Hardware Support for ACID Transactions in Persistent Memory Systems

Arpit Joshi

ARM Research Summit, 2018

THE UNIVERSITY *of* EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

intel

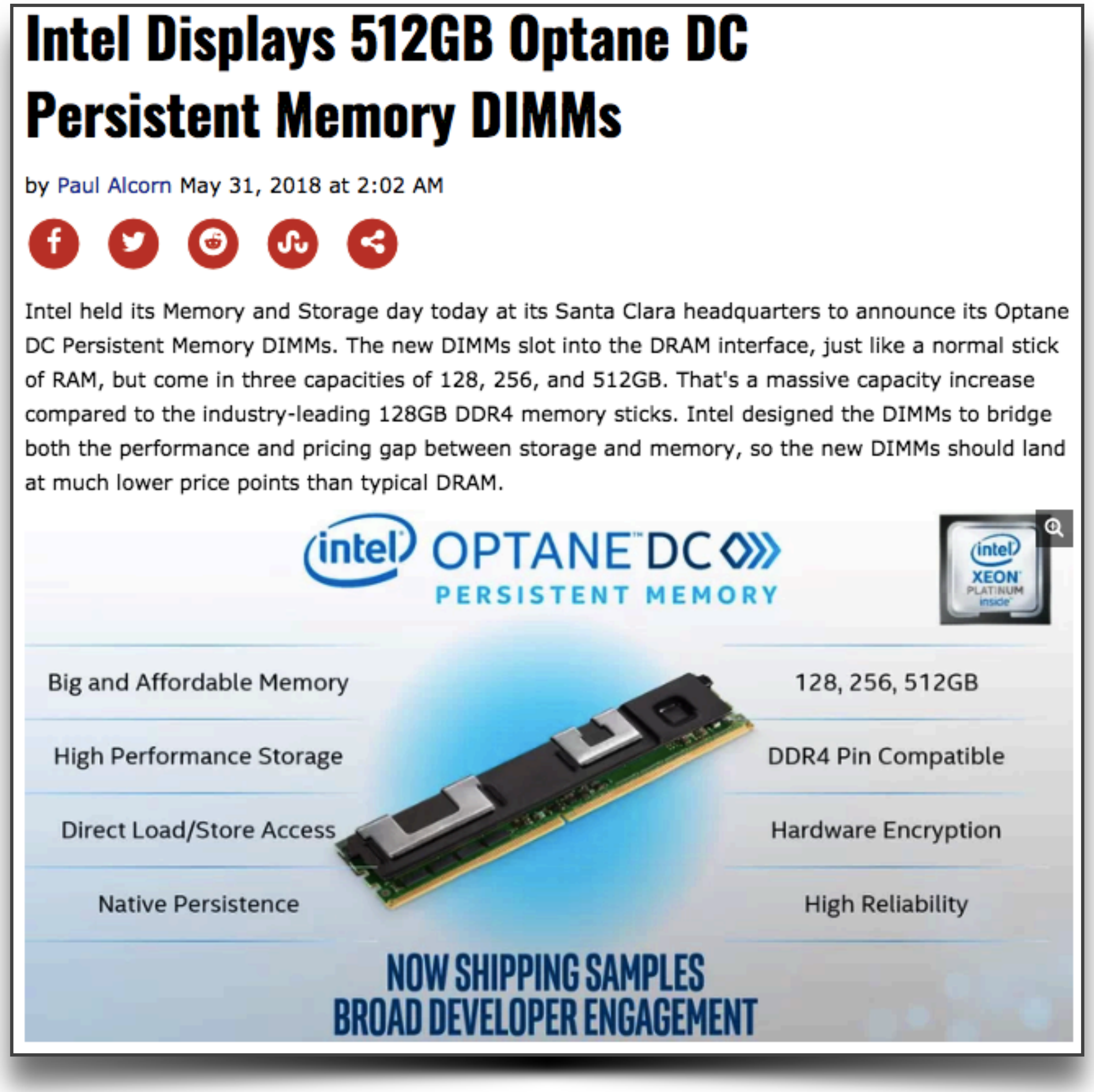
Google

Collaborators

- Vijay Nagarajan
- Marcelo Cintra
- Stratis Viglas

Persistent Memory is here…

Persistent Memory is here…



Intel Displays 512GB Optane DC Persistent Memory DIMMs

by Paul Alcorn May 31, 2018 at 2:02 AM

Intel held its Memory and Storage day today at its Santa Clara headquarters to announce its Optane DC Persistent Memory DIMMs. The new DIMMs slot into the DRAM interface, just like a normal stick of RAM, but come in three capacities of 128, 256, and 512GB. That's a massive capacity increase compared to the industry-leading 128GB DDR4 memory sticks. Intel designed the DIMMs to bridge both the performance and pricing gap between storage and memory, so the new DIMMs should land at much lower price points than typical DRAM.

intel OPTANE DC PERSISTENT MEMORY

intel XEON PLATINUM inside

Big and Affordable Memory	128, 256, 512GB
High Performance Storage	DDR4 Pin Compatible
Direct Load/Store Access	Hardware Encryption
Native Persistence	High Reliability

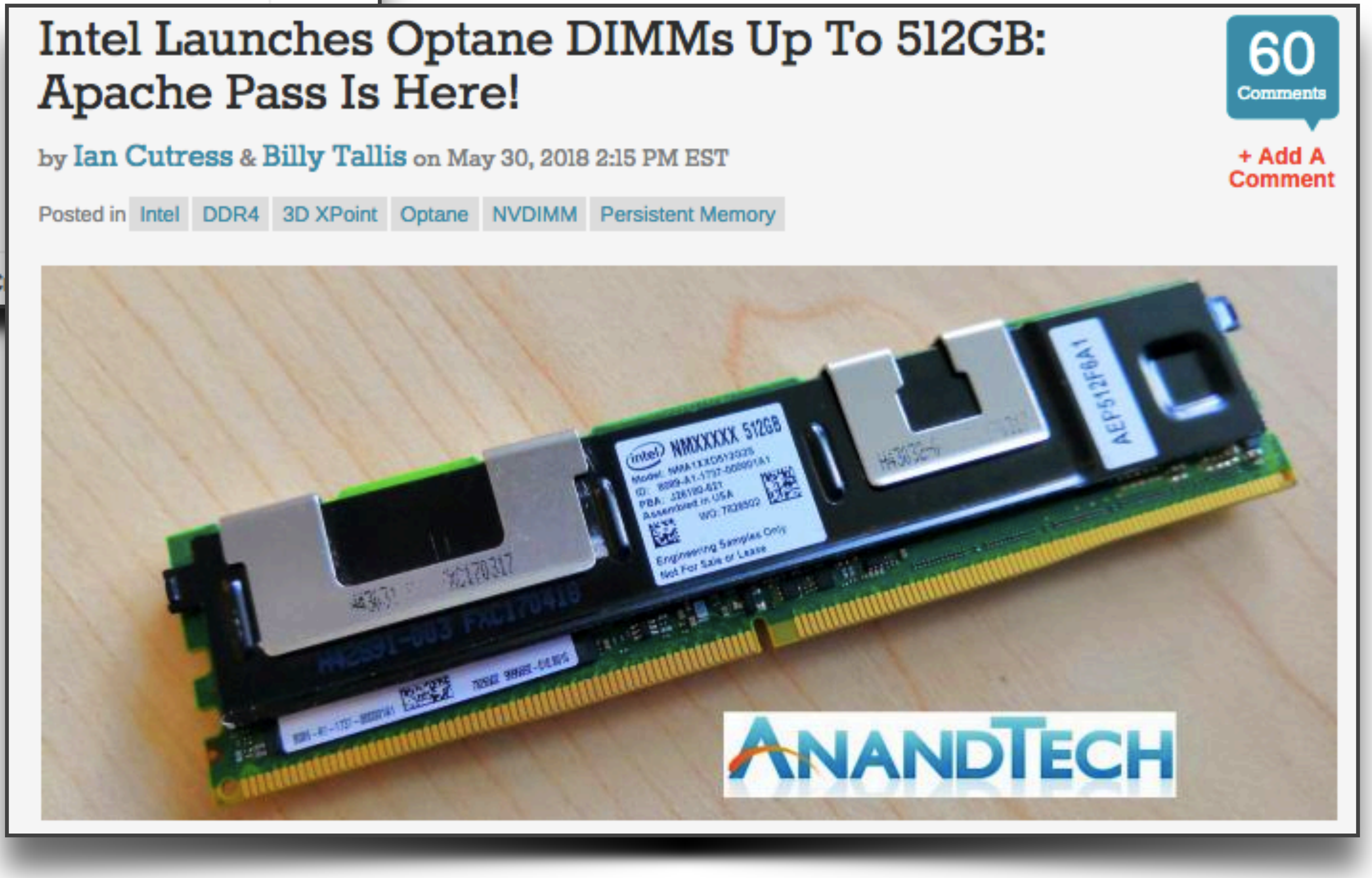
NOW SHIPPING SAMPLES
BROAD DEVELOPER ENGAGEMENT

Intel teases Optane DIMMS, but you may need a new Xeon first

128GB, 256GB and 512GB modules offered as new storage tier below RAM, above SSD

By Simon Sharwood, APAC Editor 31 May 2018 at 03:50 5 SHARE ▼

Intel's new Optane DC persistent memory DIMM. (C



Intel Launches Optane DIMMs Up To 512GB: Apache Pass Is Here!

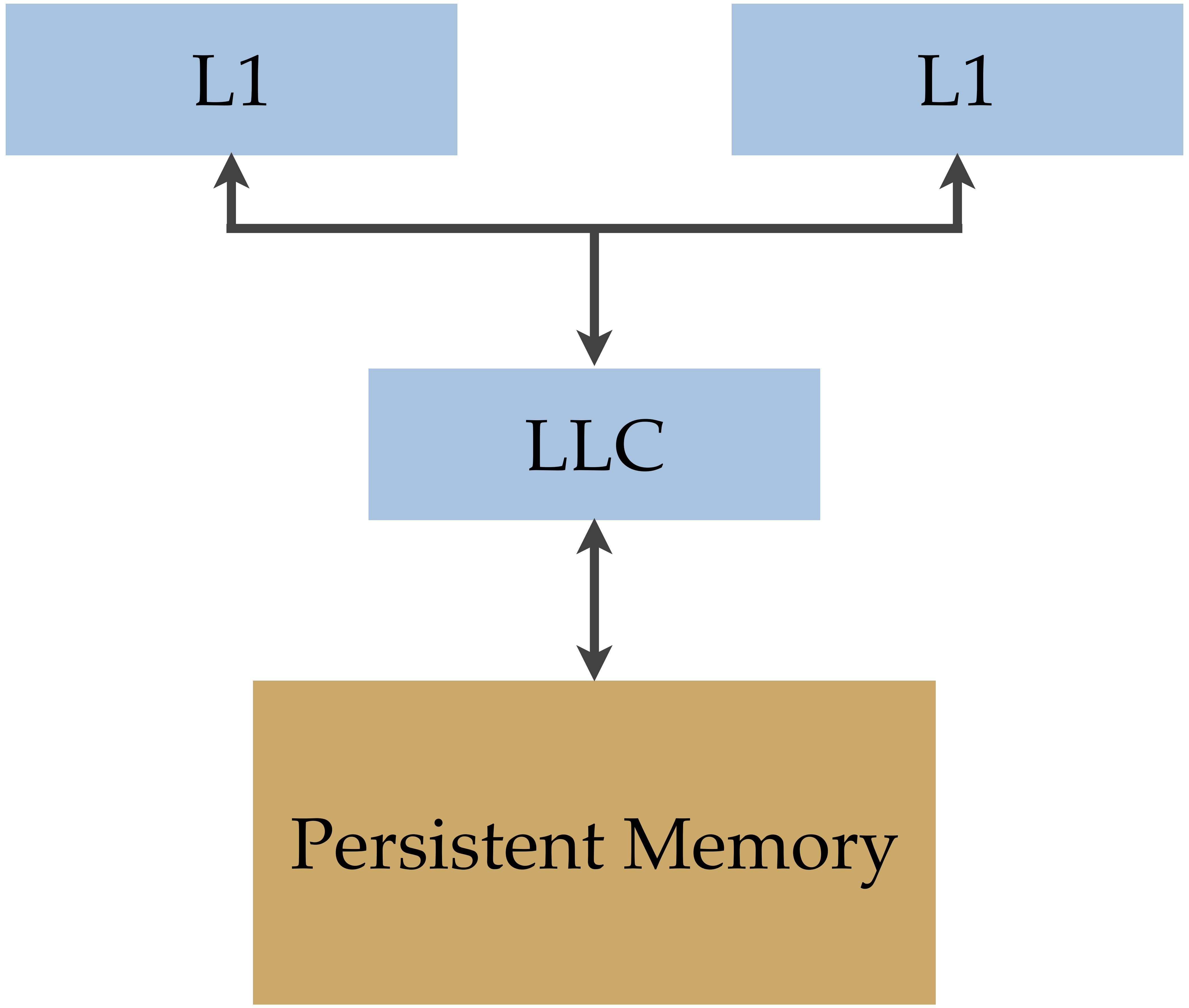
60 Comments

+ Add A Comment

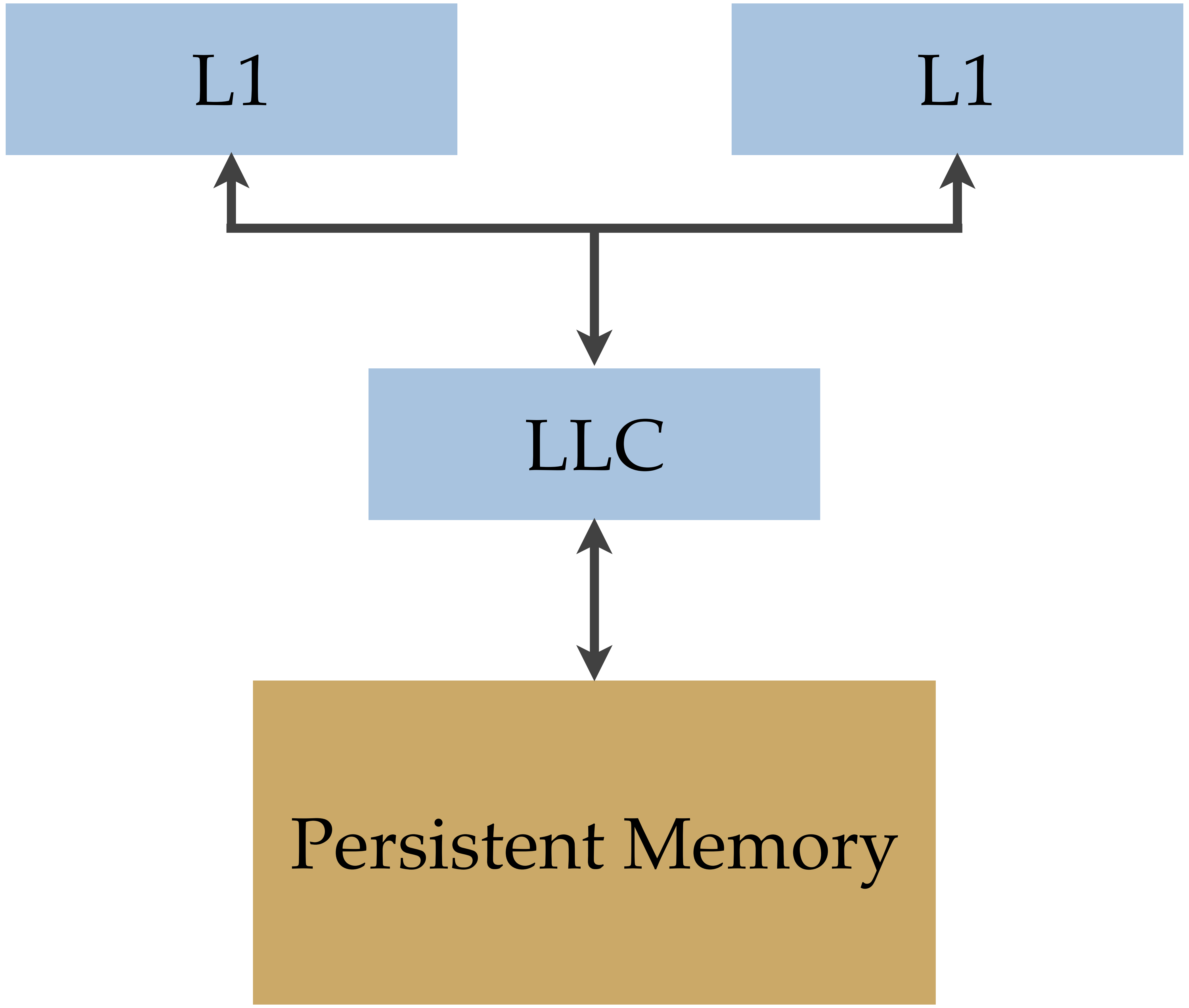
by Ian Cutress & Billy Tallis on May 30, 2018 2:15 PM EST

Posted in Intel DDR4 3D XPoint Optane NVDIMM Persistent Memory

Persistent Memory Systems

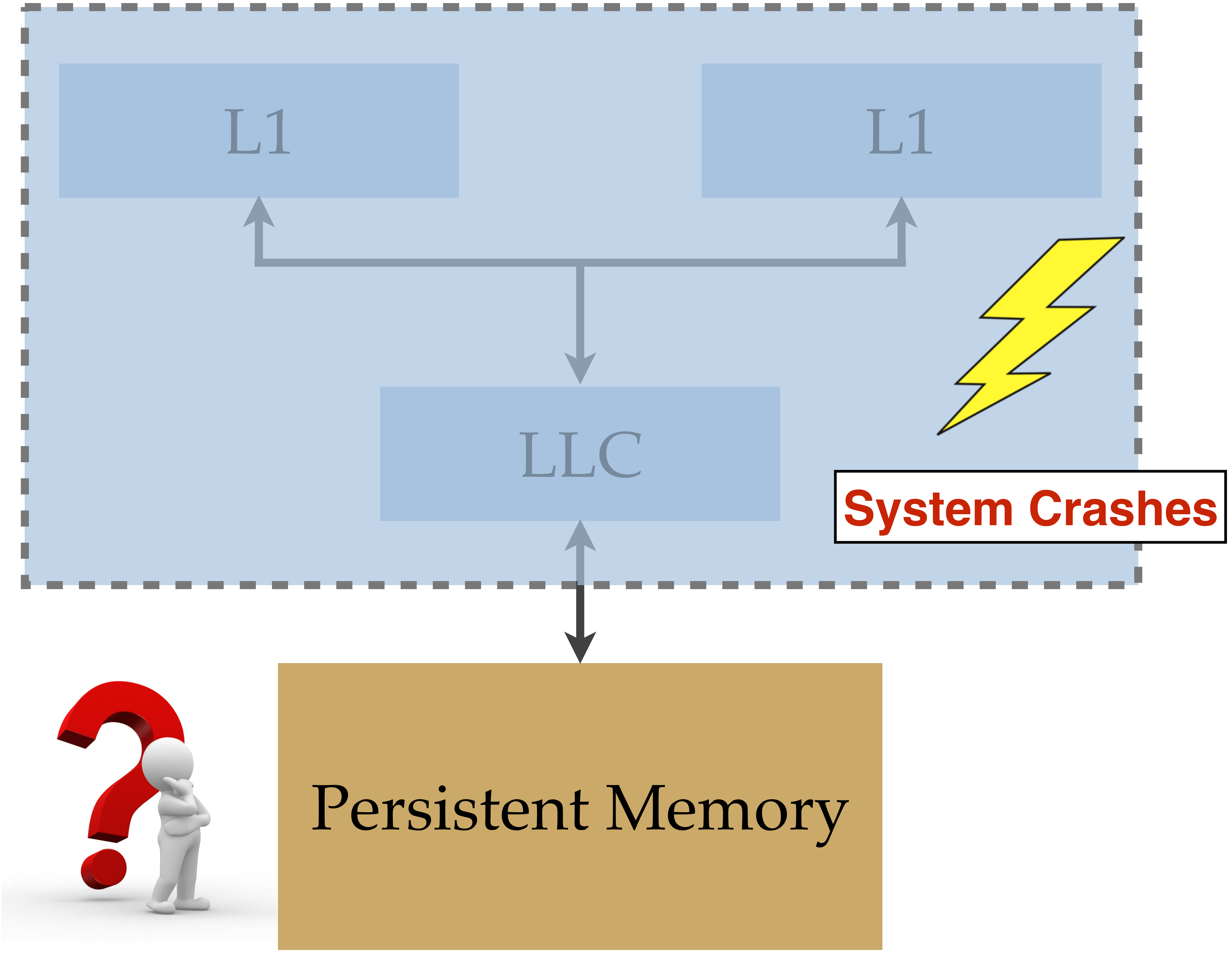


Persistent Memory Systems



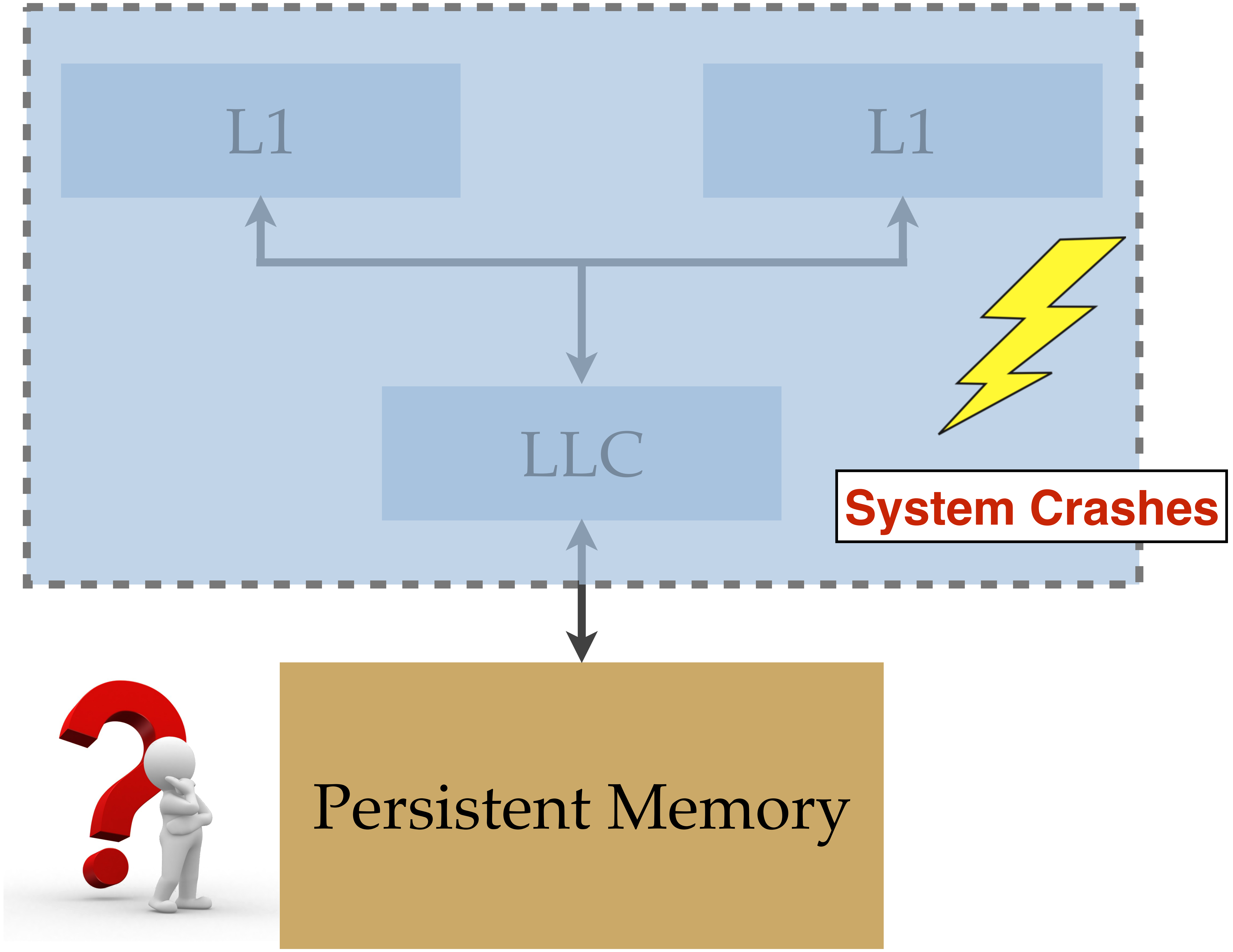
- **Persistent Memory**
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

Persistent Memory Systems



- **Persistent Memory**
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

Persistent Memory Systems



L1

L1

LLC

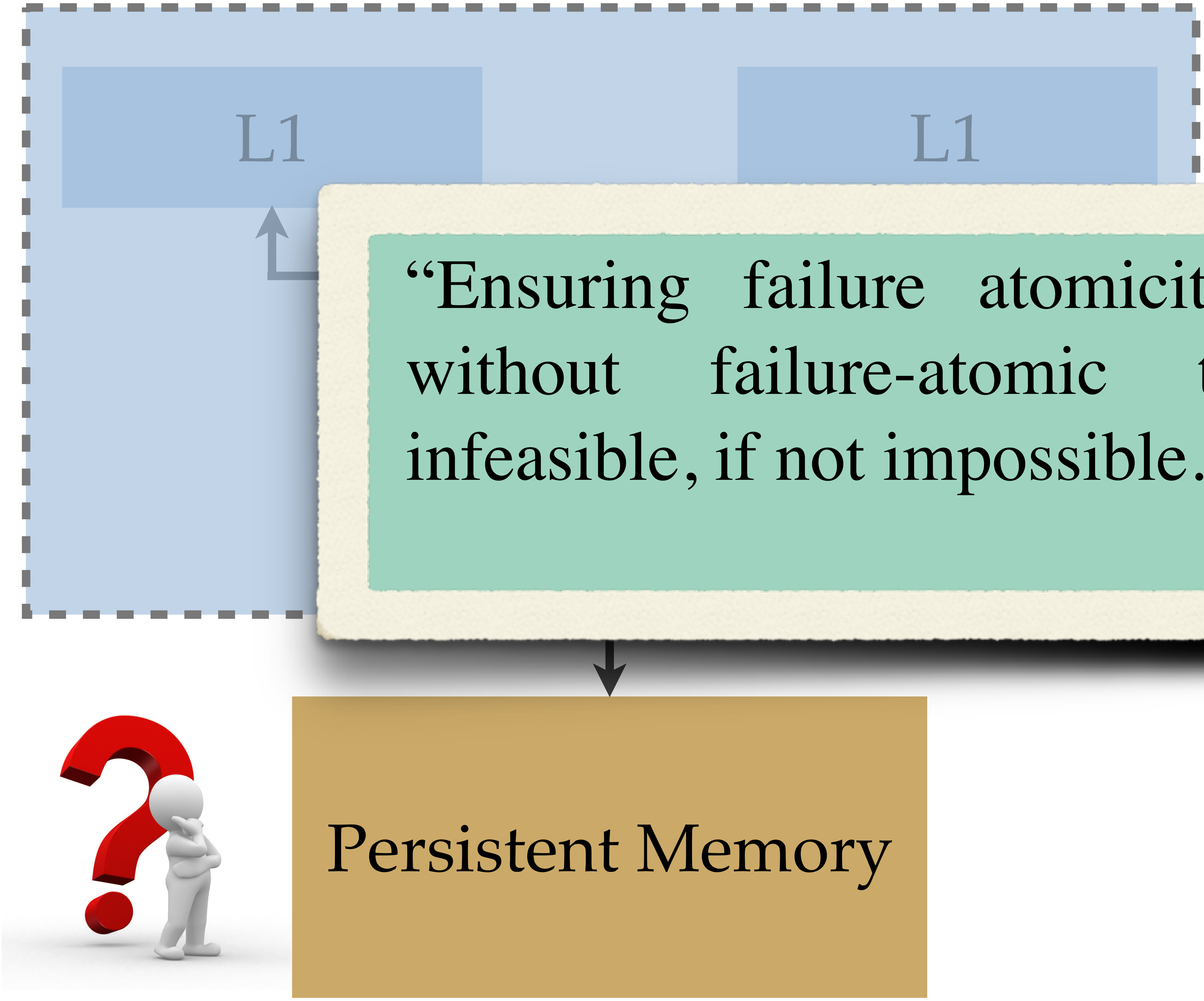
System Crashes

Persistent Memory

- **Persistent Memory**
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

- **Crash Consistency**
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions

Persistent Memory Systems



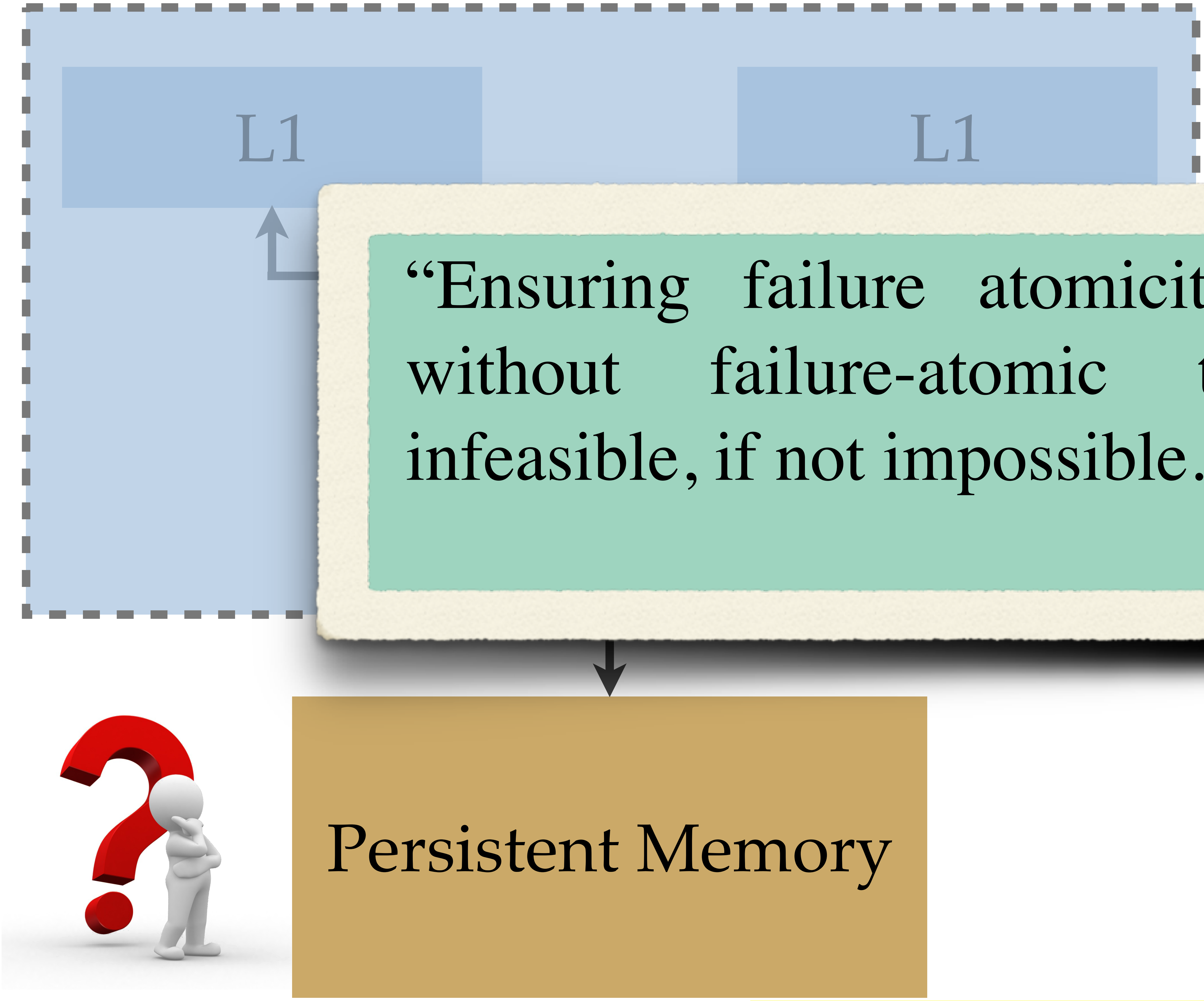
> “Ensuring failure atomicity for all this computation without failure-atomic transactions is practically infeasible, if not impossible.”
>
> Marathe et al. [HotStorage’17]

mory bus

rsistent data

- Crash Consistency
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions

Persistent Memory Systems



> “Ensuring failure atomicity for all this computation without failure-atomic transactions is practically infeasible, if not impossible.”
>
> Marathe et al. [HotStorage’17]

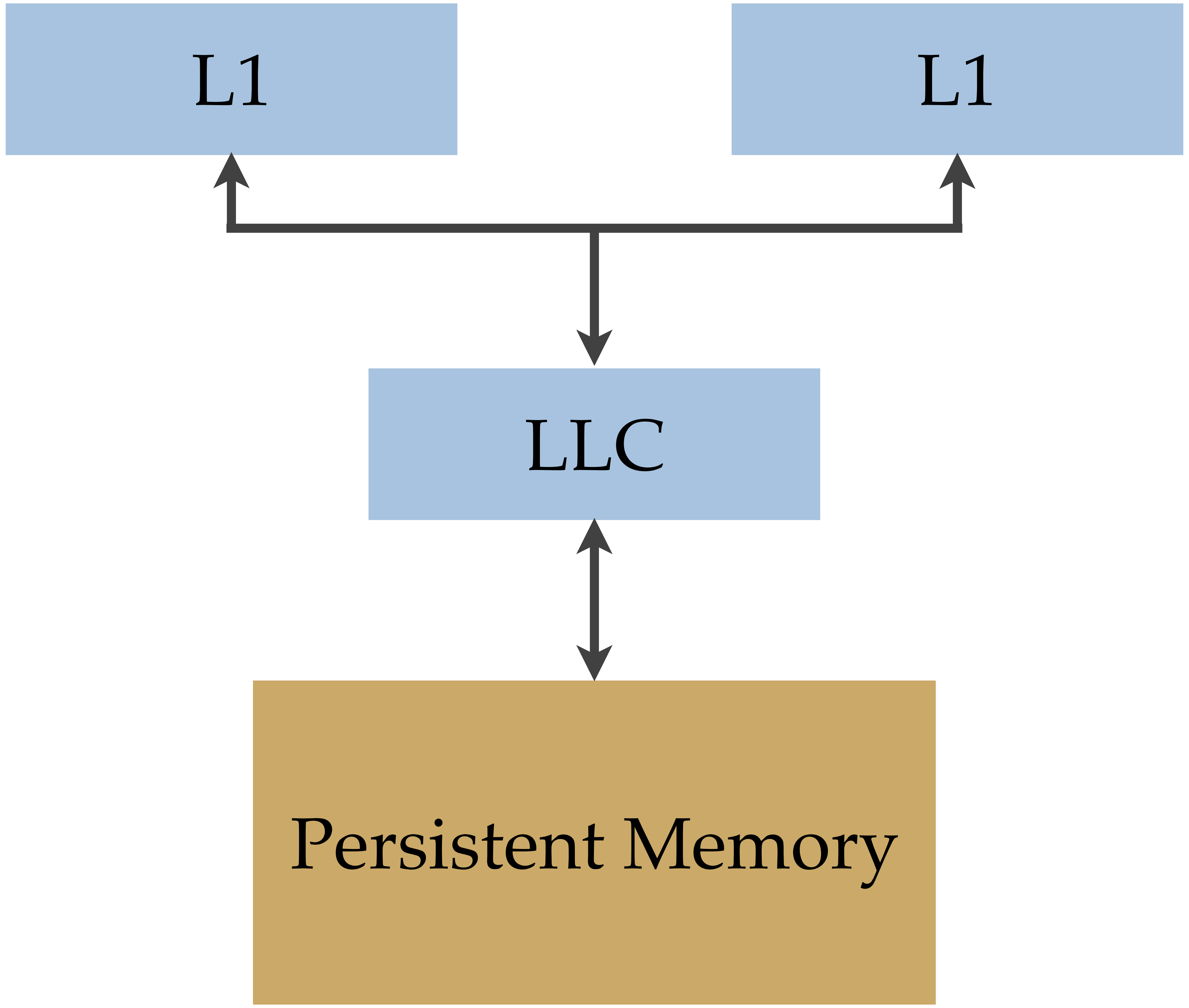
mory bus

rsistent data

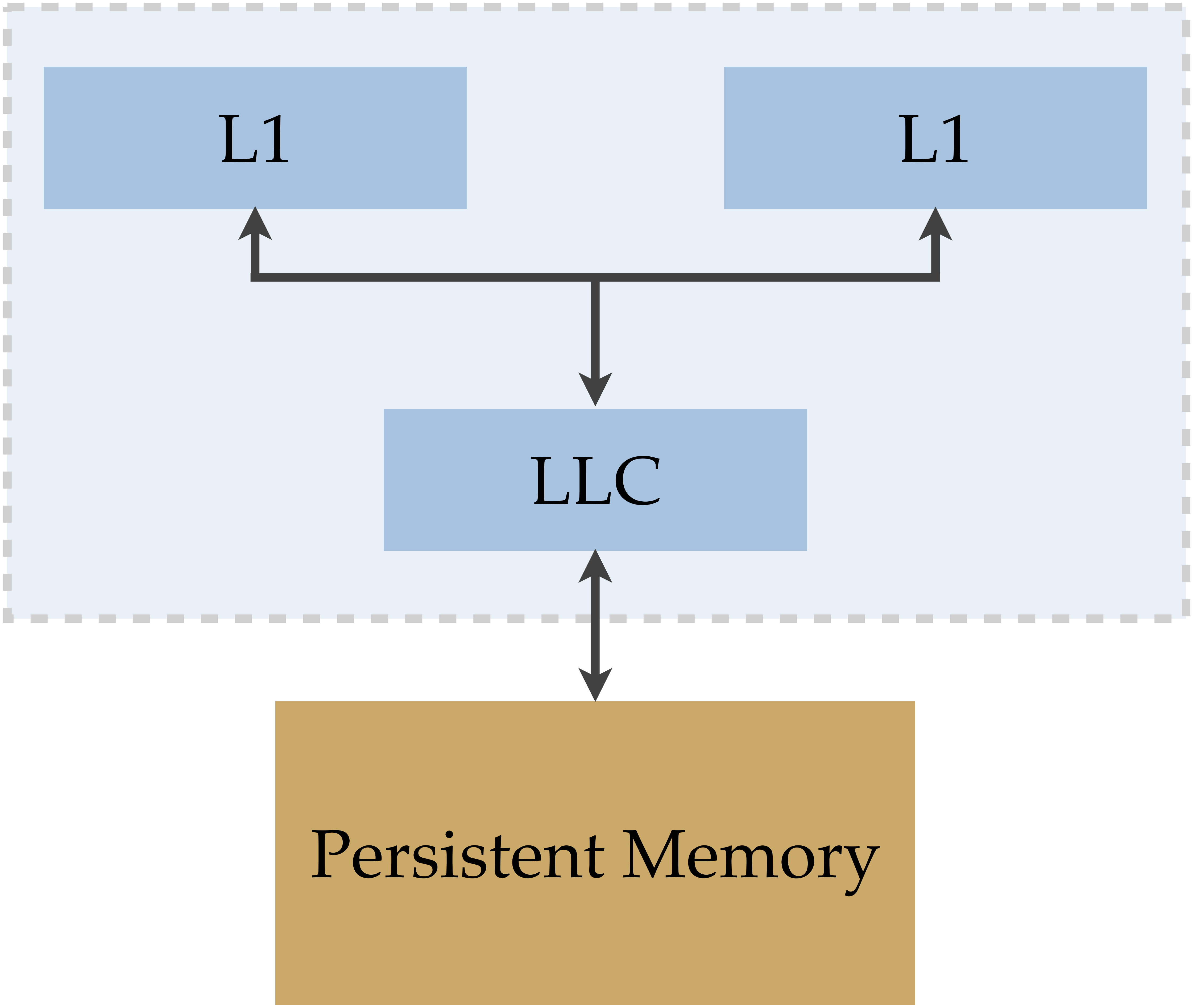
- Crash Consistency
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions

How fast can we support ACID?

ACID Transactions

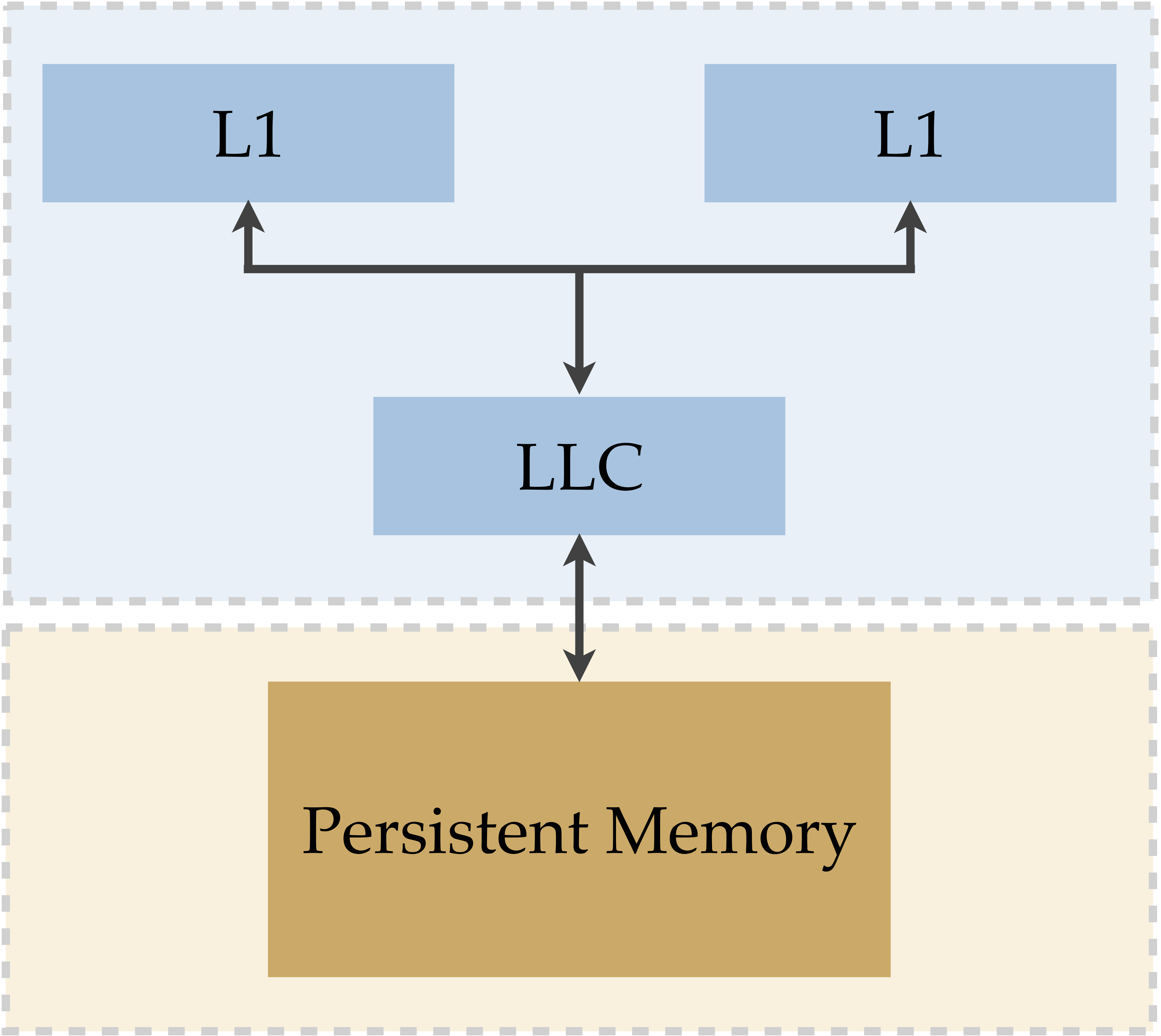


ACID Transactions



Atomic Visibility

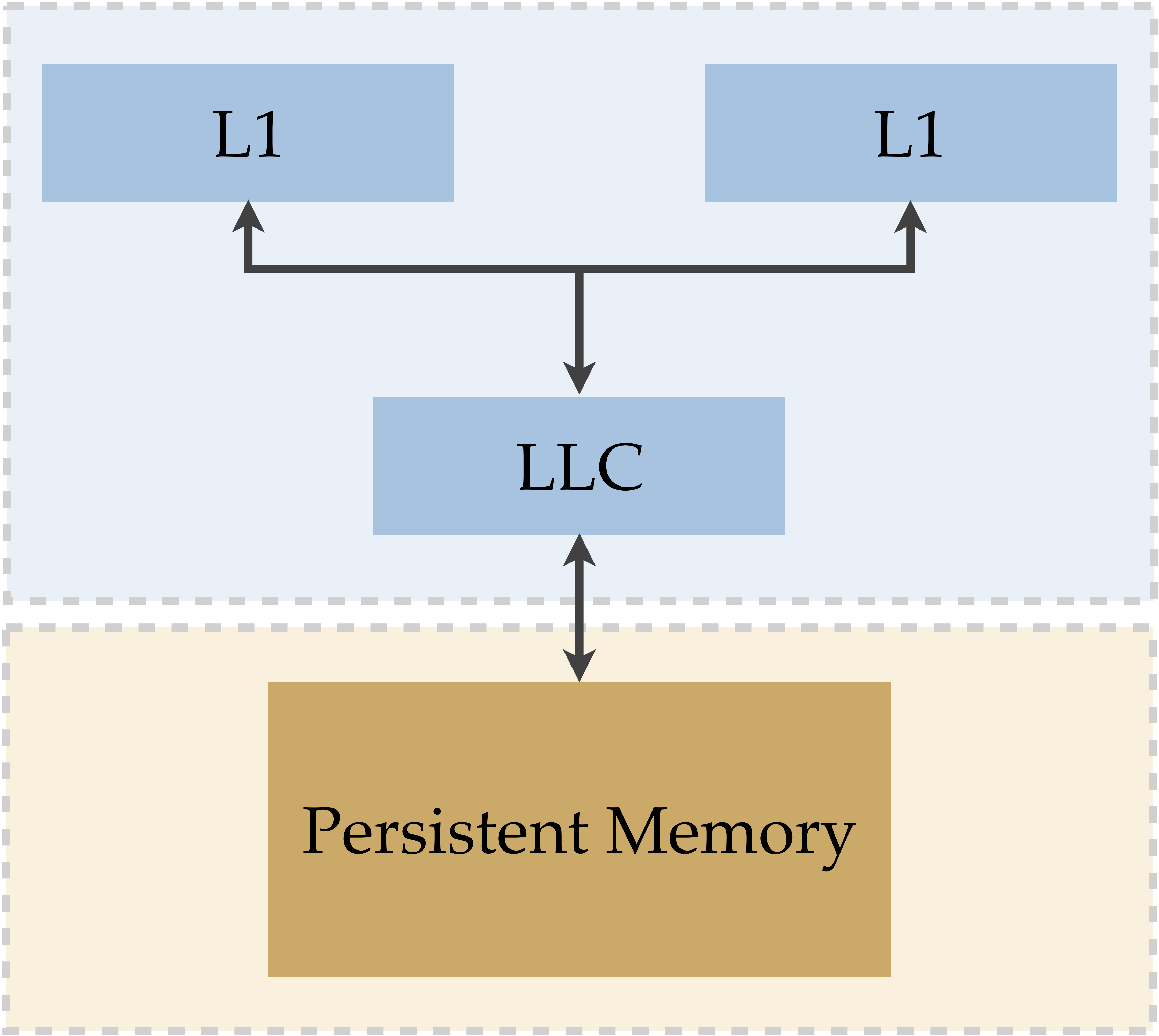
ACID Transactions

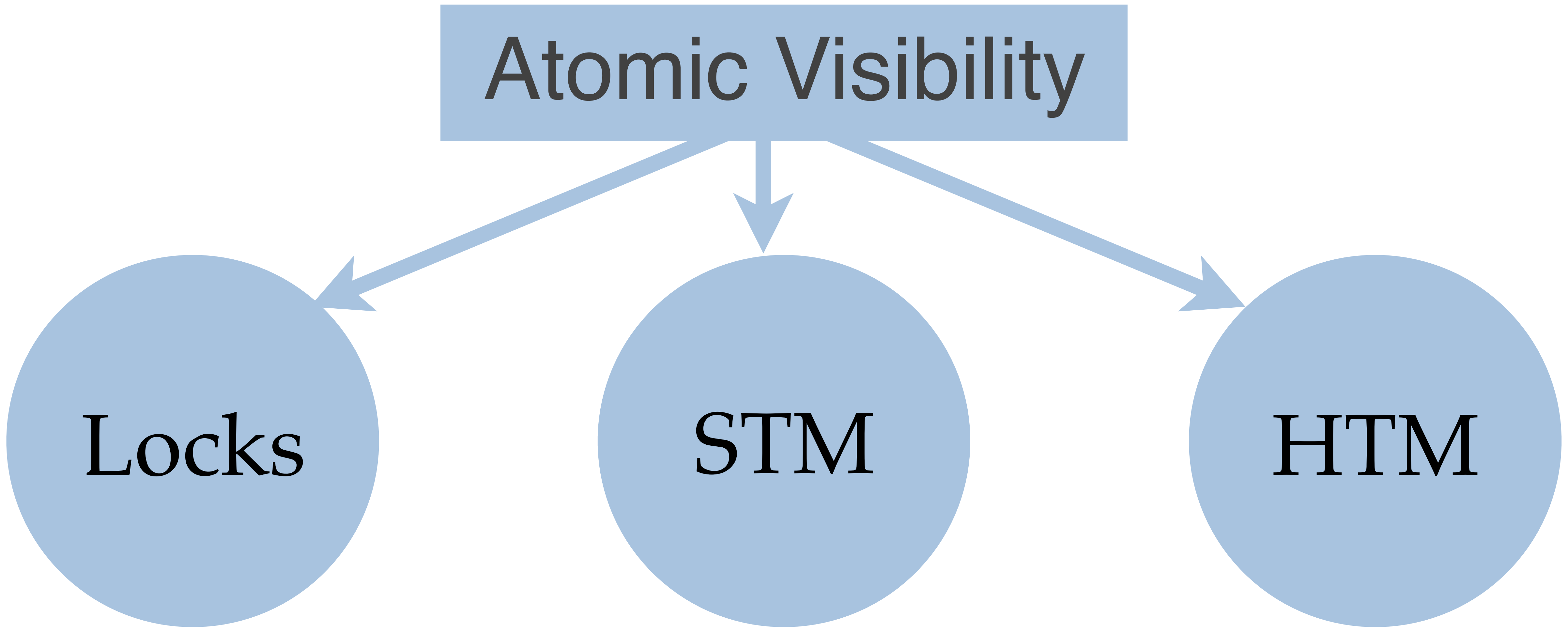


Atomic Visibility

Atomic Durability

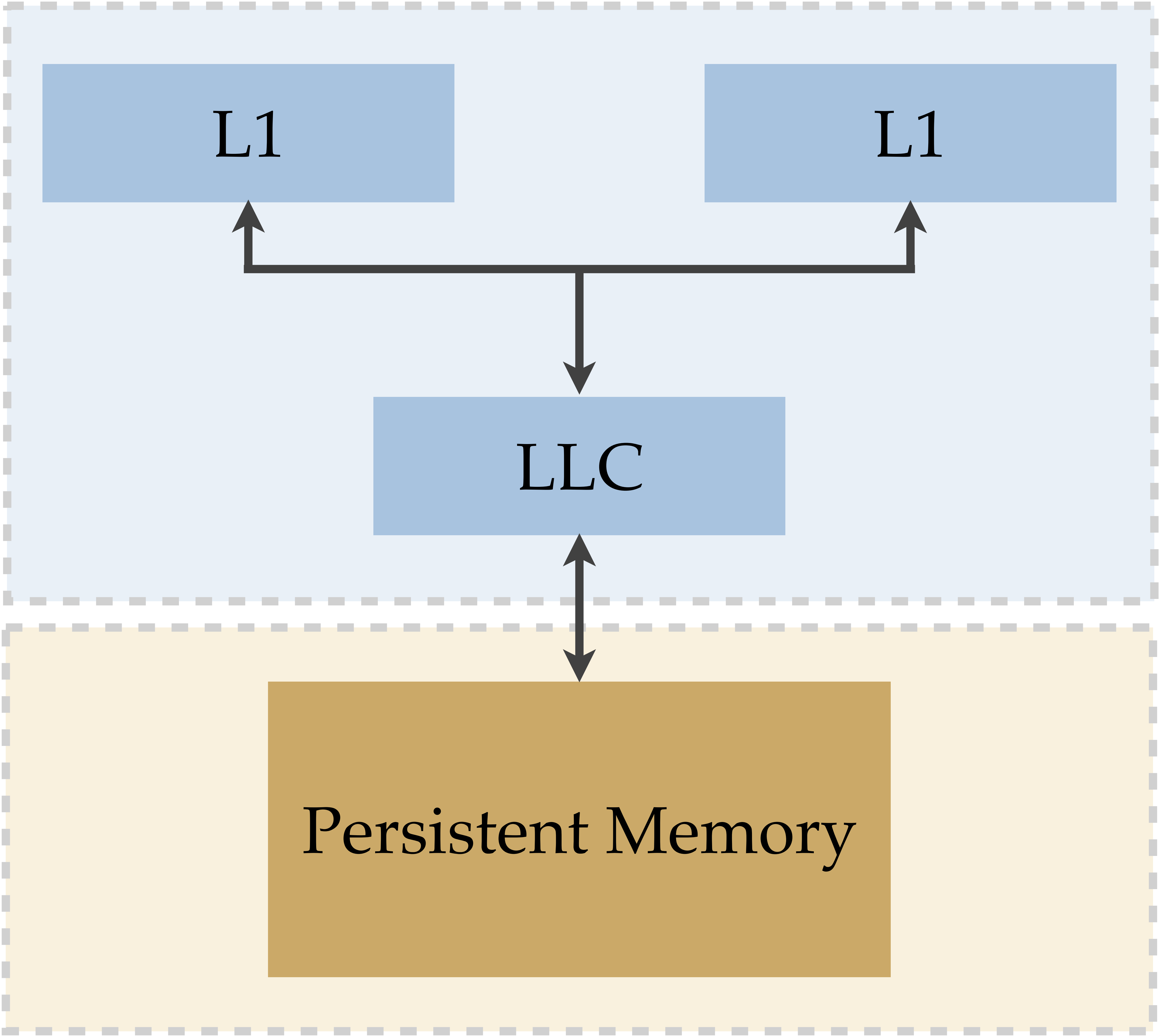
ACID Transactions





Atomic Durability

ACID Transactions

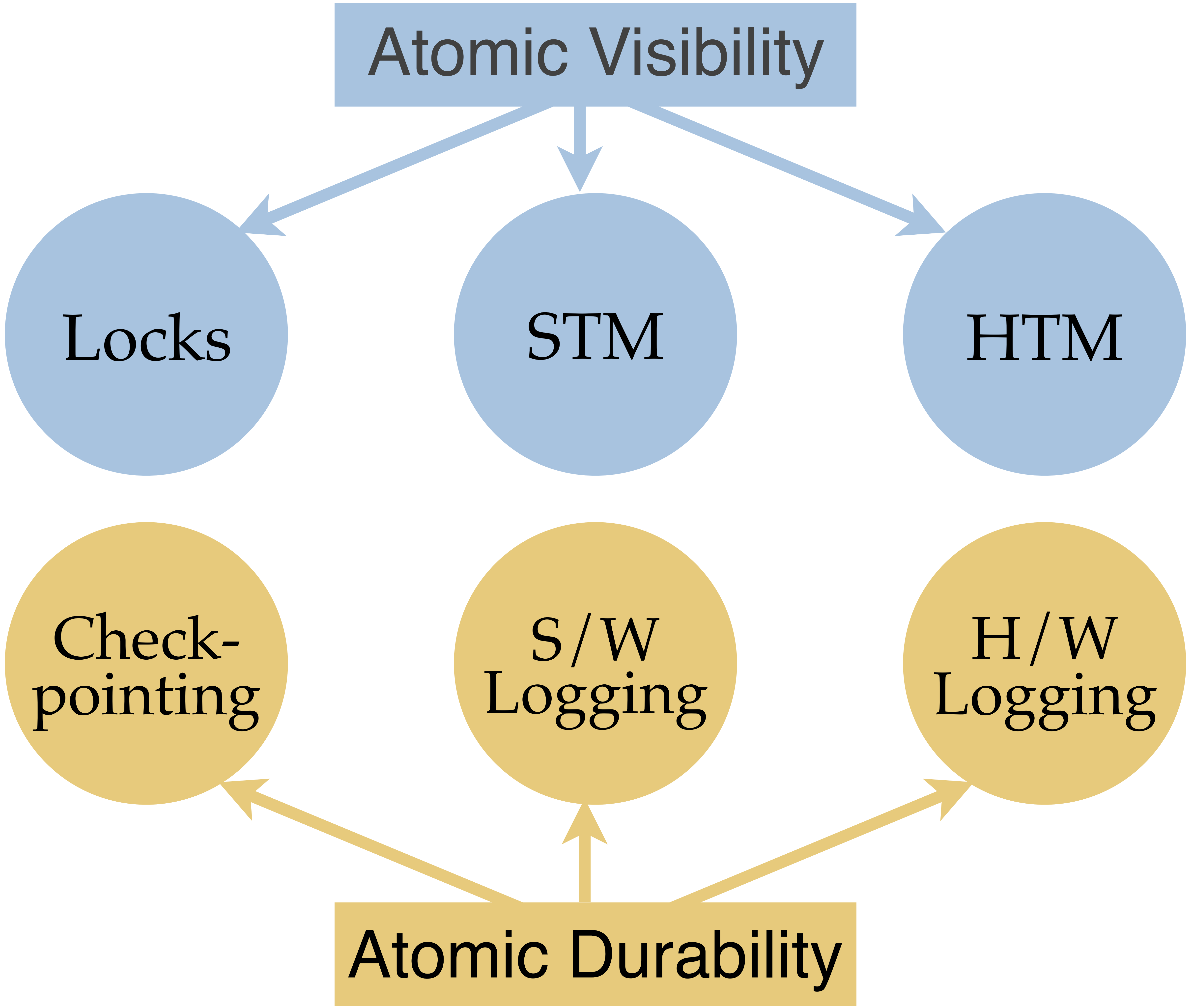


L1

L1

LLC

Persistent Memory



Atomic Visibility

Locks

STM

HTM

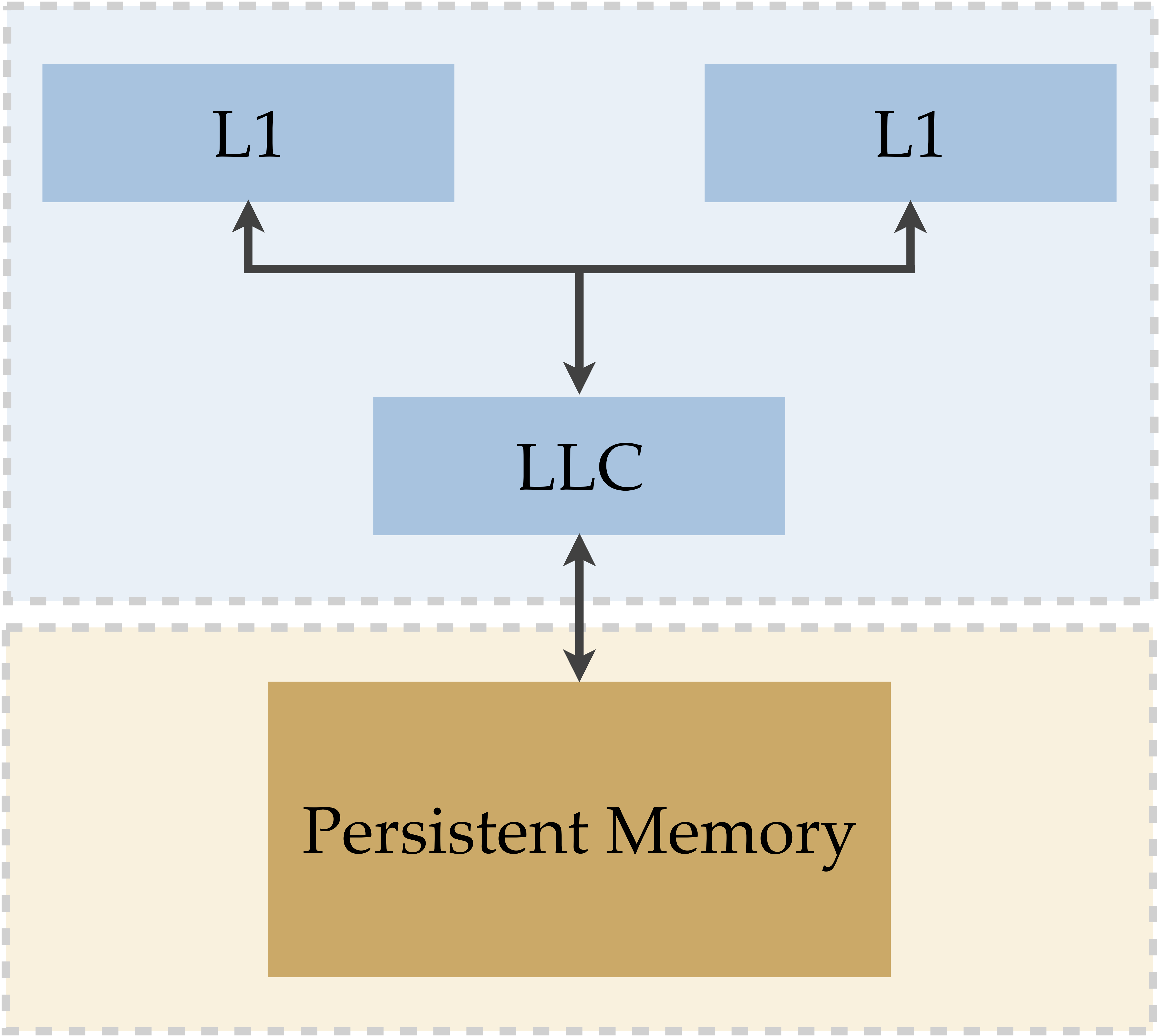
Check-pointing

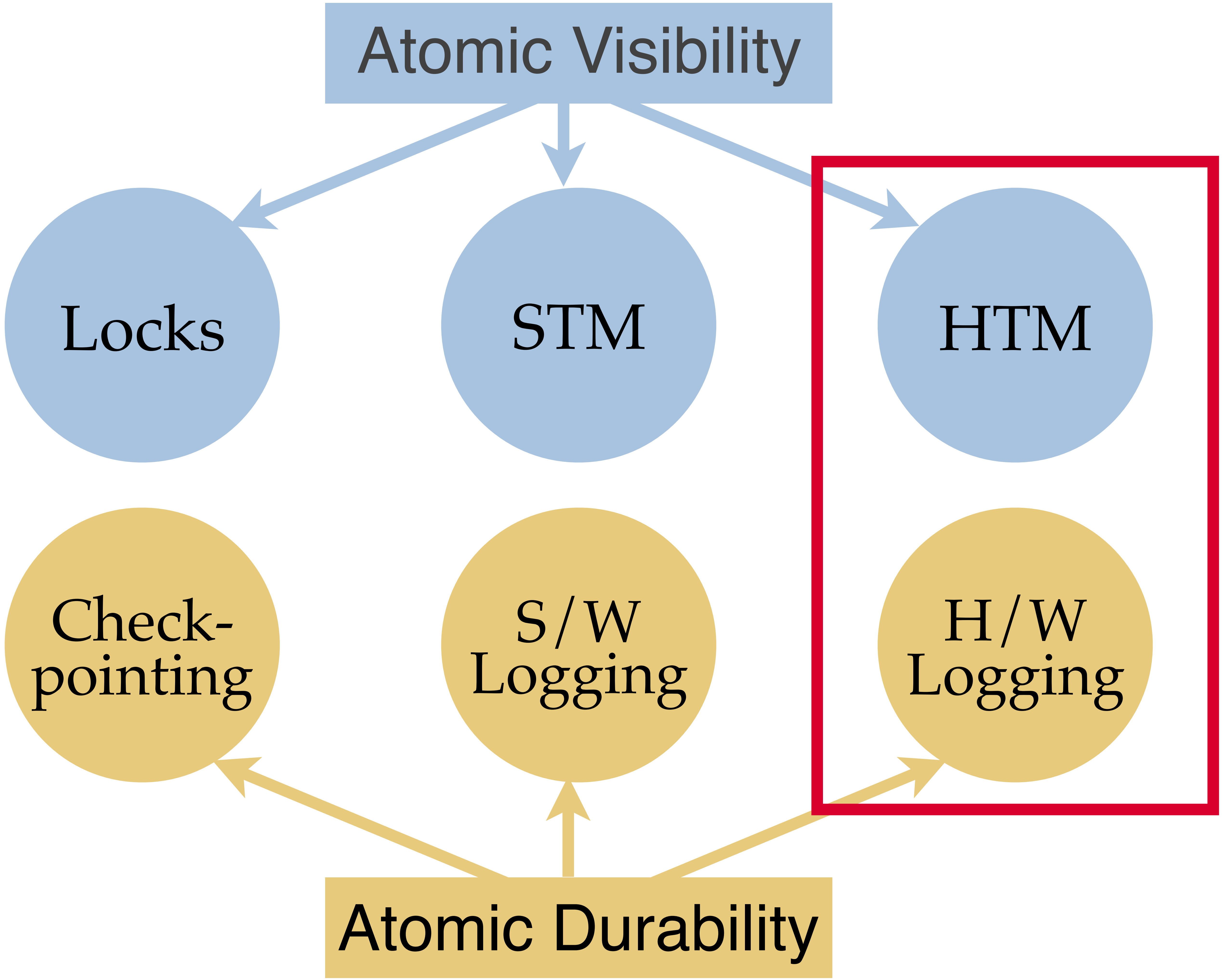
S/W Logging

H/W Logging

Atomic Durability

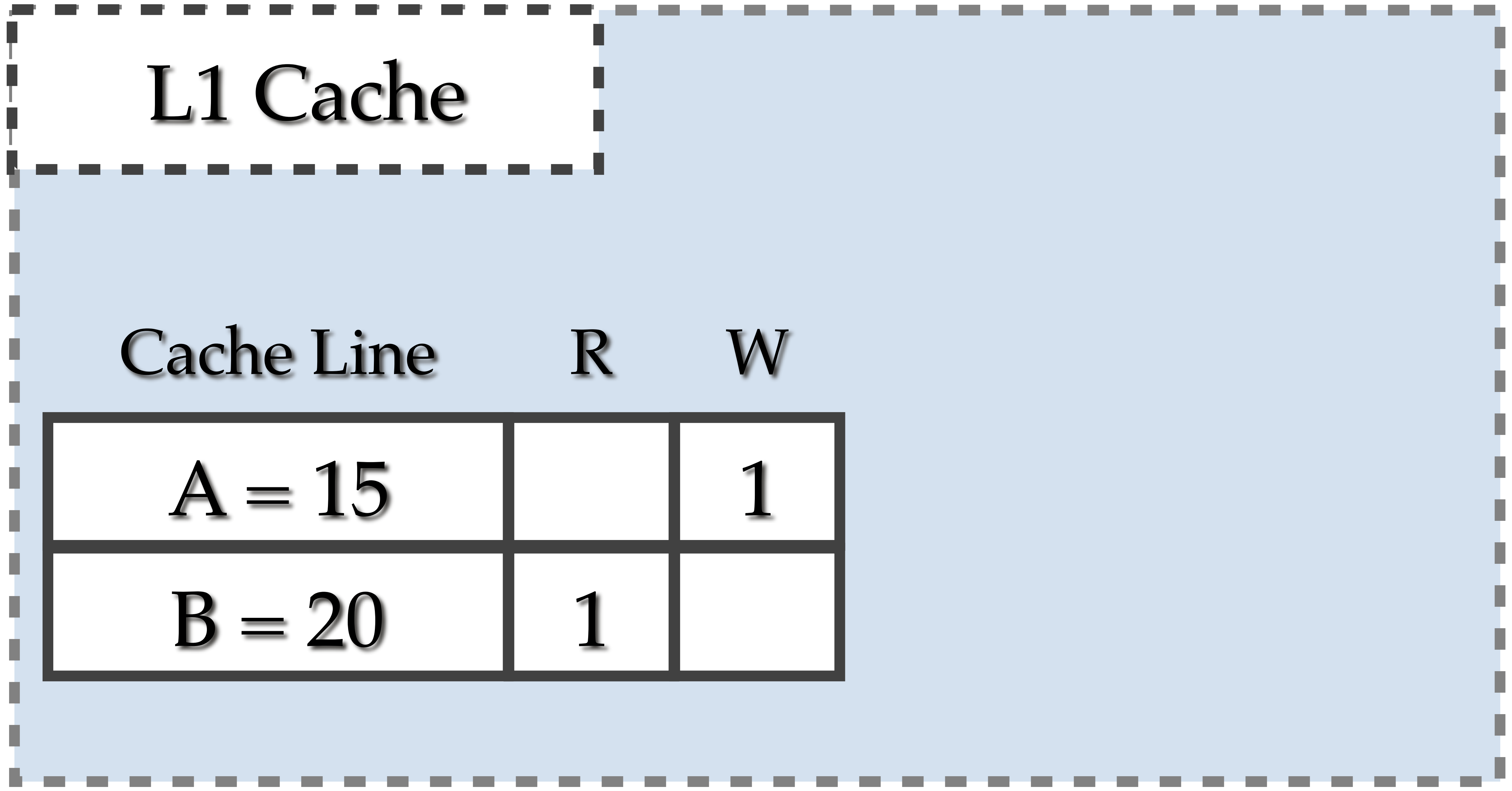
ACID Transactions





Atomic Visibility: HTM

Atomic Visibility: HTM

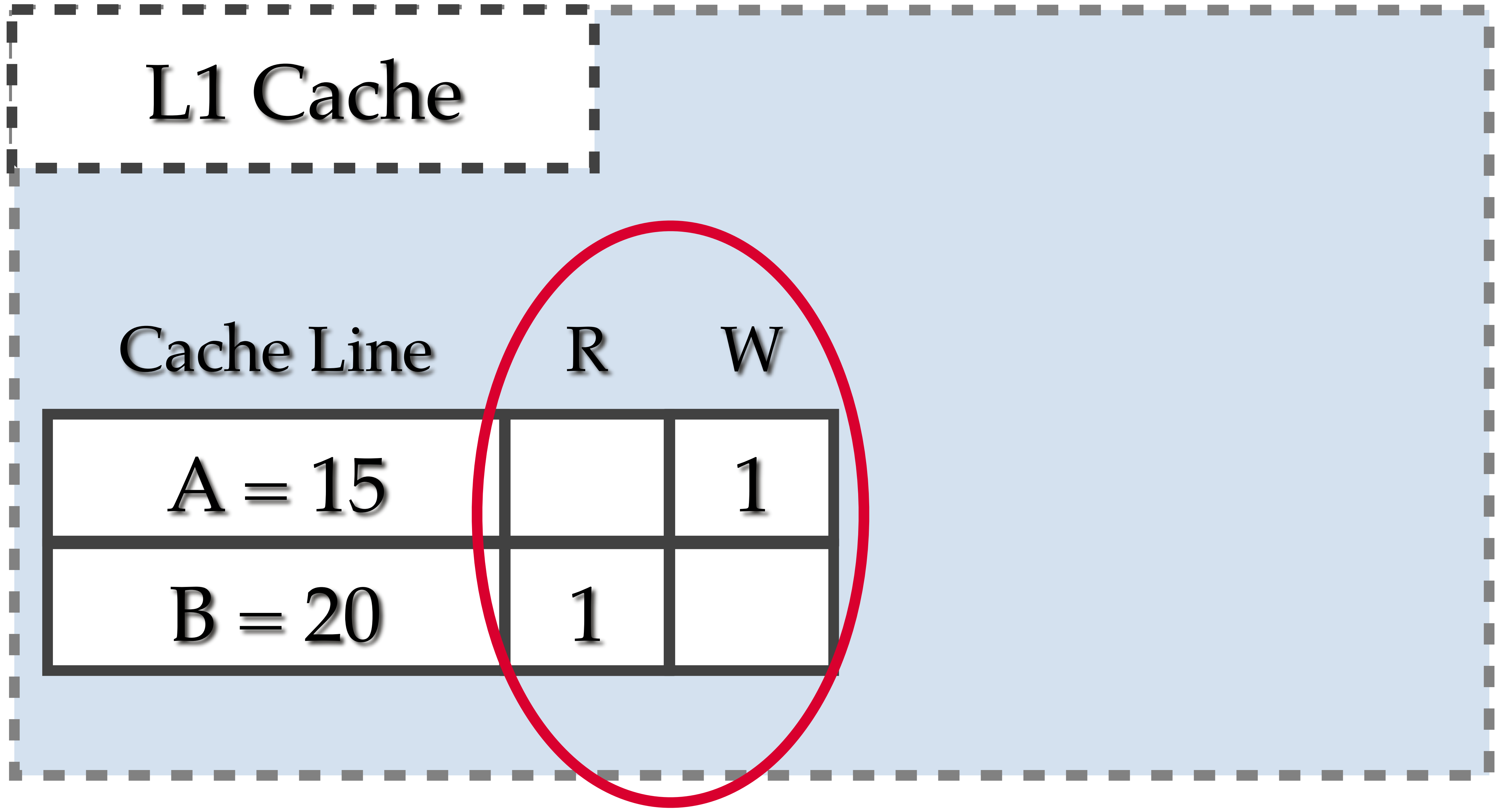


L1 Cache

Cache Line	R	W
A = 15		1
B = 20	1	

- **Commercial HTMs** [Intel, IBM]

Atomic Visibility: HTM

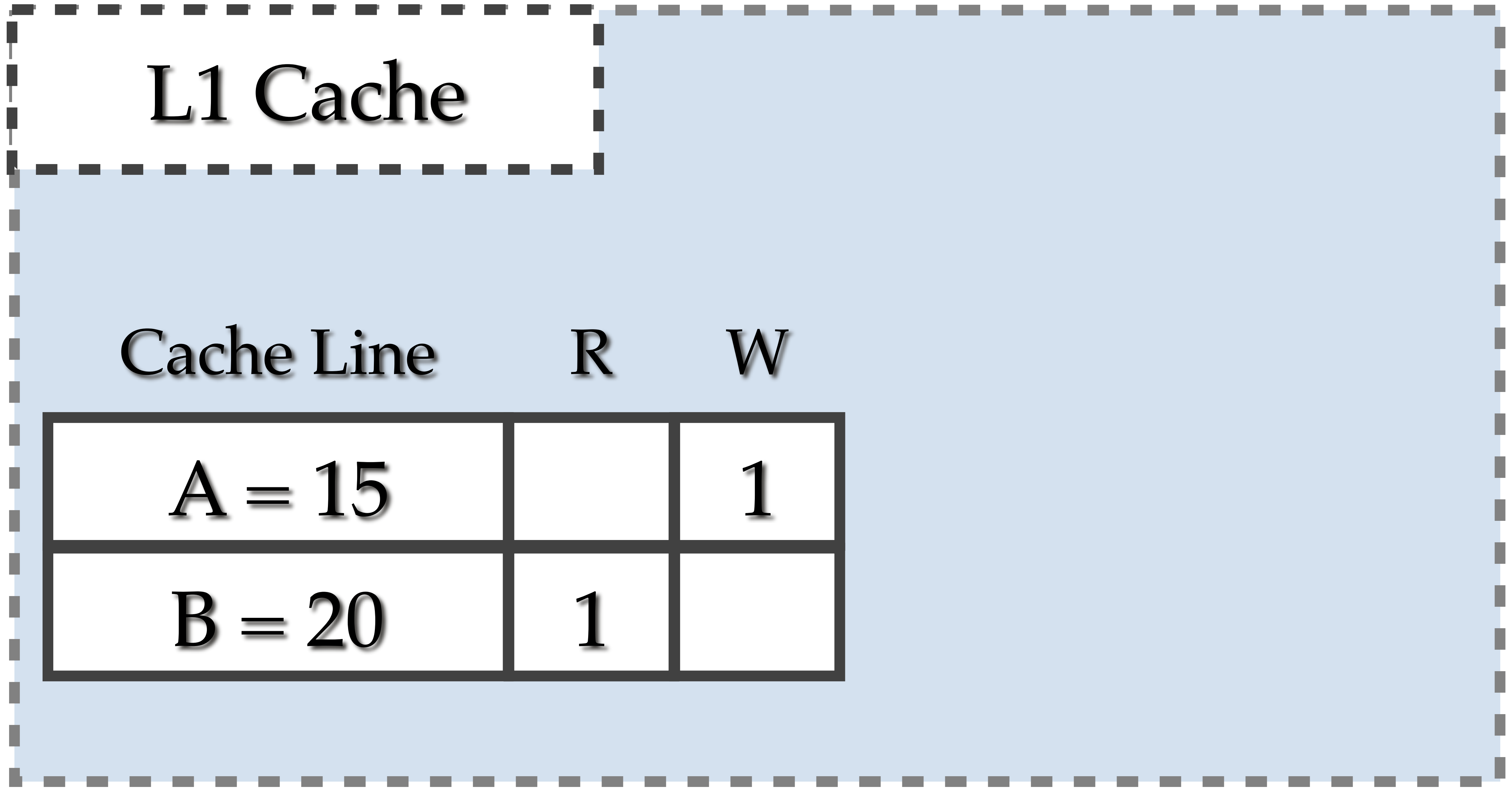


L1 Cache

Cache Line	R	W
A = 15		1
B = 20	1	

- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache

Atomic Visibility: HTM

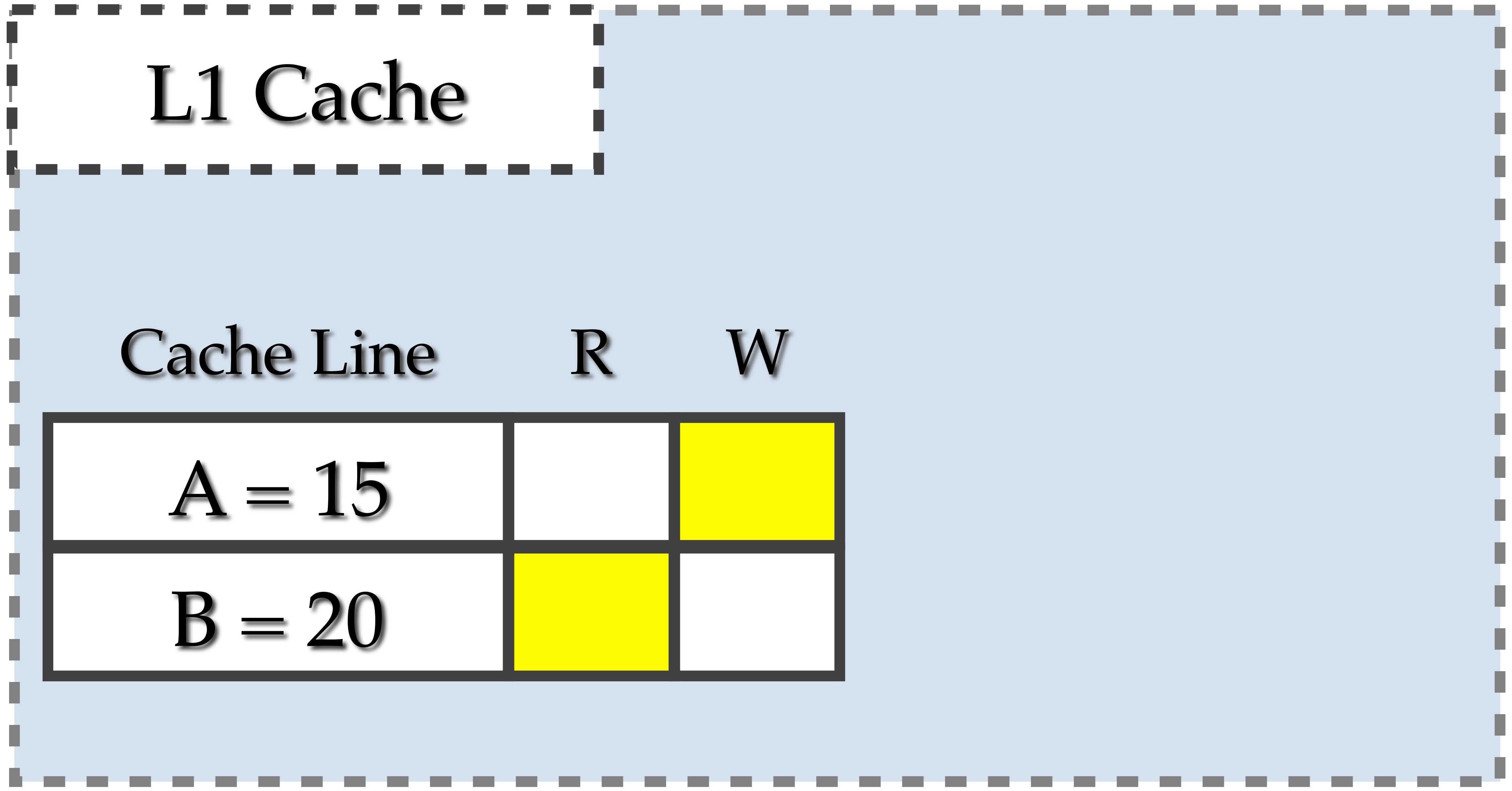


L1 Cache

Cache Line	R	W
A = 15		1
B = 20	1	

- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol

Atomic Visibility: HTM

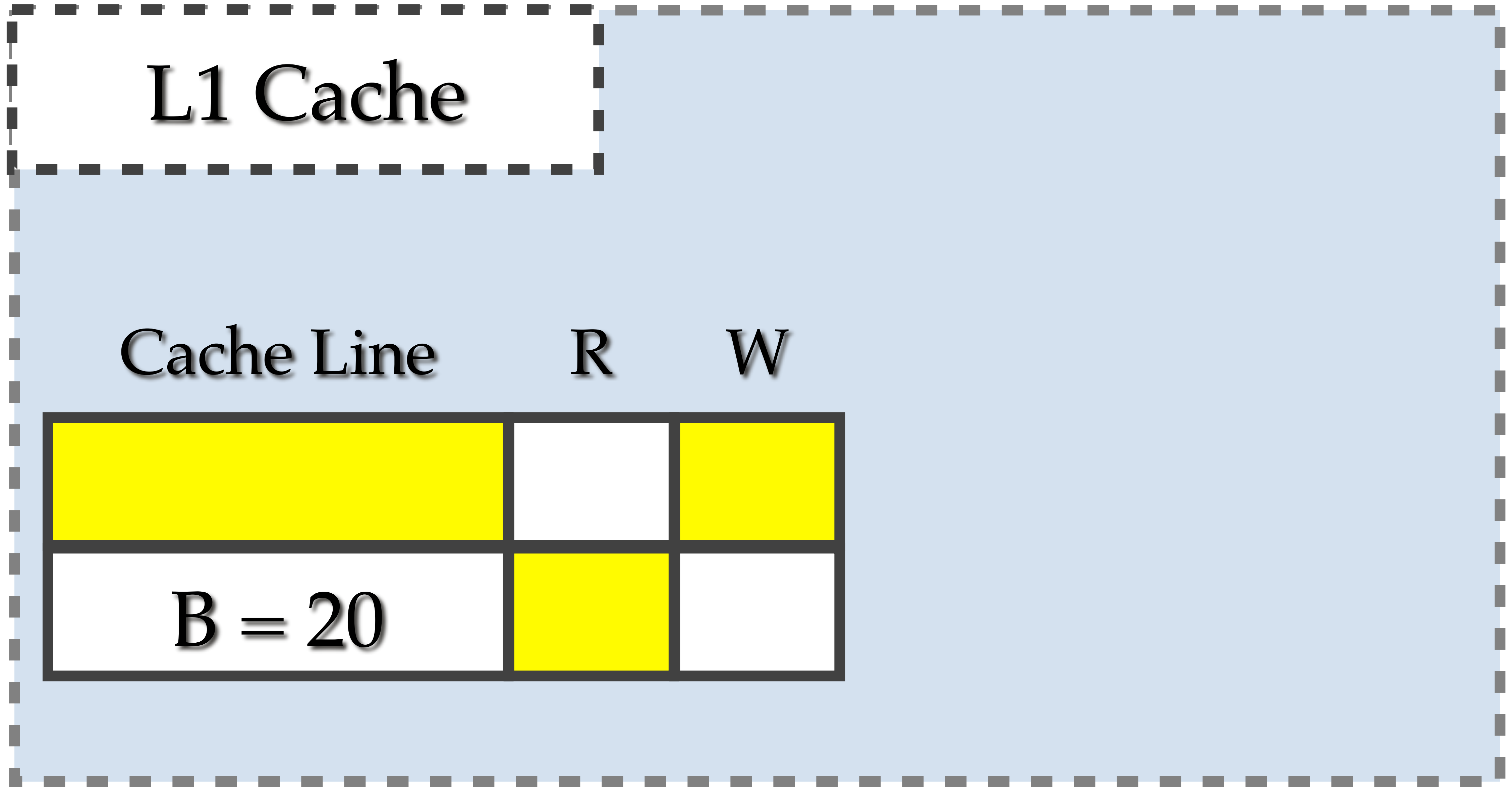


L1 Cache

Cache Line	R	W
A = 15		🟨
B = 20	🟨	

- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol
  - **Commit**: make updates non-speculative

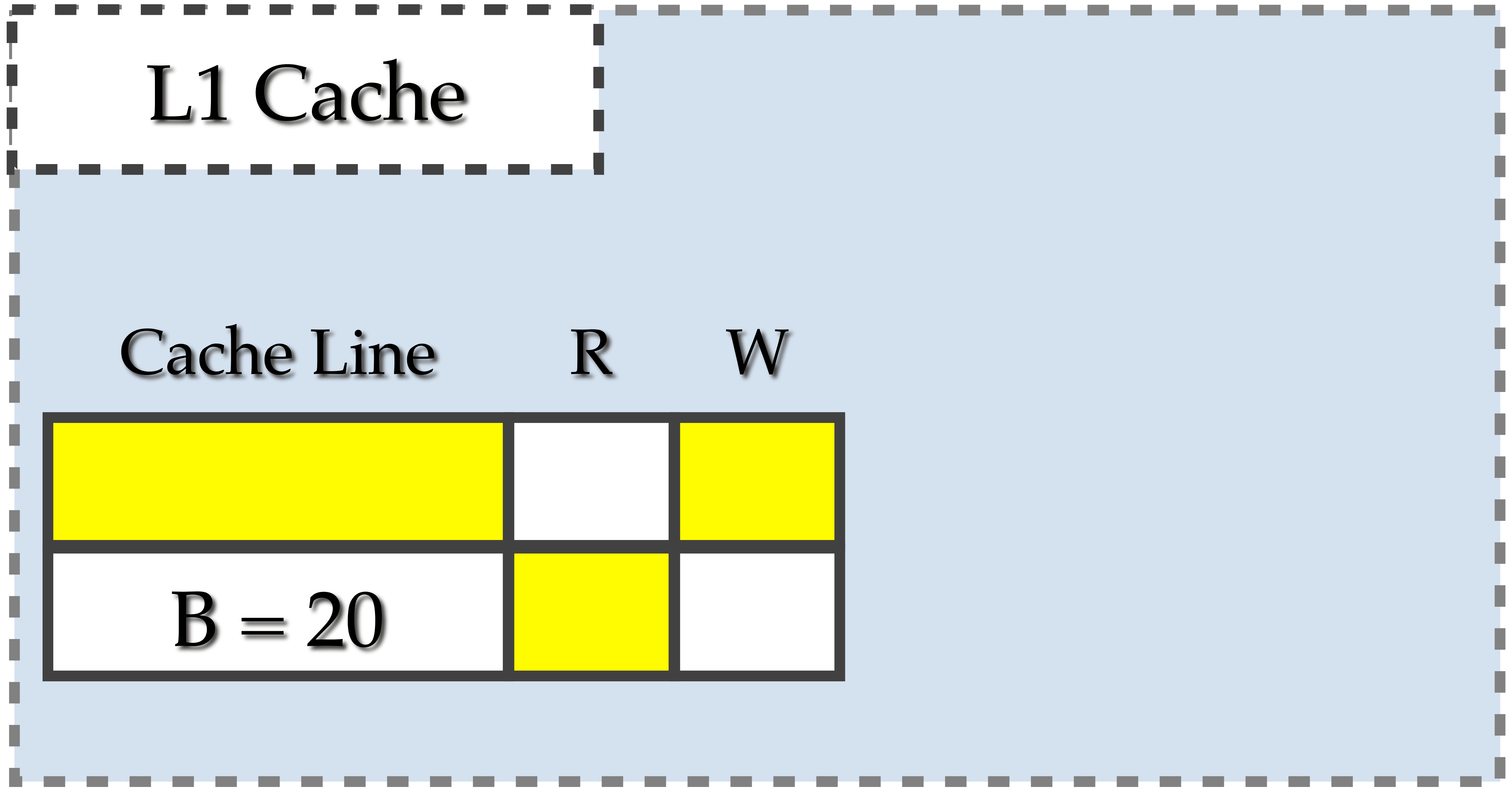
Atomic Visibility: HTM



Cache Line	R	W
🟨		🟨
B = 20	🟨	

- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol
  - **Commit**: make updates non-speculative
  - **Abort**: invalidate write set

Atomic Visibility: HTM

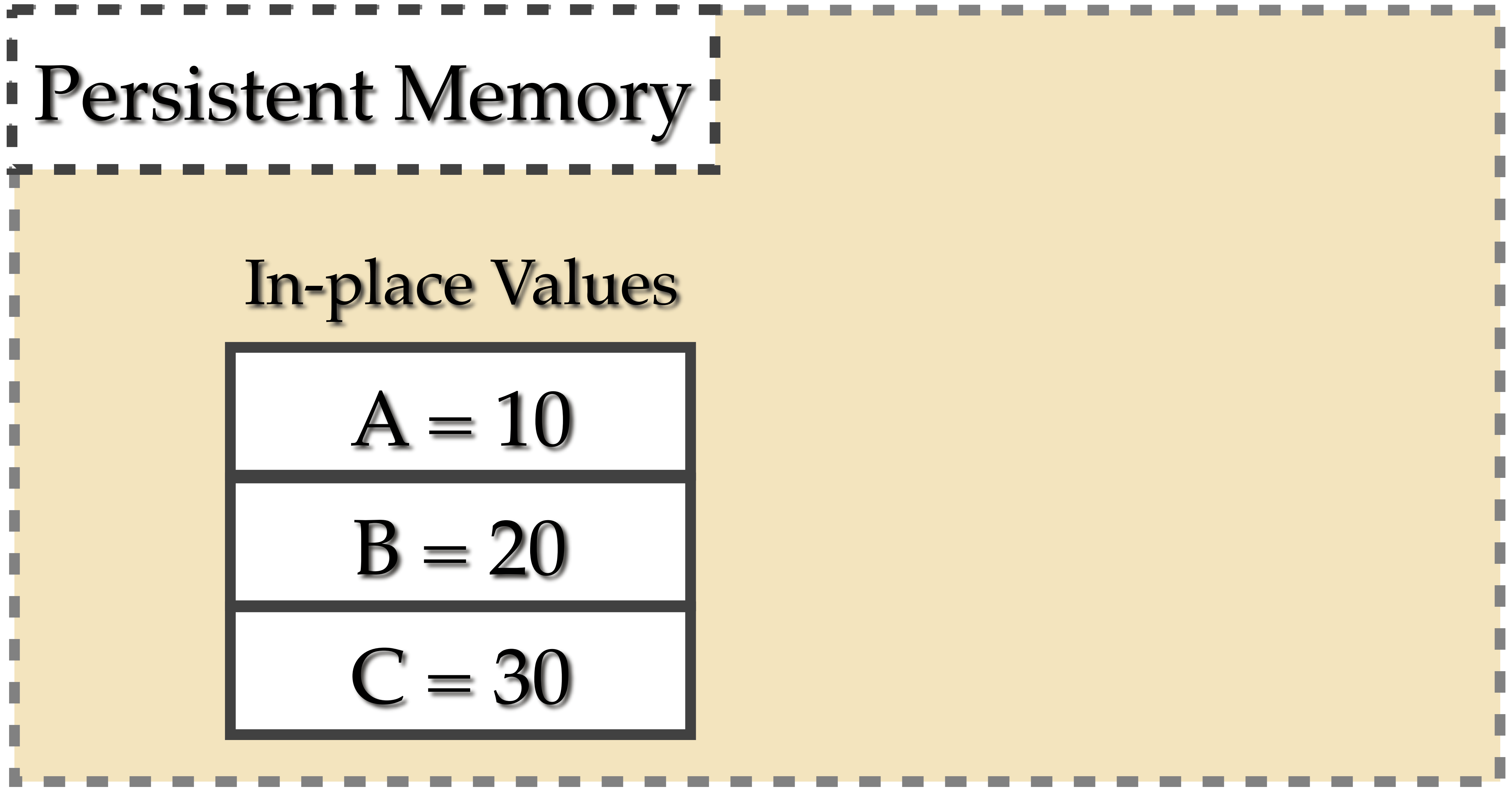


- **Commercial HTMs** [Intel, IBM]
    - **Version Management**: read/write sets in L1 cache
    - **Conflict Detection**: piggy back on the coherence protocol
    - **Commit**: make updates non-speculative
    - **Abort**: invalidate write set

✗ **Write-sets in commercial HTMs limited by the size of the L1 cache.**

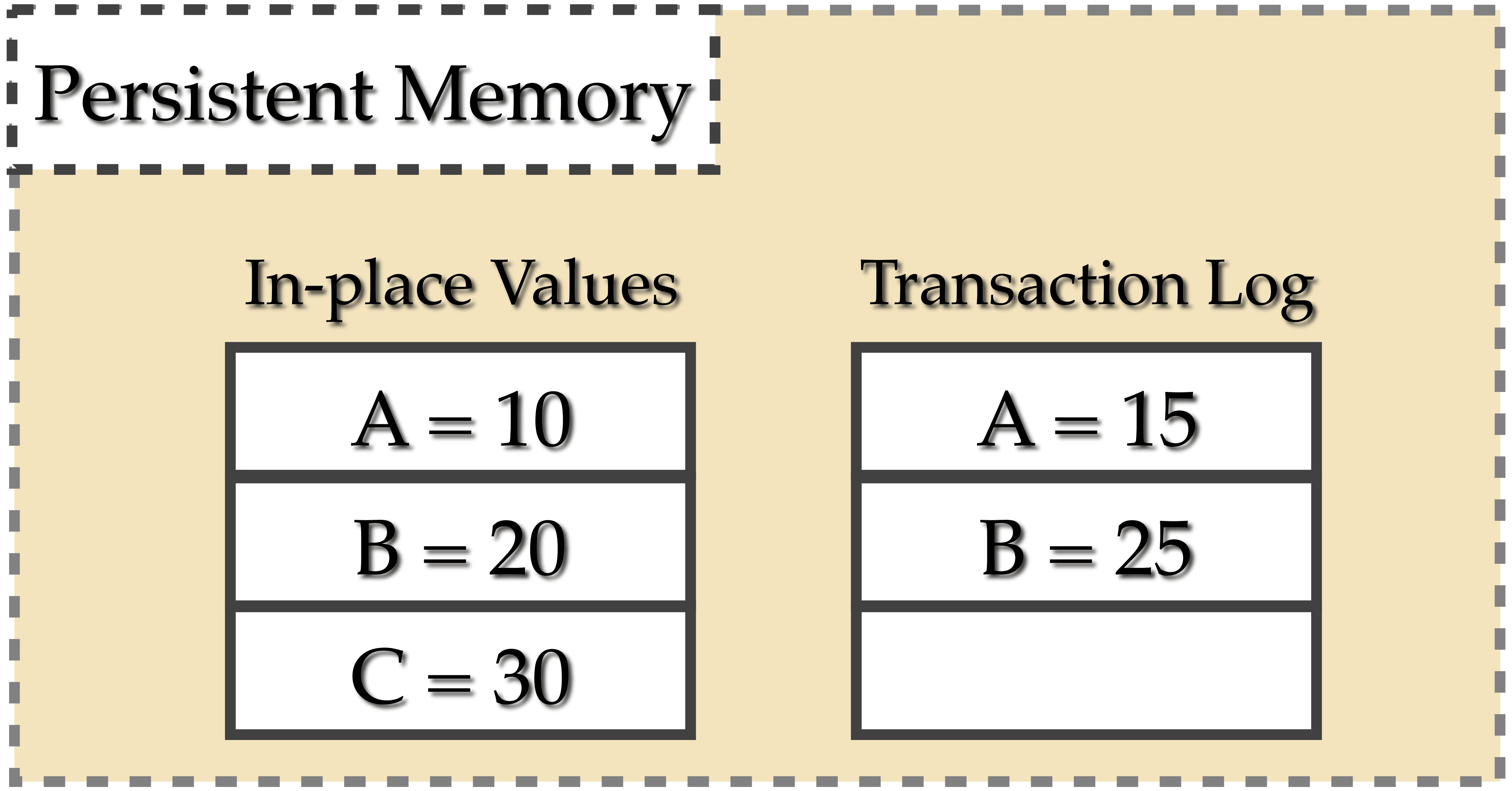
Atomic Durability: Logging

Atomic Durability: Logging



- **Logging for durability** [Doshi'16, Joshi'17, Shin'17, Ogleari'18]

Atomic Durability: Logging

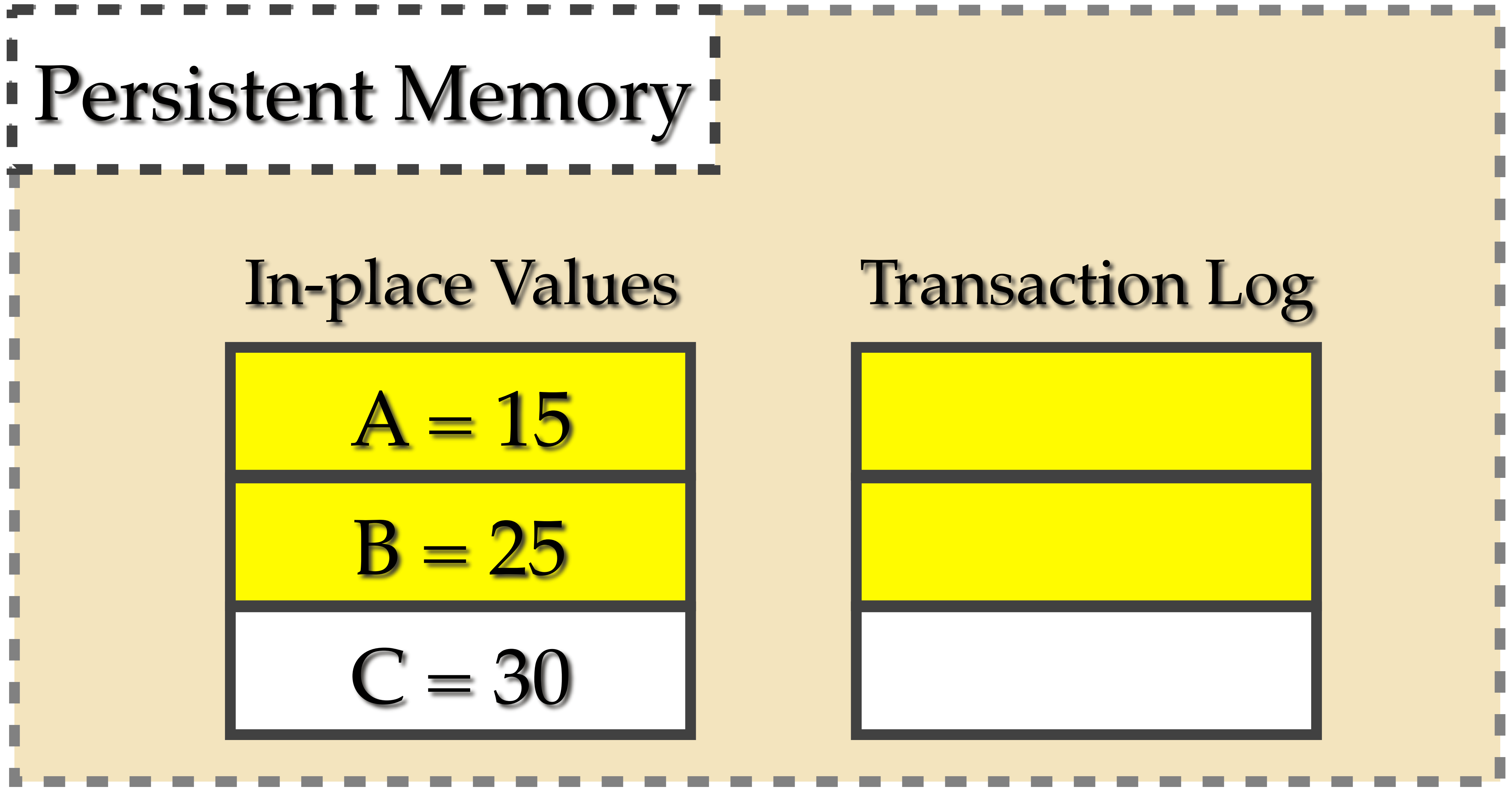


Persistent Memory

In-place Values	Transaction Log
A = 10	A = 15
B = 20	B = 25
C = 30	

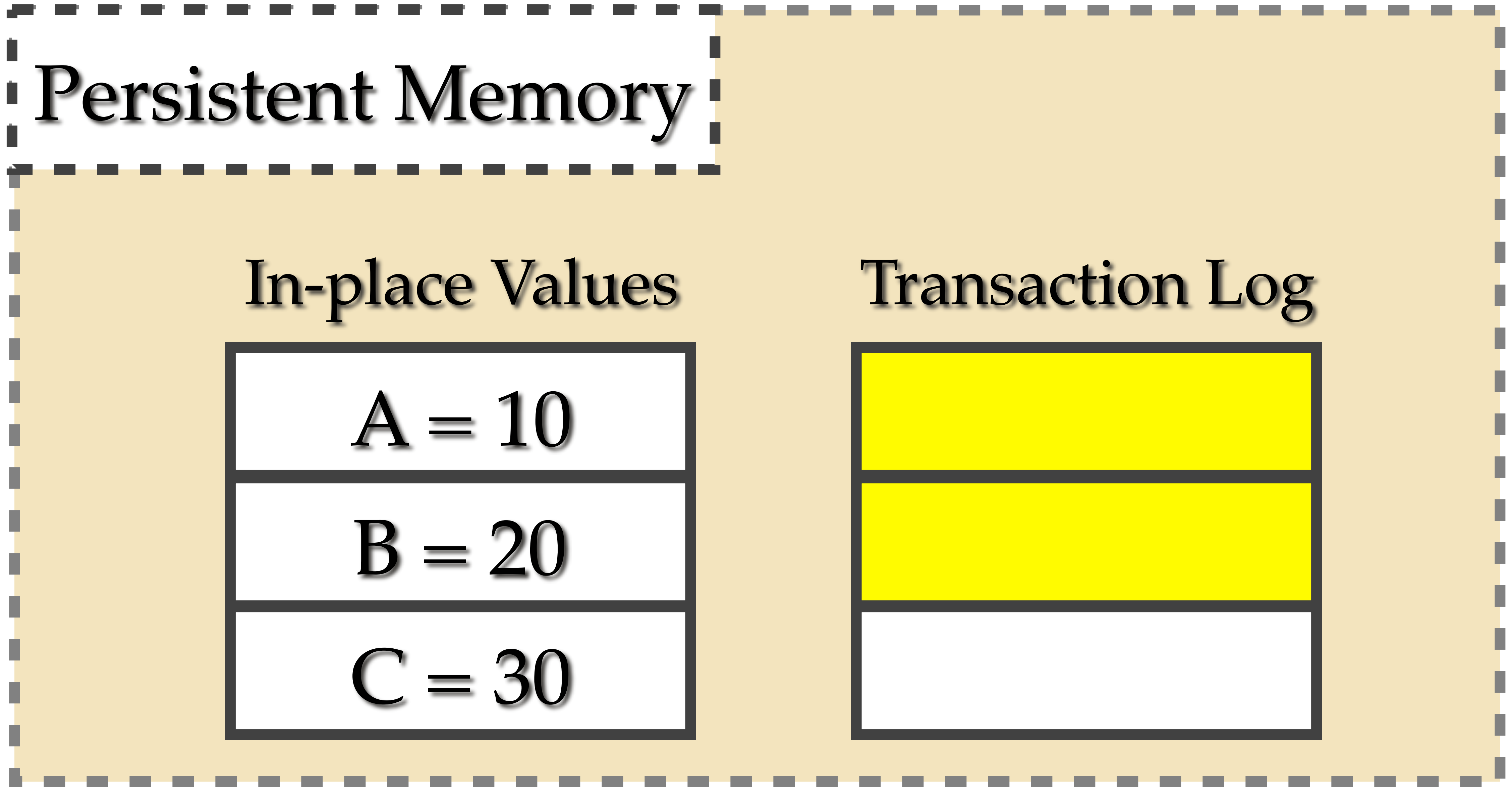
- **Logging for durability** [Doshi’16, Joshi’17, Shin’17, Ogleari’18]
  - Write a log entry for every update

Atomic Durability: Logging



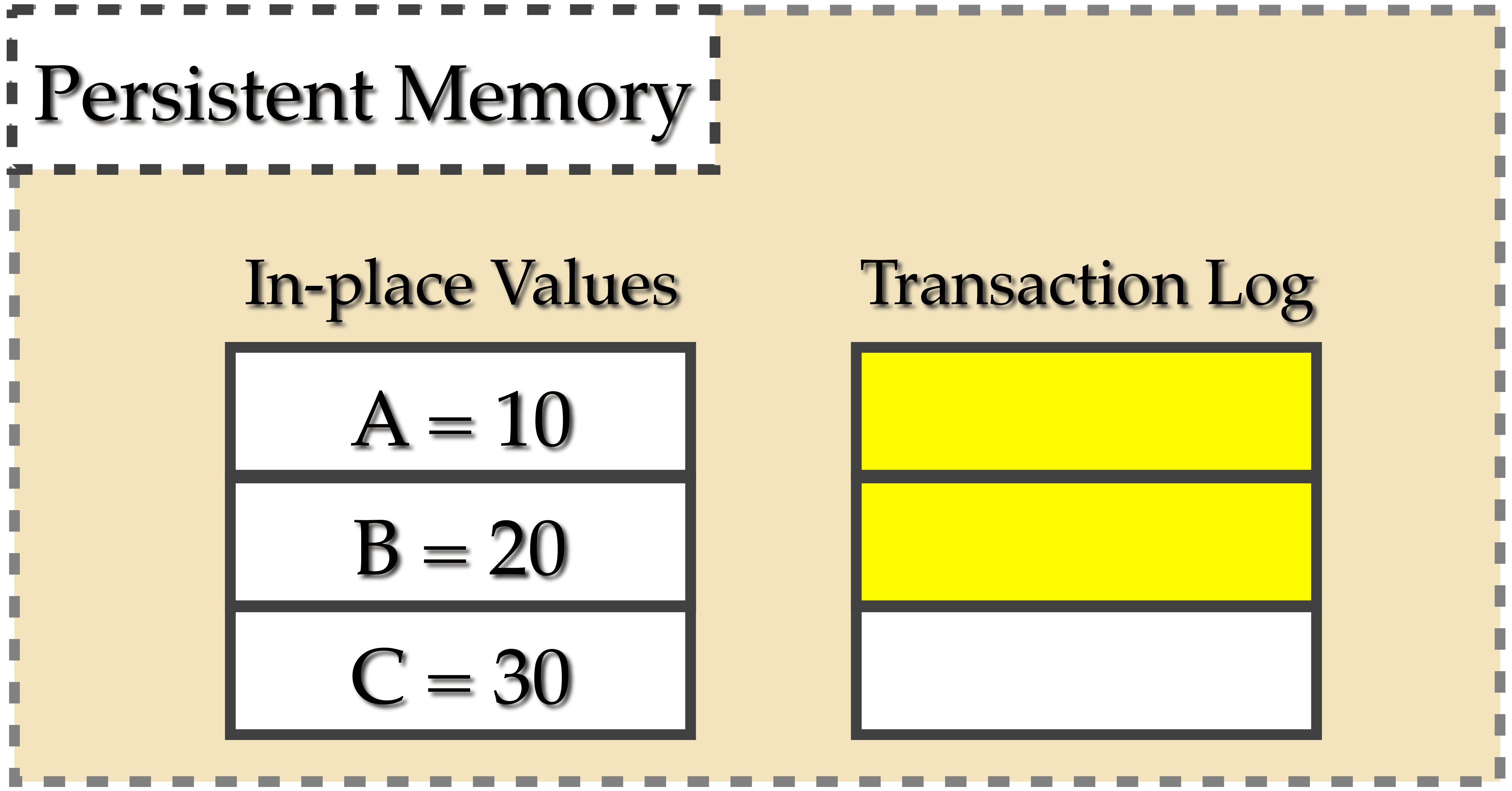
- **Logging for durability** [Doshi’16, Joshi’17, Shin’17, Ogleari’18]
    - Write a log entry for every update
    - **Commit**: Update the values in-place

Atomic Durability: Logging



- **Logging for durability** [Doshi'16, Joshi'17, Shin'17, Ogleari'18]
  - Write a log entry for every update
  - **Commit**: Update the values in-place
  - **Abort**: Undo any in-place updates

Atomic Durability: Logging



- **Logging for durability** [Doshi'16, Joshi'17, Shin'17, Ogleari'18]
  - Write a log entry for every update
  - **Commit**: Update the values in-place
  - **Abort**: Undo any in-place updates

✗ In-place updates in the critical path of commit

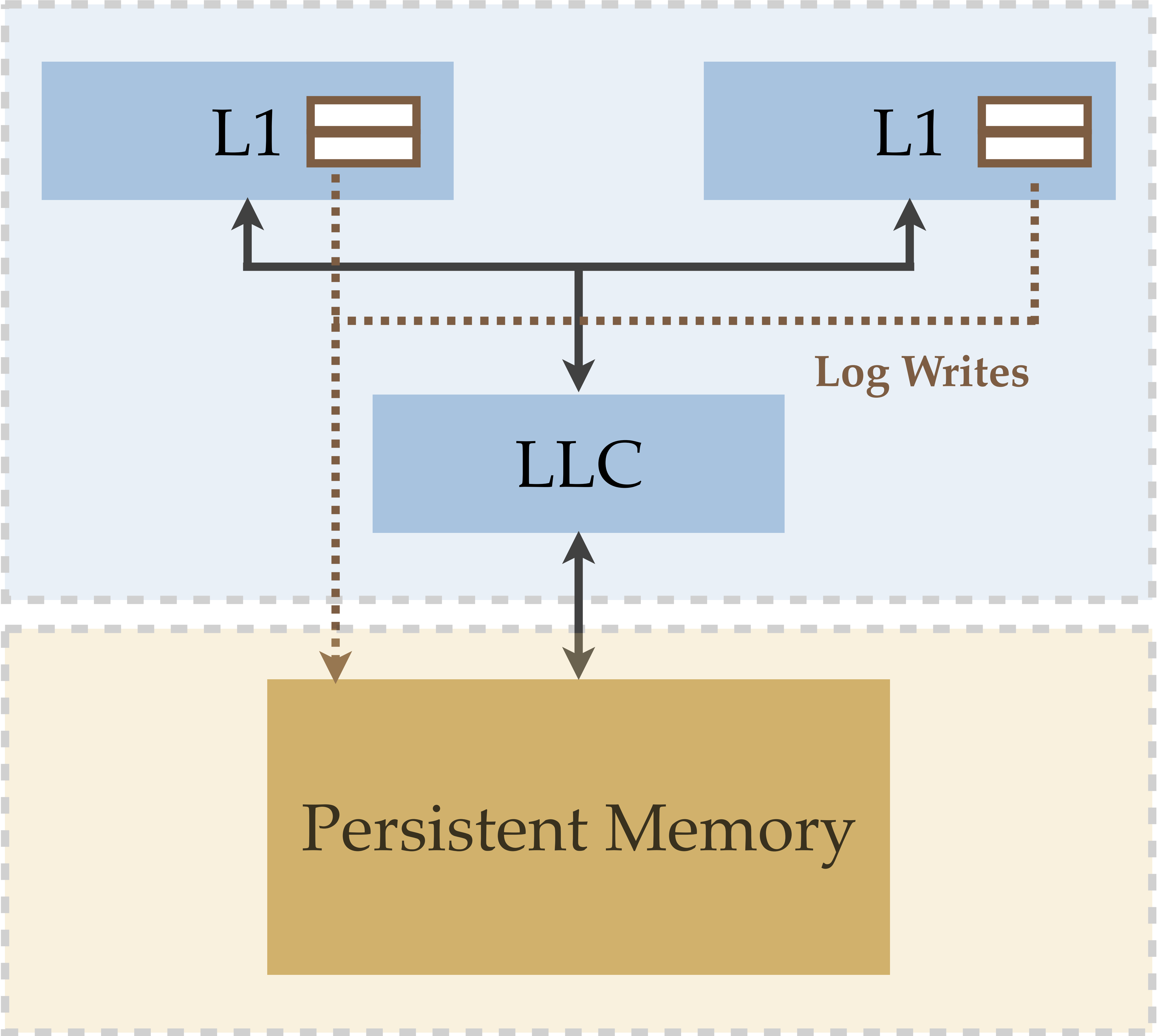
✗ High memory write bandwidth requirement

ACID = HTM + Logging

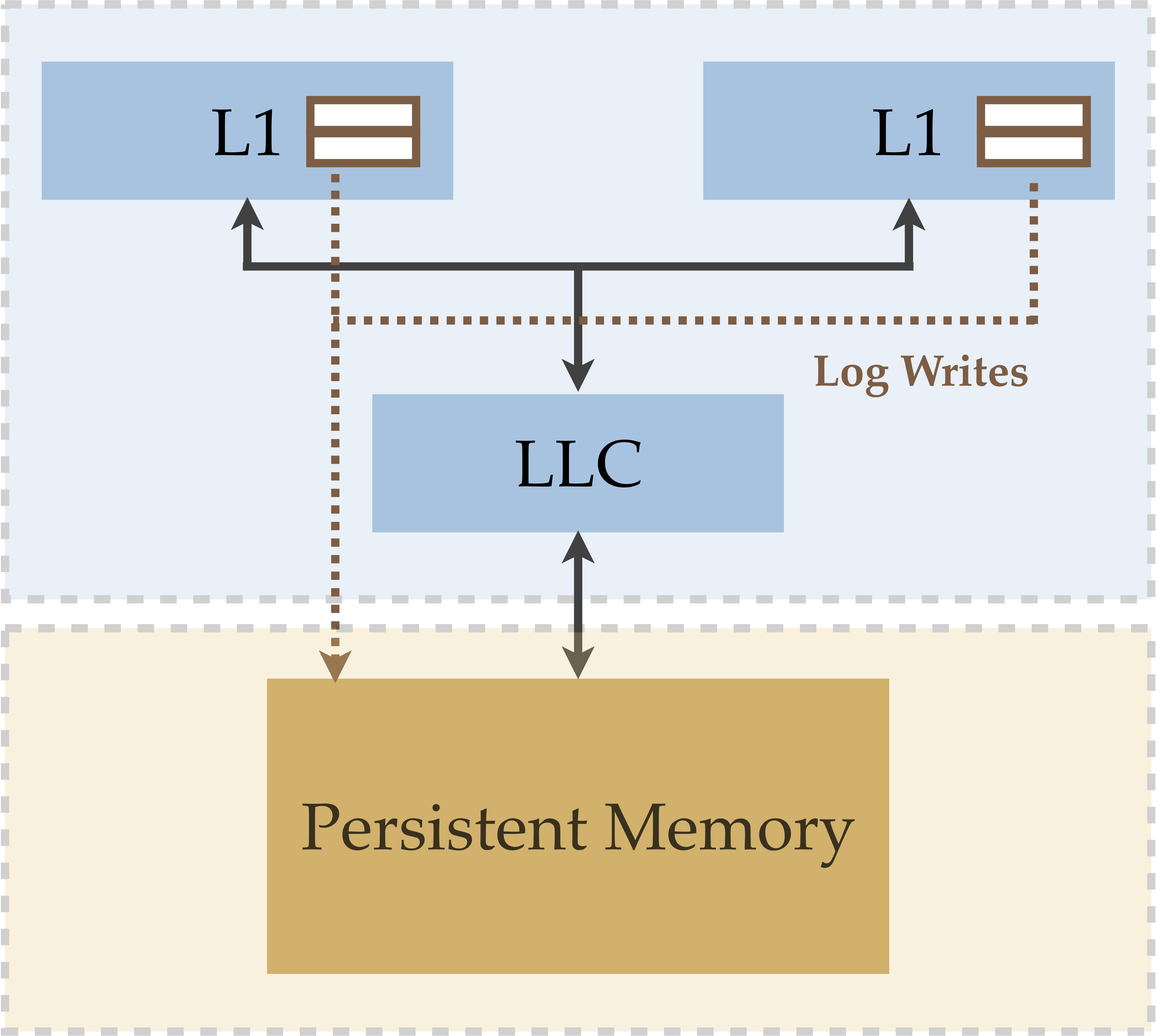
Goals:

- Support fast commits
- Minimise memory bandwidth consumption
- Extend the supported transaction size
- Maintain the simplicity of commercial HTMs

DHTM: Durable Hardware Transactional Memory

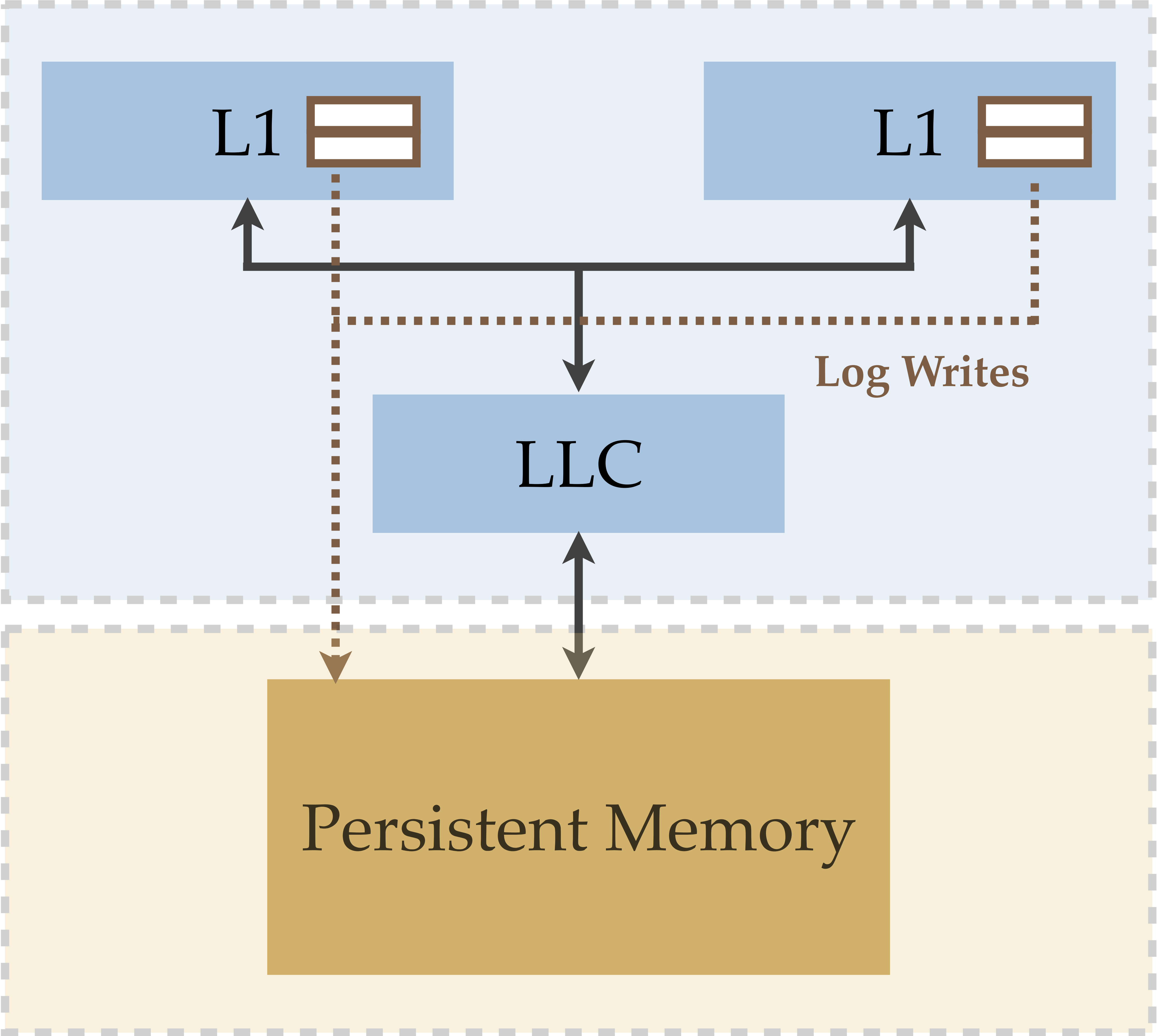


DHTM: Durable Hardware Transactional Memory



Commercial HTM + Hardware Redo Log

DHTM: Durable Hardware Transactional Memory



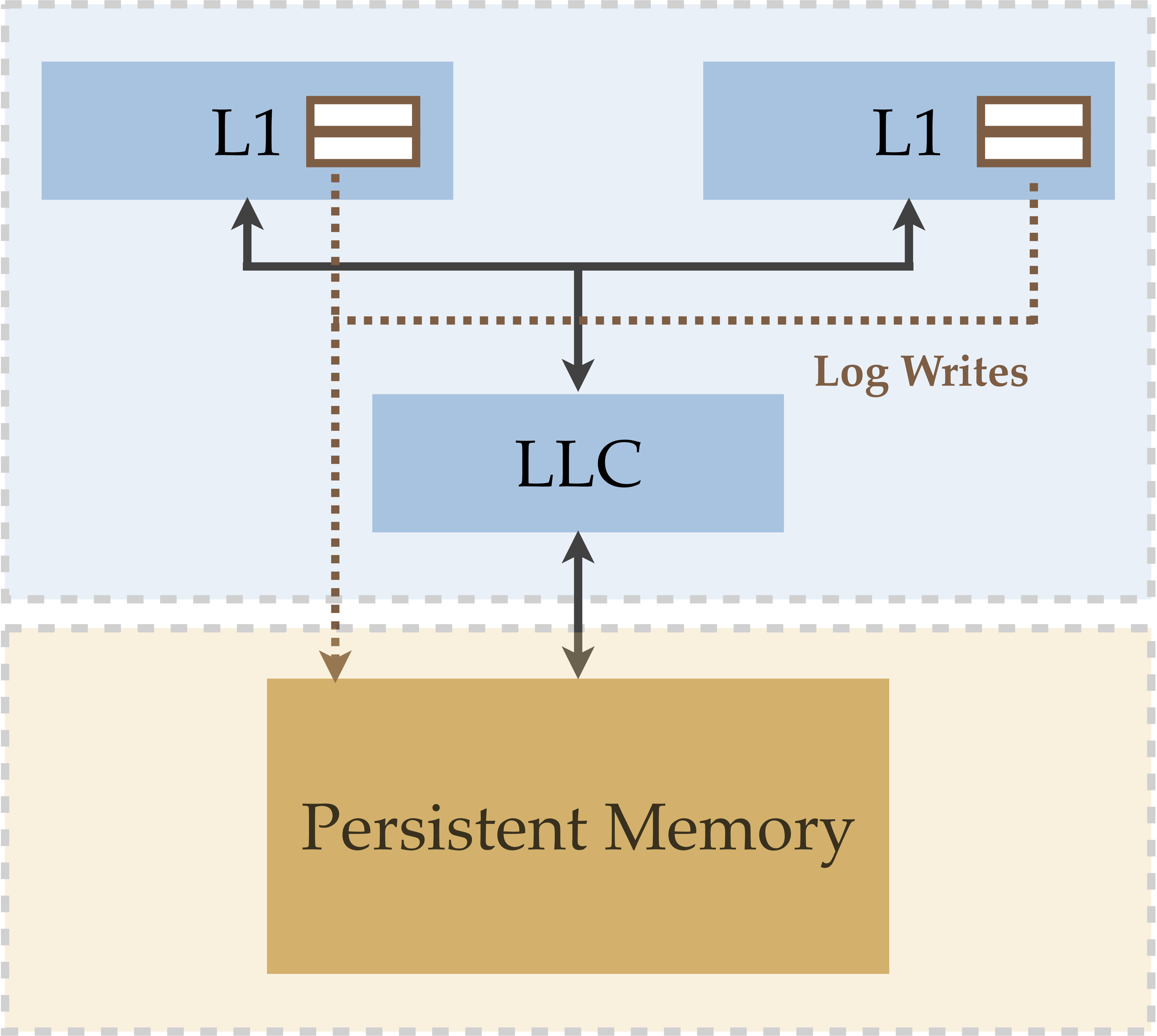
Commercial HTM + **Hardware Redo Log**

- H/W Redo Log + Log Buffer

✓ Reduced memory bandwidth

✓ Fast commits

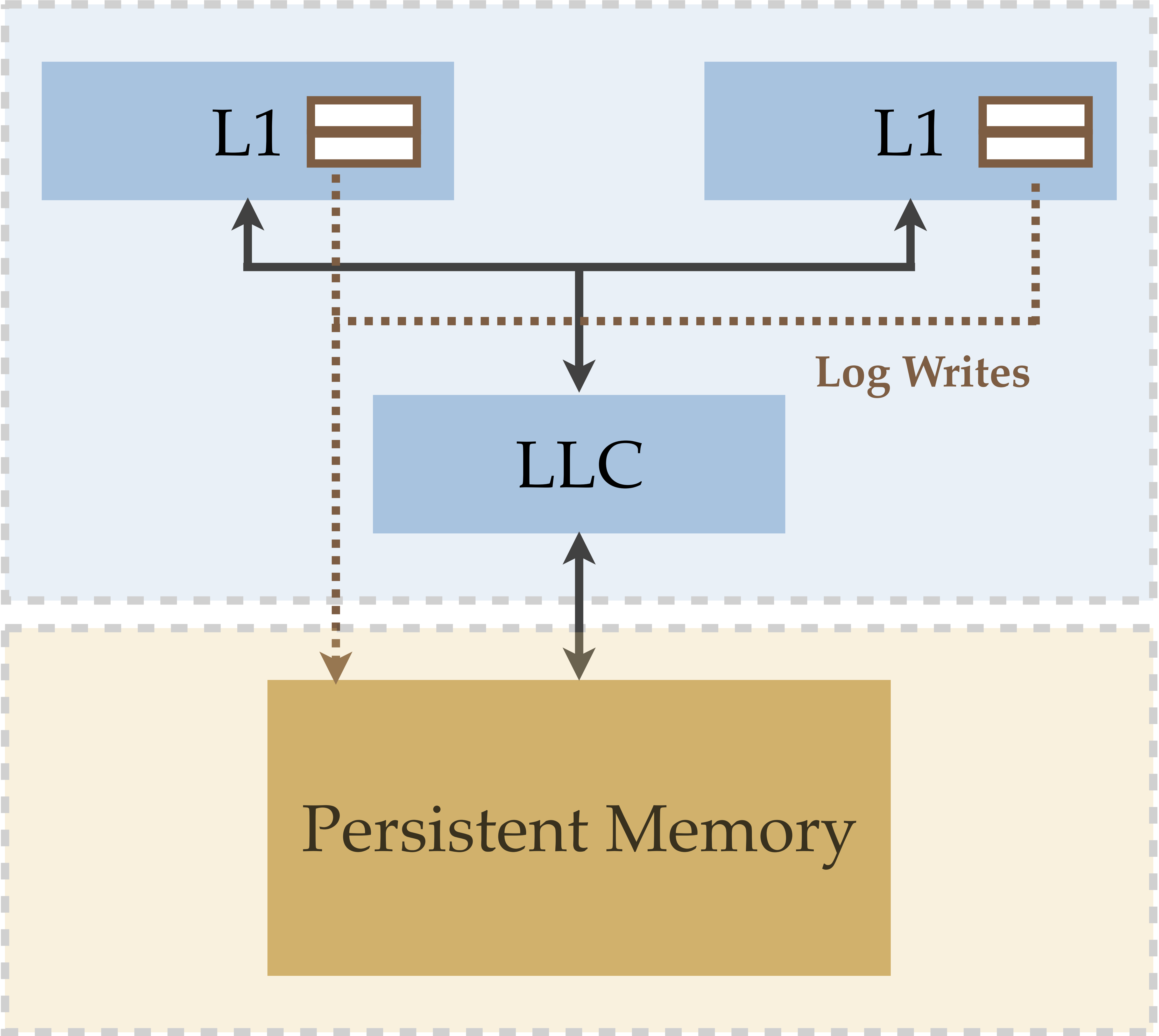
DHTM: Durable Hardware Transactional Memory



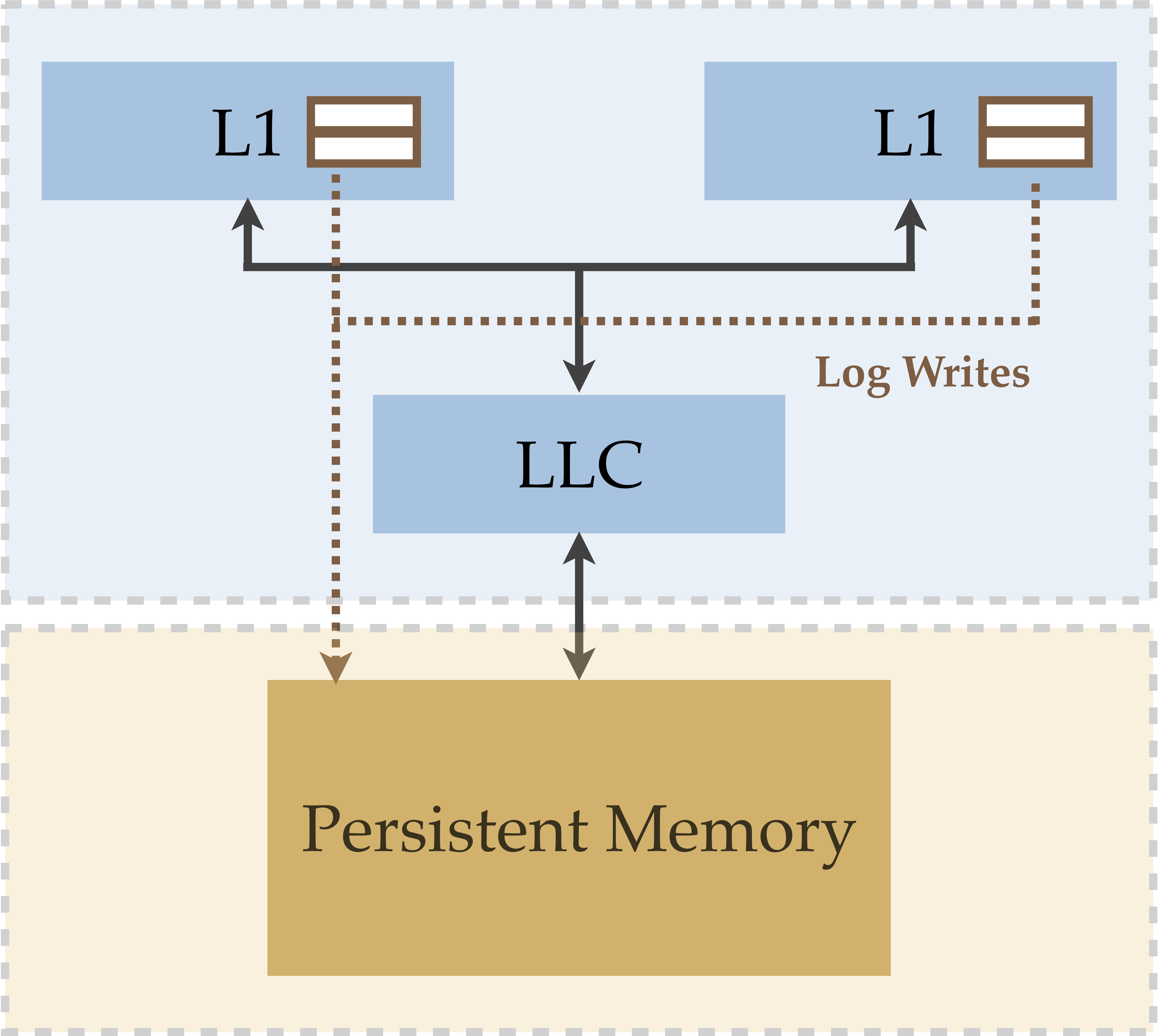
Commercial HTM + Hardware Redo Log

- H/W Redo Log + Log Buffer
  - ✓ Reduced memory bandwidth
  - ✓ Fast commits
- H/W Log + Sticky State
  - ✓ Extended transaction size to the LLC
  - ✓ Simplicity of commercial HTM

DHTM: Log Buffer

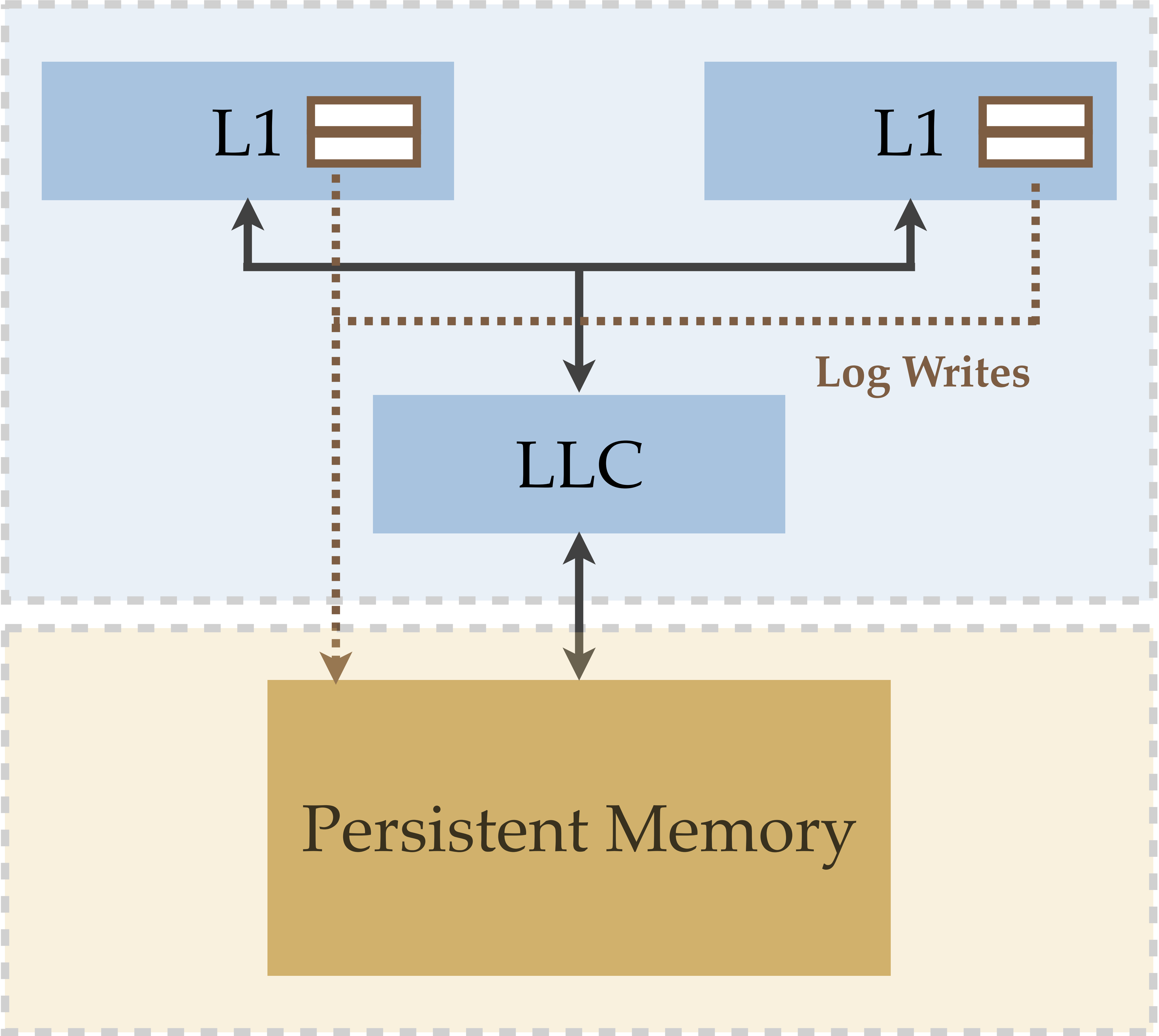


DHTM: Log Buffer



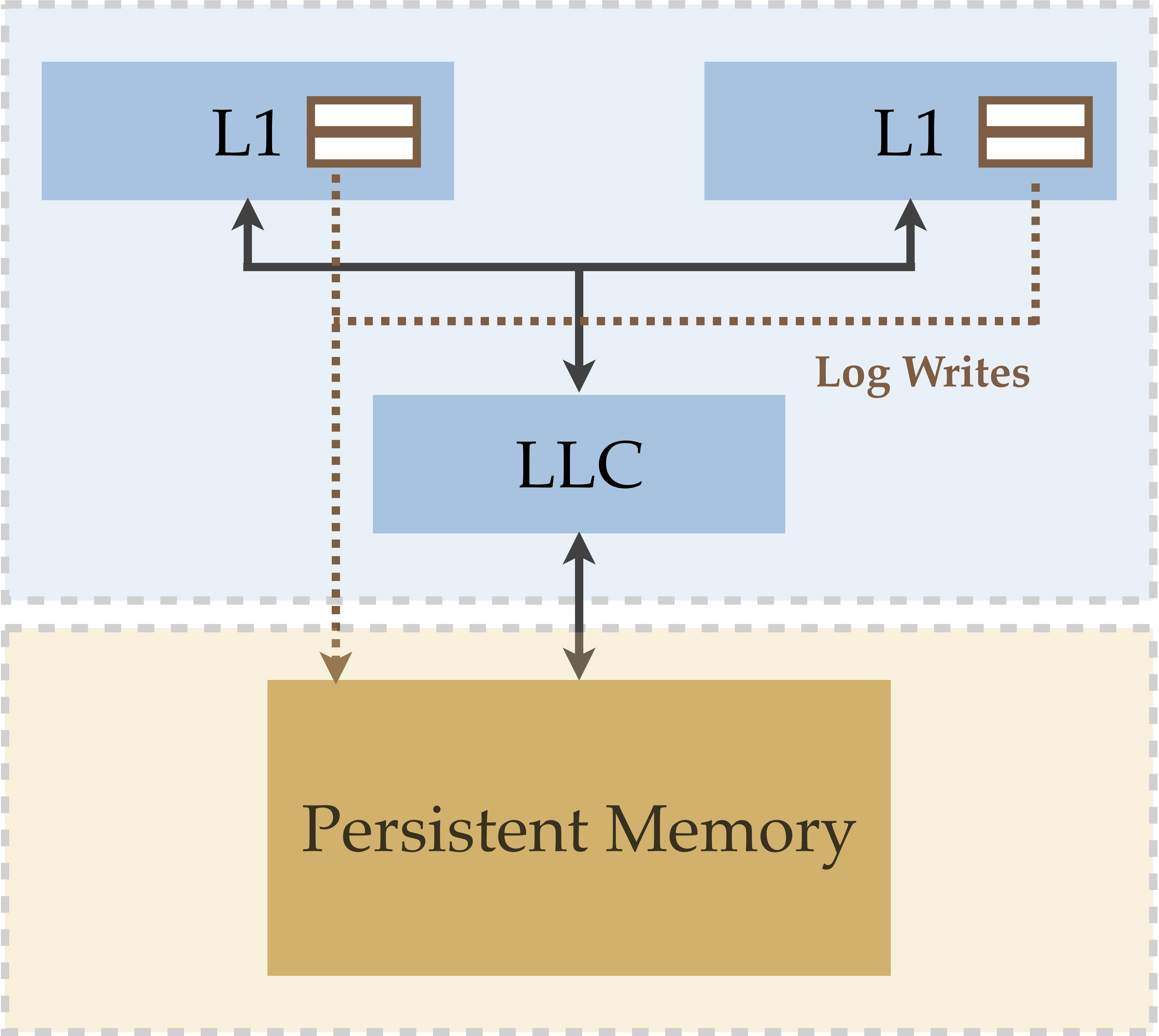
- **Redo Log Bandwidth Problem**

DHTM: Log Buffer



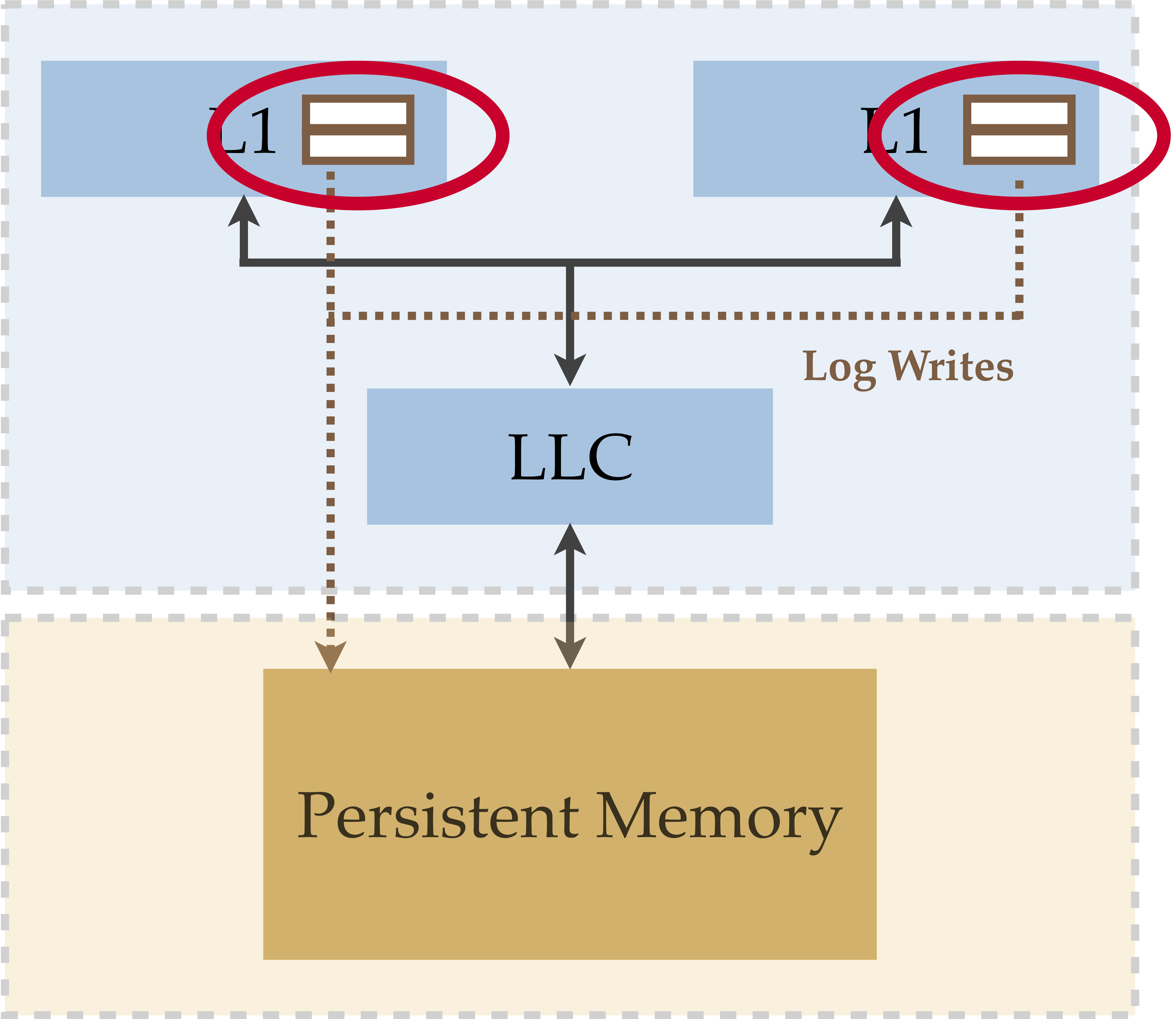
- **Redo Log Bandwidth Problem**
    - write a log entry for every store

DHTM: Log Buffer



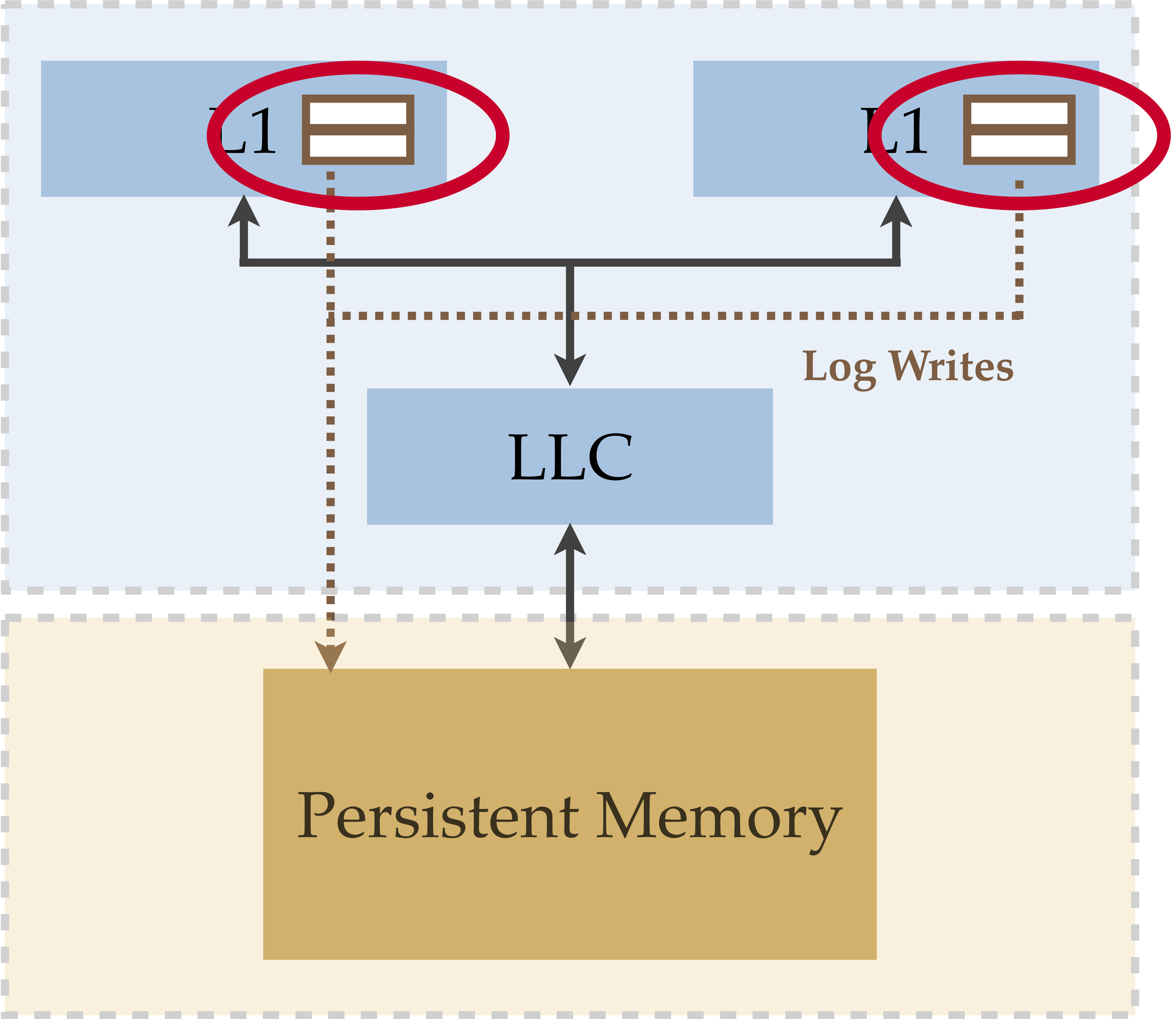
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries

DHTM: Log Buffer



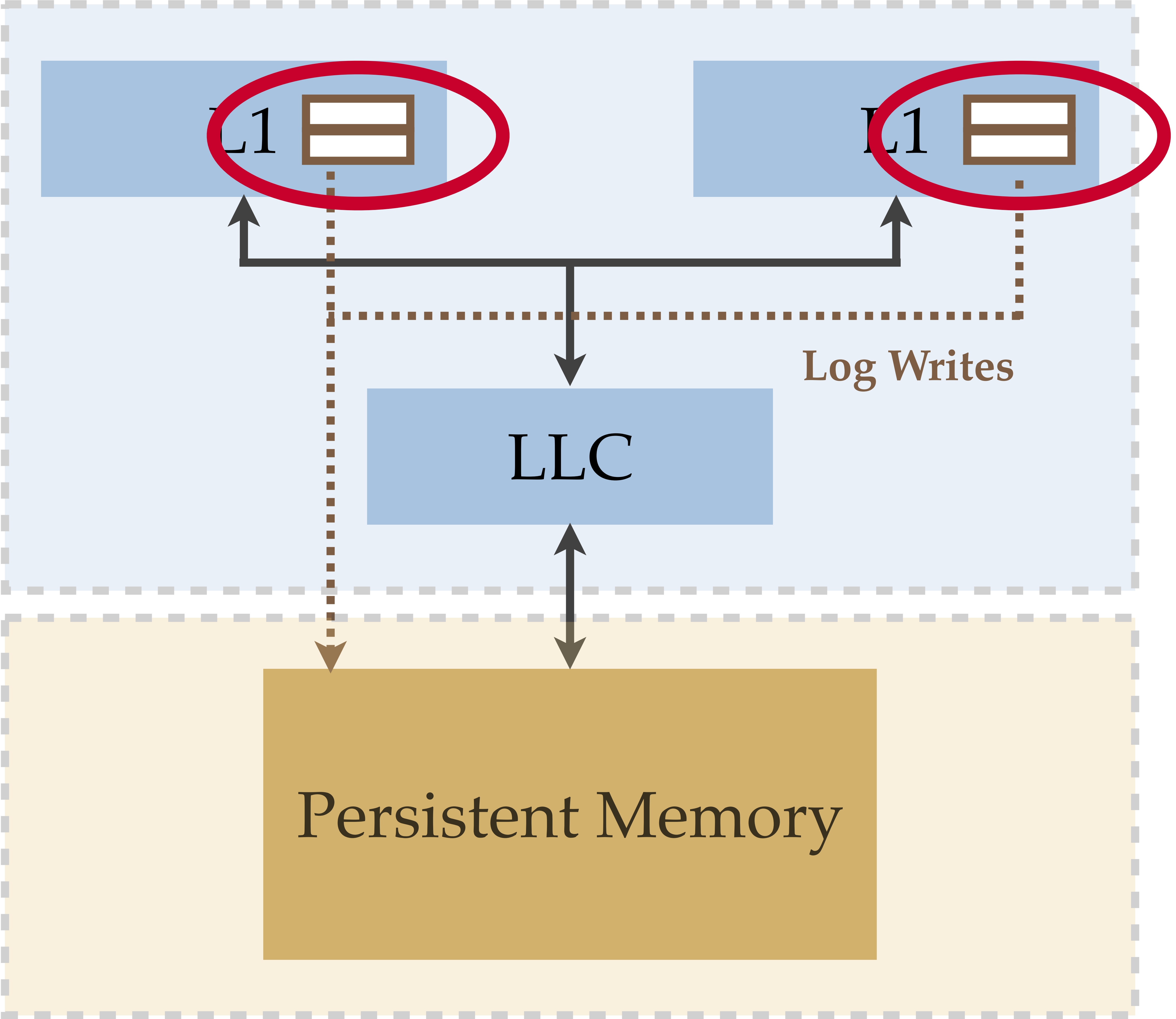
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries
- **Solution: Log Buffer**

DHTM: Log Buffer



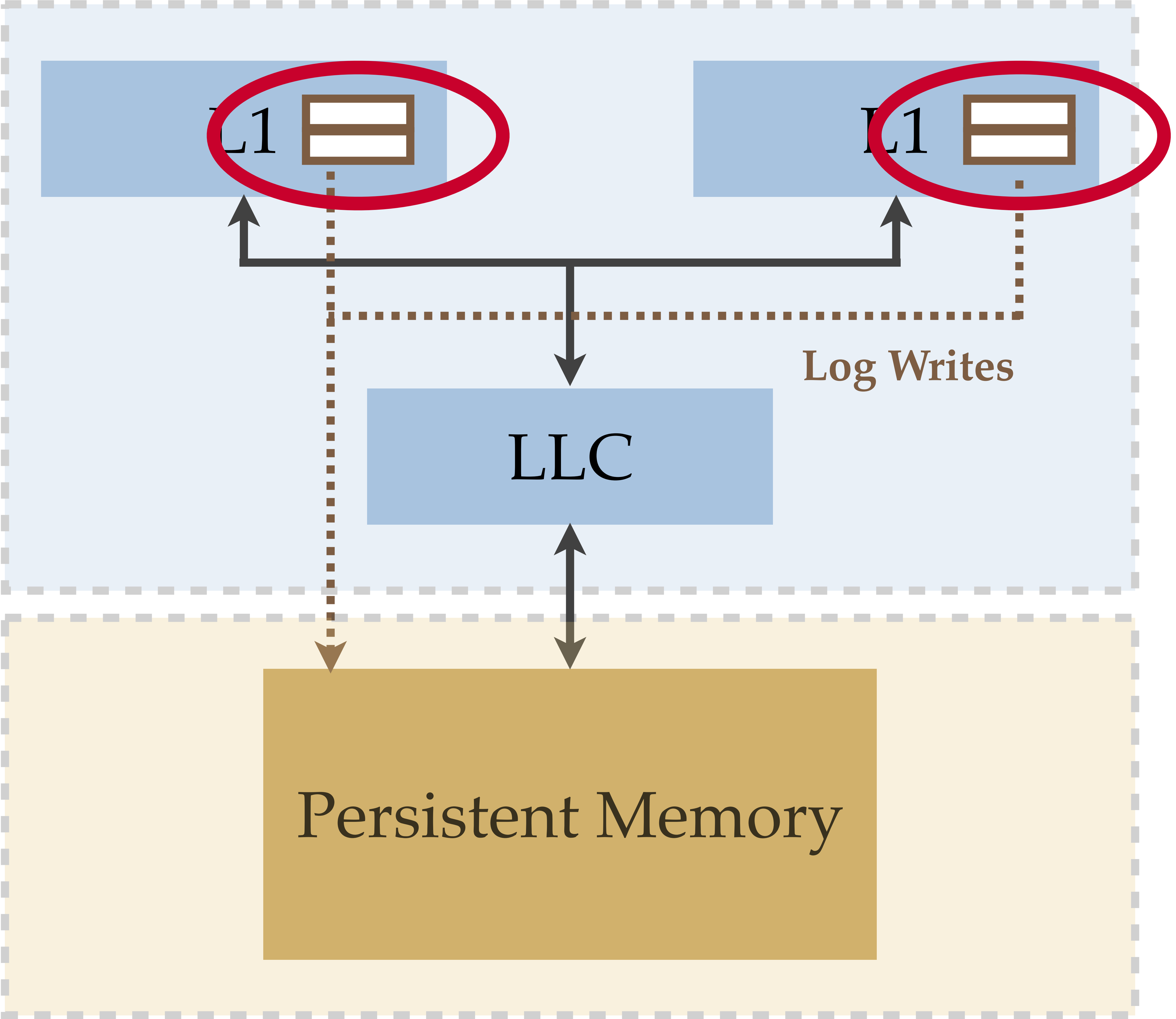
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries
- **Solution: Log Buffer**
  - track cache lines being modified

DHTM: Log Buffer



- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries
- **Solution: Log Buffer**
  - track cache lines being modified
  - multiple writes coalesced in a log entry

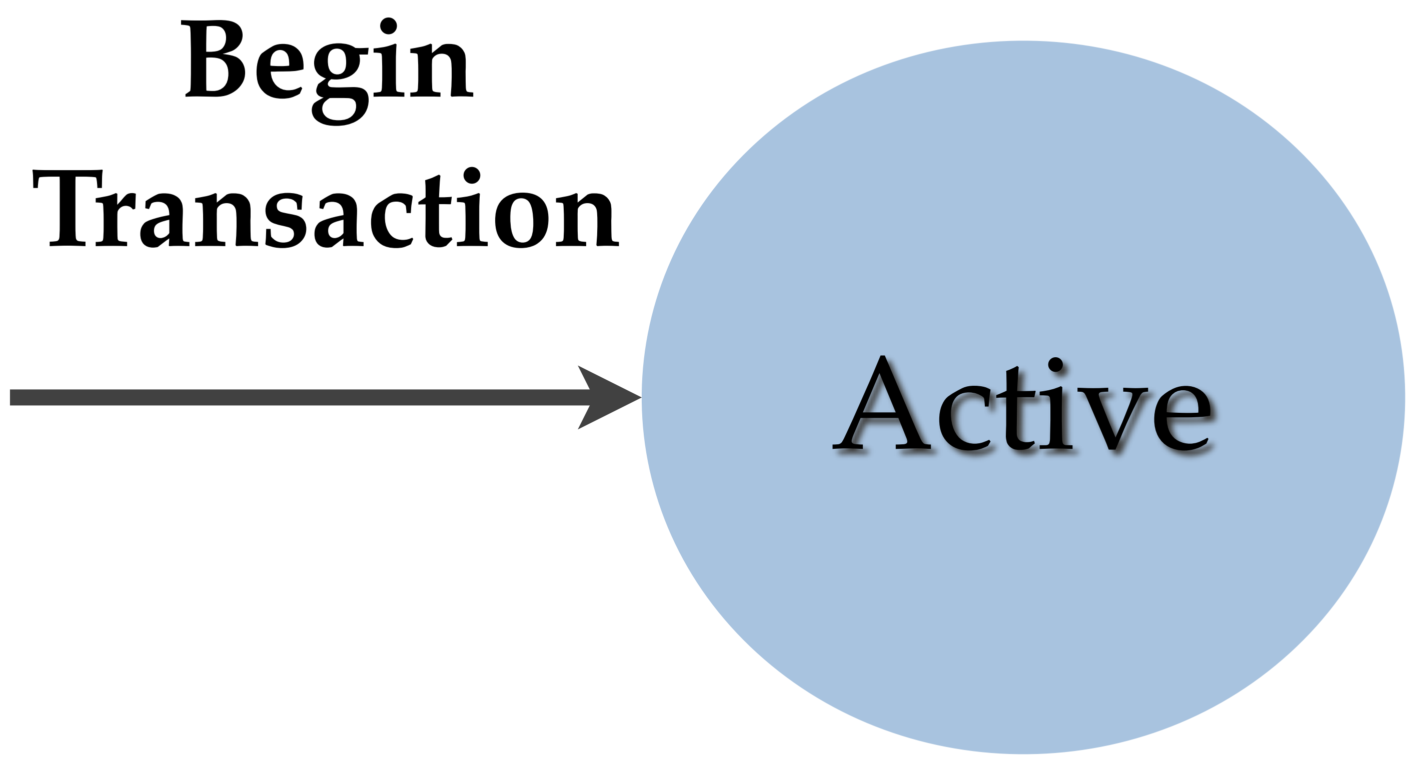
DHTM: Log Buffer



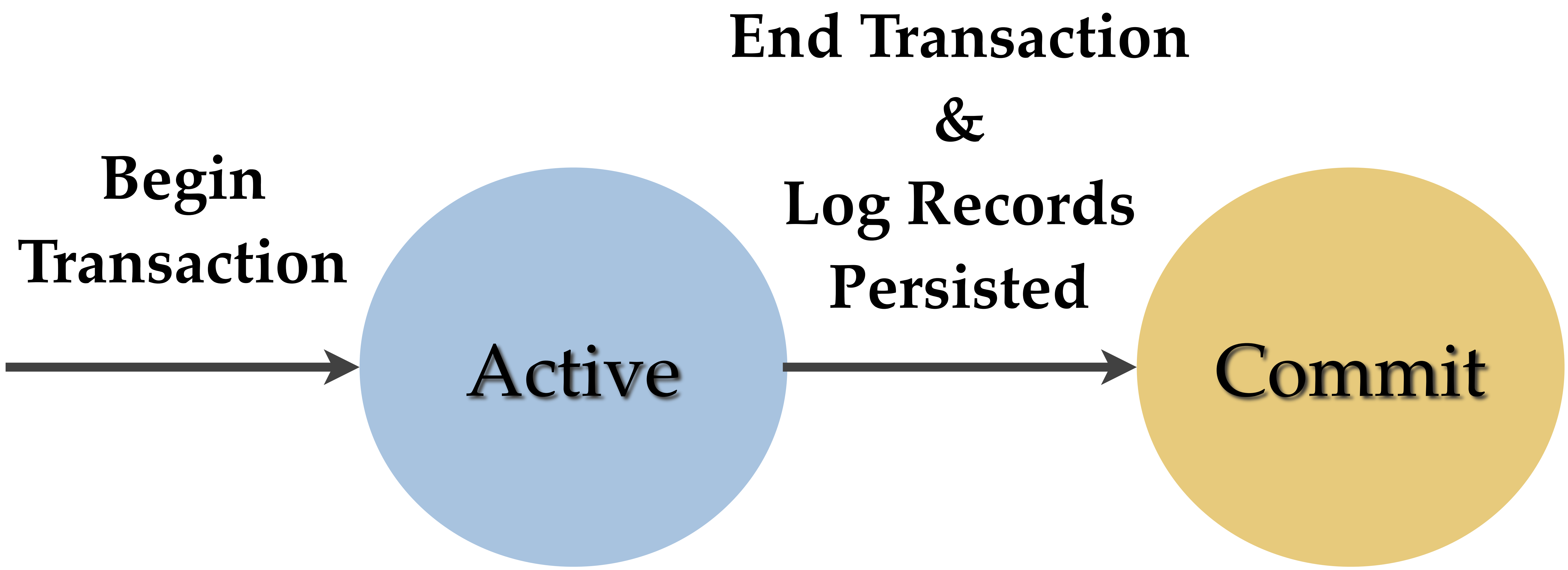
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries
- **Solution: Log Buffer**
  - track cache lines being modified
  - multiple writes coalesced in a log entry
  - log entry written to persistent memory on eviction from log buffer

DHTM: Transaction States

DHTM: Transaction States



DHTM: Transaction States



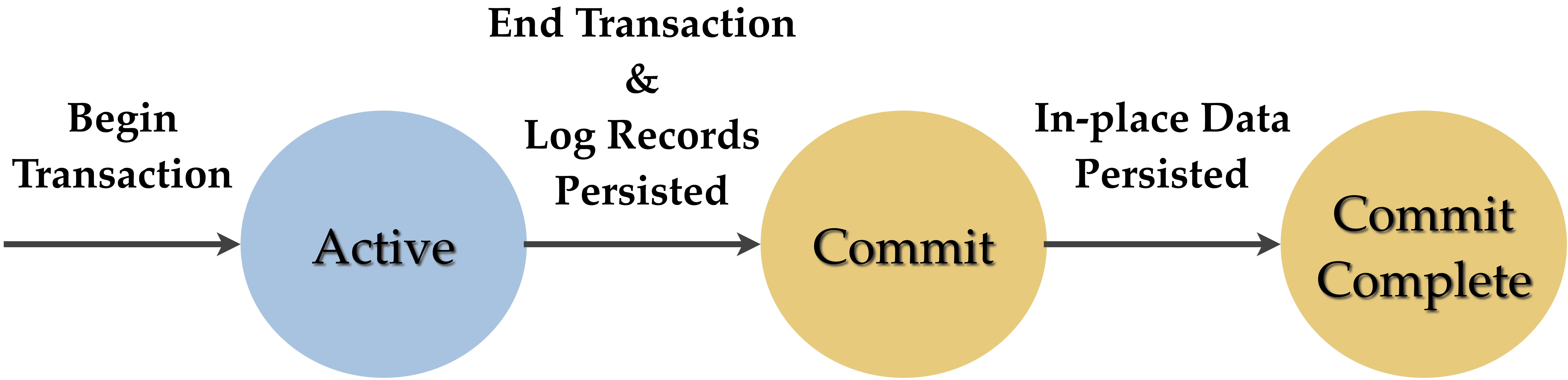
Begin Transaction

Active

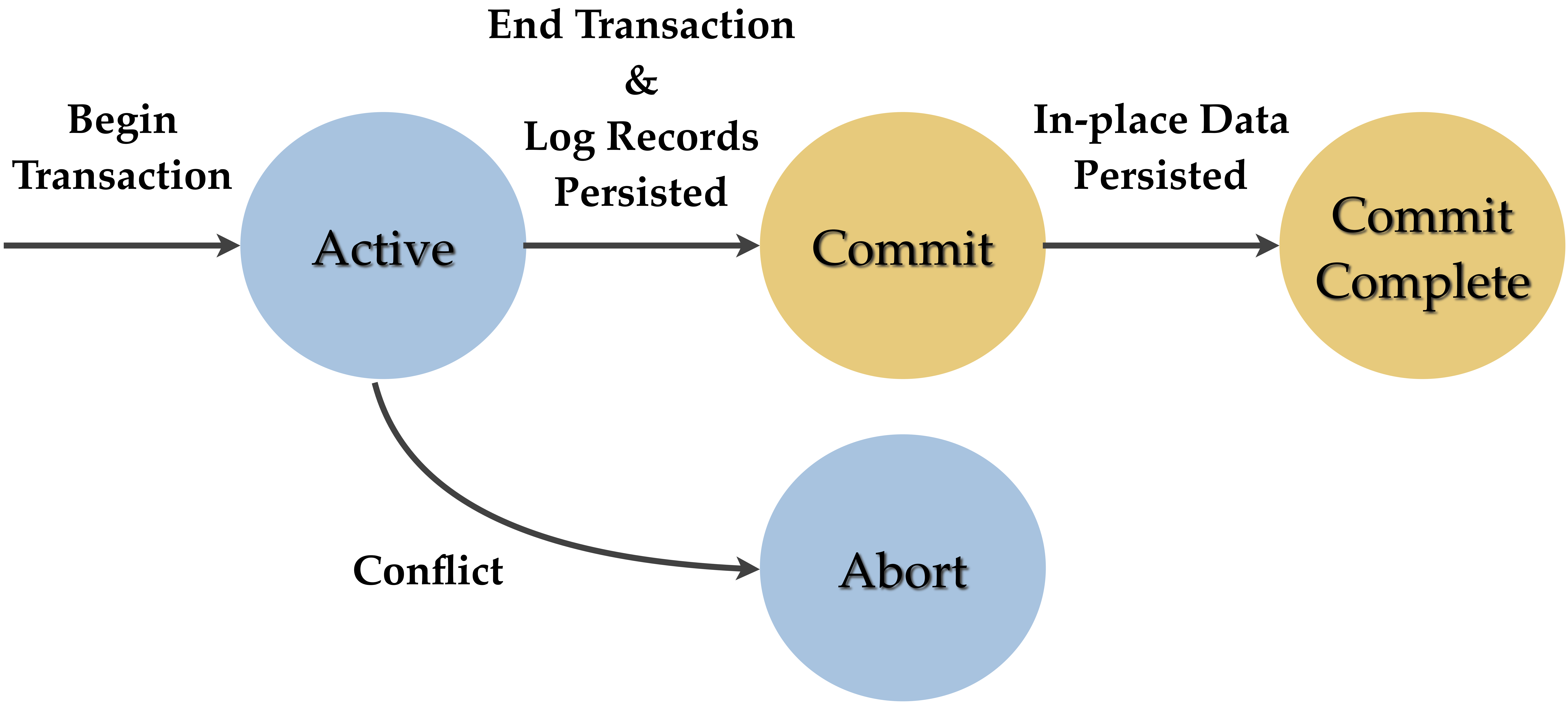
End Transaction & Log Records Persisted

Commit

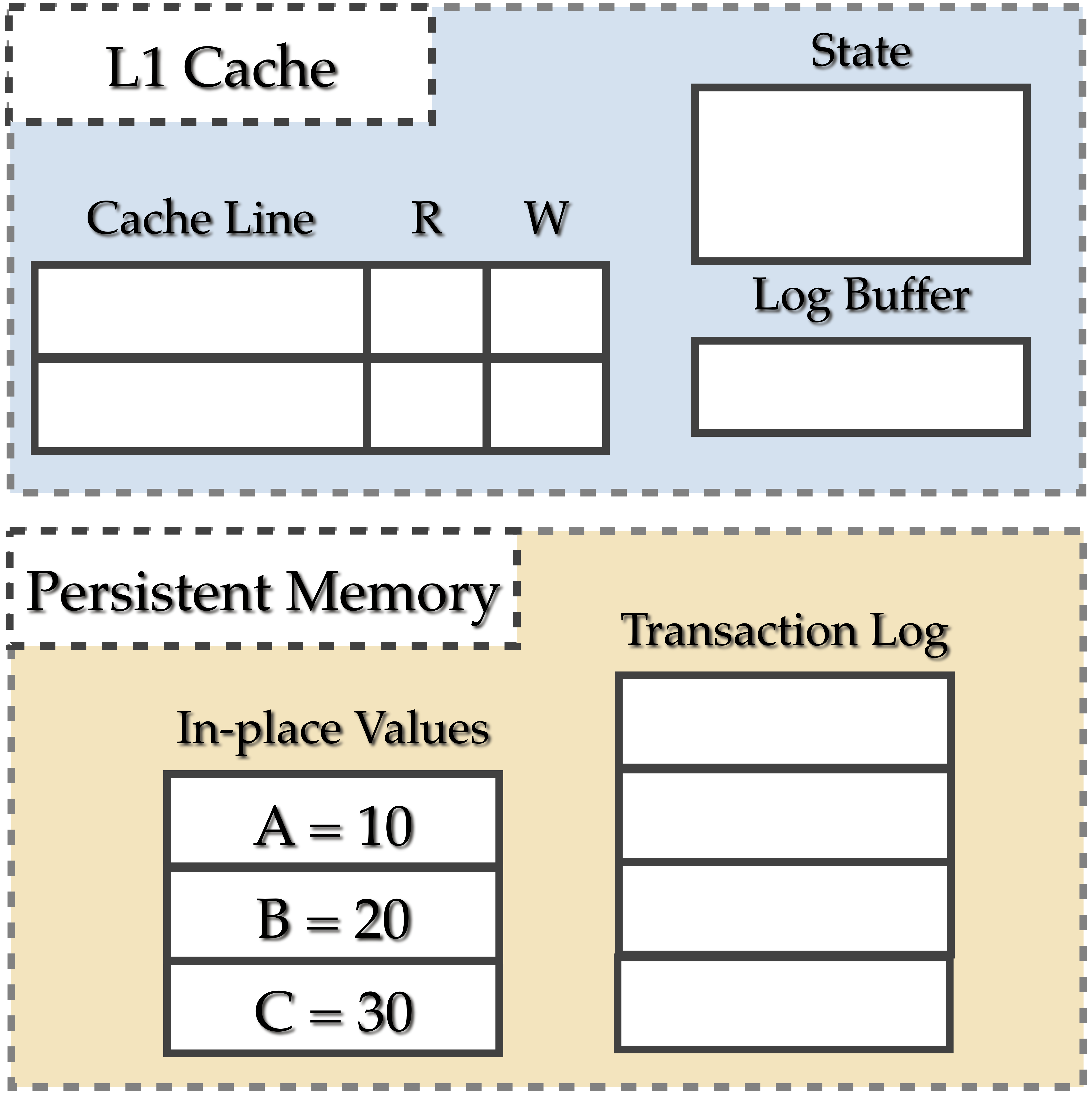
DHTM: Transaction States



DHTM: Transaction States

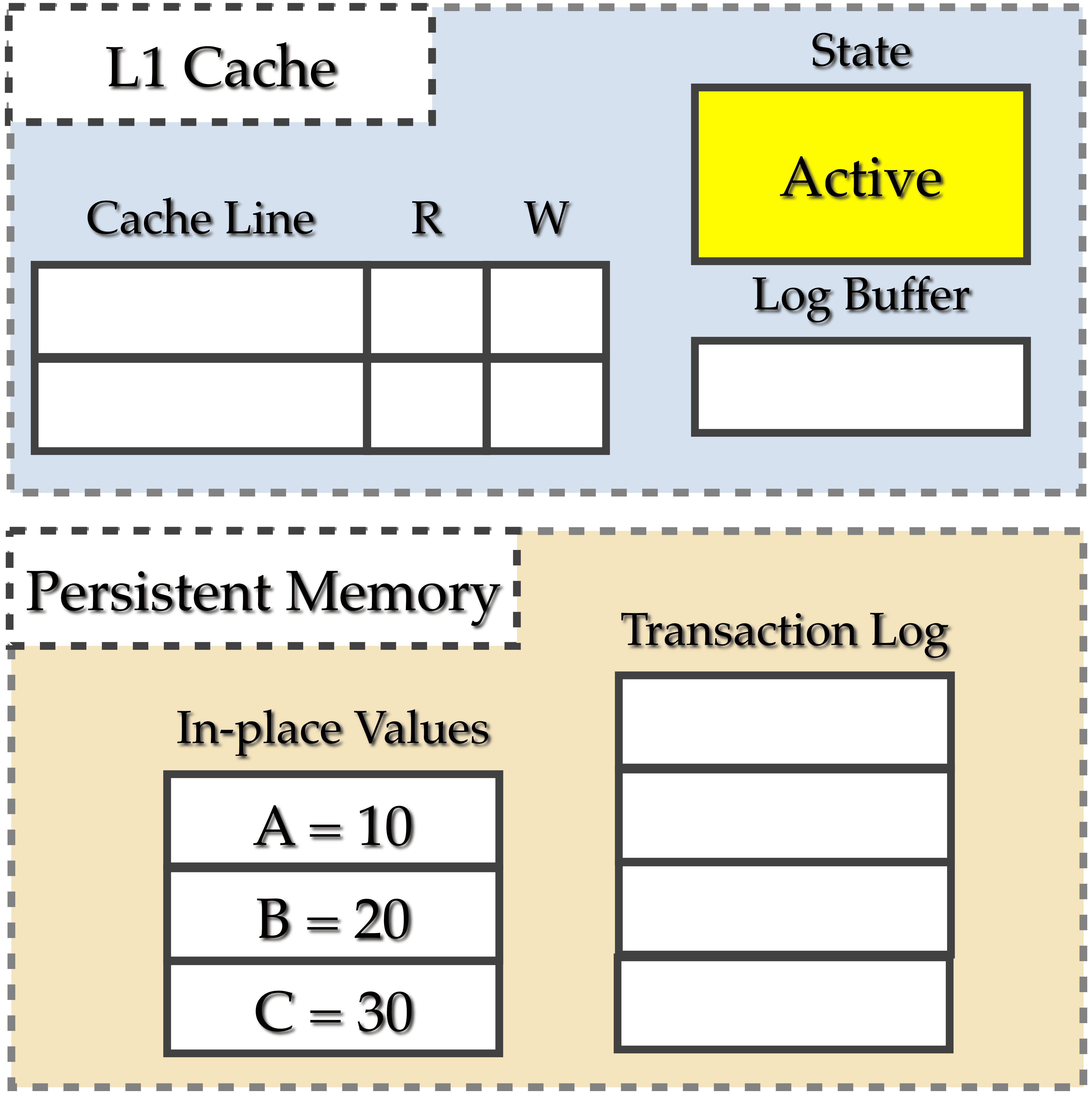


DHTM: Commit Example



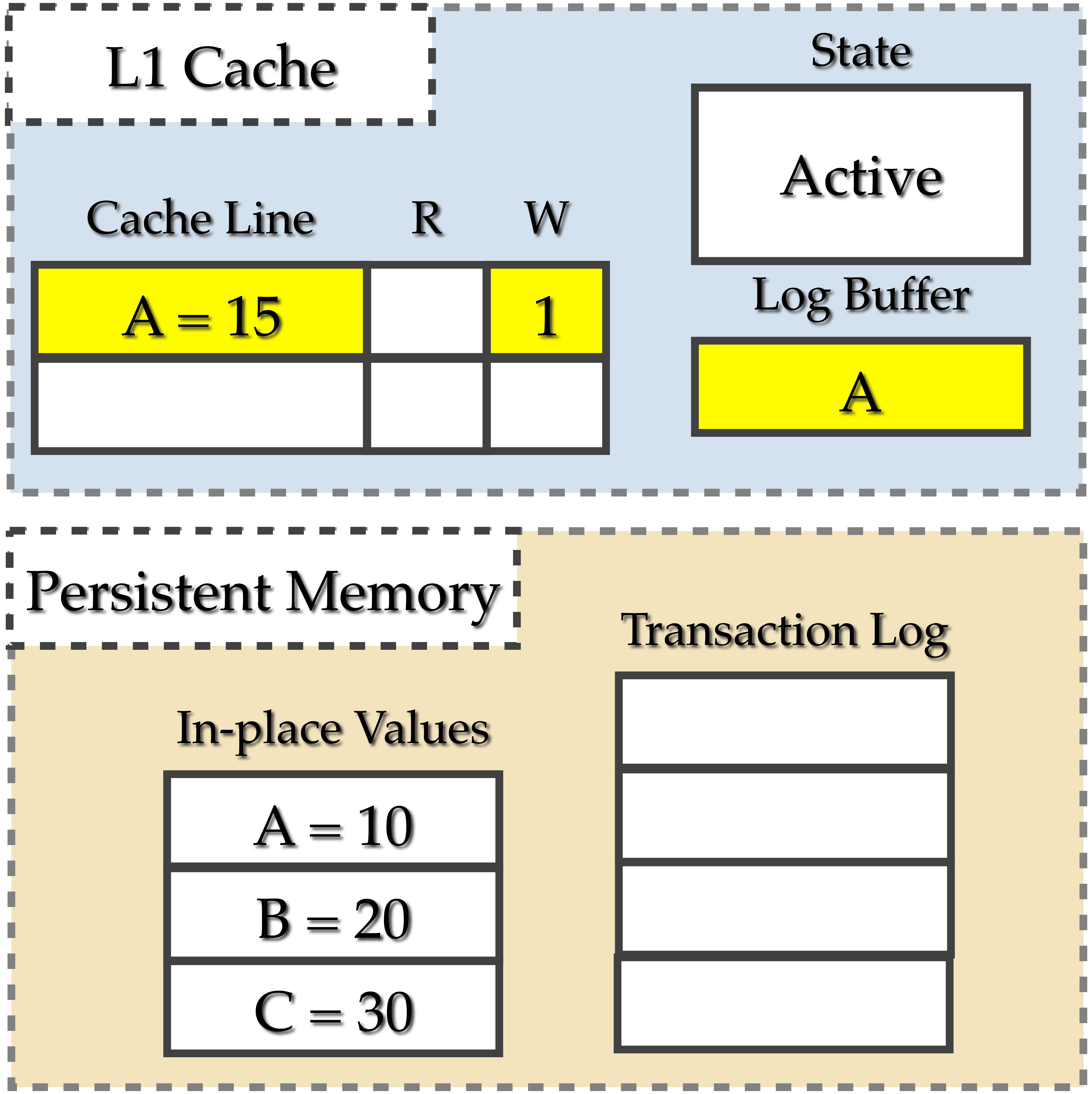
```
Begin_Transaction
          Write (A=15)
          Read  (B)
          Write (B=25)
End_Transaction
```

DHTM: Commit Example



```
Begin_Transaction
        Write (A=15)
        Read  (B)
        Write (B=25)
End_Transaction
```

DHTM: Commit Example



L1 Cache

Cache Line	R	W
A = 15		1

State

Active

Log Buffer

A

Persistent Memory

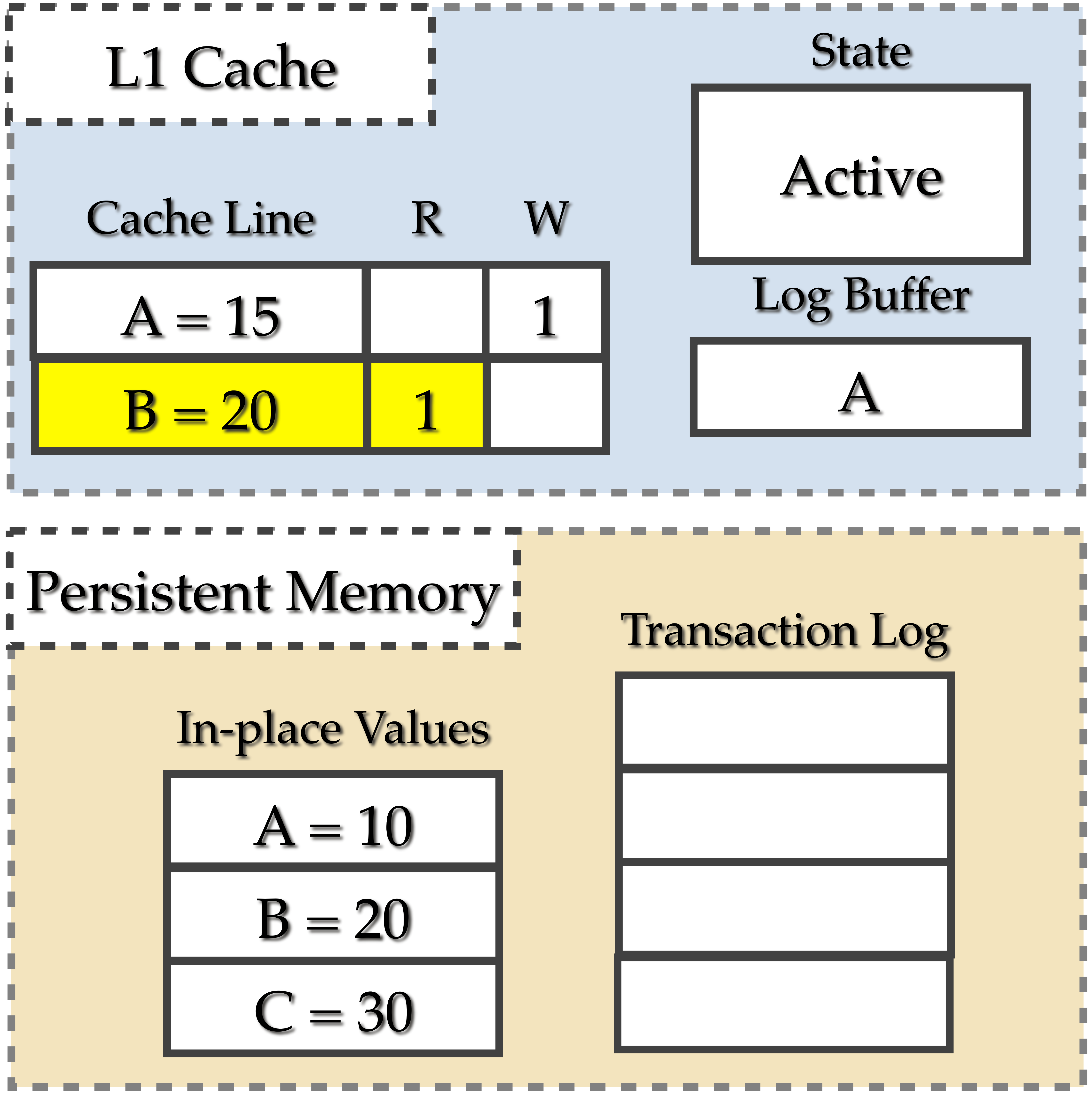
In-place Values

A = 10
B = 20
C = 30

Transaction Log

```
Begin_Transaction
        Write (A=15)
        Read  (B)
        Write (B=25)
End_Transaction
```

DHTM: Commit Example



L1 Cache

Cache Line	R	W
A = 15		1
B = 20	1	

State

Active

Log Buffer

A

Persistent Memory

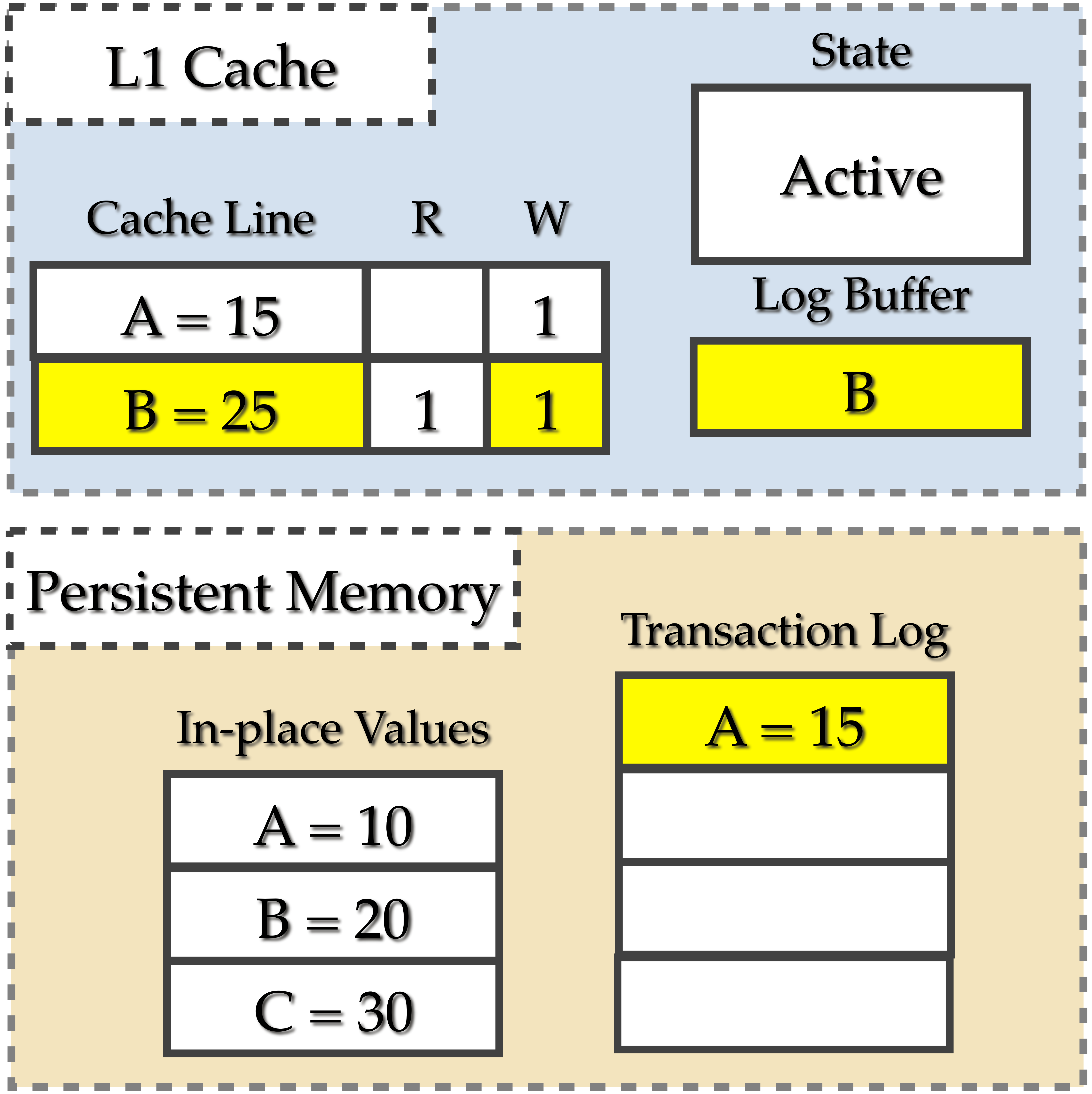
In-place Values

A = 10
B = 20
C = 30

Transaction Log

```
Begin_Transaction
     Write (A=15)
     Read  (B)
     Write (B=25)
End_Transaction
```

DHTM: Commit Example



L1 Cache

Cache Line	R	W
A = 15		1
B = 25	1	1

State

Active

Log Buffer

B

Persistent Memory

In-place Values

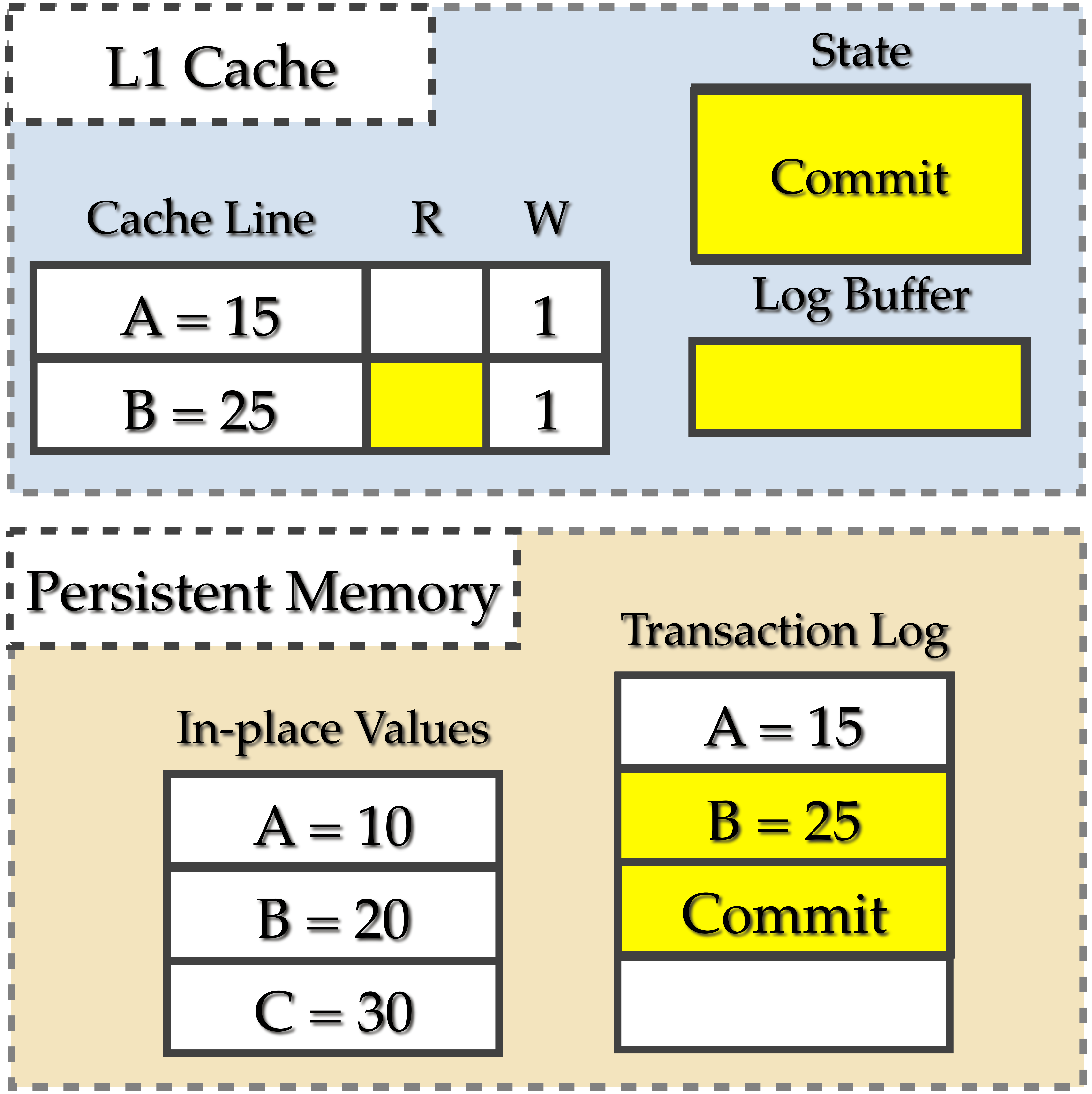
A = 10
B = 20
C = 30

Transaction Log

A = 15

```
Begin_Transaction
      Write (A=15)
      Read  (B)
      Write (B=25)
End_Transaction
```

DHTM: Commit Example



L1 Cache

Cache Line	R	W
A = 15		1
B = 25		1

State

Commit

Log Buffer

Persistent Memory

In-place Values

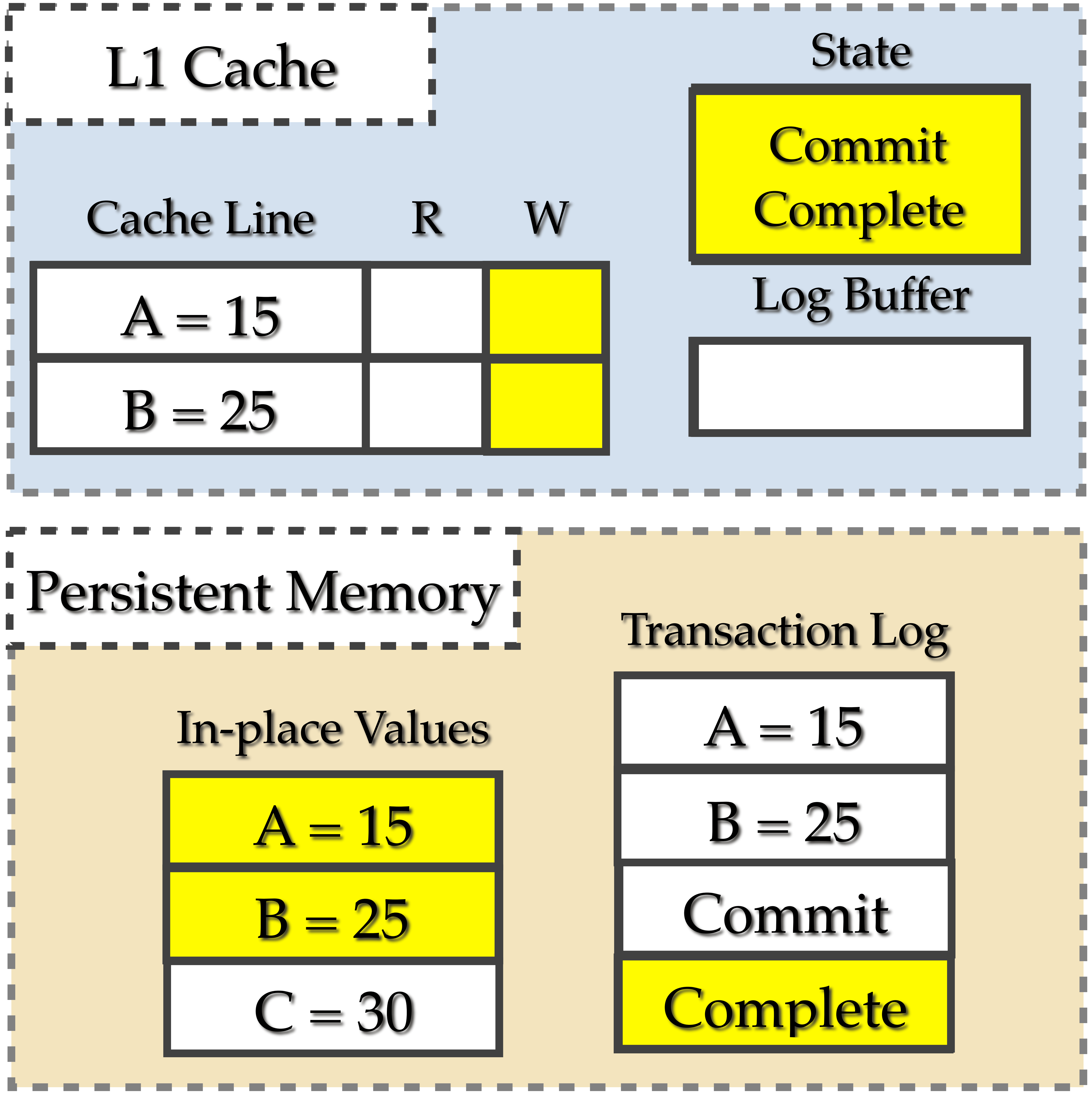
A = 10
B = 20
C = 30

Transaction Log

A = 15
B = 25
Commit

```
Begin_Transaction
     Write (A=15)
     Read  (B)
     Write (B=25)
End_Transaction
```

DHTM: Commit Example



L1 Cache

Cache Line	R	W
A = 15		
B = 25		

State

Commit Complete

Log Buffer

Persistent Memory

In-place Values

A = 15

B = 25

C = 30

Transaction Log

A = 15

B = 25

Commit

Complete

```
Begin_Transaction
     Write (A=15)
     Read  (B)
     Write (B=25)
End_Transaction
```

DHTM: Supporting Overflow

DHTM: Supporting Overflow

- **Problems with Overflow:**

DHTM: Supporting Overflow

- **Problems with Overflow:**
  - **Version Management**:
    - global operation on write-set on a commit/abort
    - overhead infeasible in larger caches (beyond L1)

DHTM: Supporting Overflow

- **Problems with Overflow:**
  - **Version Management**:
    - global operation on write-set on a commit/abort
    - overhead infeasible in larger caches (beyond L1)
  - **Conflict Detection**:
    - additional metadata to detect conflicts
    - increased complexity due to NACK based protocols

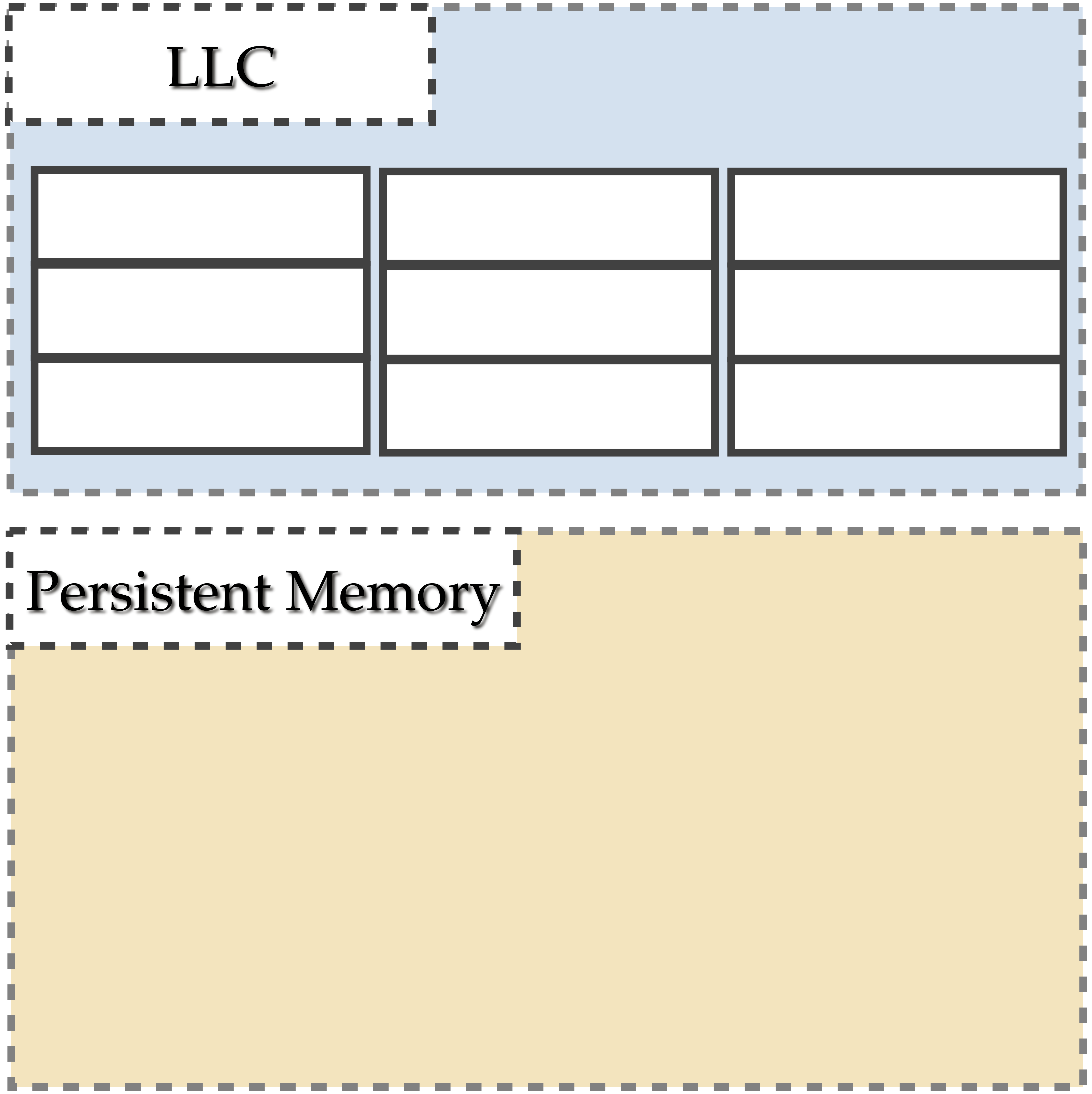
DHTM: Supporting Overflow

DHTM: Supporting Overflow

- Solution

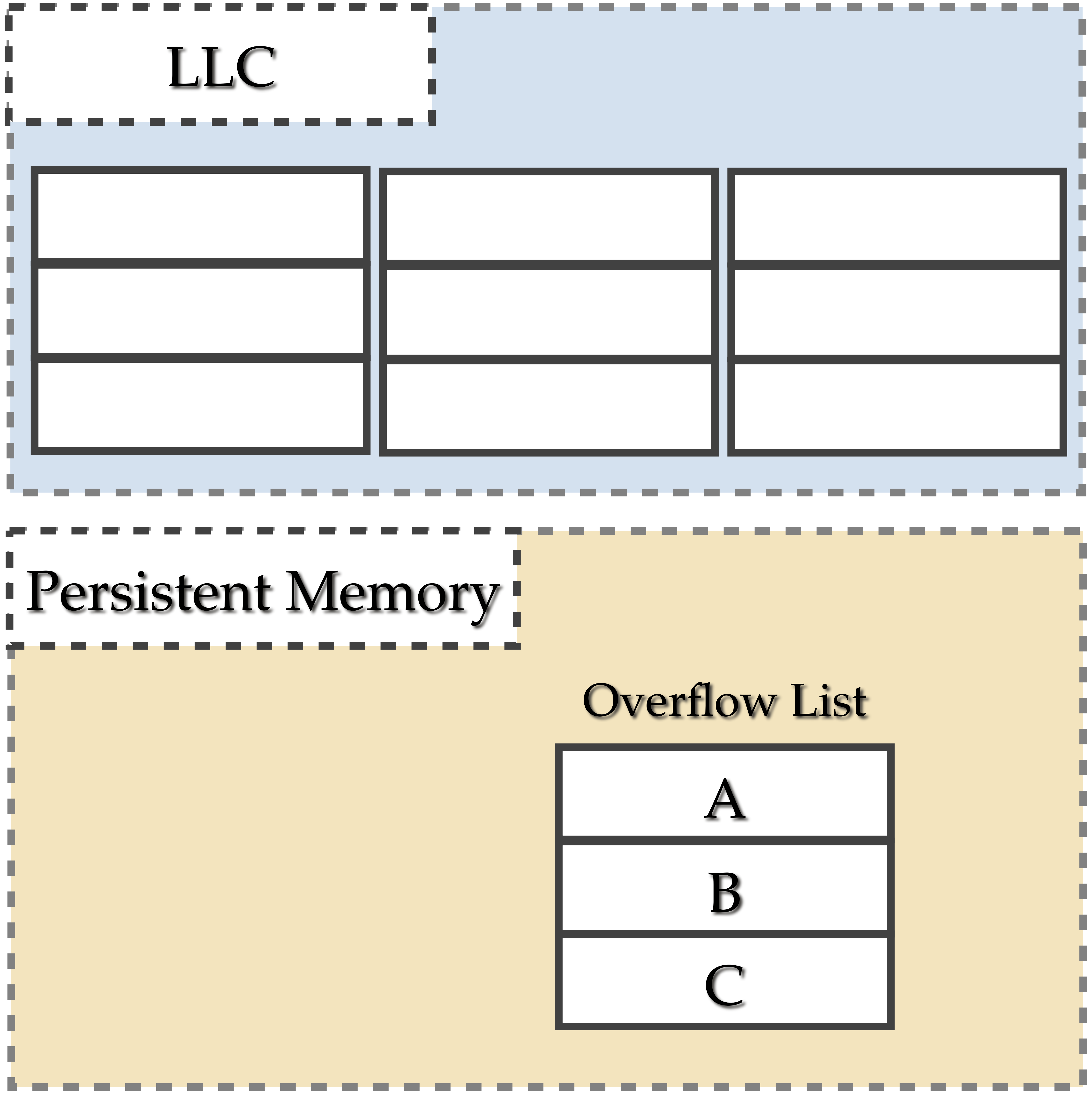
DHTM: Supporting Overflow

- **Solution**
  - **Version Management**:
    - Overflow List



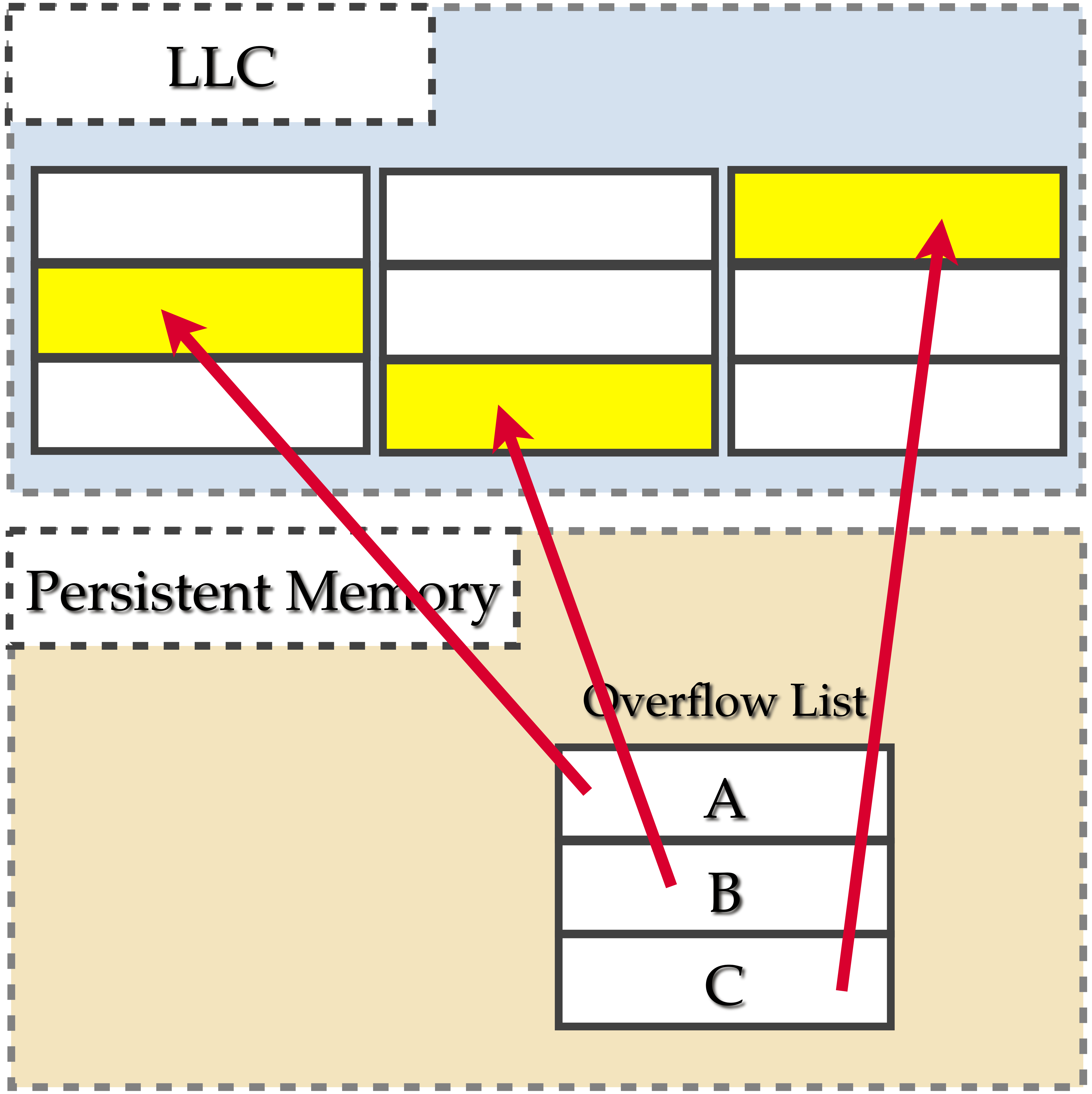
DHTM: Supporting Overflow

- **Solution**
  - **Version Management**:
    - Overflow List



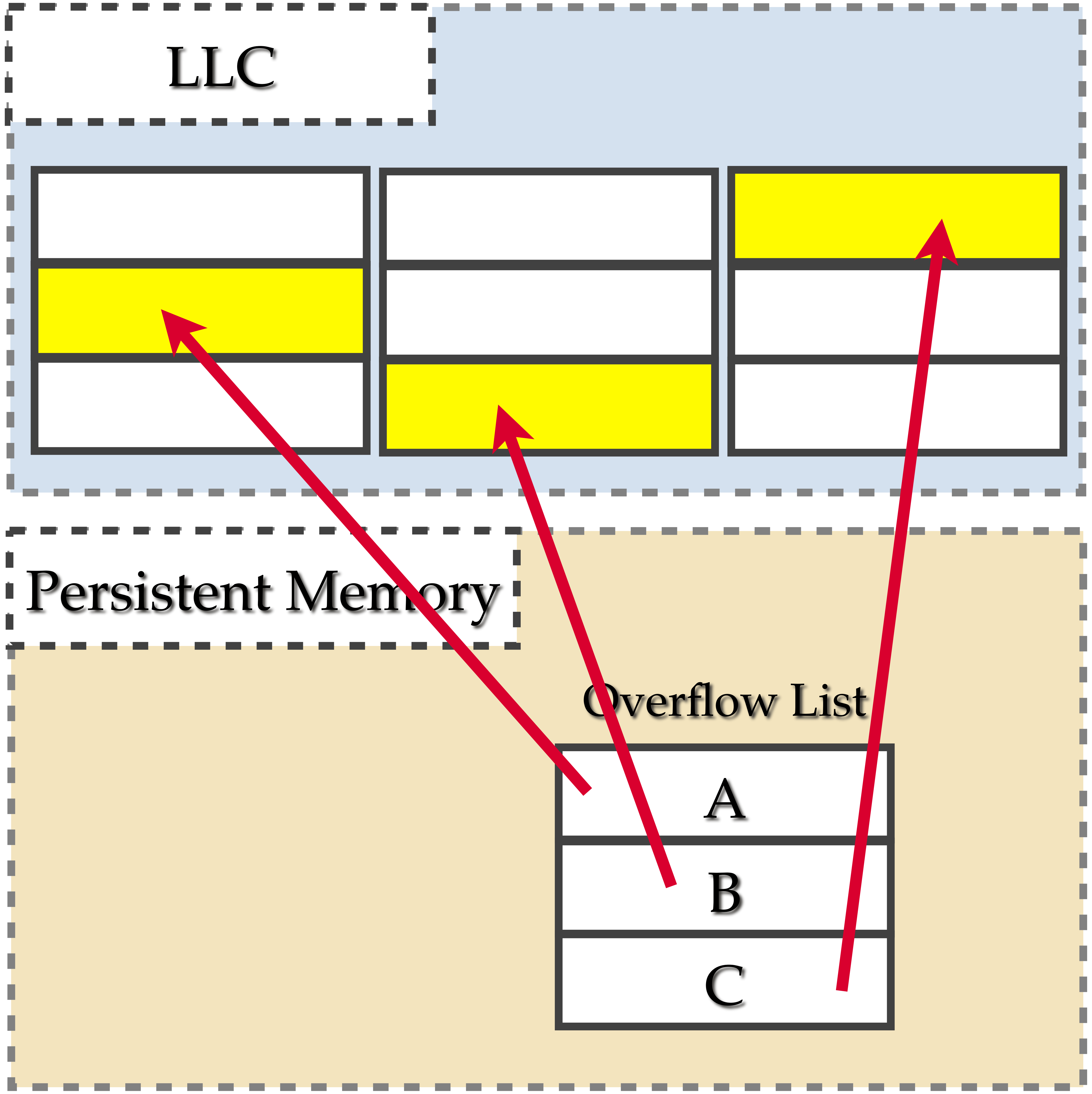
DHTM: Supporting Overflow

- **Solution**
  - **Version Management:**
    - Overflow List



DHTM: Supporting Overflow

- **Solution**
  - **Version Management**:
    - Overflow List
  - **Conflict Detection**:
    - maintain sticky state on overflow (similar to LogTM)
    - avoid NACK by restricting overflow to LLC



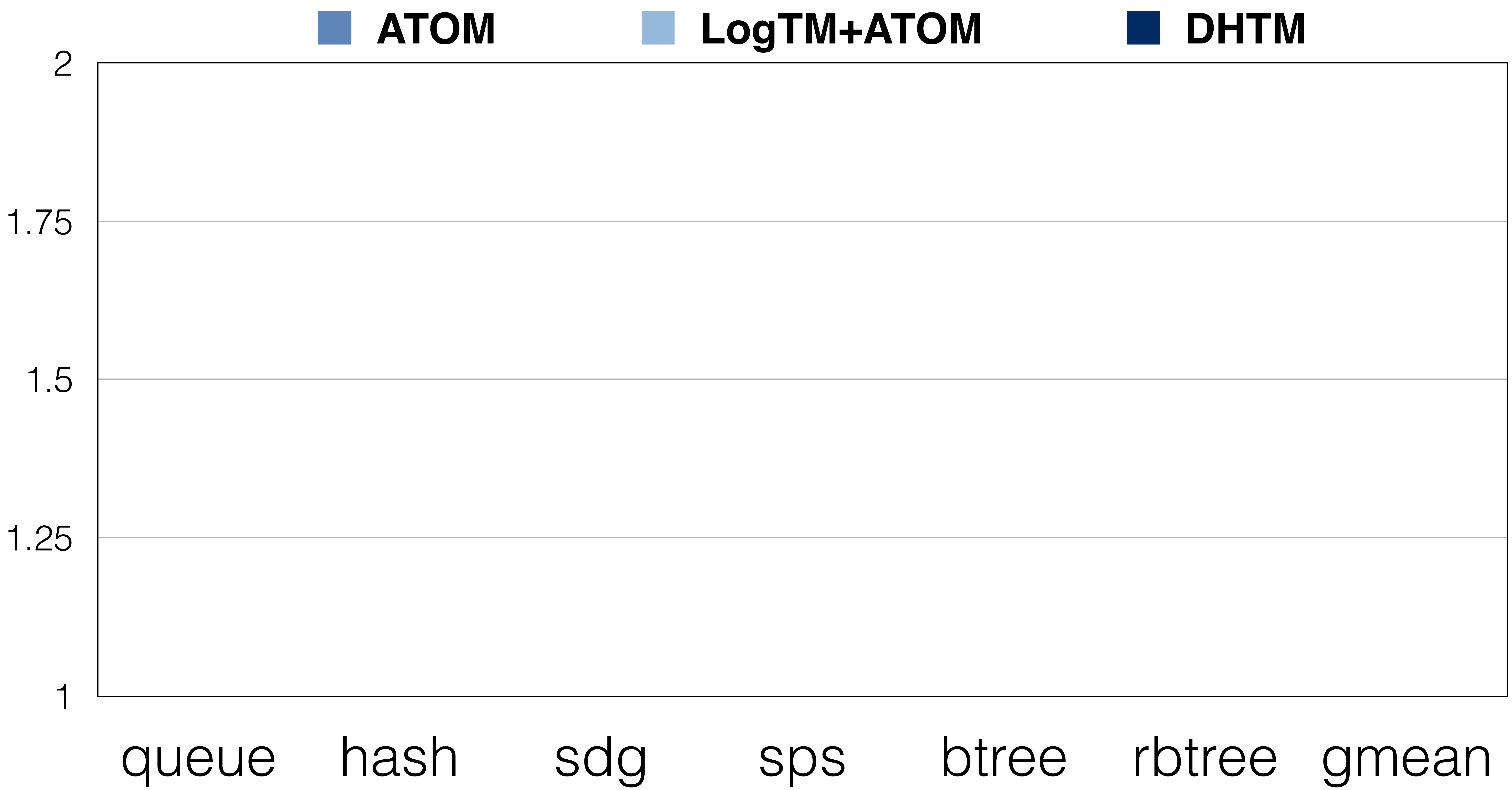
Evaluation

	Atomic Visibility	Atomic Durability
ATOM	Locks	Hardware Undo Log
LogTM+ATOM	HTM (LogTM)	Hardware Undo Log
DHTM	HTM	Hardware Redo Log (Log Buffer)

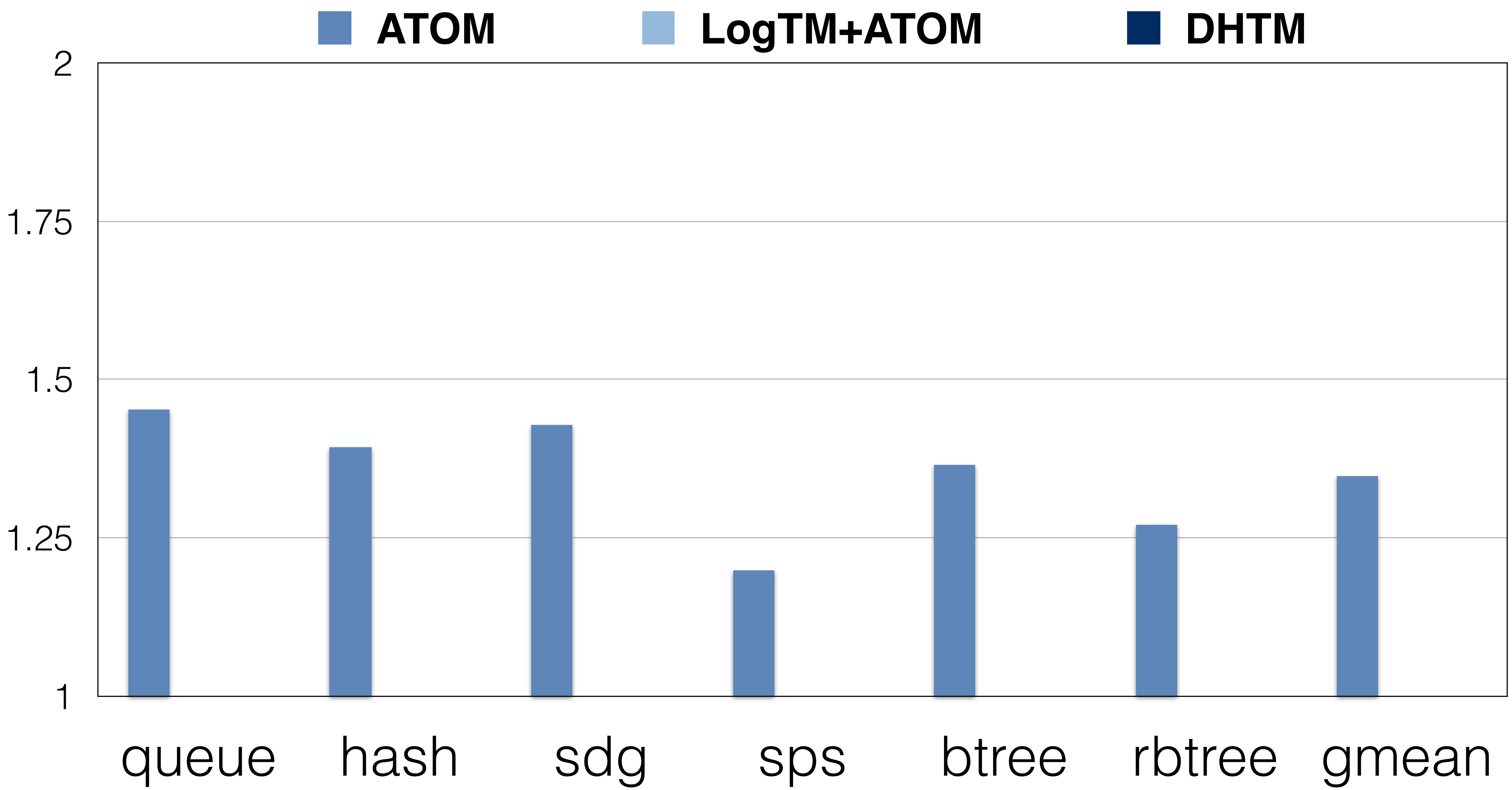
- **System Configuration**
  - We evaluate an 8-core machine with a 2-level cache hierarchy
  - HTM's implement (first) writer wins conflict resolution policy

Evaluation

Evaluation

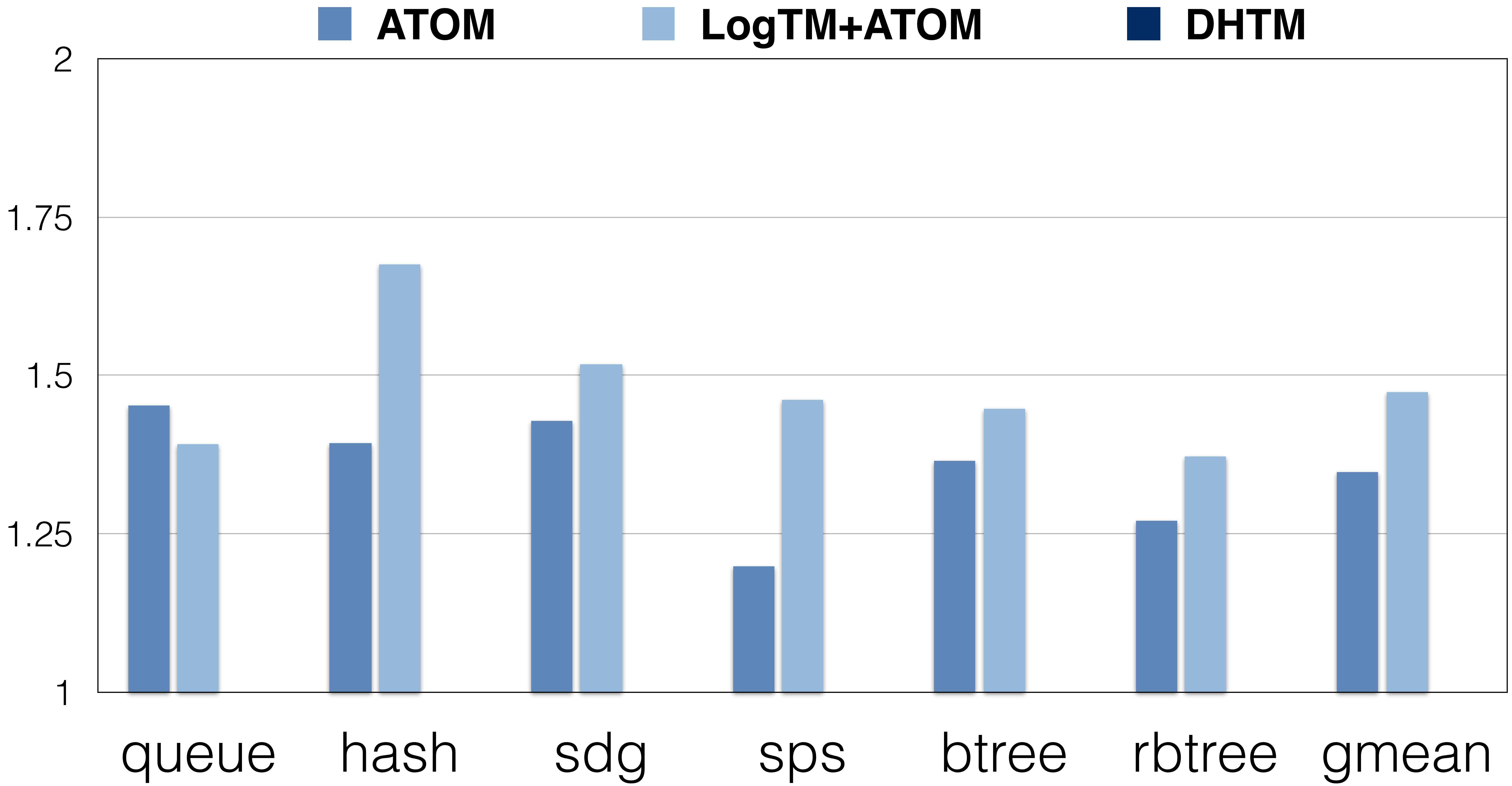

■ ATOM
■ LogTM+ATOM
■ DHTM
2
1.75
1.5
1.25
1
queue
hash
sdg
sps
btree
rbtree
gmean

Evaluation

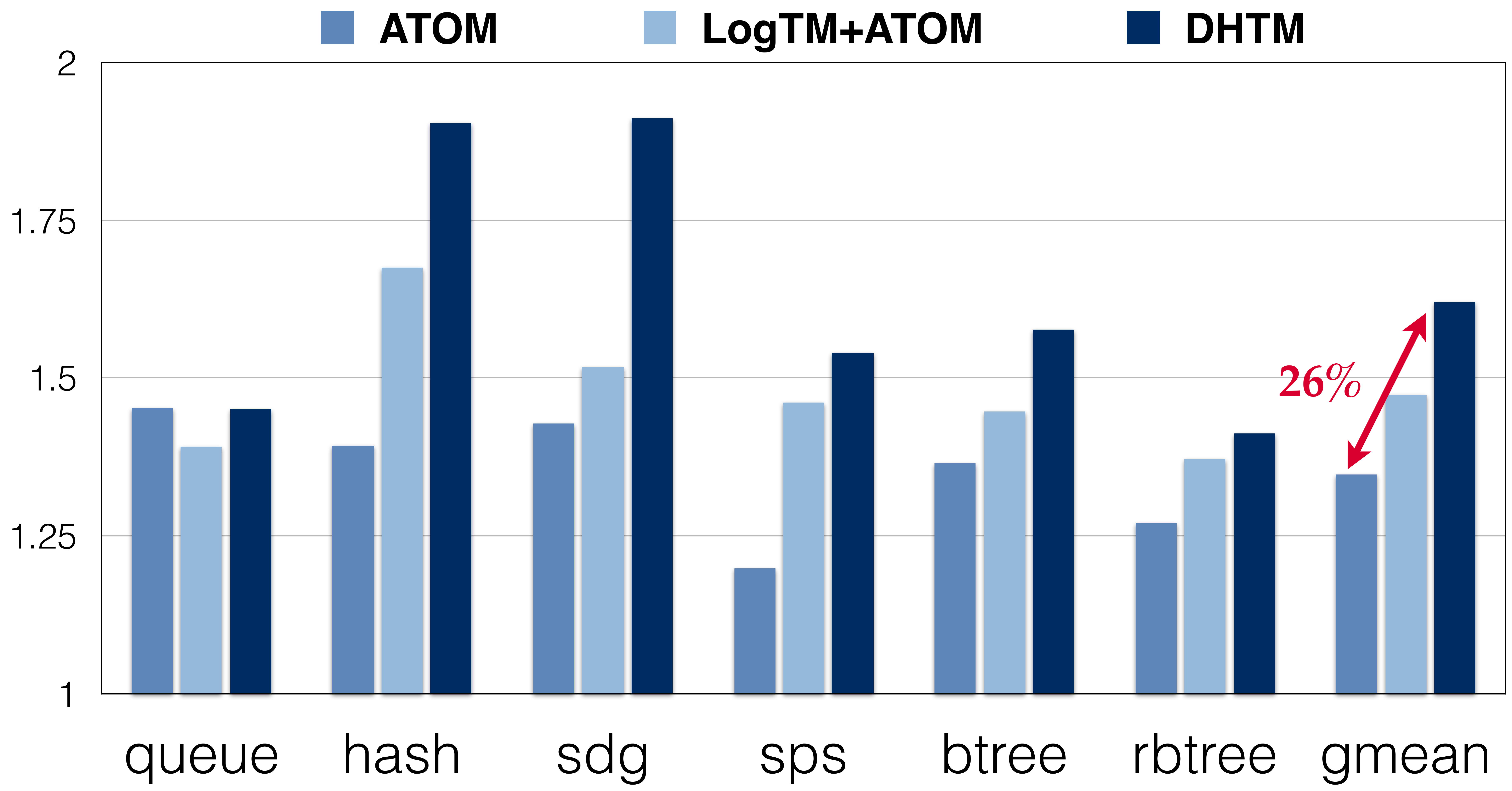


ATOM
LogTM+ATOM
DHTM
2
1.75
1.5
1.25
1
queue
hash
sdg
sps
btree
rbtree
gmean

Evaluation

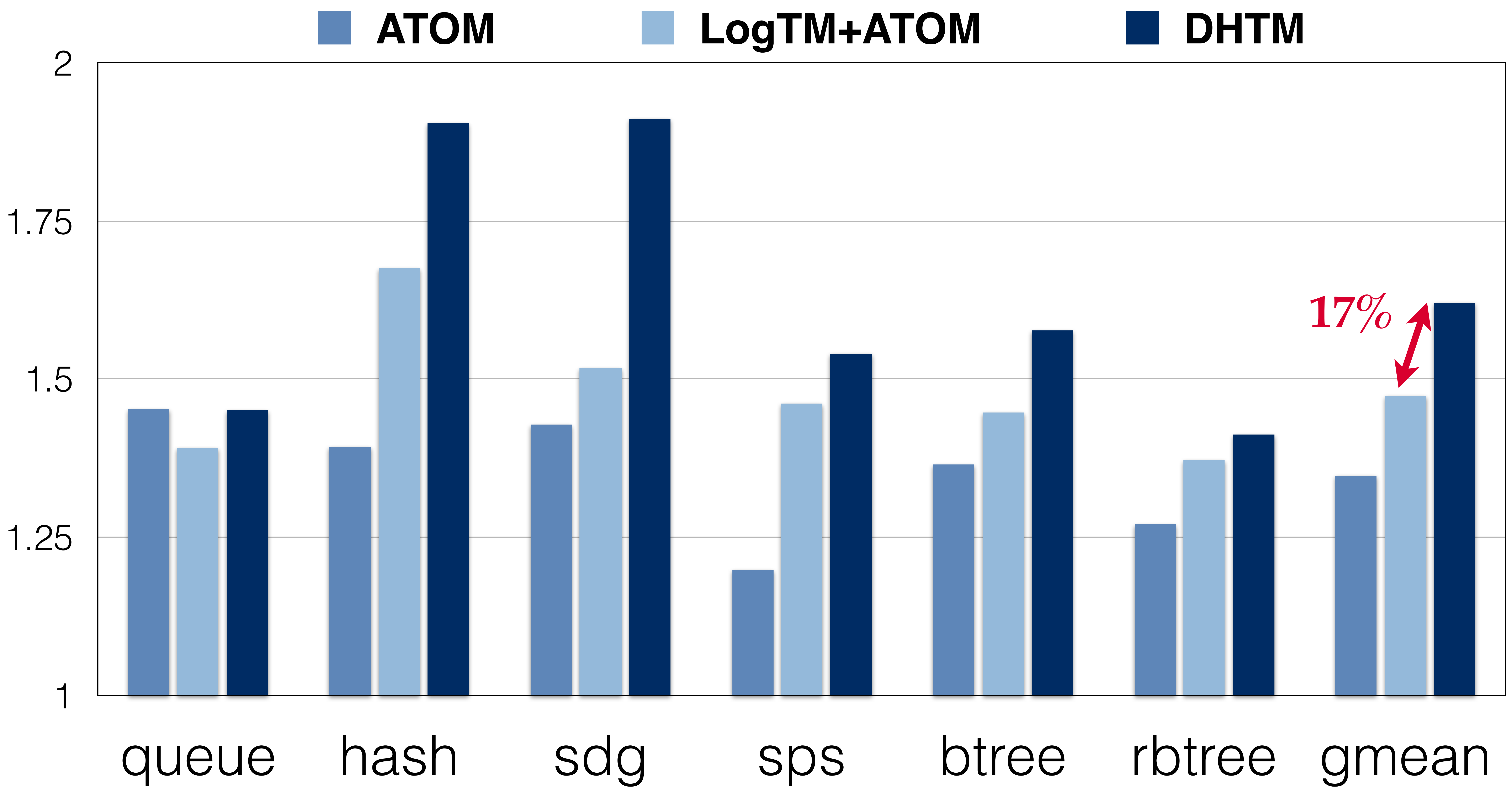

ATOM
LogTM+ATOM
DHTM
2
1.75
1.5
1.25
1
queue
hash
sdg
sps
btree
rbtree
gmean

Evaluation



ATOM
LogTM+ATOM
DHTM
2
1.75
1.5
1.25
1
queue
hash
sdg
sps
btree
rbtree
gmean
26%

Evaluation


ATOM
LogTM+ATOM
DHTM
2
1.75
1.5
1.25
1
queue
hash
sdg
sps
btree
rbtree
gmean
17%

Conclusion

- Persistent memory systems require crash consistency
- ACID Transactions: widely used crash consistency mechanism
- DHTM: ACID transactions in hardware
    - Atomic Visibility: commercial HTM
    - Atomic Durability: bandwidth optimized hardware redo log
    - Leverage hardware logging to extend transaction size unto LLC

Hardware Support for ACID Transactions in Persistent Memory Systems

Arpit Joshi

ARM Research Summit, 2018

THE UNIVERSITY of EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

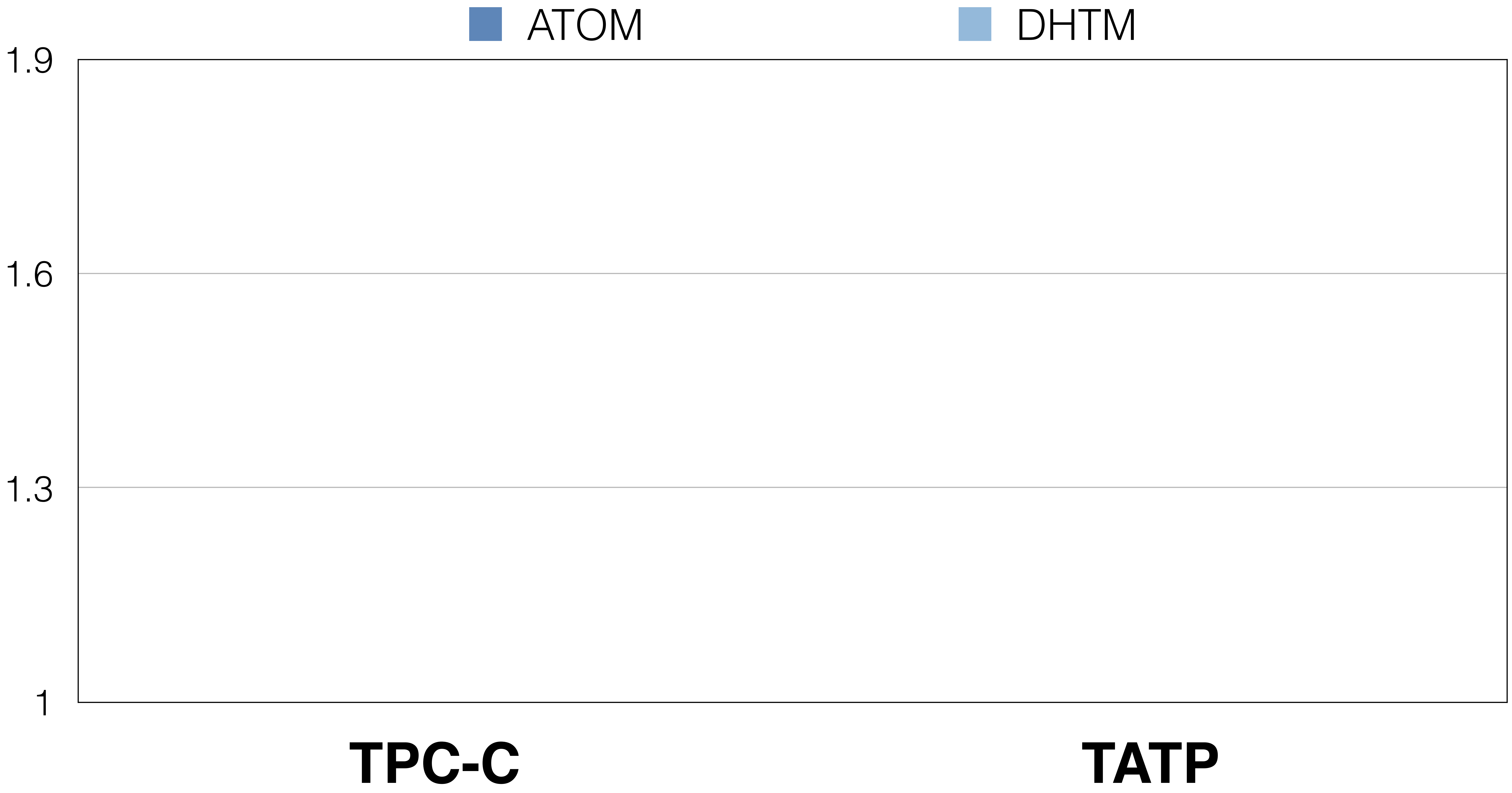
intel

Google

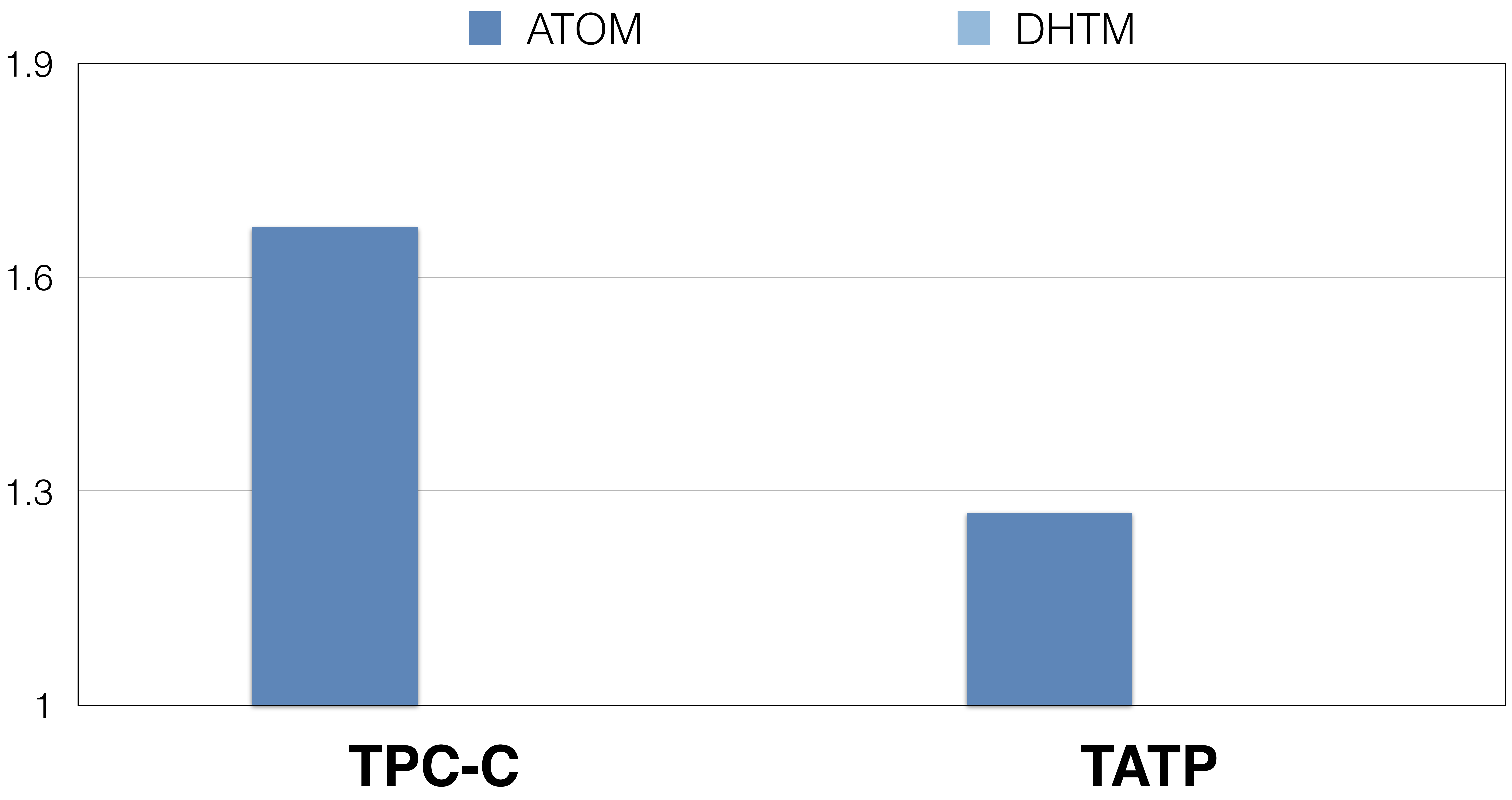
BACK UP

Evaluation

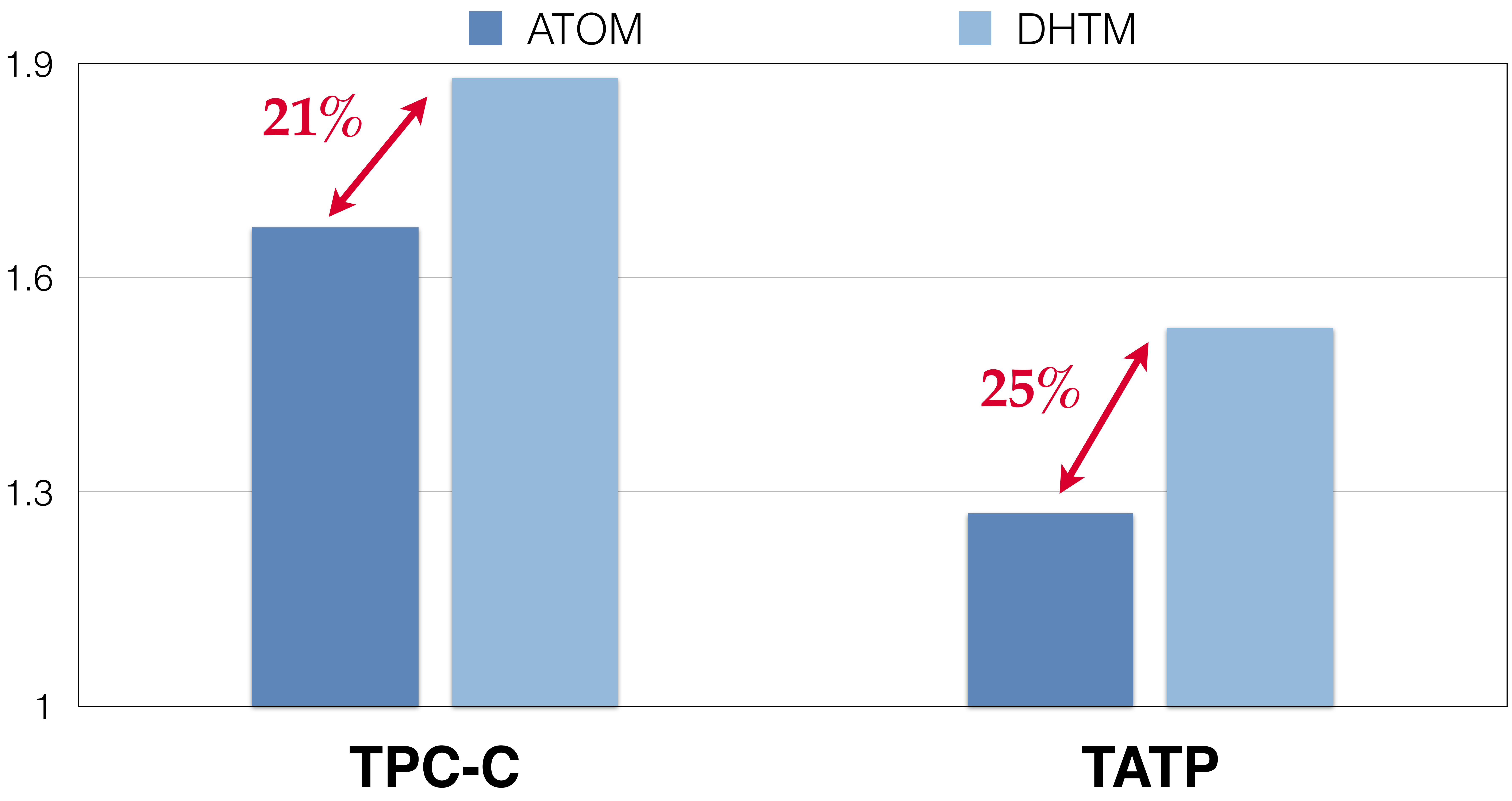
Evaluation



Evaluation



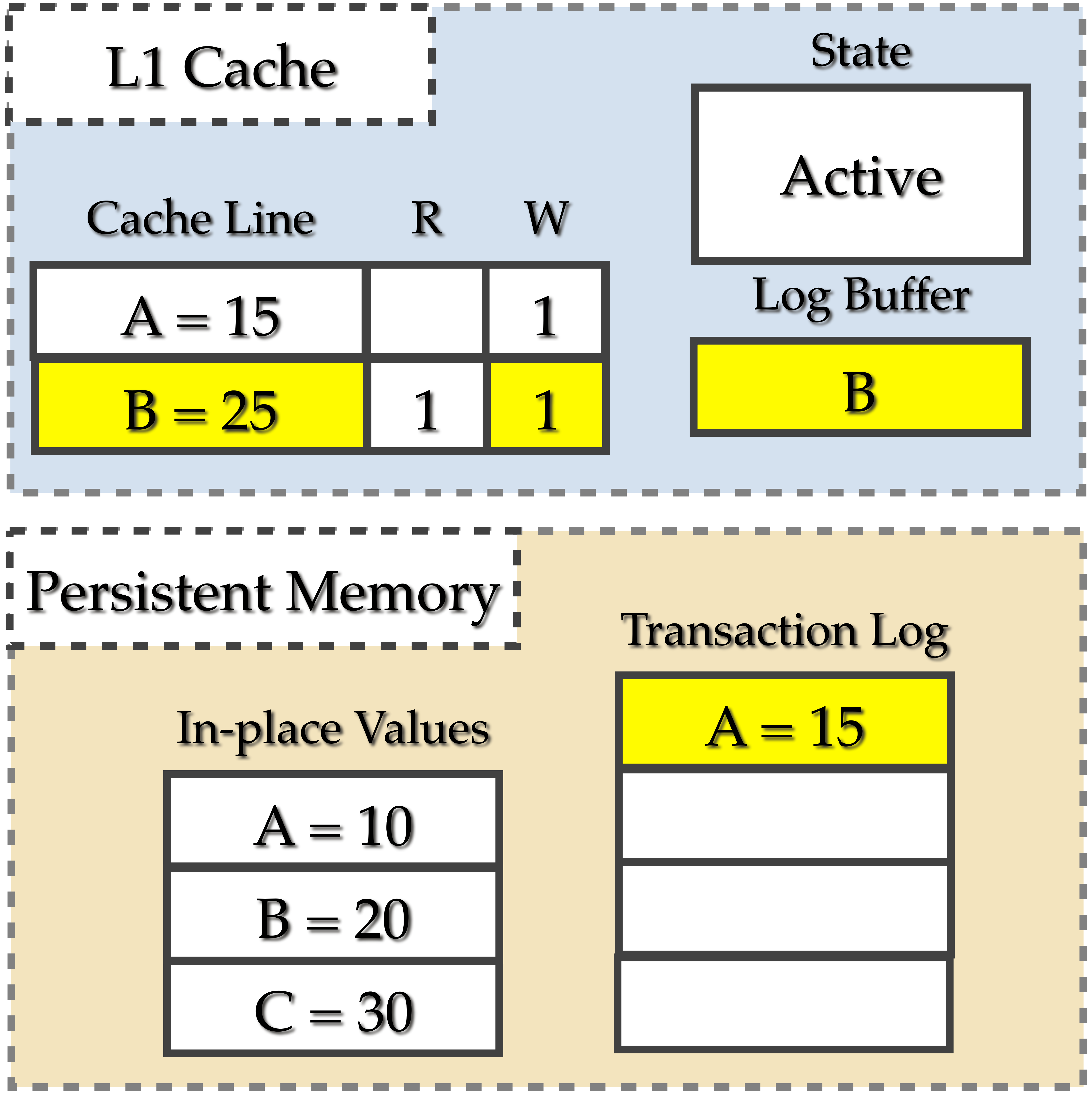
Evaluation



Unbounded HTMs

- Do not support atomic durability
- Retrofitting atomic durability to existing mechanisms not optimal (eg. LogTM+ATOM)
- Complex NACK based coherence protocols for conflict detection

DHTM: Abort Example



L1 Cache

Cache Line	R	W
A = 15		1
B = 25	1	1

State

Active

Log Buffer

B

Persistent Memory

In-place Values

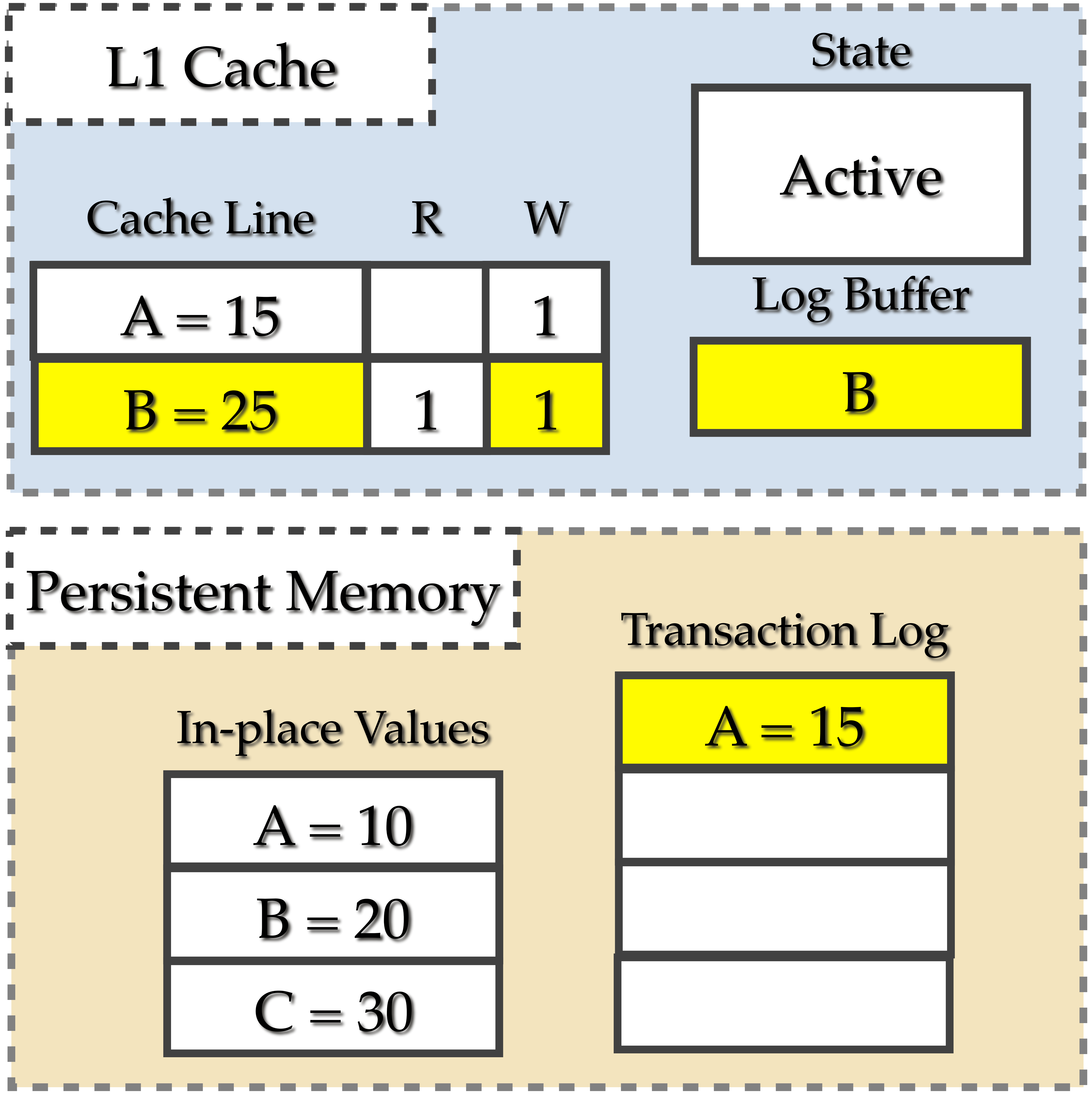
A = 10
B = 20
C = 30

Transaction Log

A = 15

```
Begin_Transaction
     Write (A=15)
     Read  (B)
     Write (B=25)
End_Transaction
```

DHTM: Abort Example



L1 Cache

Cache Line	R	W
A = 15		1
B = 25	1	1

State

Active

Log Buffer

B

Persistent Memory

In-place Values

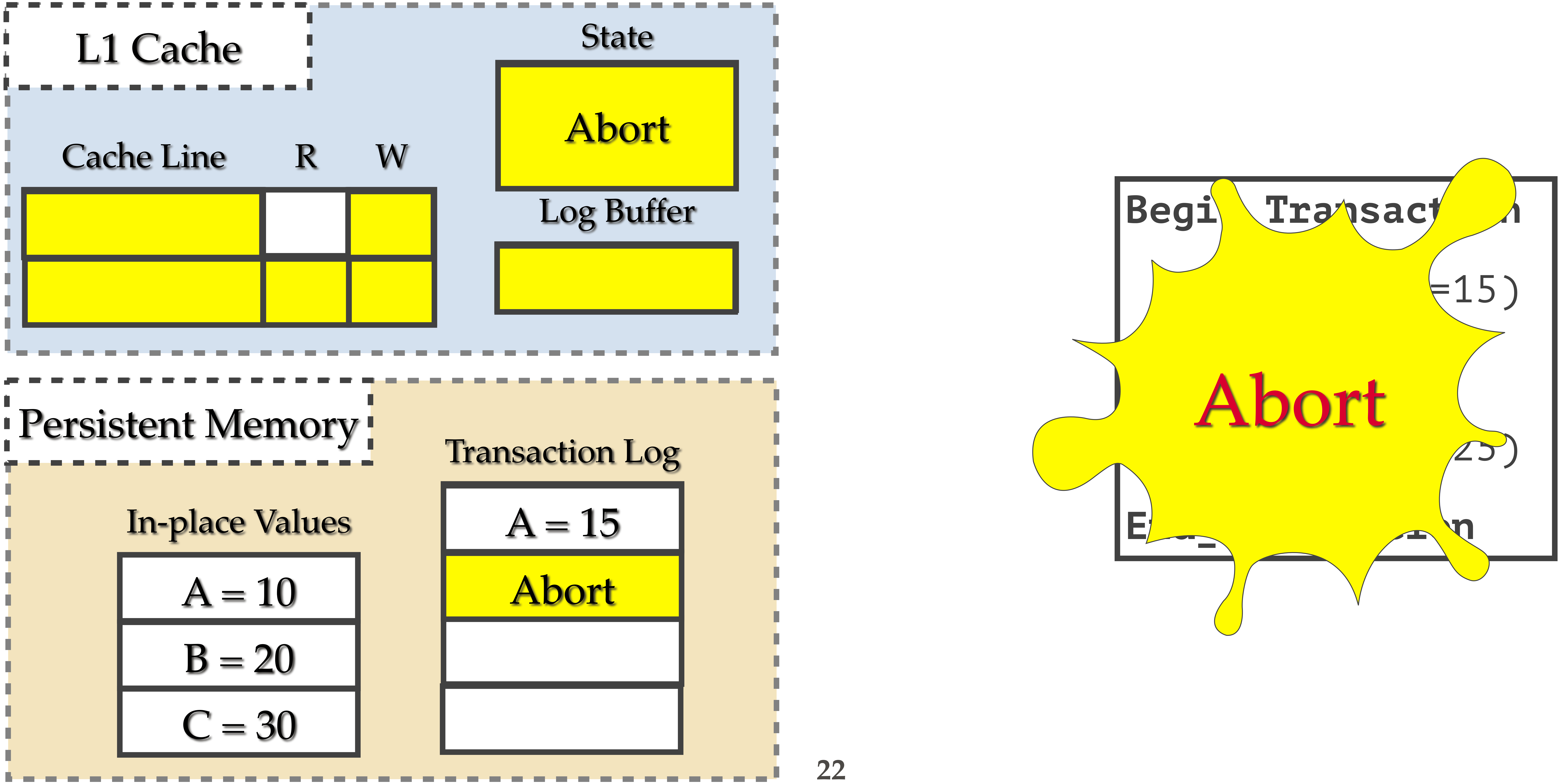
A = 10
B = 20
C = 30

Transaction Log

A = 15

```
Begin_Transaction
     Write (A=15)
     Read  (B)
     Write (B=25)
End_Transaction
```

DHTM: Abort Example



L1 Cache

State

Abort

Cache Line | R | W

Log Buffer

Persistent Memory

Transaction Log

In-place Values

A = 10

B = 20

C = 30

A = 15

Abort

```
Begi   Transact  n
              =15)
               25)
E               n
```

Abort

Hardware Support for ACID Transactions in Persistent Memory Systems

Arpit Joshi

ARM Research Summit, 2018

THE UNIVERSITY *of* EDINBURGH informatics

icsa | Institute for Computing Systems Architecture

intel®

Google