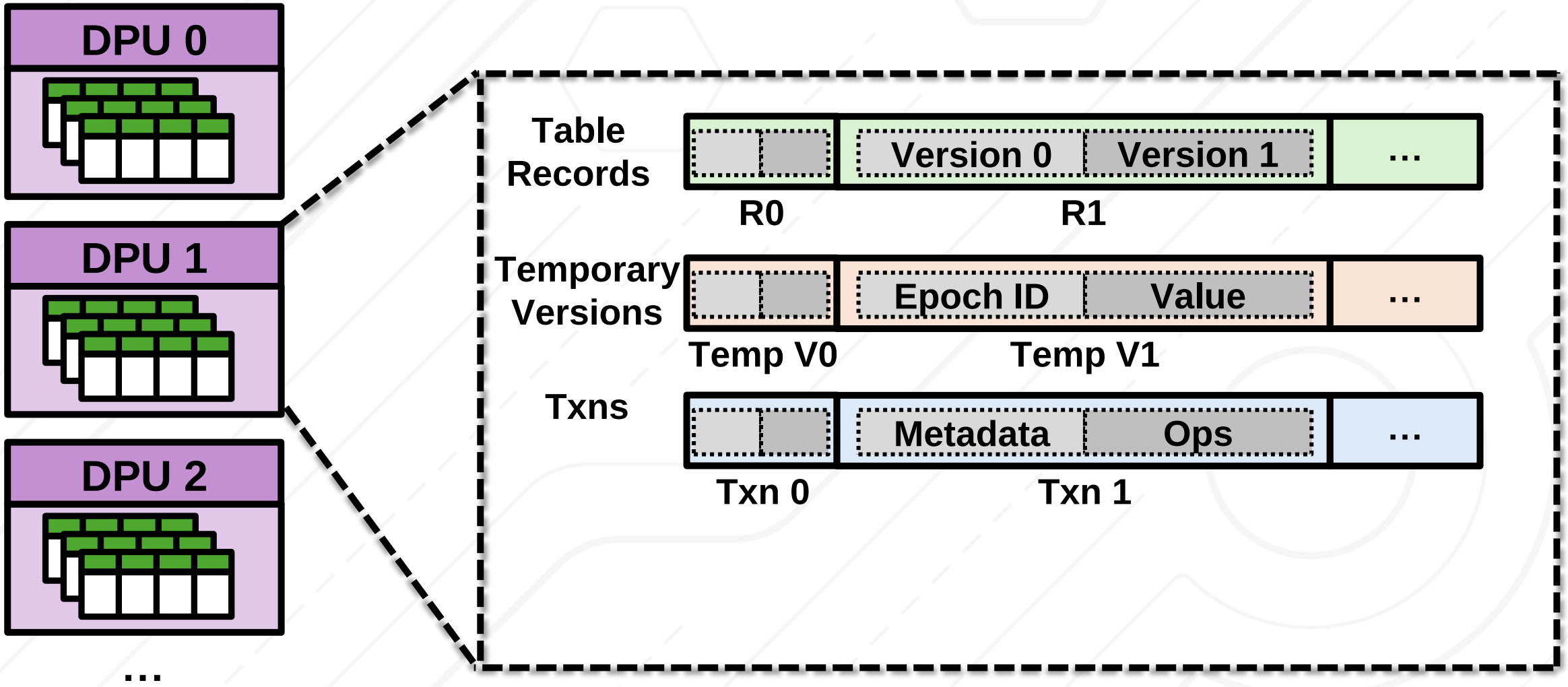
Multi-version Storage Scheme



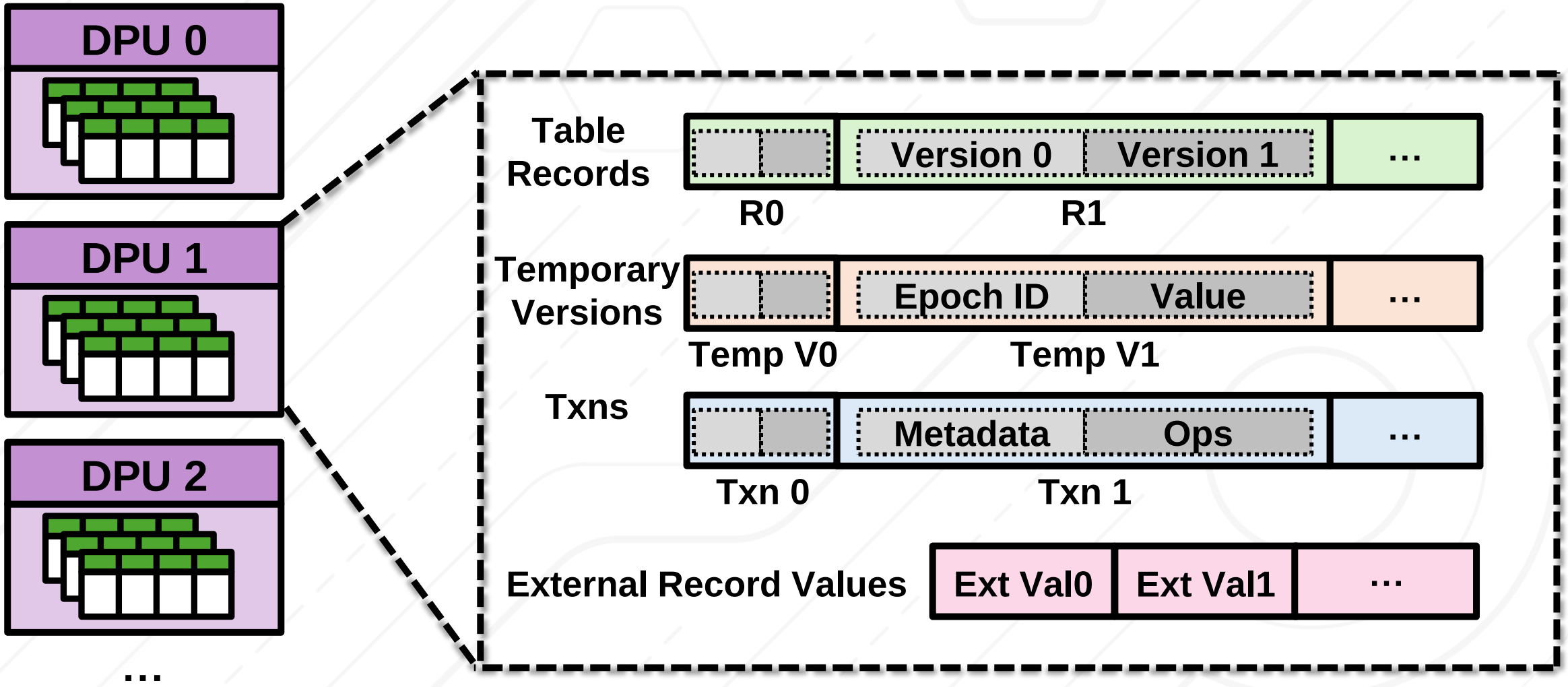
Multi-version Storage Scheme



Multi-version Storage Scheme



Multi-version Storage Scheme



Multi-versioning Initialization

Txn 0 READ A, B
WRITE A = A + B

Txn 1 READ A, C, D
WRITE A = A - D
WRITE C = C + D

Txn 2 READ B, D
WRITE B = B + 1
WRITE D = D * B

Key	Col 0	Col 1	…
A	A.c0	A.c1	
C	C.c0	C.c1	
…			

…

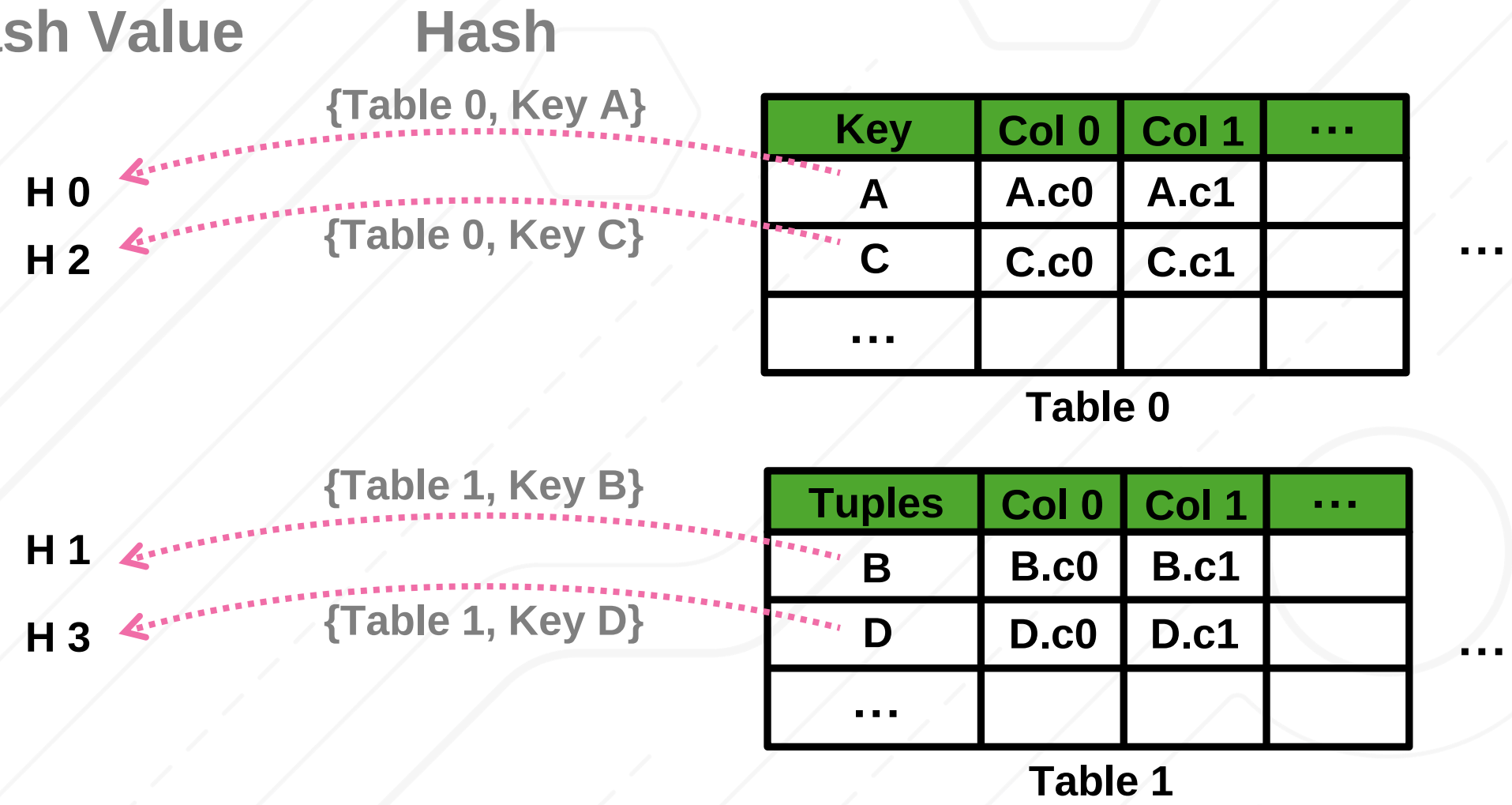
Table 0

Tuples	Col 0	Col 1	…
B	B.c0	B.c1	
D	D.c0	D.c1	
…			

…

Table 1

Multi-versioning Initialization



Multi-versioning Initialization

Keys	A	B	C	D
Hash Value	H0	H1	H2	H3
DPU IDs	0	0	1	1
Record IDs	0	1	2	3

Key	Col 0	Col 1	…
A	A.c0	A.c1	
C	C.c0	C.c1	
…			

…

Table 0

Tuples	Col 0	Col 1	…
B	B.c0	B.c1	
D	D.c0	D.c1	
…			

…

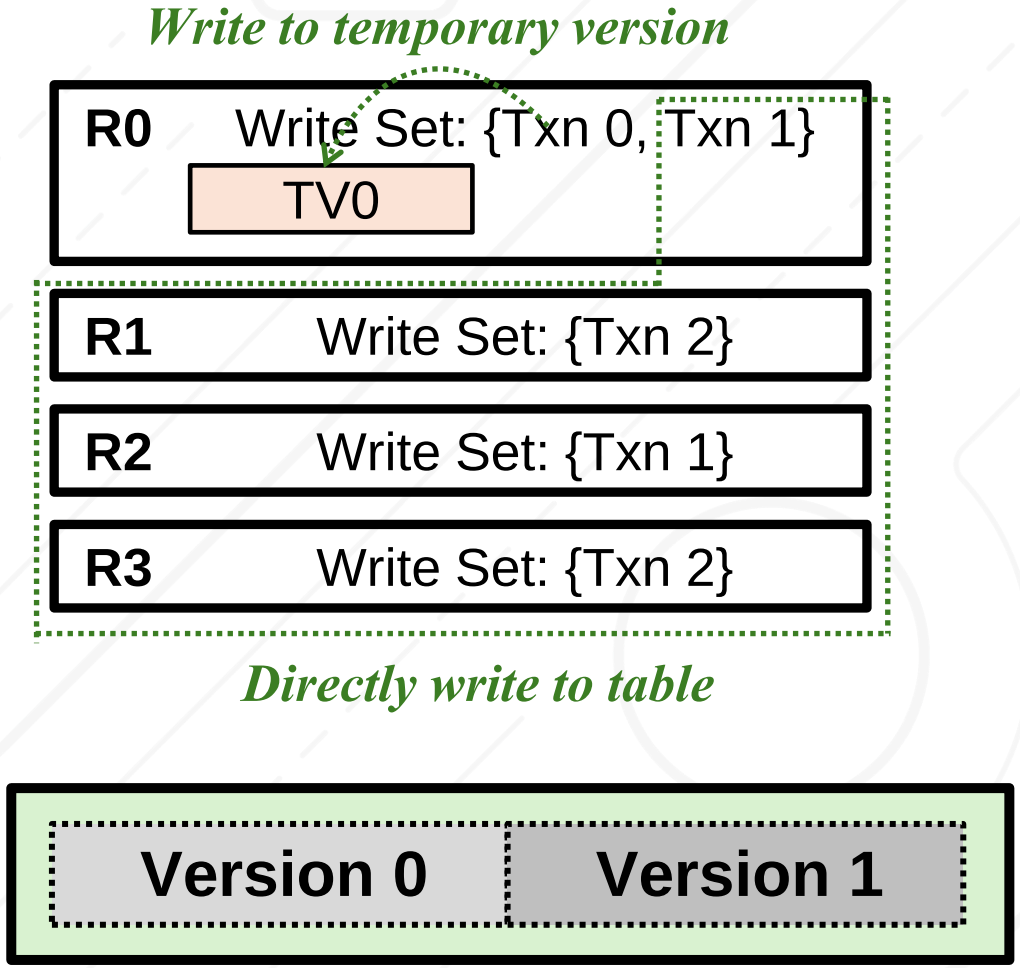
Table 1

Multi-versioning Initialization

Txn 0 READ A, B
WRITE A = A + B

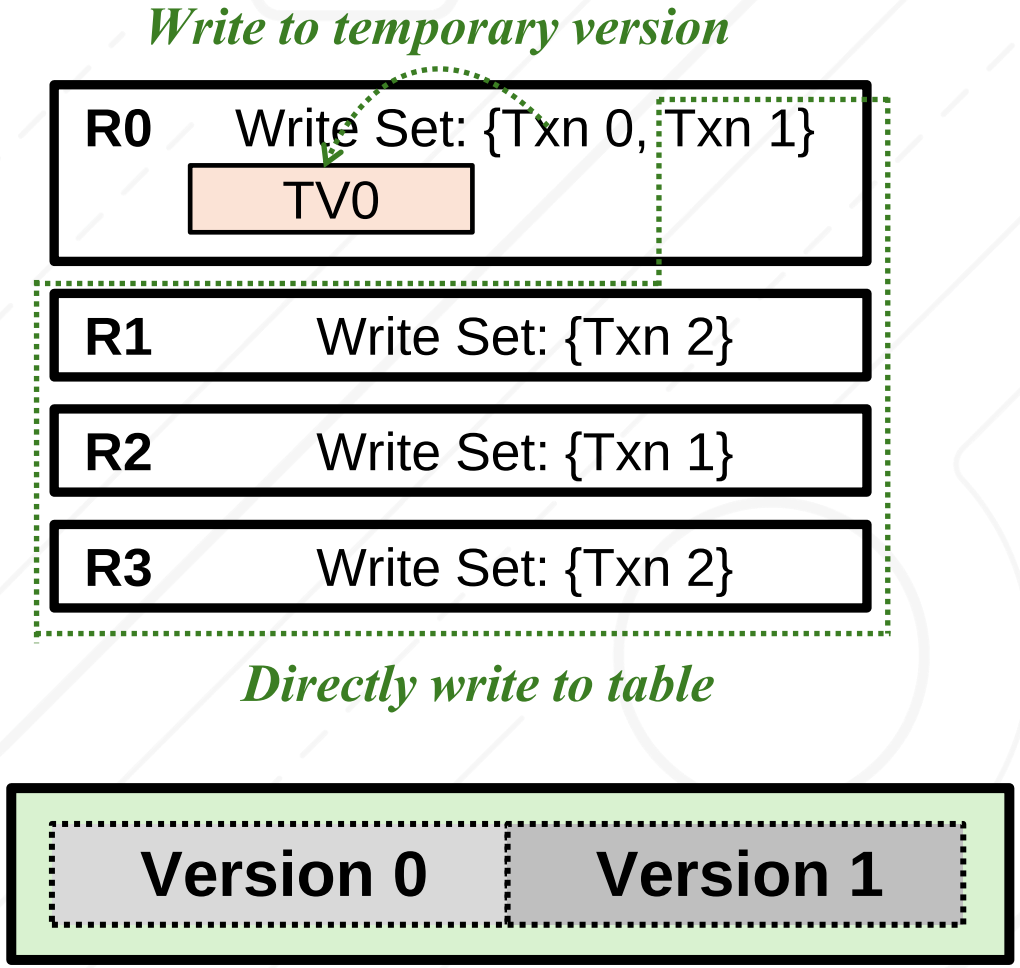
Txn 1 READ A, C, D
WRITE A = A - D
WRITE C = C + D

Txn 2 READ B, D
WRITE B = B + 1
WRITE D = D * B



Multi-versioning Initialization

Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1

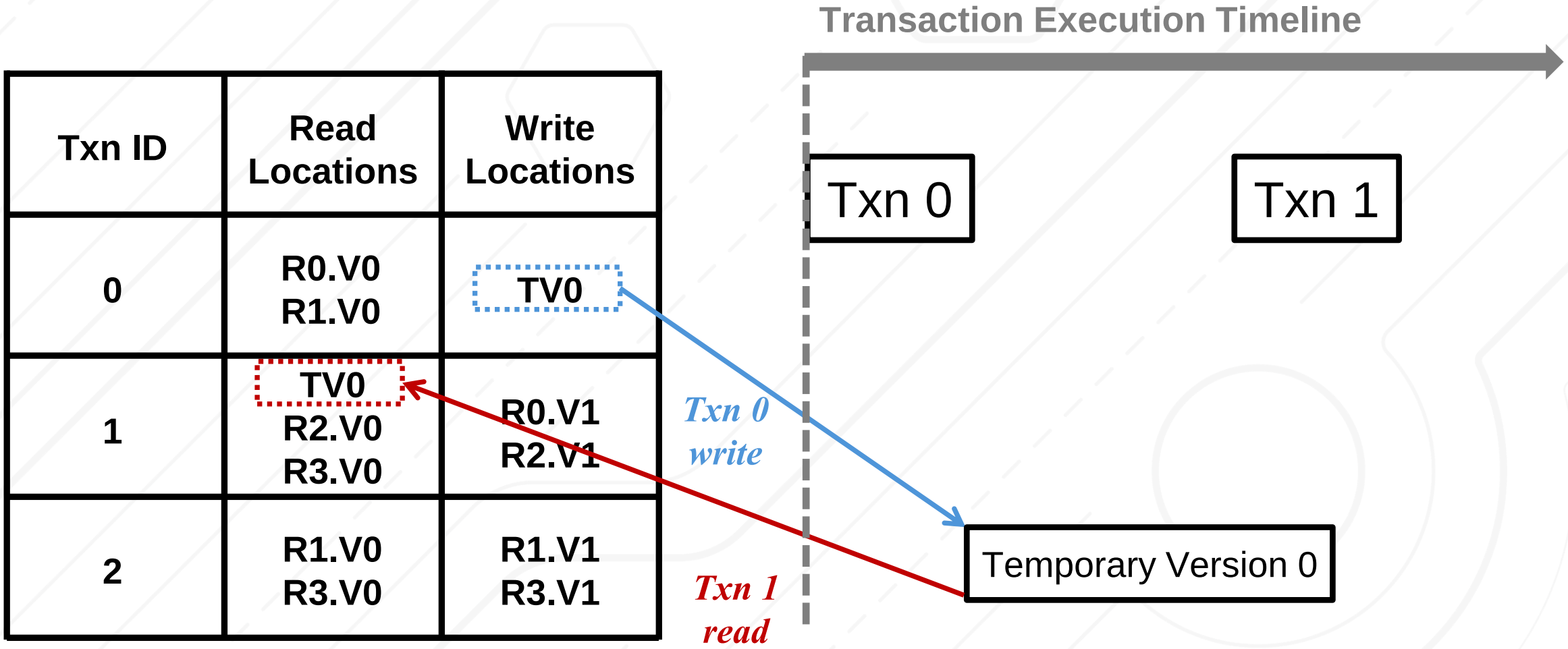


Multi-versioning Initialization

Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1

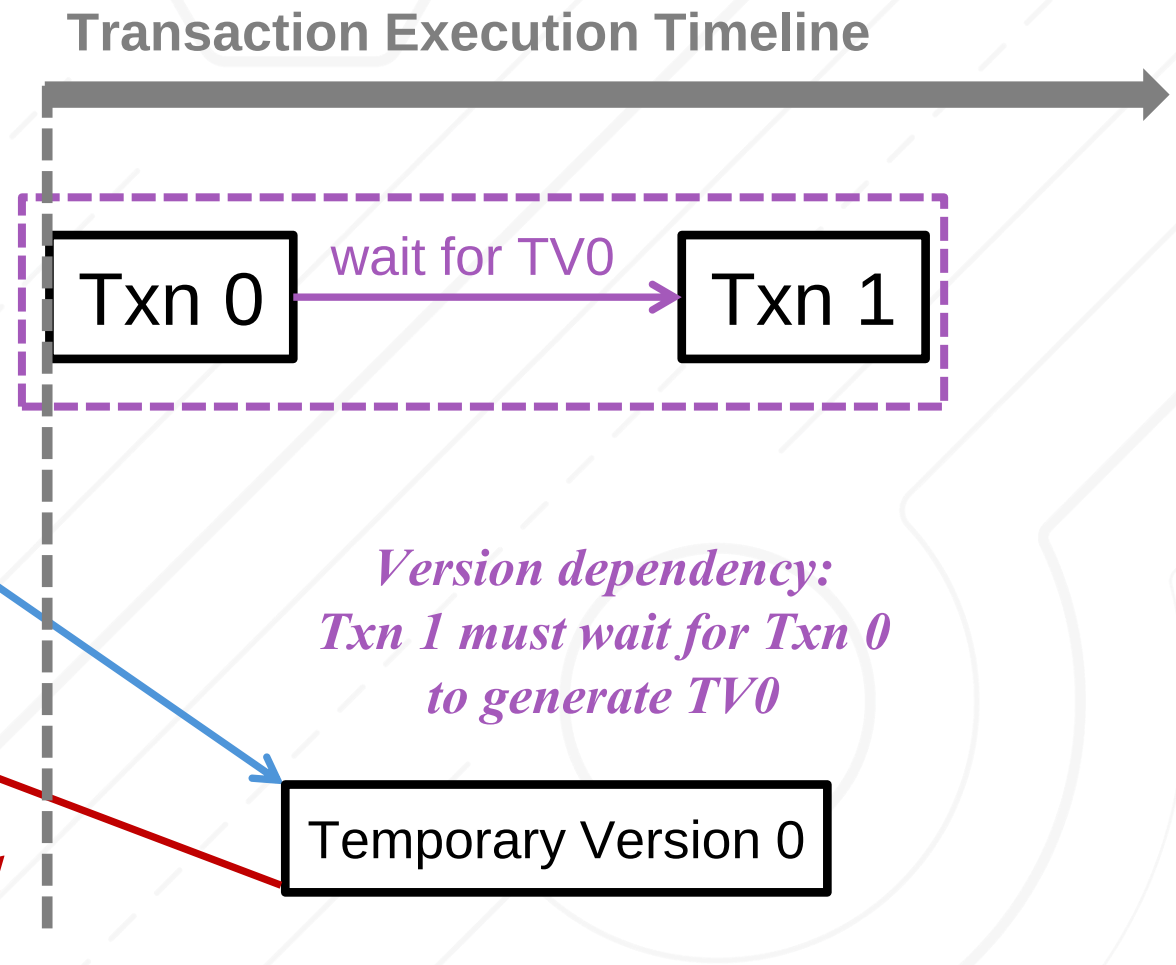
Multi-versioning Initialization

Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



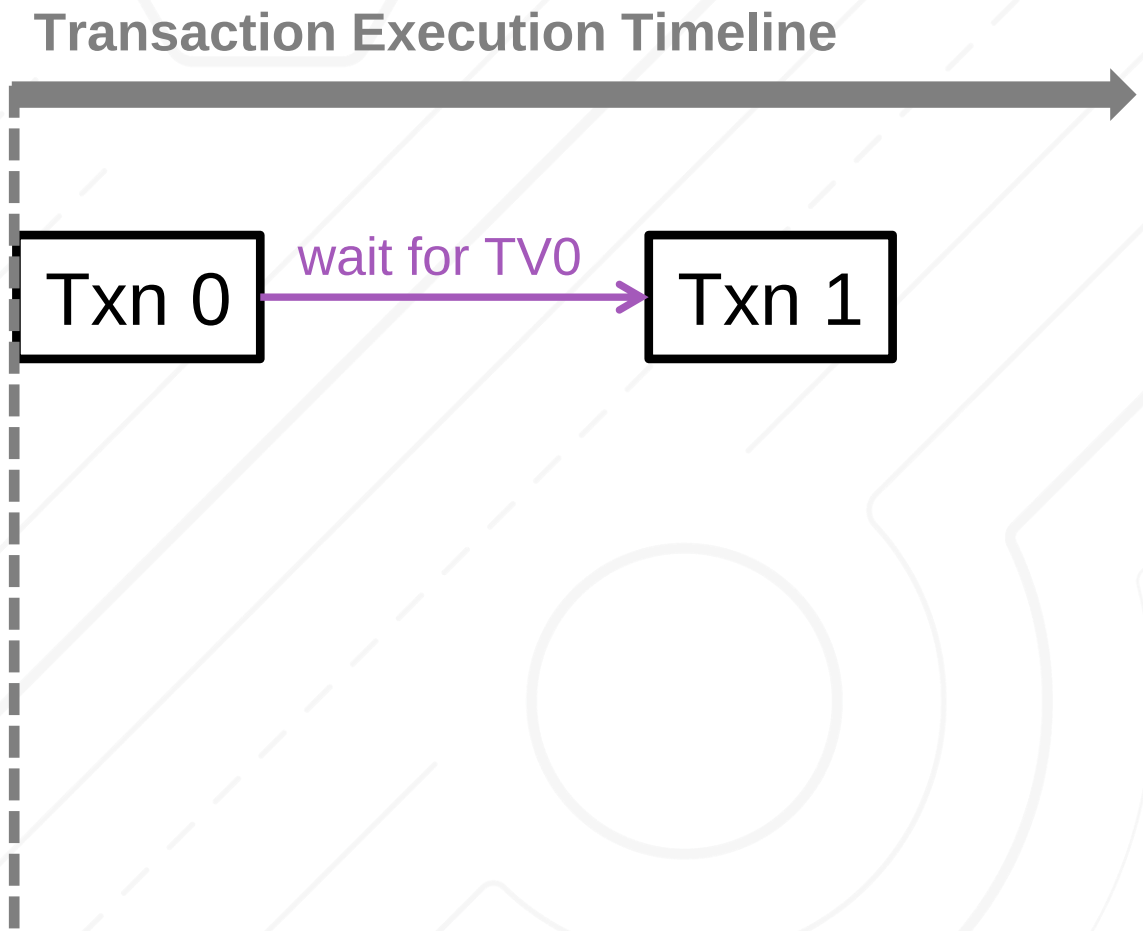
Multi-versioning Initialization

Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



Multi-versioning Initialization

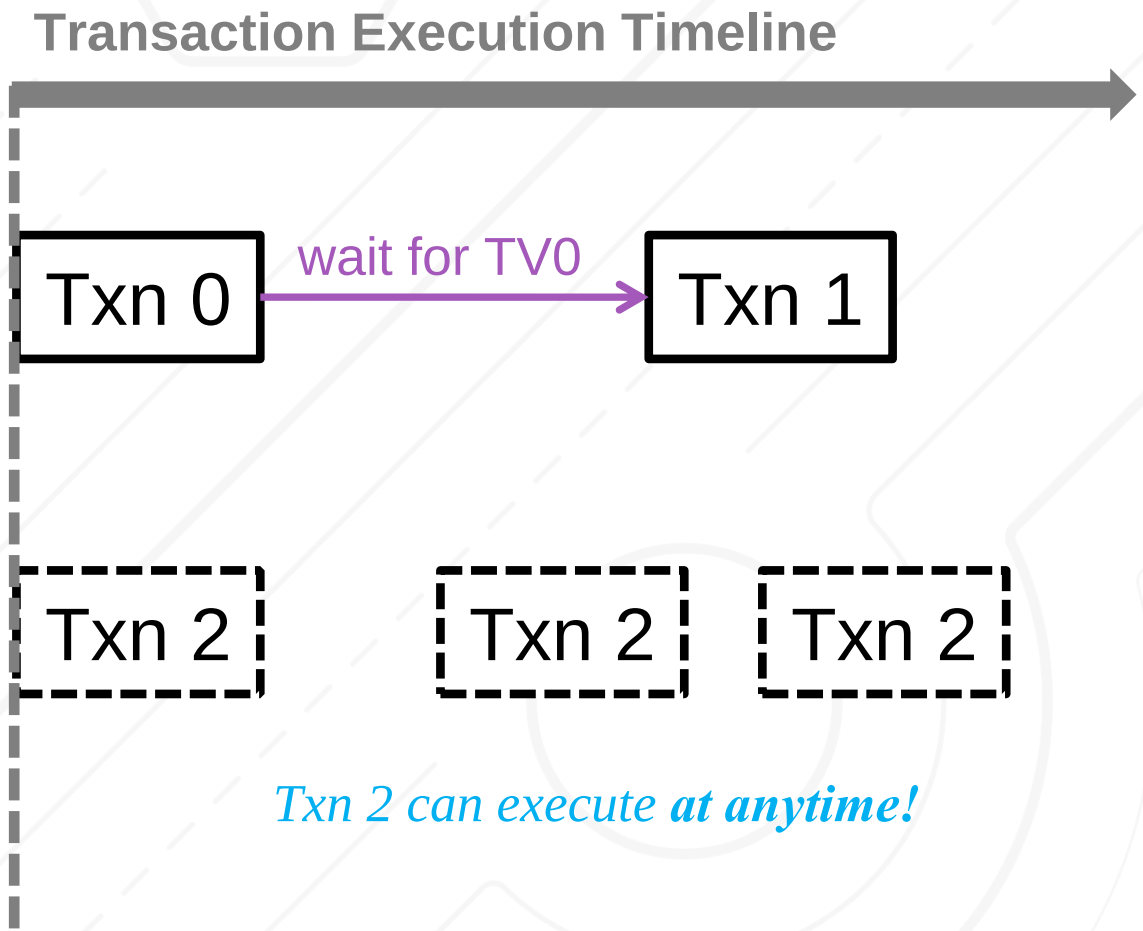
Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



No Version Dependency between Txn 2 and Txn 0 / Txn 1

Multi-versioning Initialization

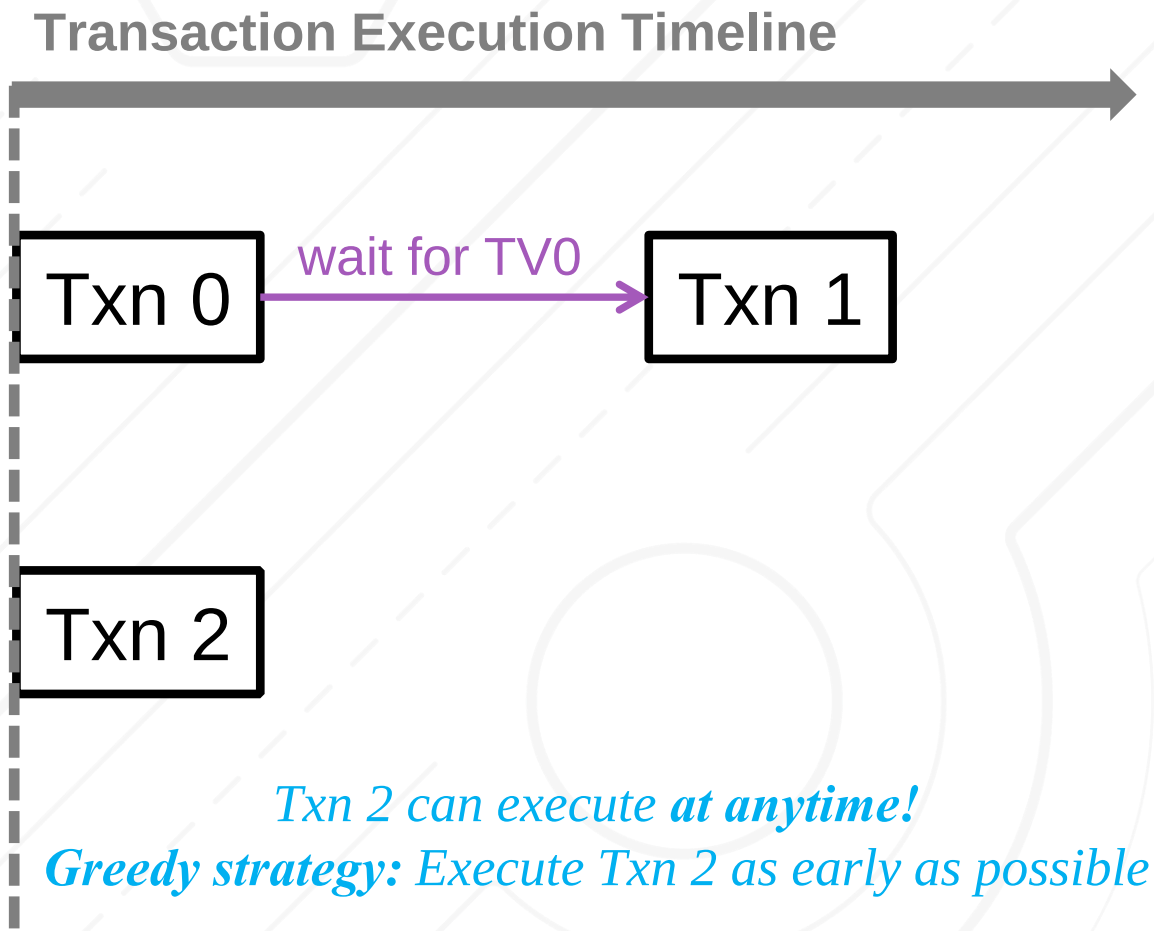
Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



No Version Dependency between Txn 2 and Txn 0 / Txn 1

Multi-versioning Initialization

Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



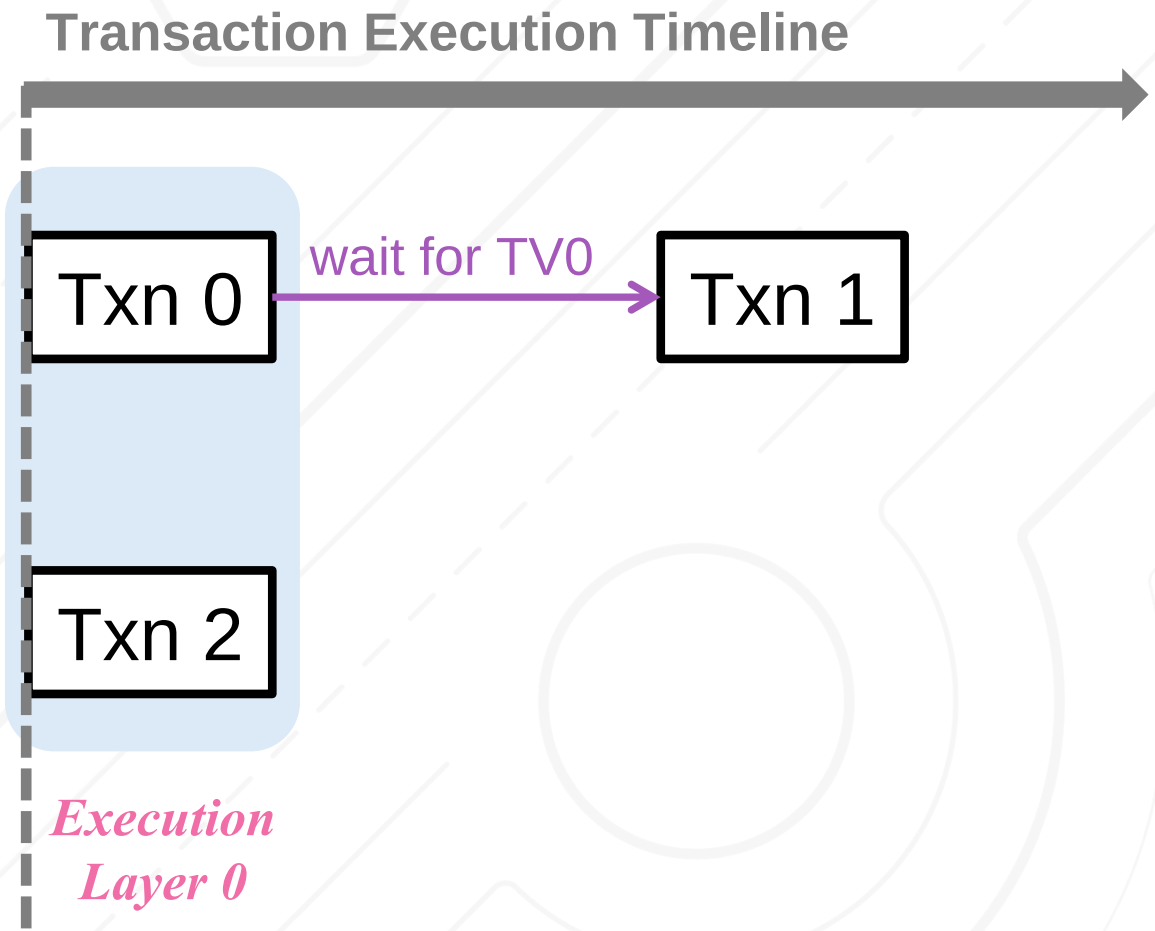
*Txn 2 can execute **at anytime!***

***Greedy strategy:** Execute Txn 2 as early as possible*

***No Version Dependency** between Txn 2 and Txn 0 / Txn 1*

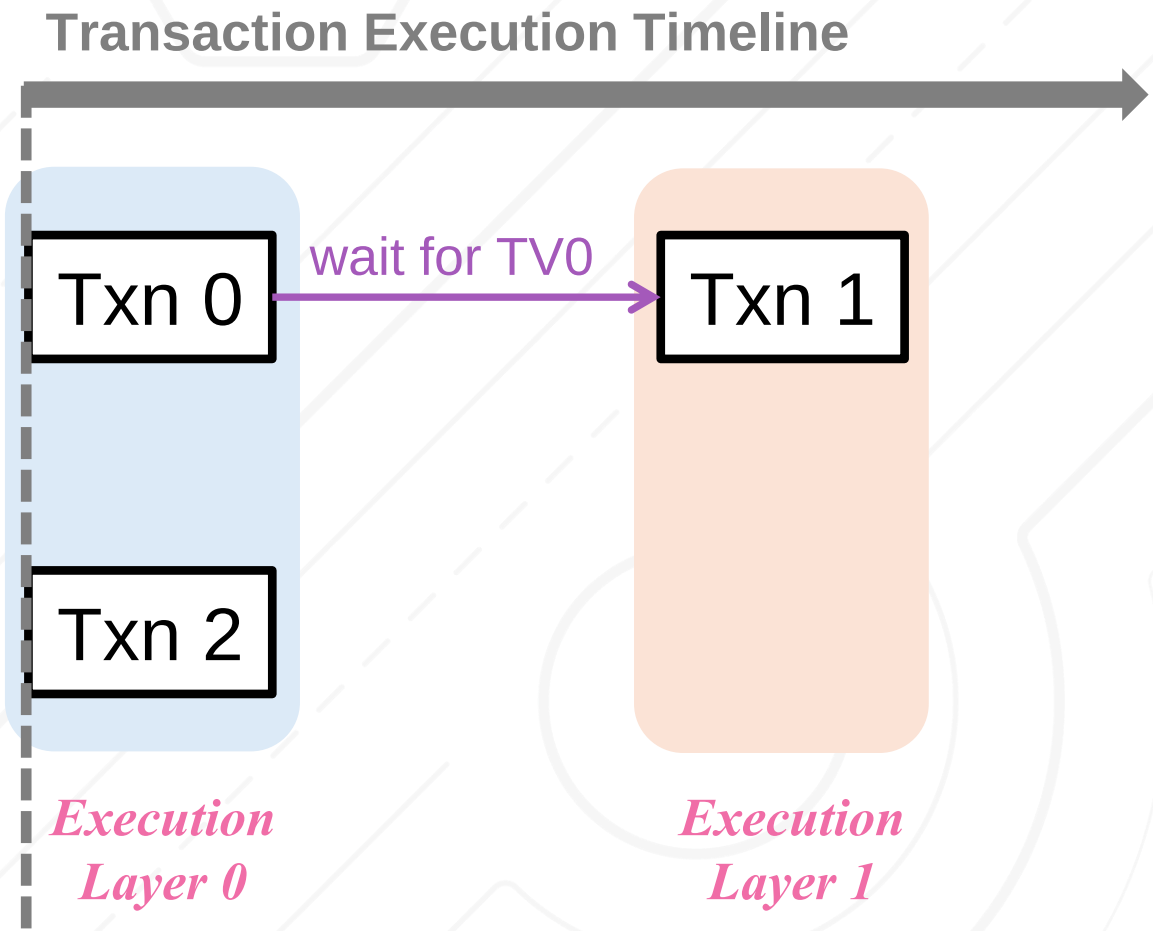
Multi-versioning Initialization

Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1

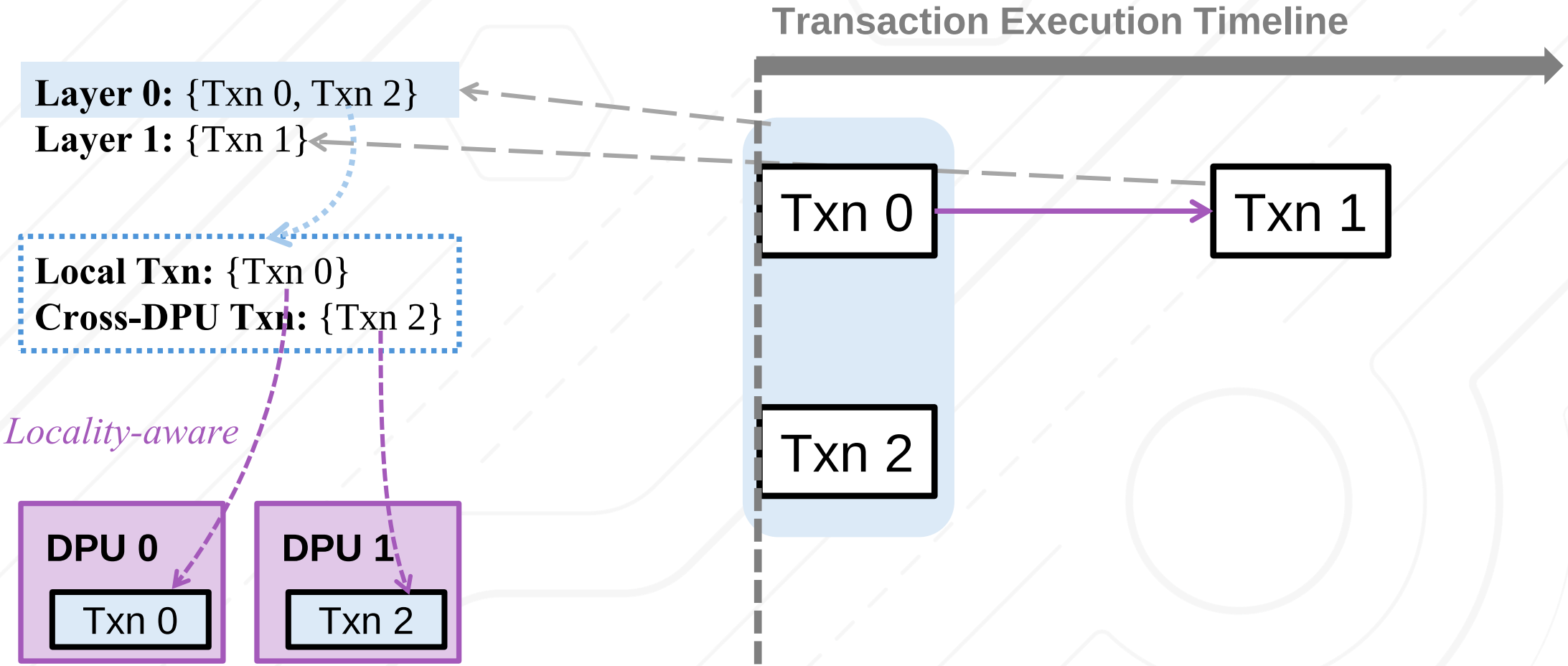


Multi-versioning Initialization

Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



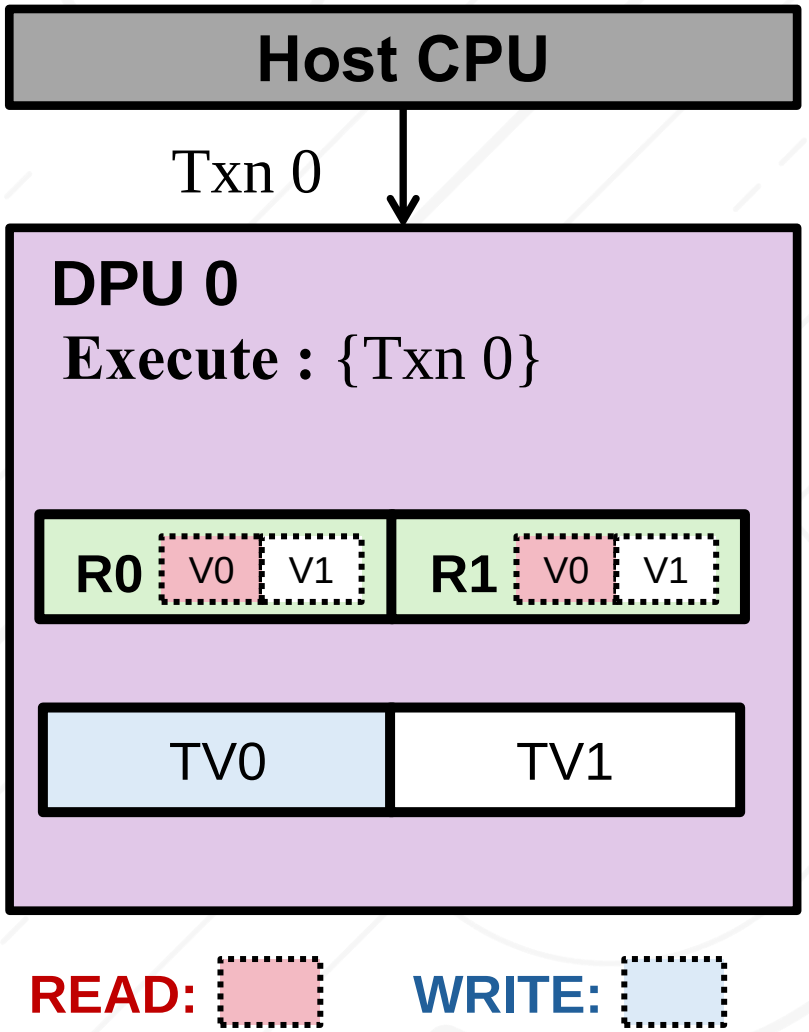
Locality-Aware Execution



Locality-Aware Execution

Local Txn Execution

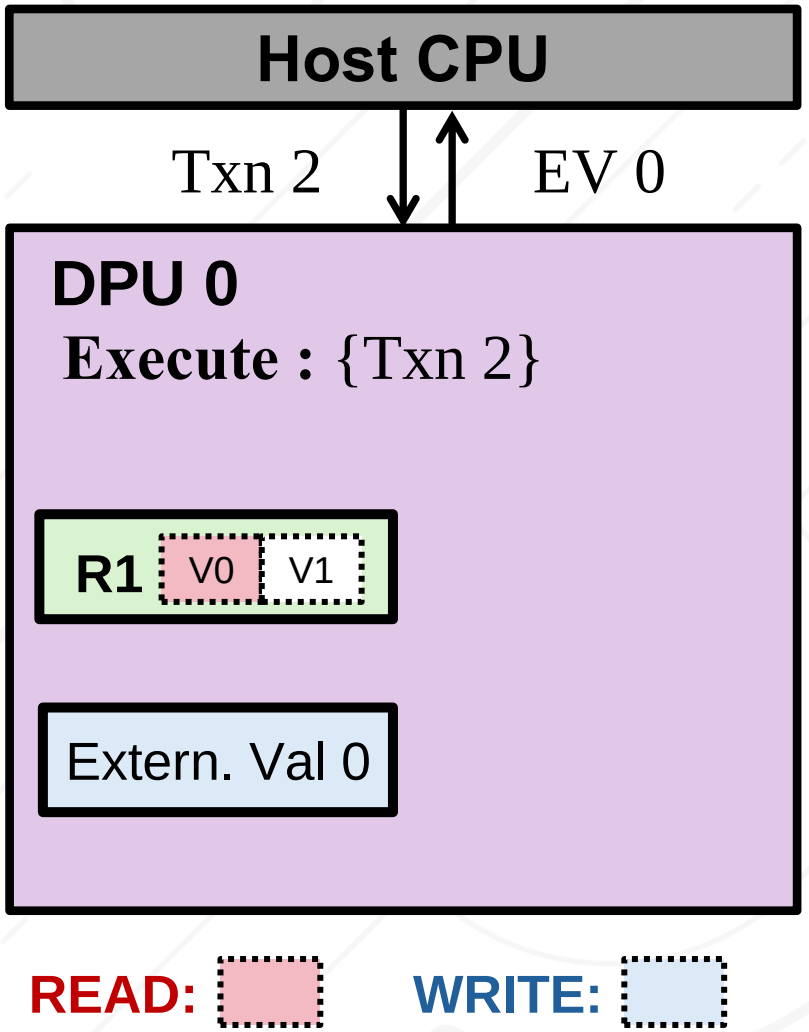
Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



Locality-Aware Execution

Cross-DPU Txn Read Phase

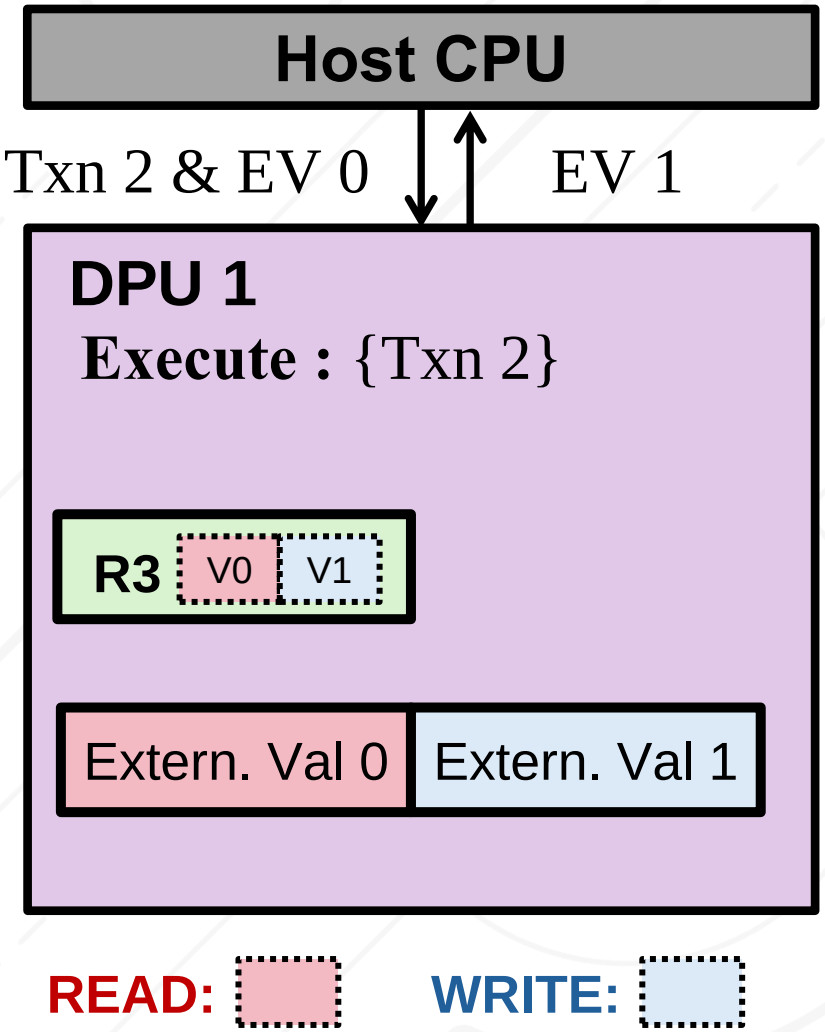
Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



Locality-Aware Execution

Cross-DPU Txn Compute Phase

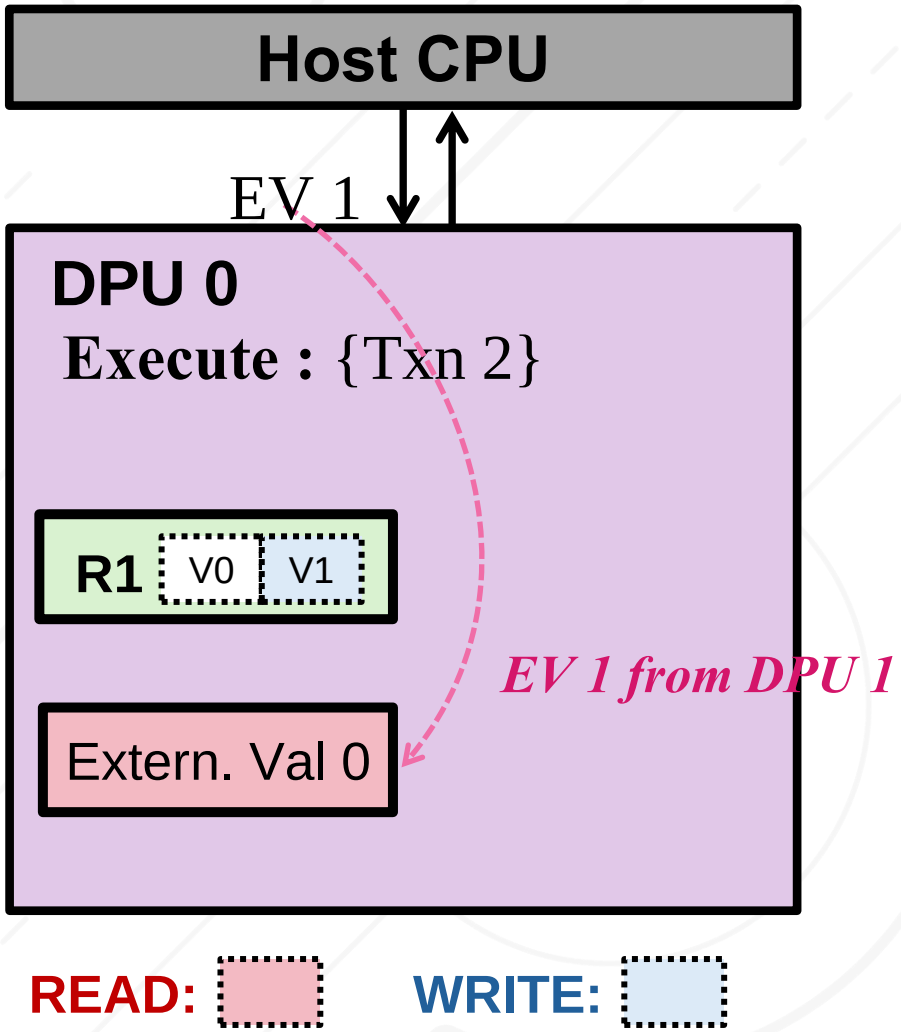
Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



Locality-Aware Execution

Cross-DPU Txn Write Phase

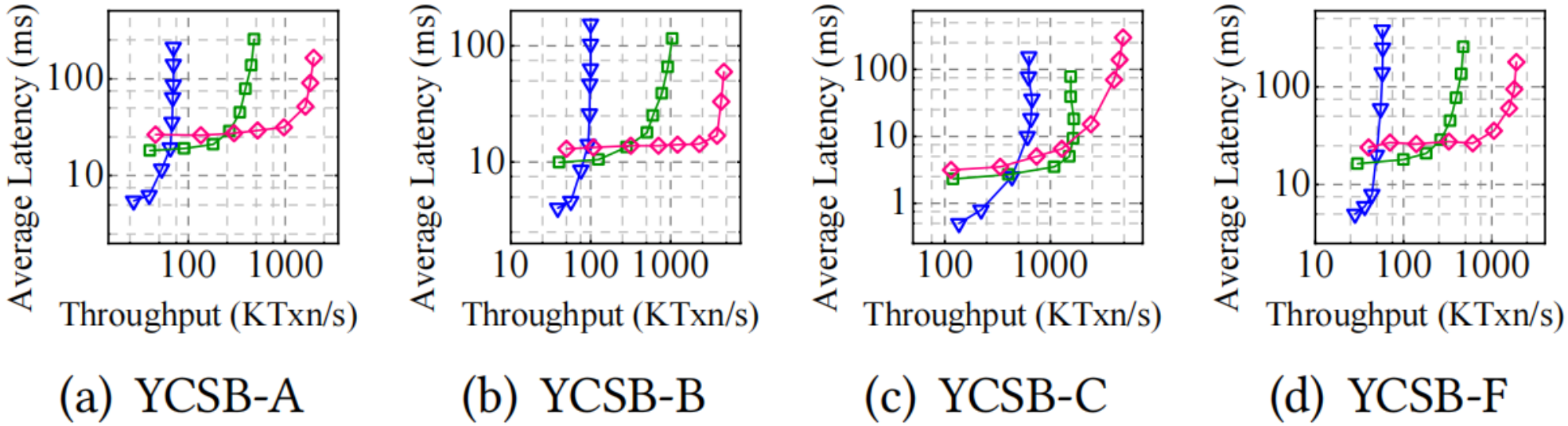
Txn ID	Read Locations	Write Locations
0	R0.V0 R1.V0	TV0
1	TV0 R2.V0 R3.V0	R0.V1 R2.V1
2	R1.V0 R3.V0	R1.V1 R3.V1



Evaluation

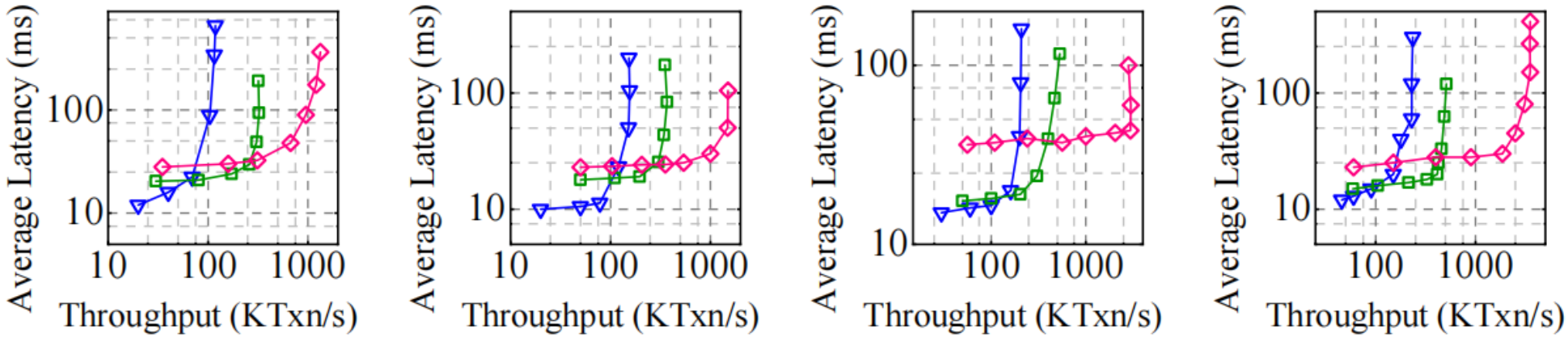
- **Platform**
  - A Real PIM Prototype
    - 8 UPMEM PIM DIMMs
    - 2 Intel Xeon Silver 4216 CPUs
    - 192 GB of DRAM memory
- **Comparisons**
  - Single-version system: Calvin[Sigmod' 12]
  - Multi-version system: Caracal[SOSP' 21]
- **Workloads**
  - YCSB benchmark: single database table containing 1 million records
  - TPC-C benchmark: complex OLTP environment with 9 database tables

End-to-End Performance



By fully leveraging PIM bandwidth & parallelism via orchestrated workflow, Onyx improves the throughput by 4.62x on average

End-to-End Performance



(e) TPC-C-W1 (f) TPC-C-W4 (g) TPC-C-W16 (h) TPC-C-W64

Onyx improves the throughput by 8.13x on average

Conclusion

- Existing in-memory OLTP systems suffer from severe data movements
  - *The **CPU-Memory bandwidth** fundamentally limits throughputs*
- **Onyx:** an efficient transaction processing system
  - *Leveraging a real-world PIM prototype*
  - *Multi-versioning Initialization*
  - *Locality-aware Execution*

Thanks! Q&A