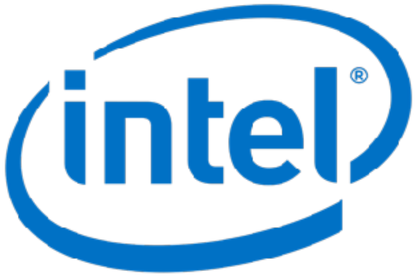
DHTM: Durable Hardware Transactional Memory

Arpit Joshi, *Vijay Nagarajan, Marcelo Cintra, Stratis Viglas*

ISCA 2018

THE UNIVERSITY *of* EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture



Google

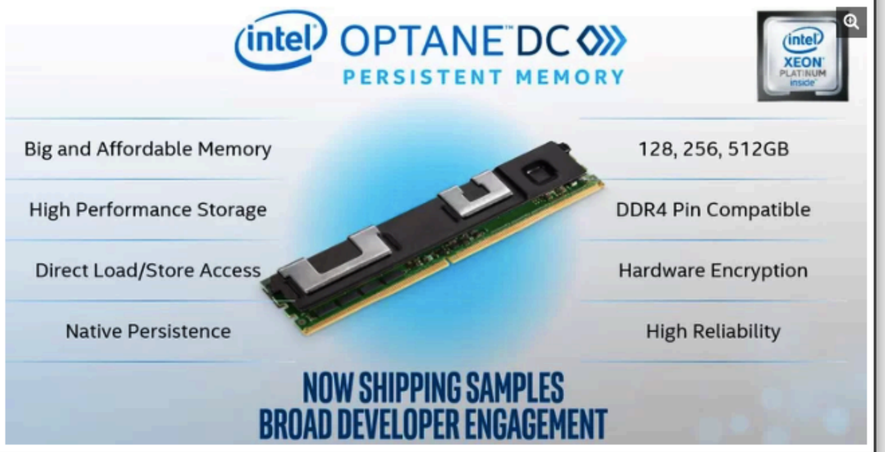
Persistent Memory is here…

Persistent Memory is here…

Intel Displays 512GB Optane DC Persistent Memory DIMMs

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

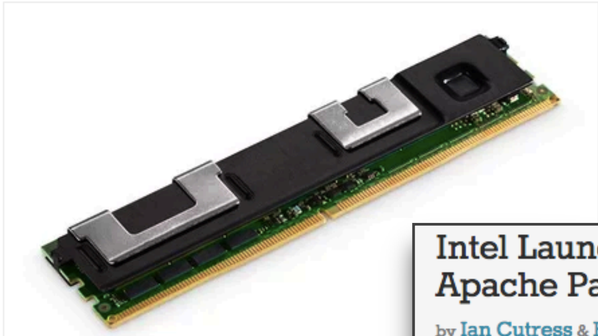
Intel held its Memory and Storage day today at its Santa Clara headquarters to announce its Optane DC Persistent Memory DIMMs. The new DIMMs slot into the DRAM interface, just like a normal stick of RAM, but come in three capacities of 128, 256, and 512GB. That's a massive capacity increase compared 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 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!

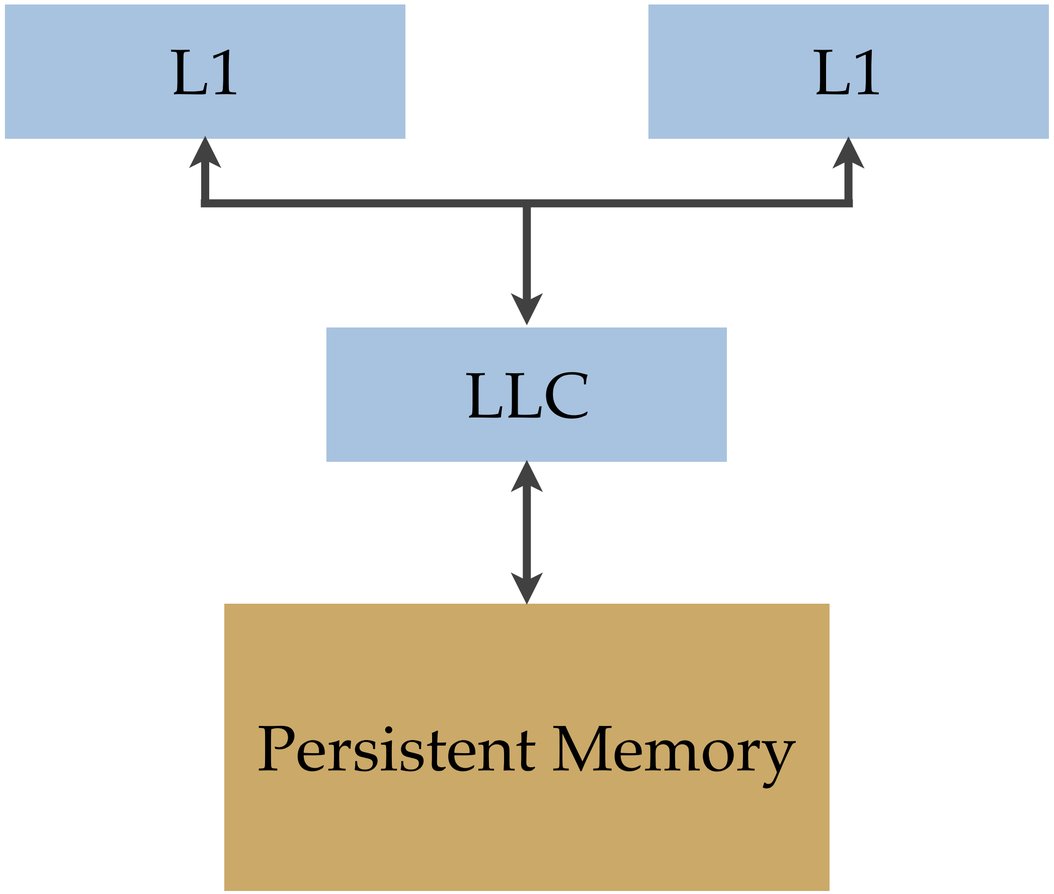
60 Comments

+ Add A Comment

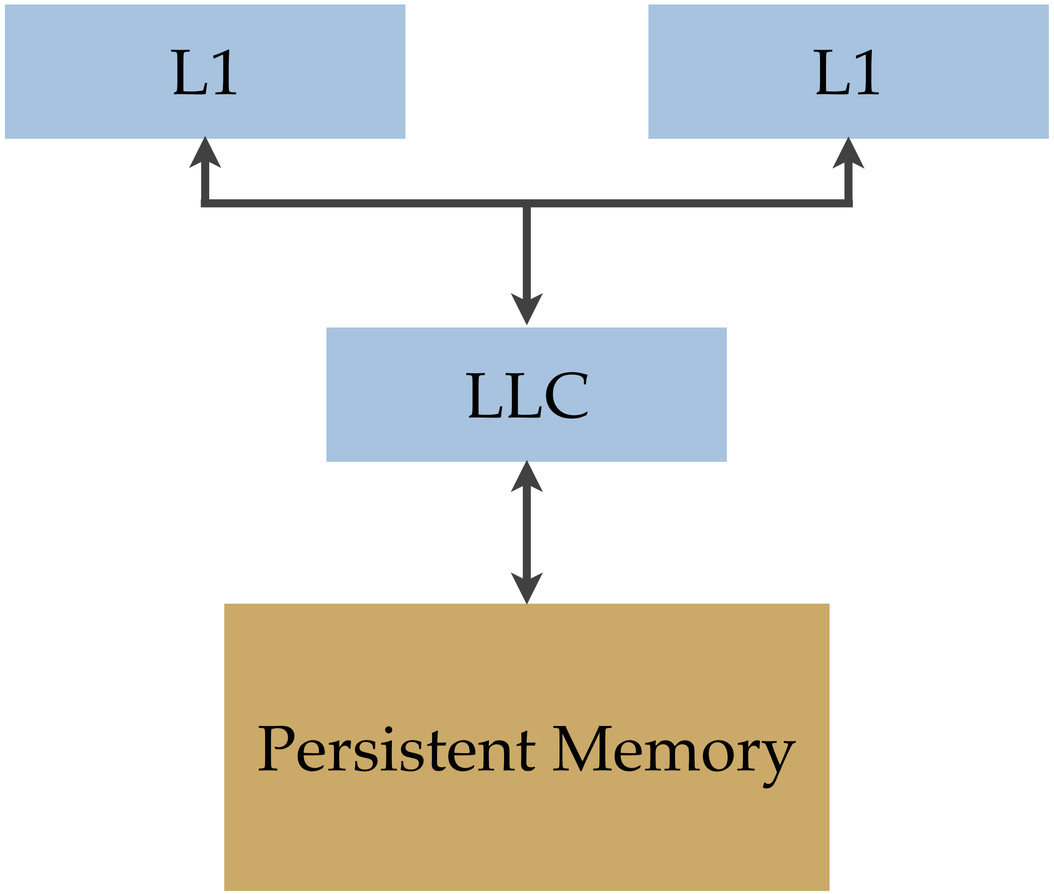
by Ian Cutress & Billy Tallis on May 30, 2018 2:15 PM EST

Posted in Intel DDR4 3D XPoint Optane NVDIMM Persistent Memory

Persistent Memory Systems

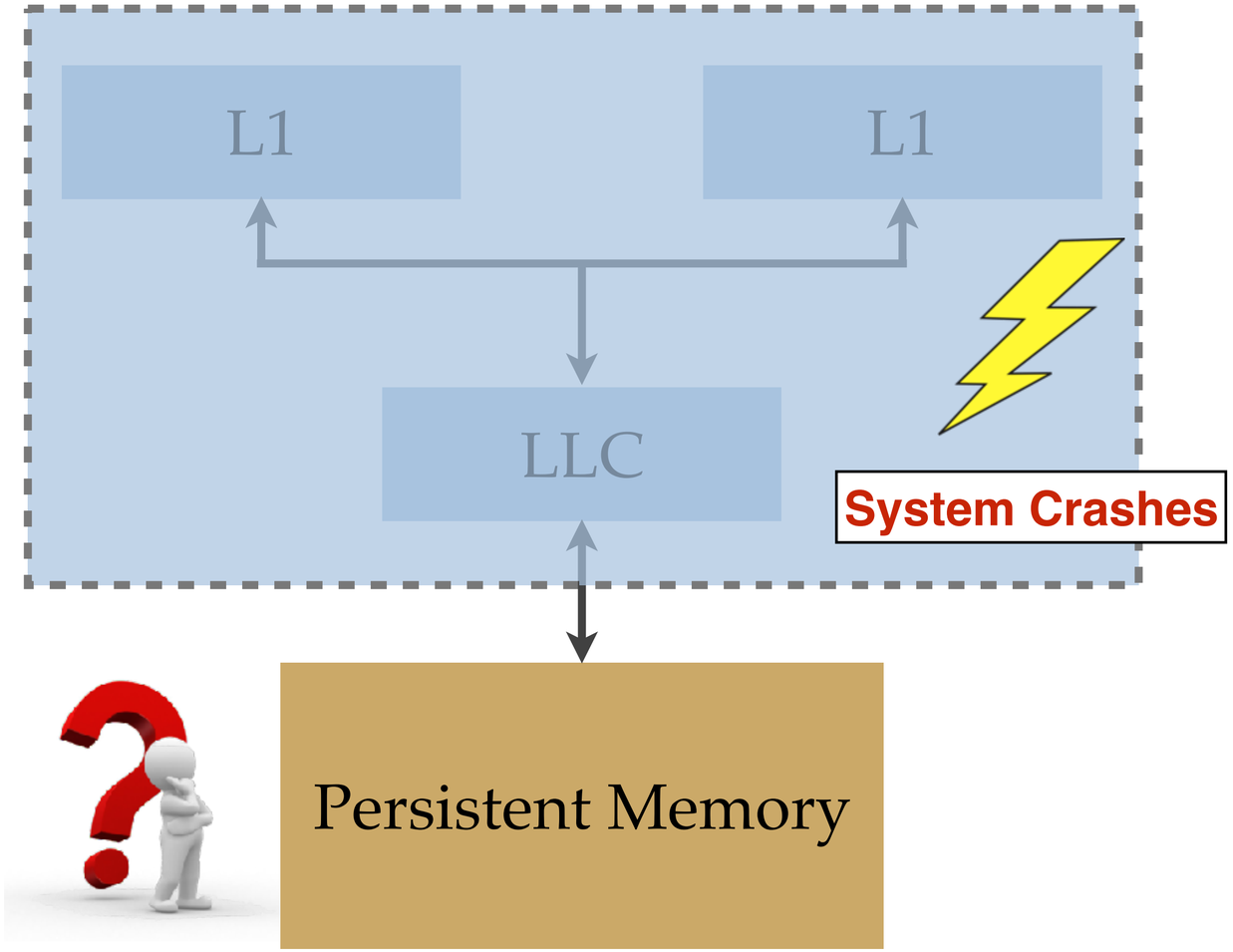


Persistent Memory Systems



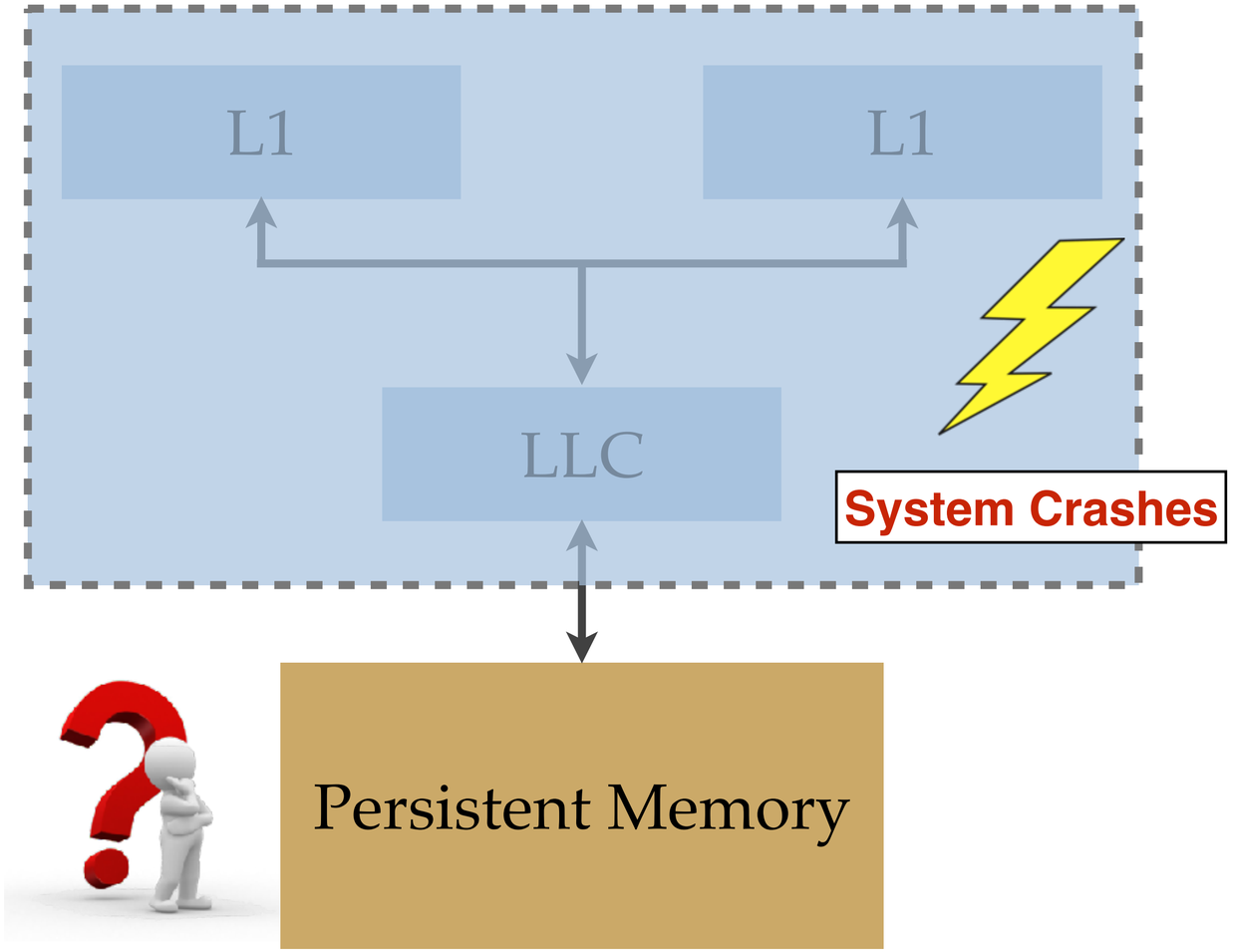
- **Persistent Memory**
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

Persistent Memory Systems



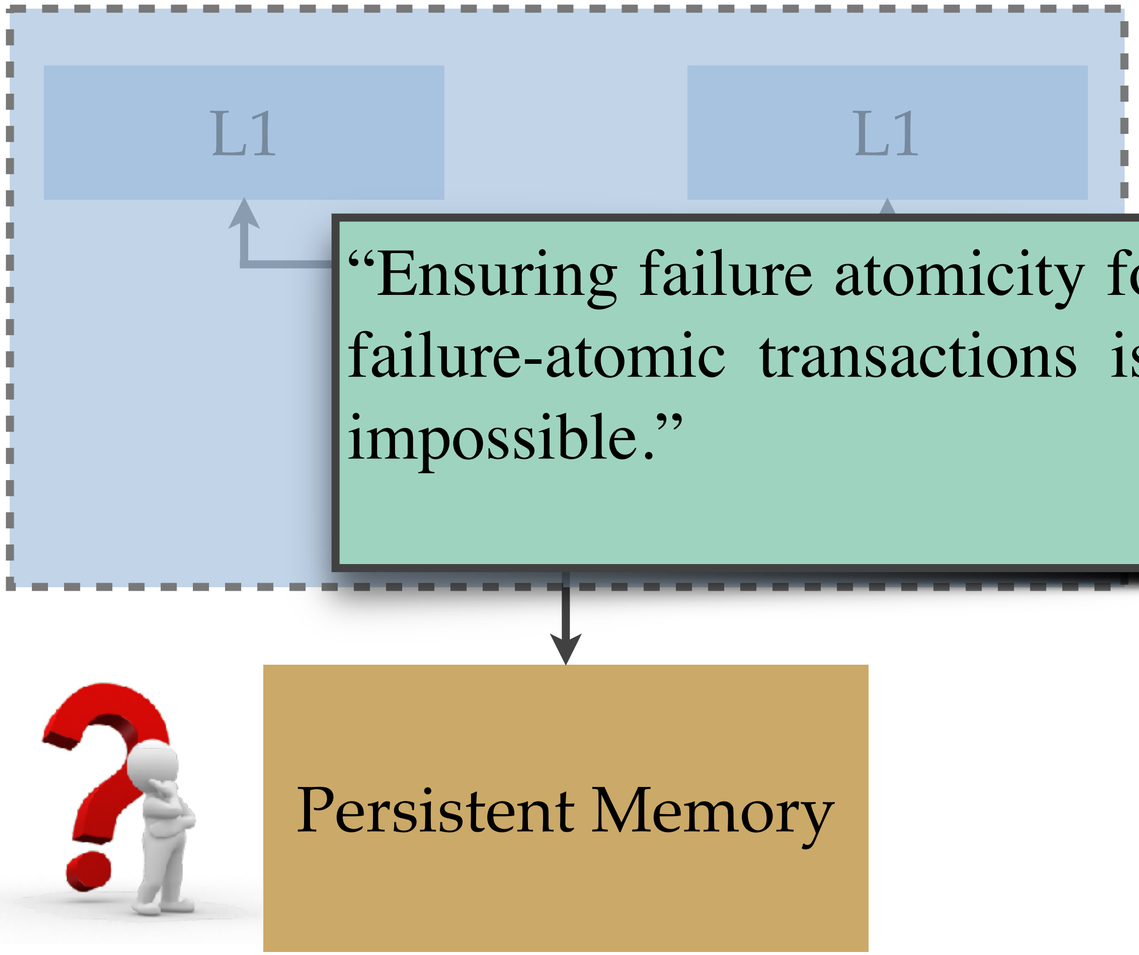
- **Persistent Memory**
  - Non-volatility over the memory bus
  - Load/Store interface to persistent data

Persistent Memory Systems



- **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

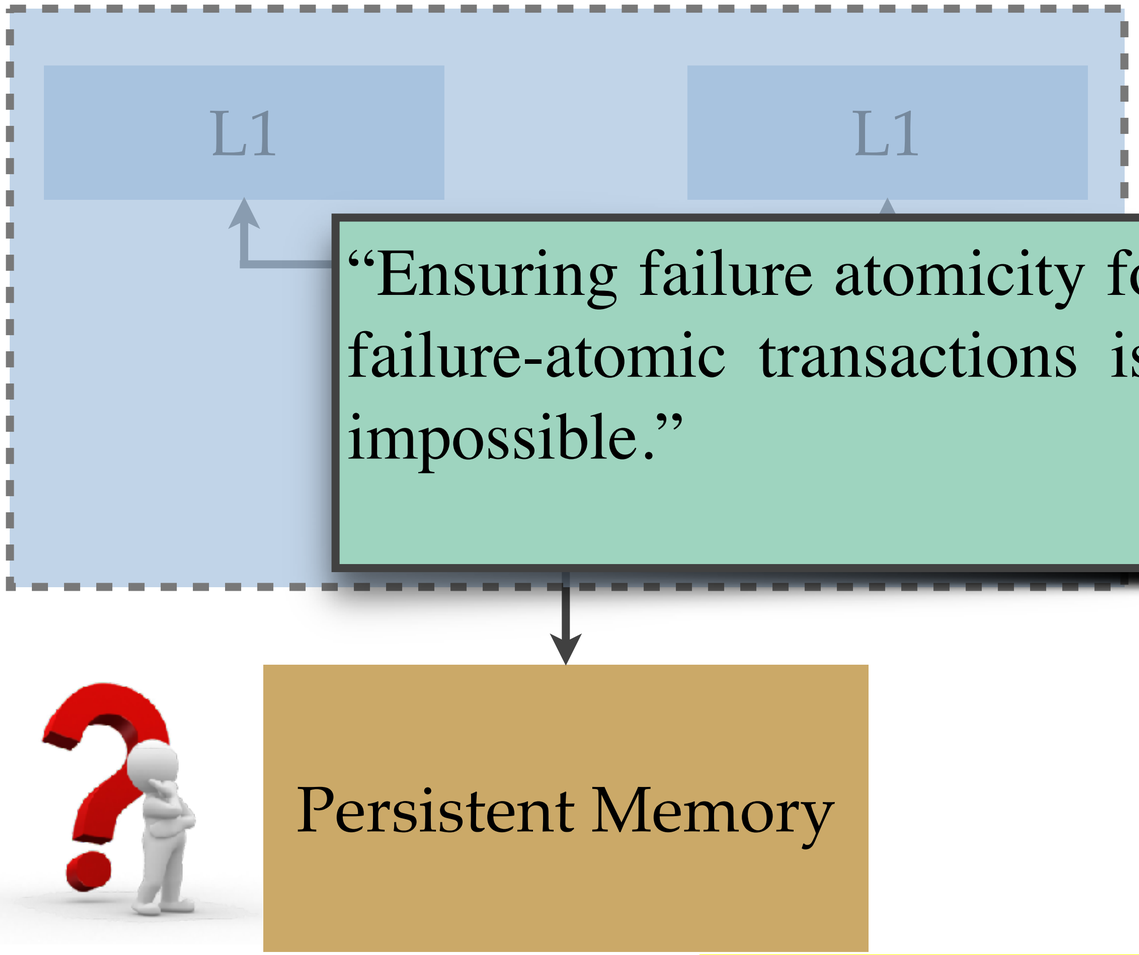


"Ensuring failure atomicity for all this computation without failure-atomic transactions is practically infeasible, if not impossible."

Marathe et al. [HotStorage'17]

- **Persistent Memory**
  - emory bus
  - ersistent data
- **Crash Consistency**
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions

Persistent Memory Systems



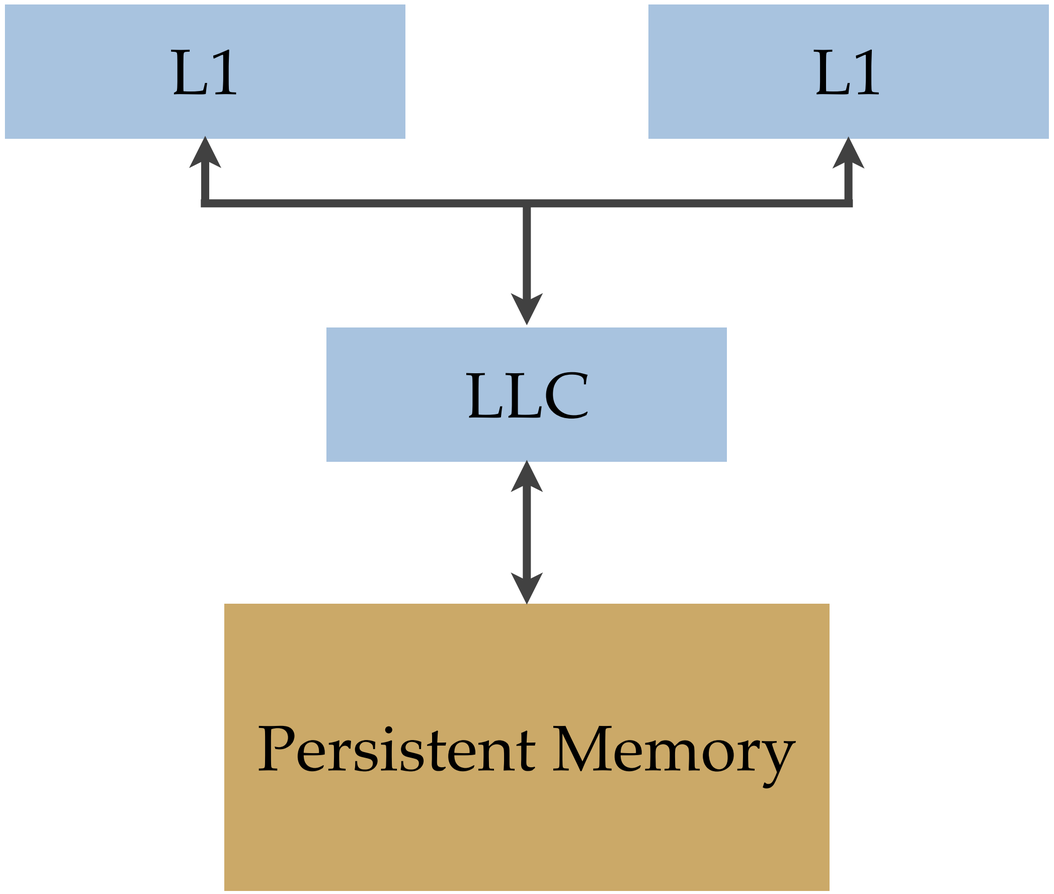
"Ensuring failure atomicity for all this computation without failure-atomic transactions is practically infeasible, if not impossible."

Marathe et al. [HotStorage'17]

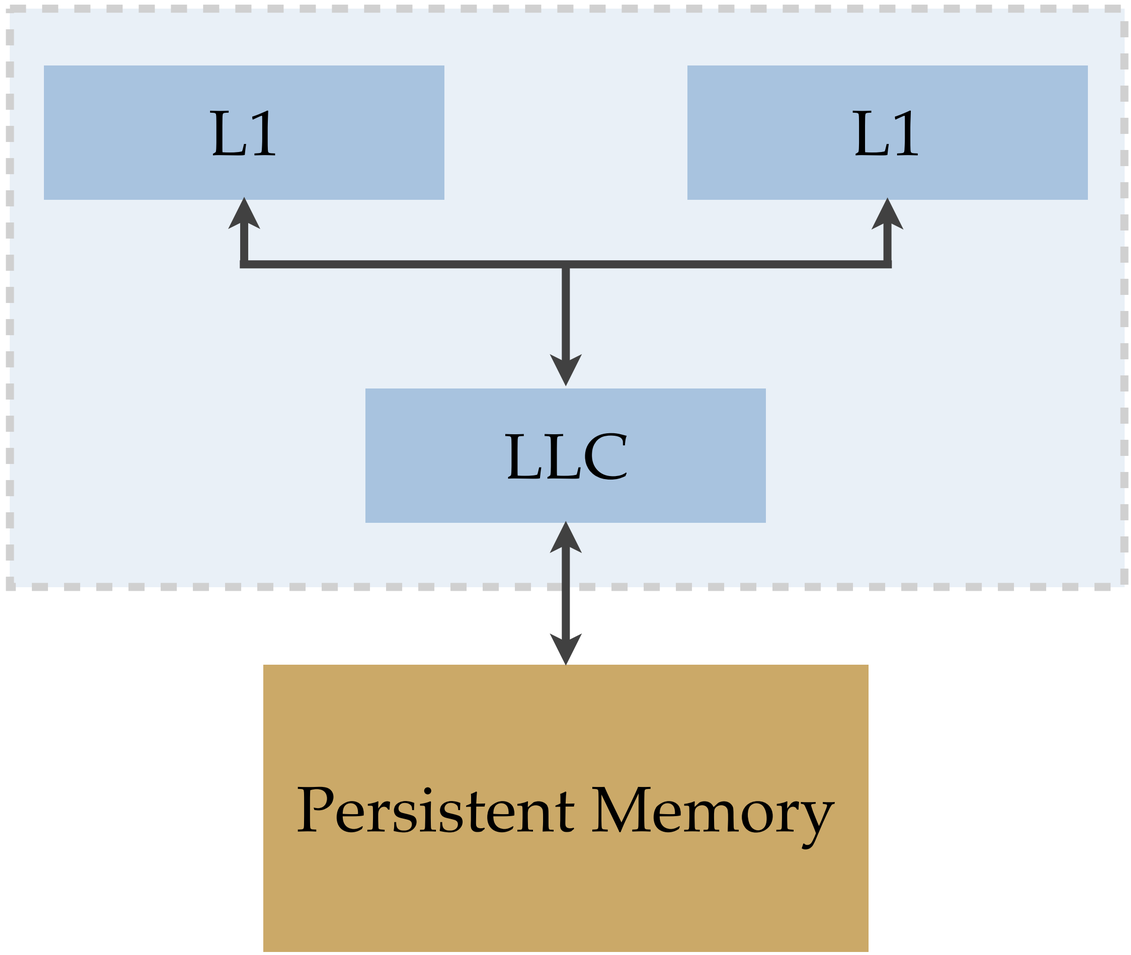
- **Persistent Memory**
  - emory bus
  - ersistent data
- **Crash Consistency**
  - Is the persistent state consistent?
  - Programming Model: ACID Transactions

How fast can we support ACID?

ACID Transactions

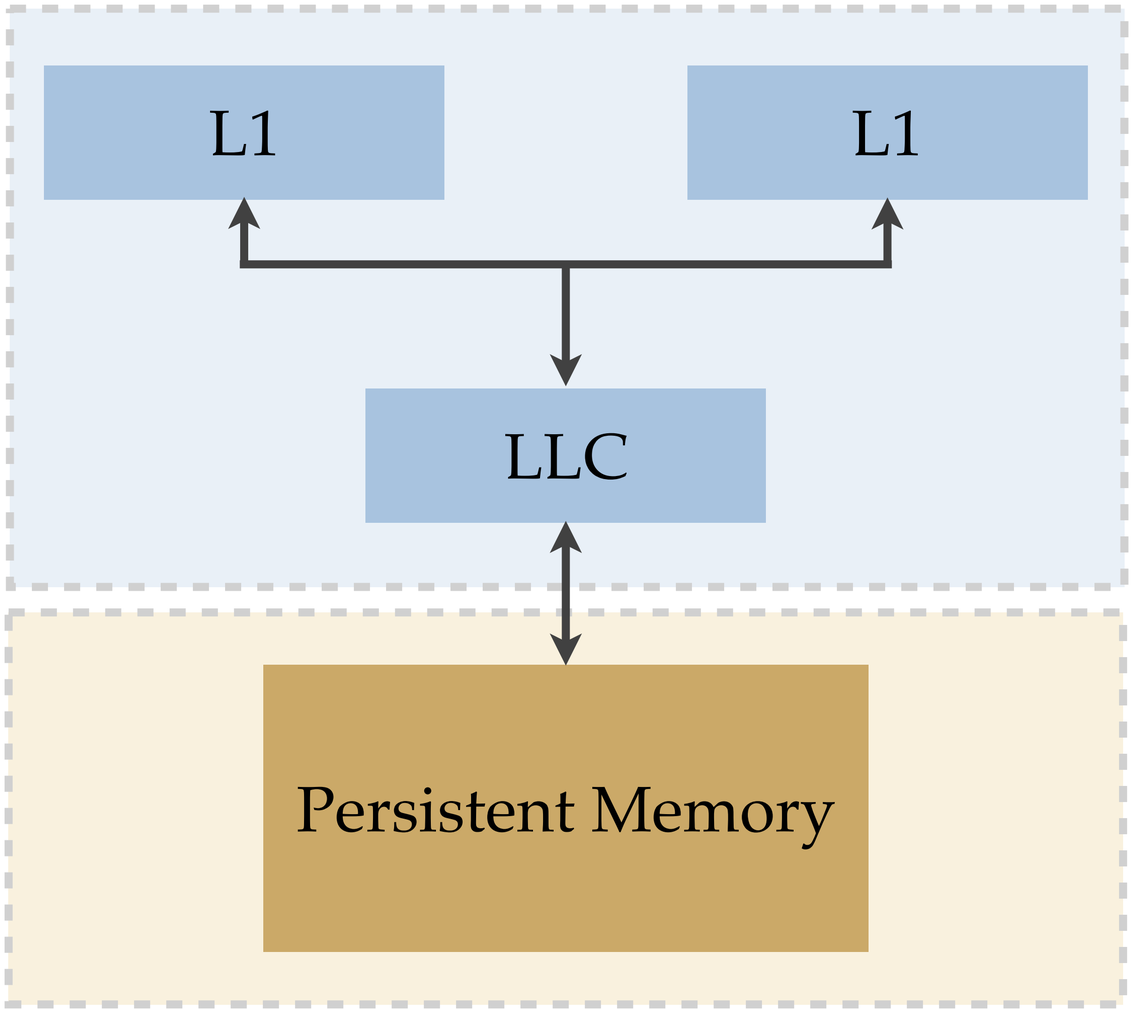


ACID Transactions



Atomic Visibility

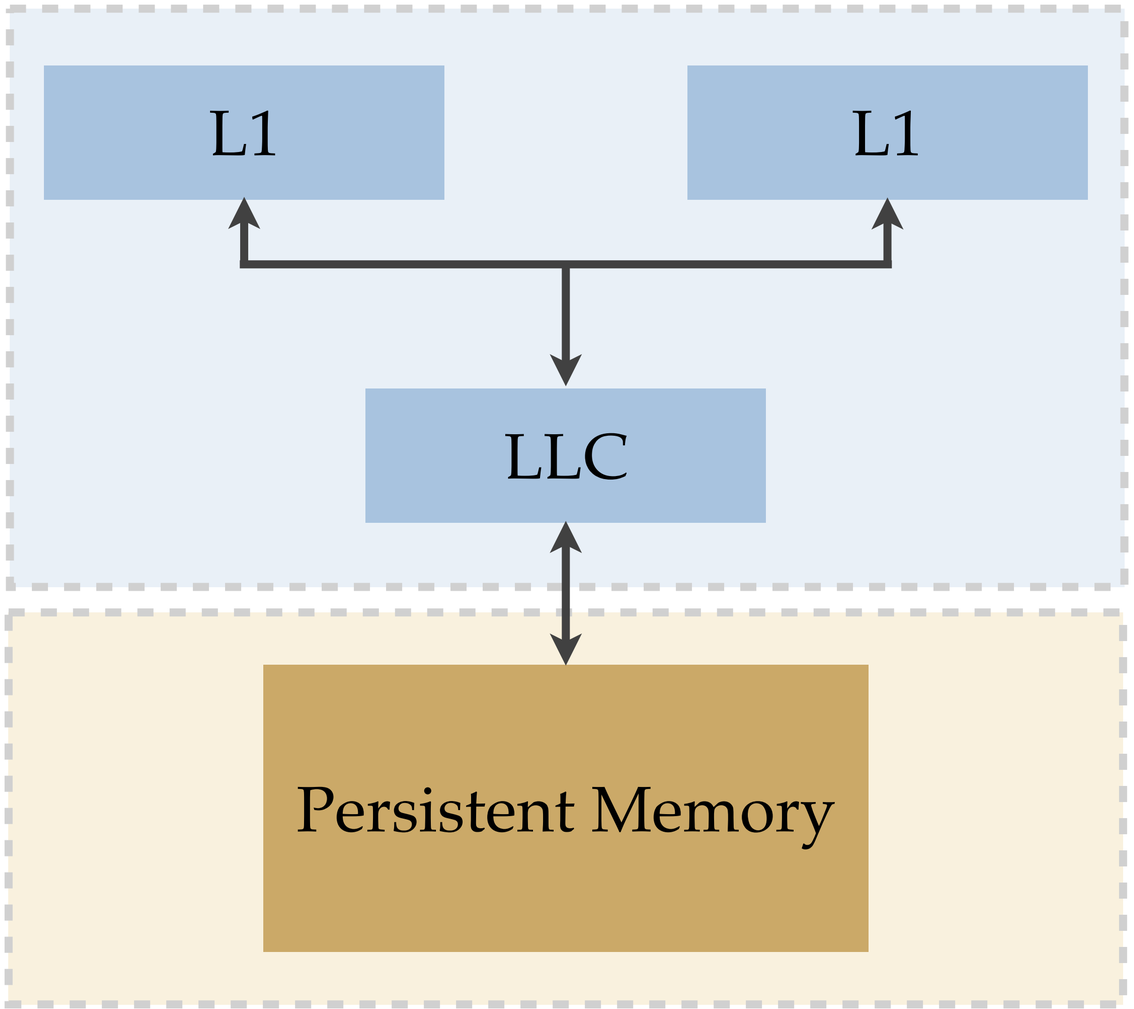
ACID Transactions

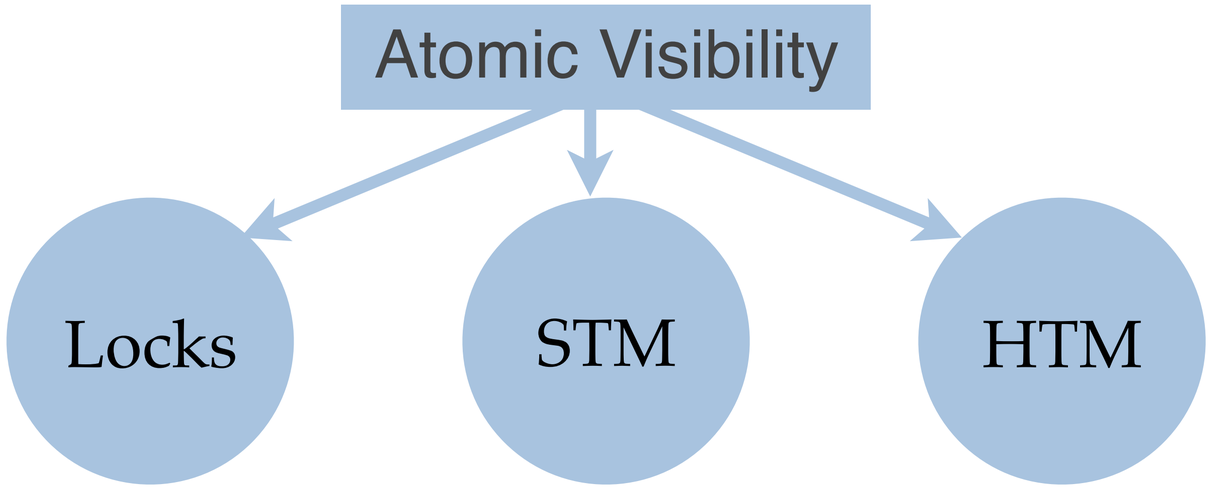


Atomic Visibility

Atomic Durability

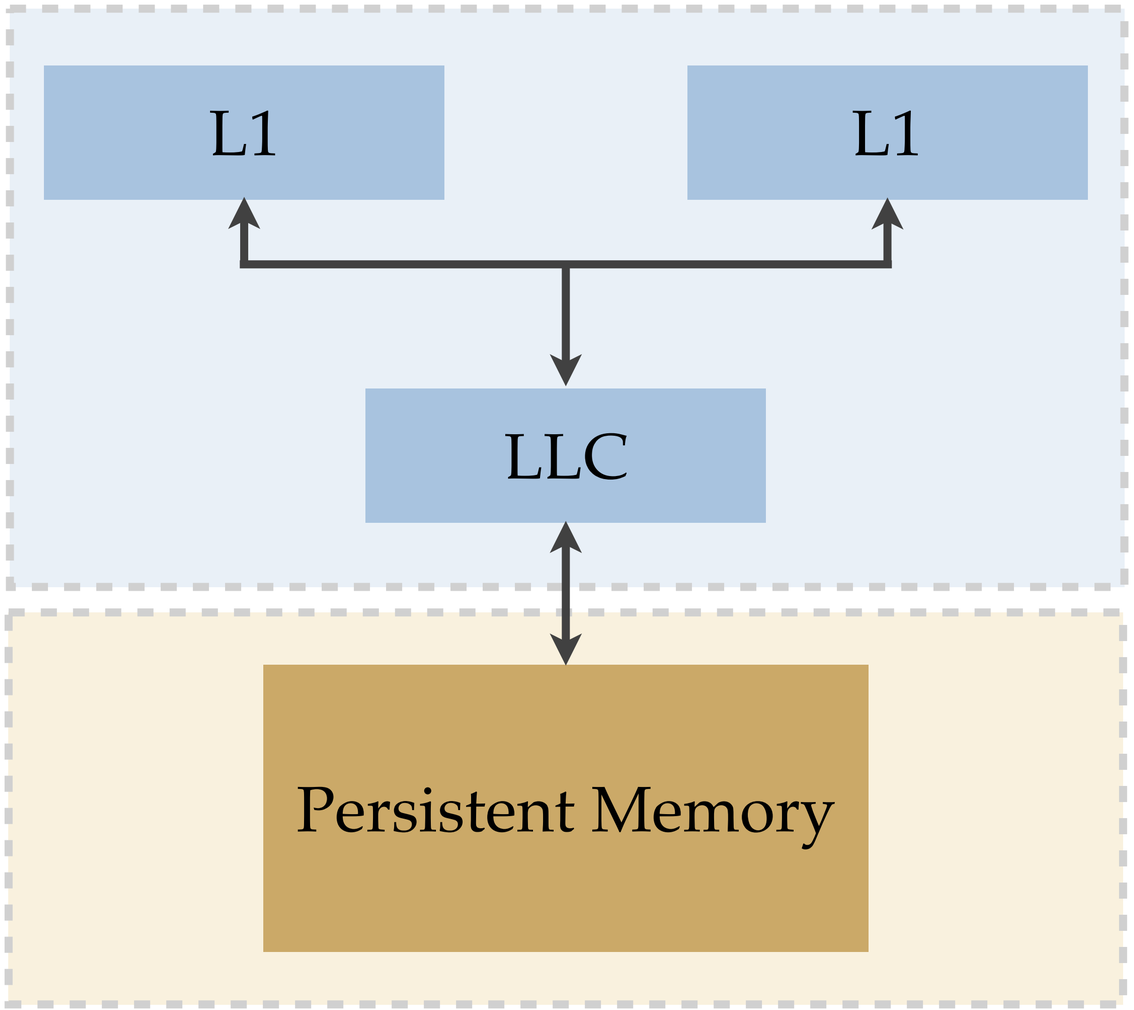
ACID Transactions

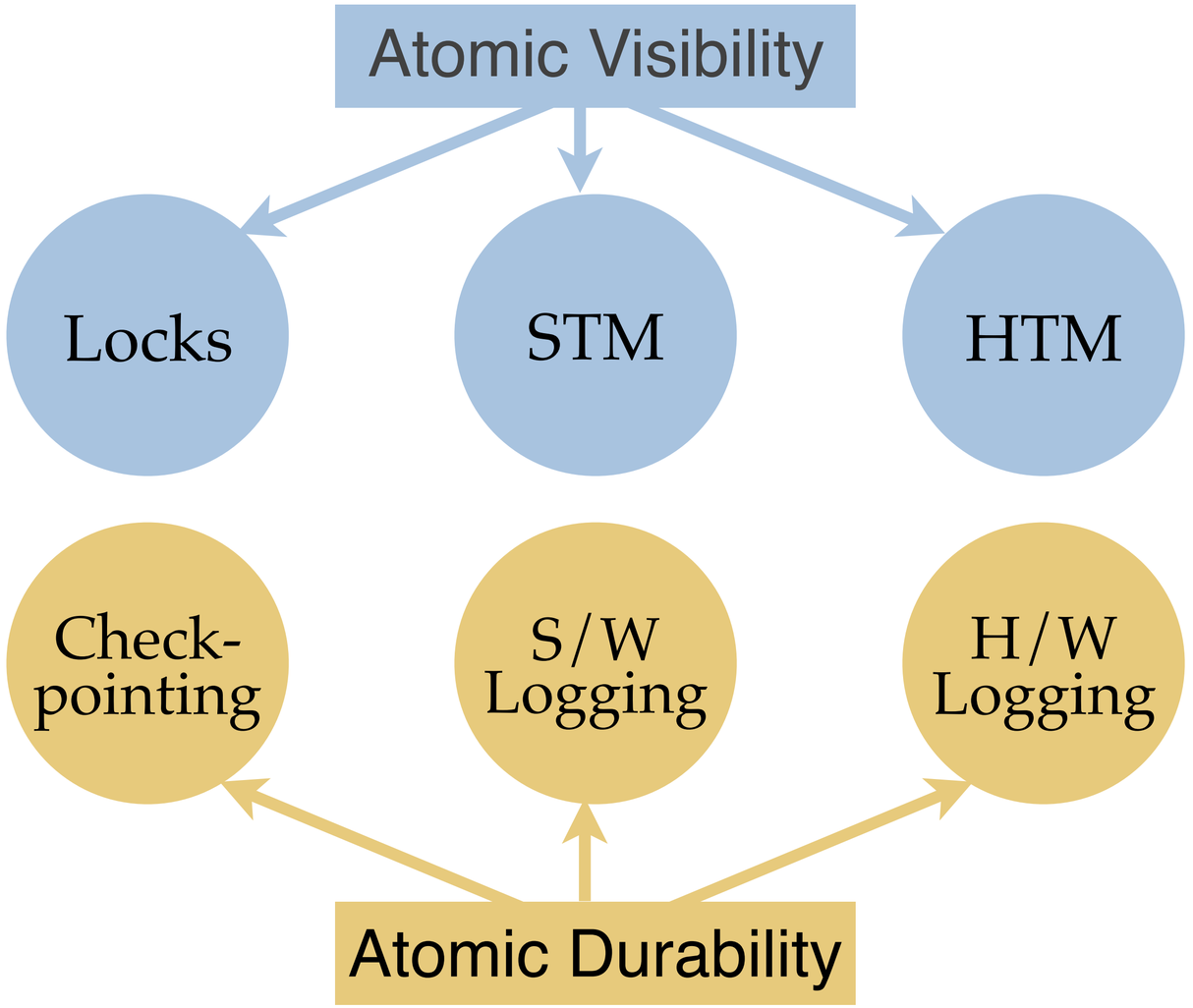




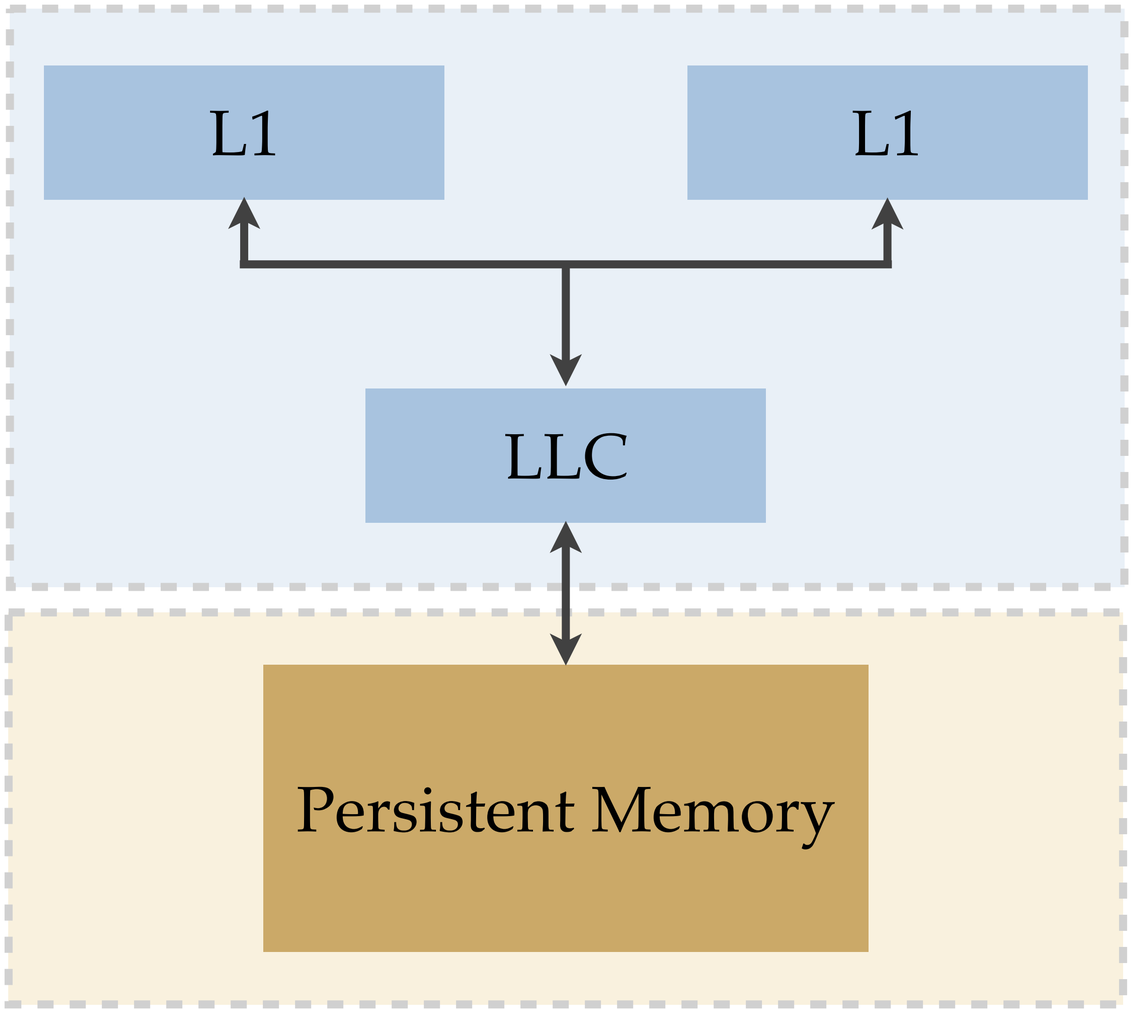
Atomic Durability

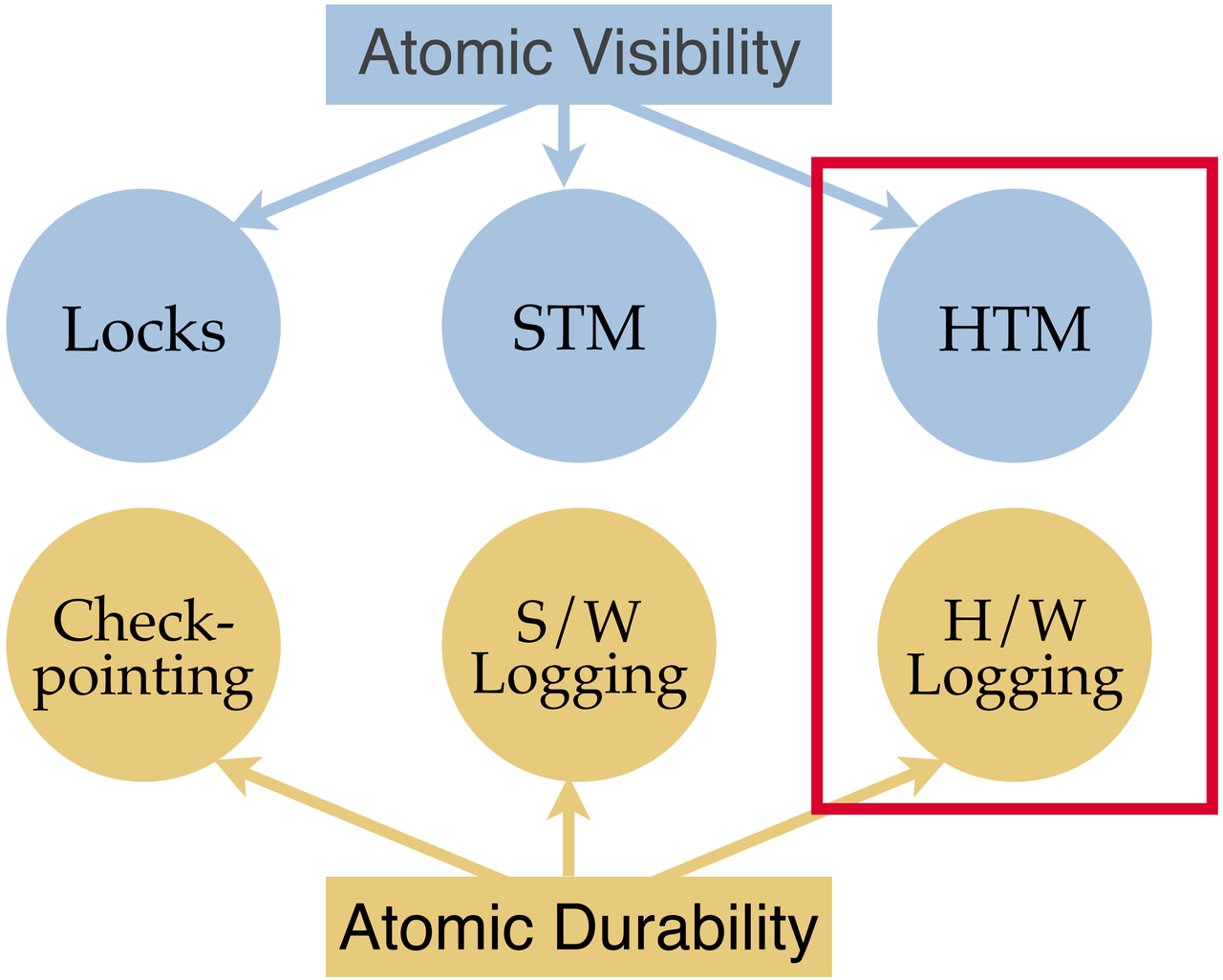
ACID Transactions





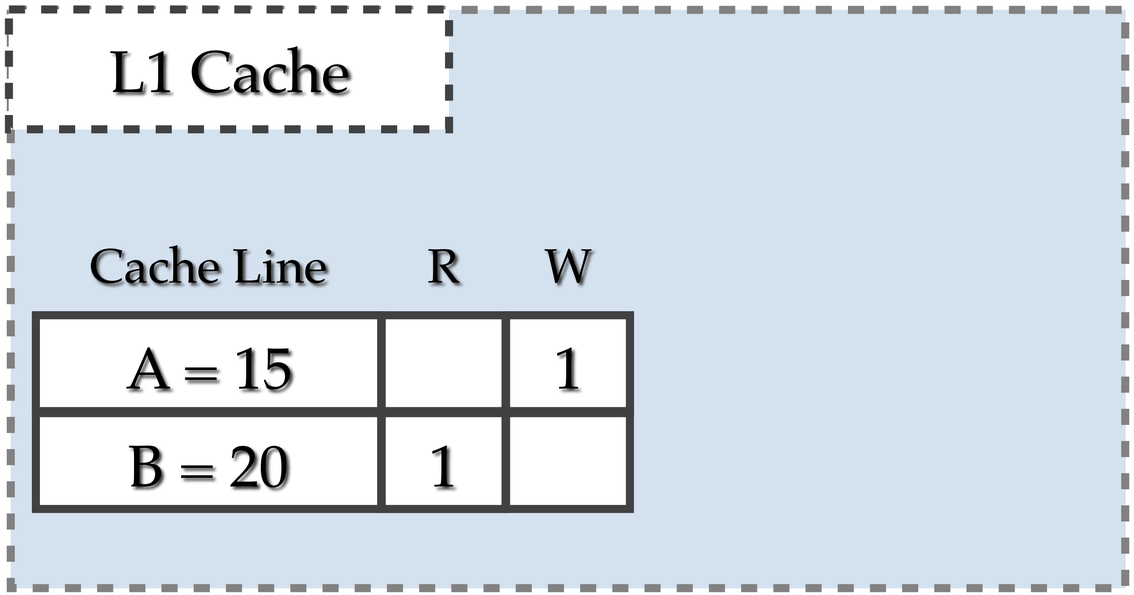
ACID Transactions





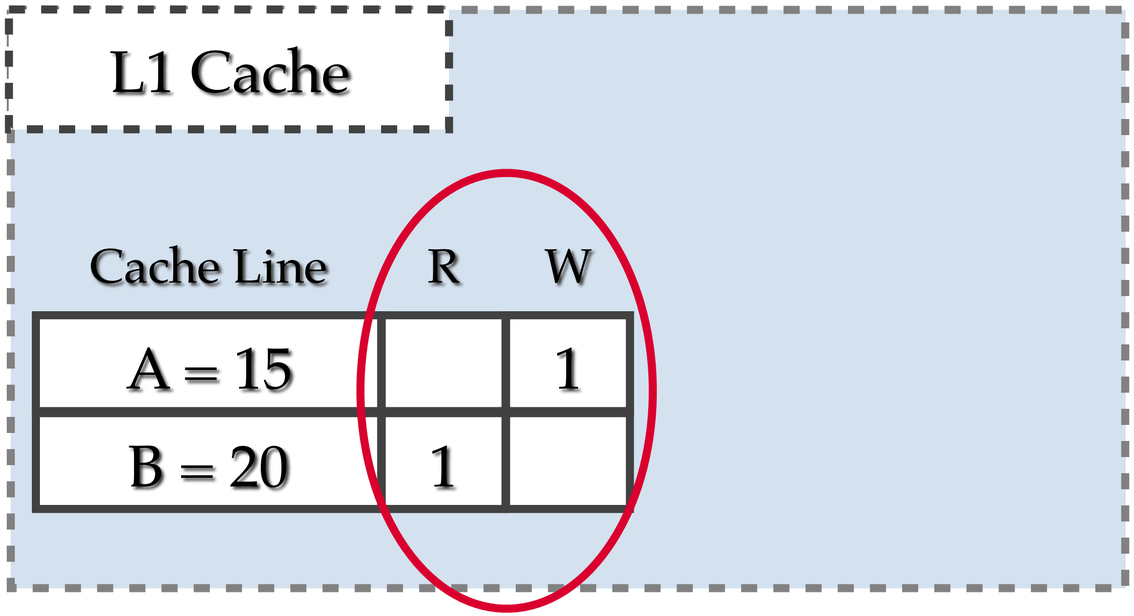
Atomic Visibility: HTM

Atomic Visibility: HTM



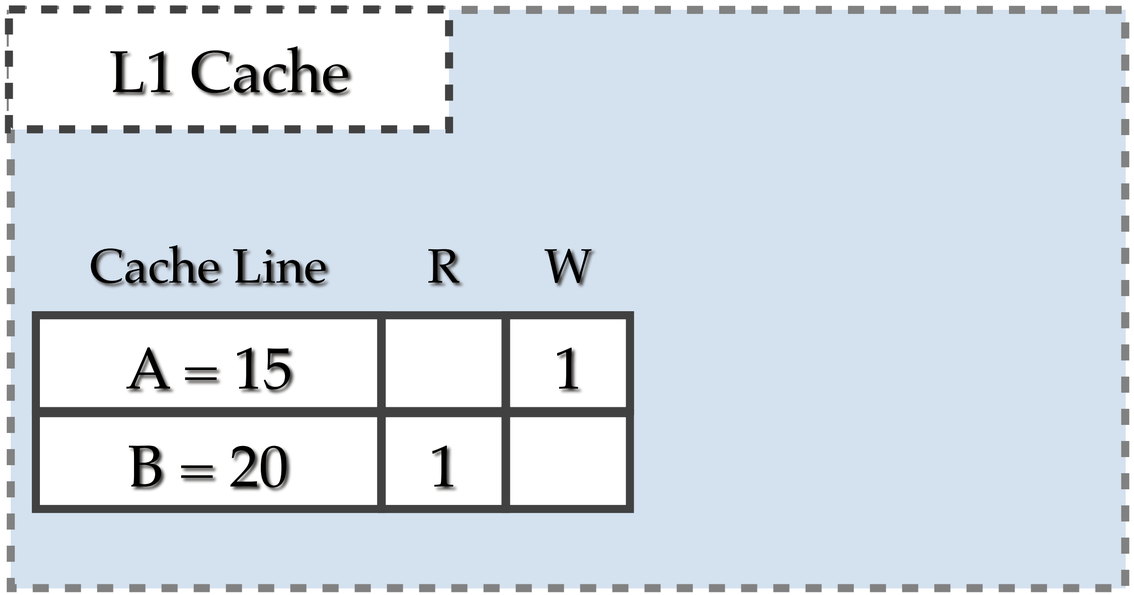
- **Commercial HTMs** [Intel, IBM]

Atomic Visibility: HTM



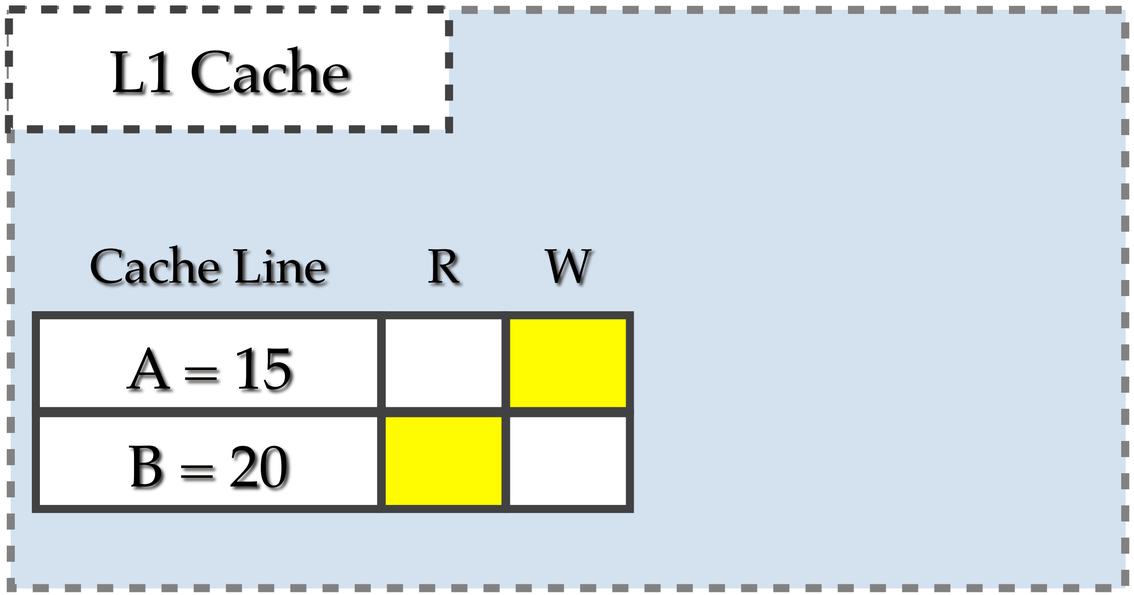
- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache

Atomic Visibility: HTM



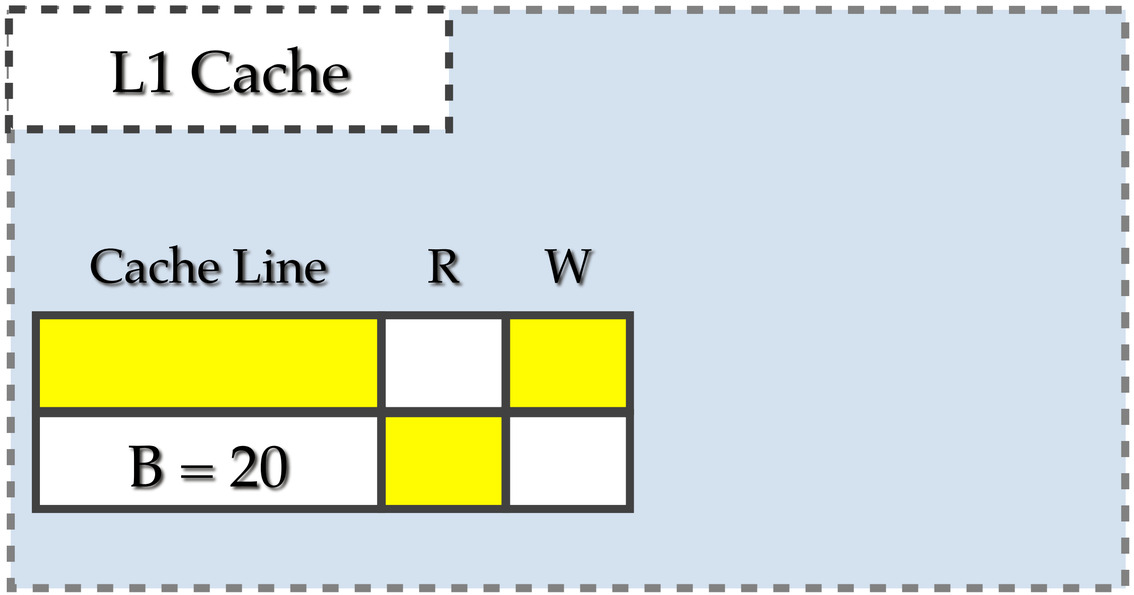
- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol

Atomic Visibility: HTM



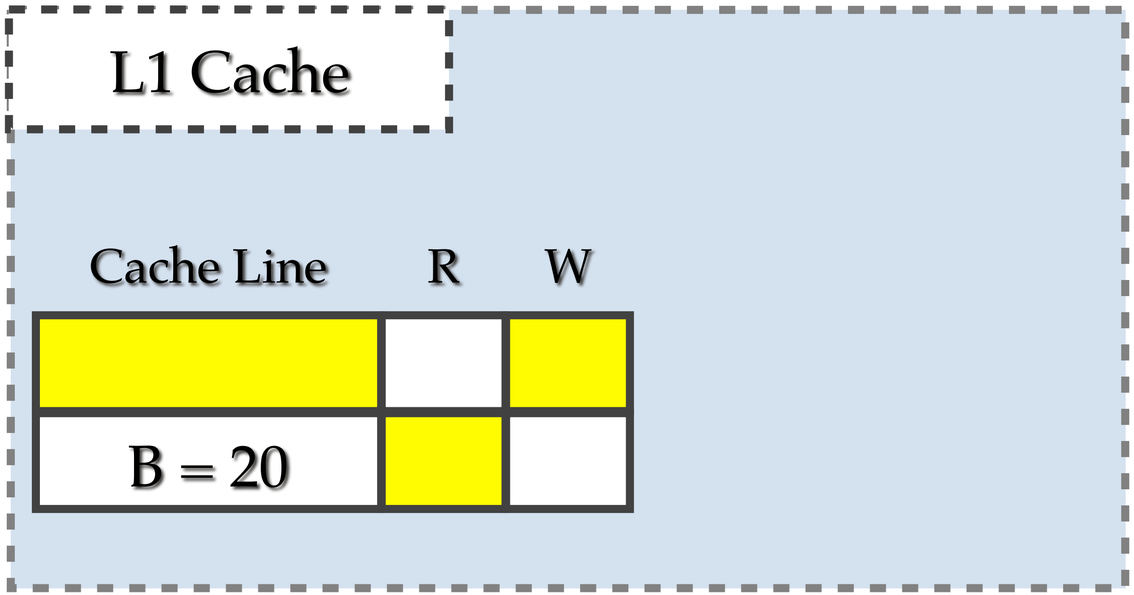
- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol
  - **Commit**: make updates non-speculative

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

Atomic Visibility: HTM

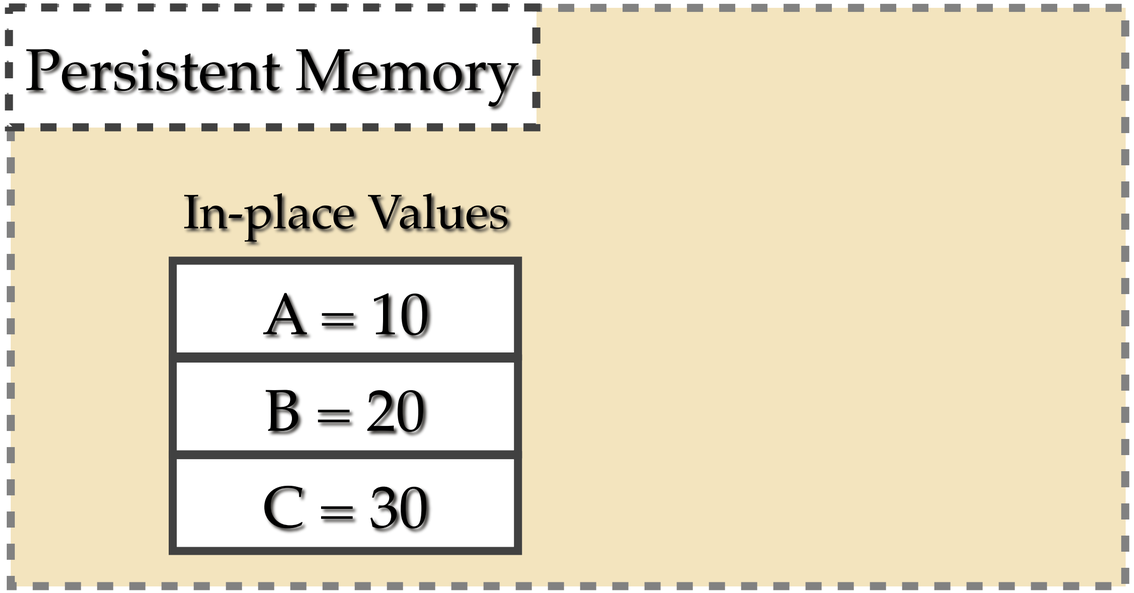


- **Commercial HTMs** [Intel, IBM]
  - **Version Management**: read/write sets in L1 cache
  - **Conflict Detection**: piggy back on the coherence protocol
  - **Commit**: make updates non-speculative
  - **Abort**: invalidate write set

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

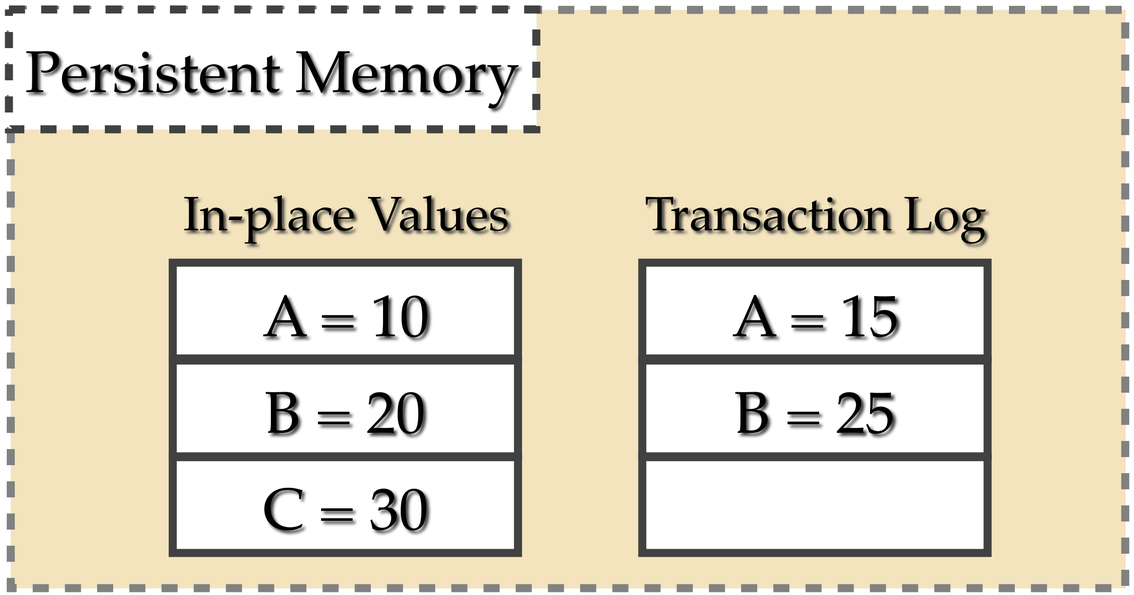
Atomic Durability: Logging

Atomic Durability: Logging



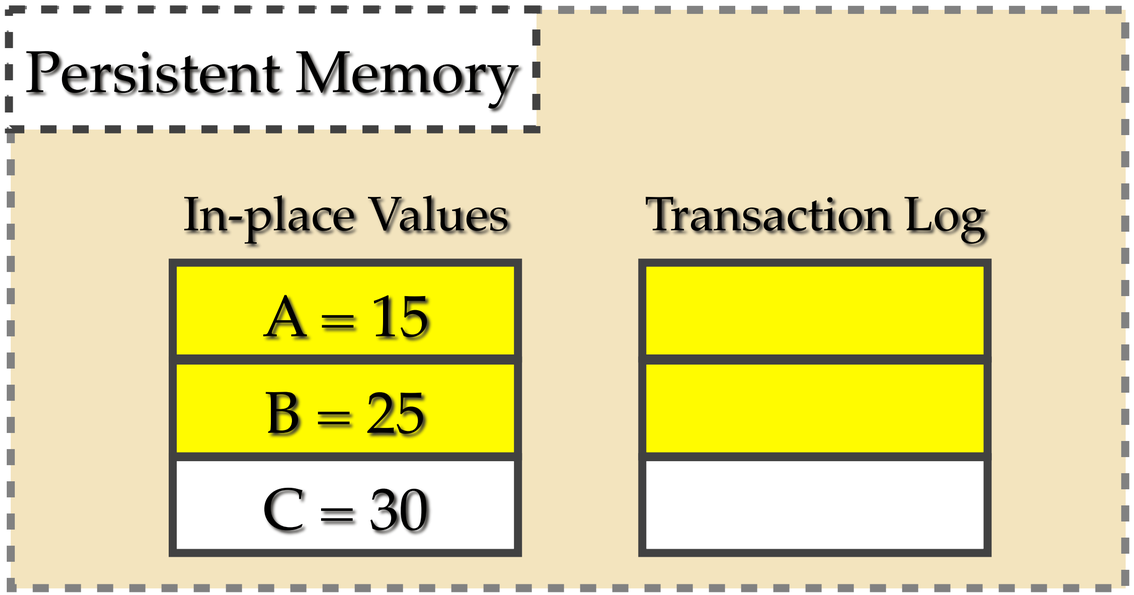
- **Logging for durability** [Doshi'16, Joshi'17, Shin'17, Ogleari'18]

Atomic Durability: Logging



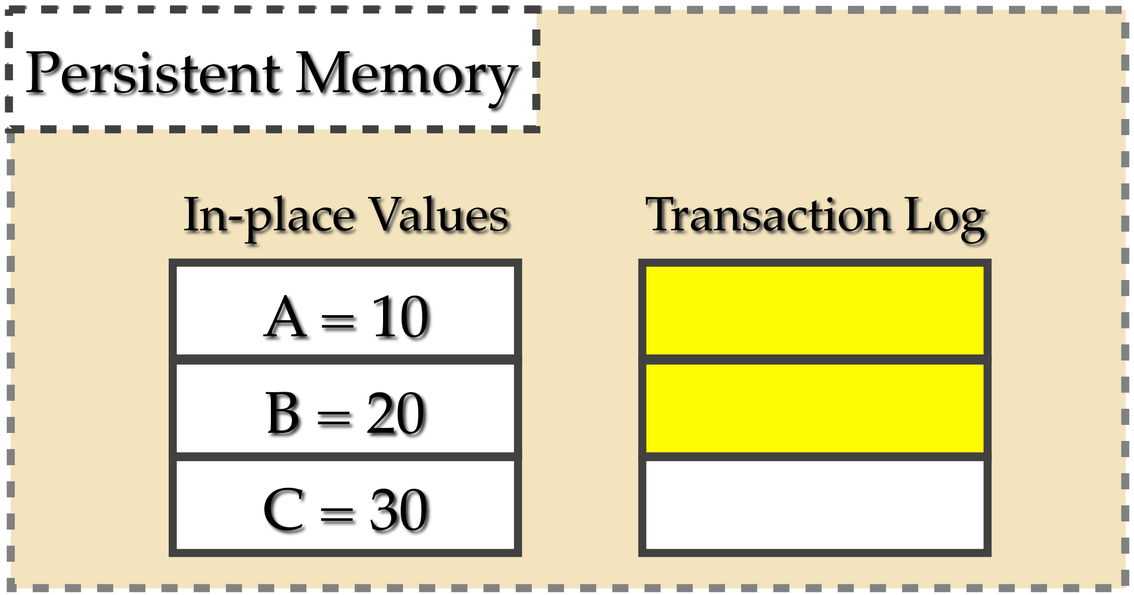
- **Logging for durability** [Doshi'16, Joshi'17, Shin'17, Ogleari'18]
  - Write a log entry for every update

Atomic Durability: Logging



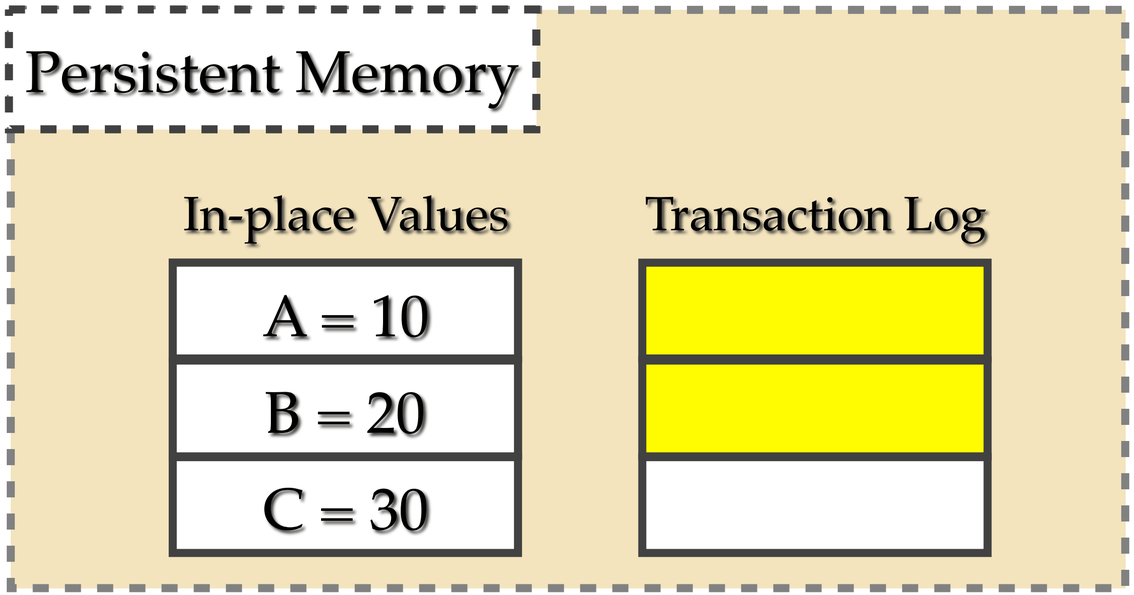
- **Logging for durability** [Doshi'16, Joshi'17, Shin'17, Ogleari'18]
  - Write a log entry for every update
  - **Commit**: Update the values in-place

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

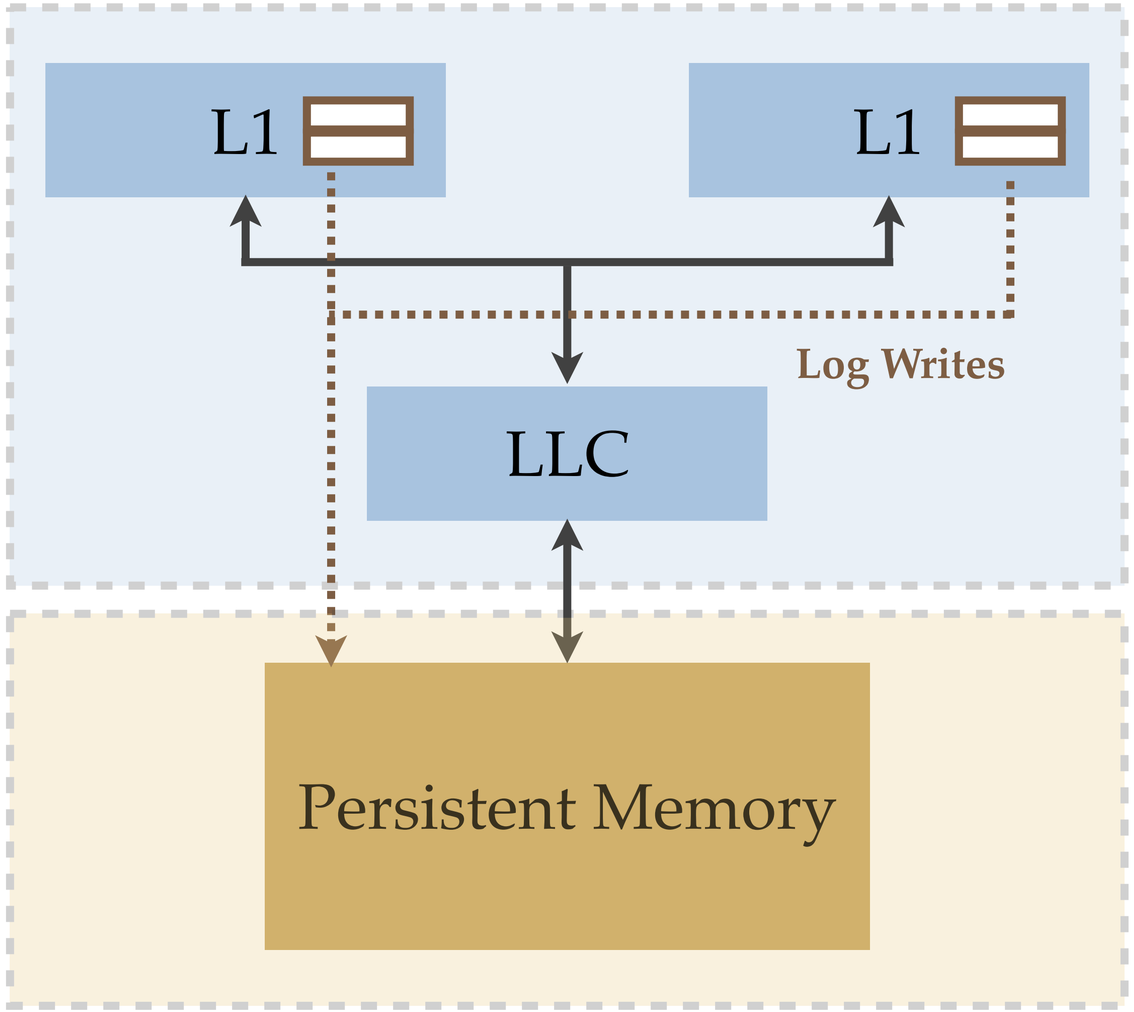
✗ In-place updates in the critical path of commit
✗ High memory write bandwidth requirement

ACID = HTM + Logging

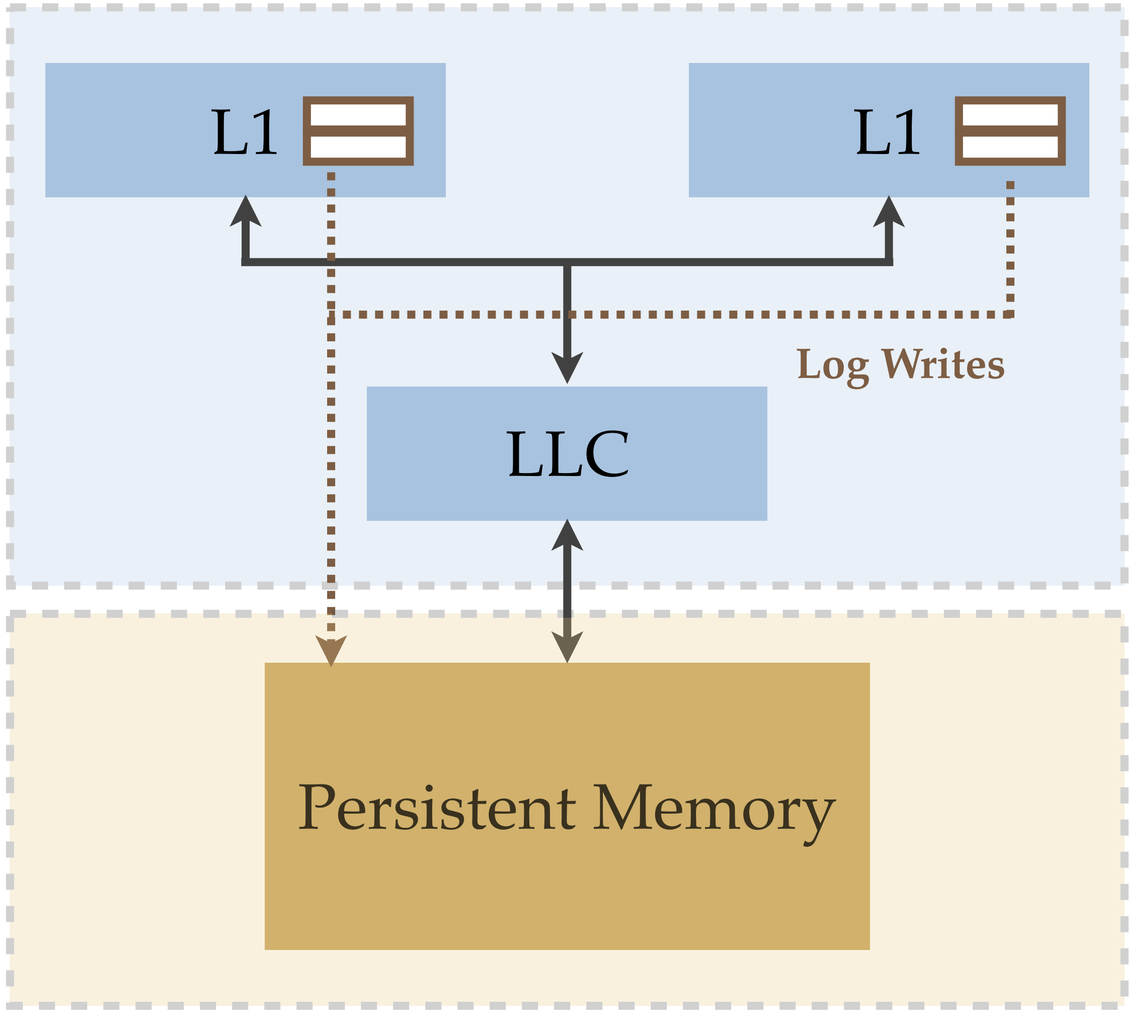
Goals:

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

DHTM: Durable Hardware Transactional Memory

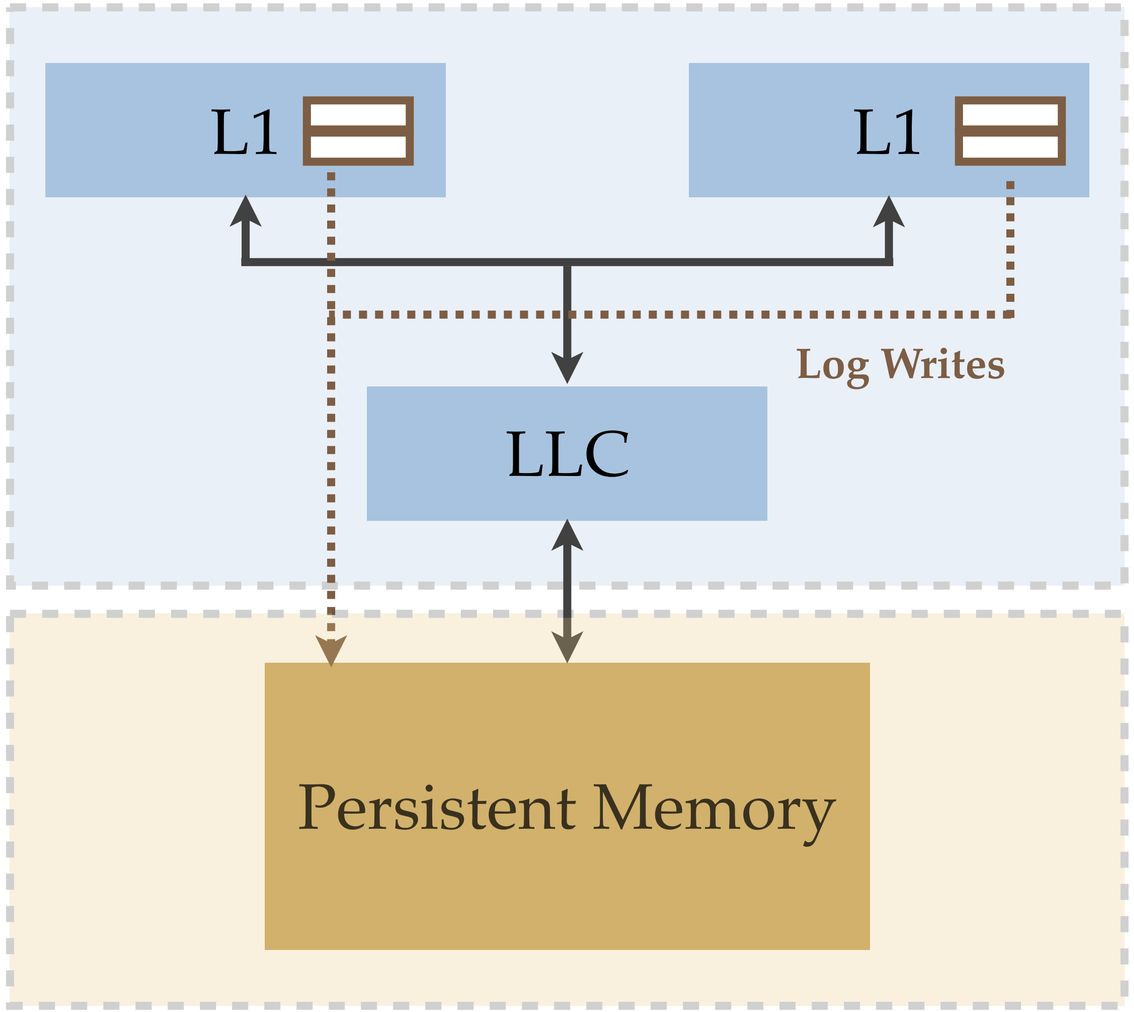


DHTM: Durable Hardware Transactional Memory



Commercial HTM + Hardware Redo Log

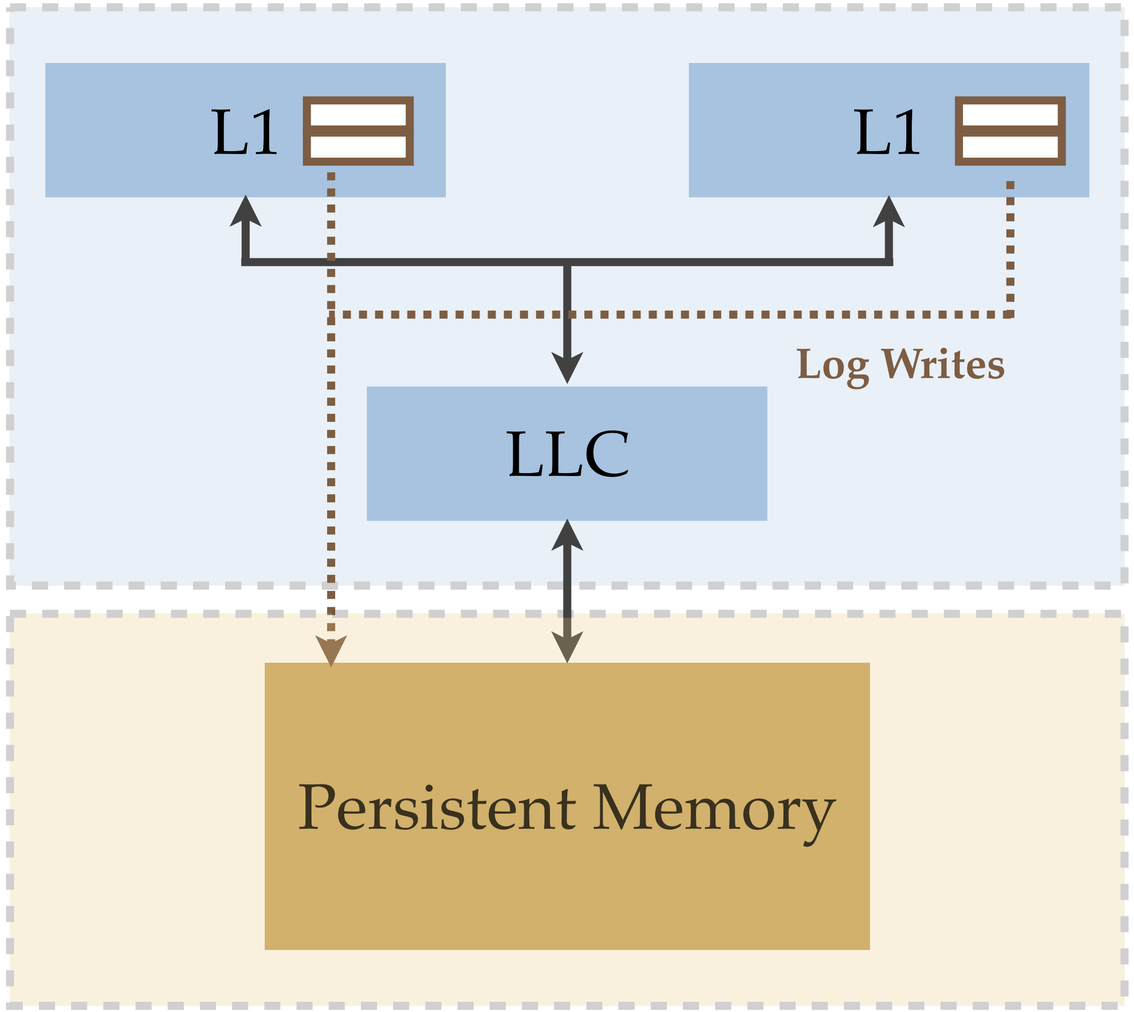
DHTM: Durable Hardware Transactional Memory



Commercial HTM + **Hardware Redo Log**

- H/W Redo Log + Log Buffer
  - ✓ Reduced memory bandwidth
  - ✓ Fast commits

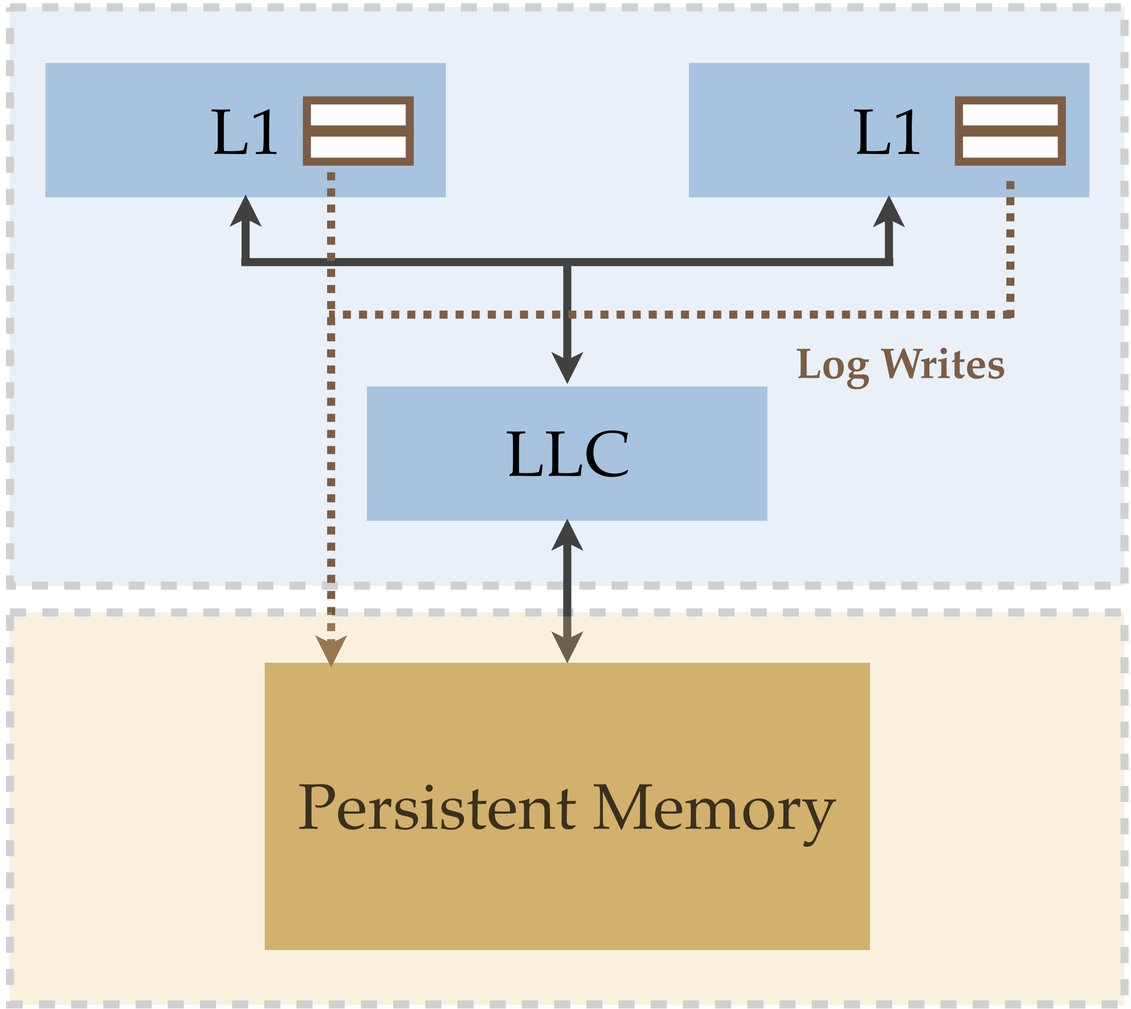
DHTM: Durable Hardware Transactional Memory



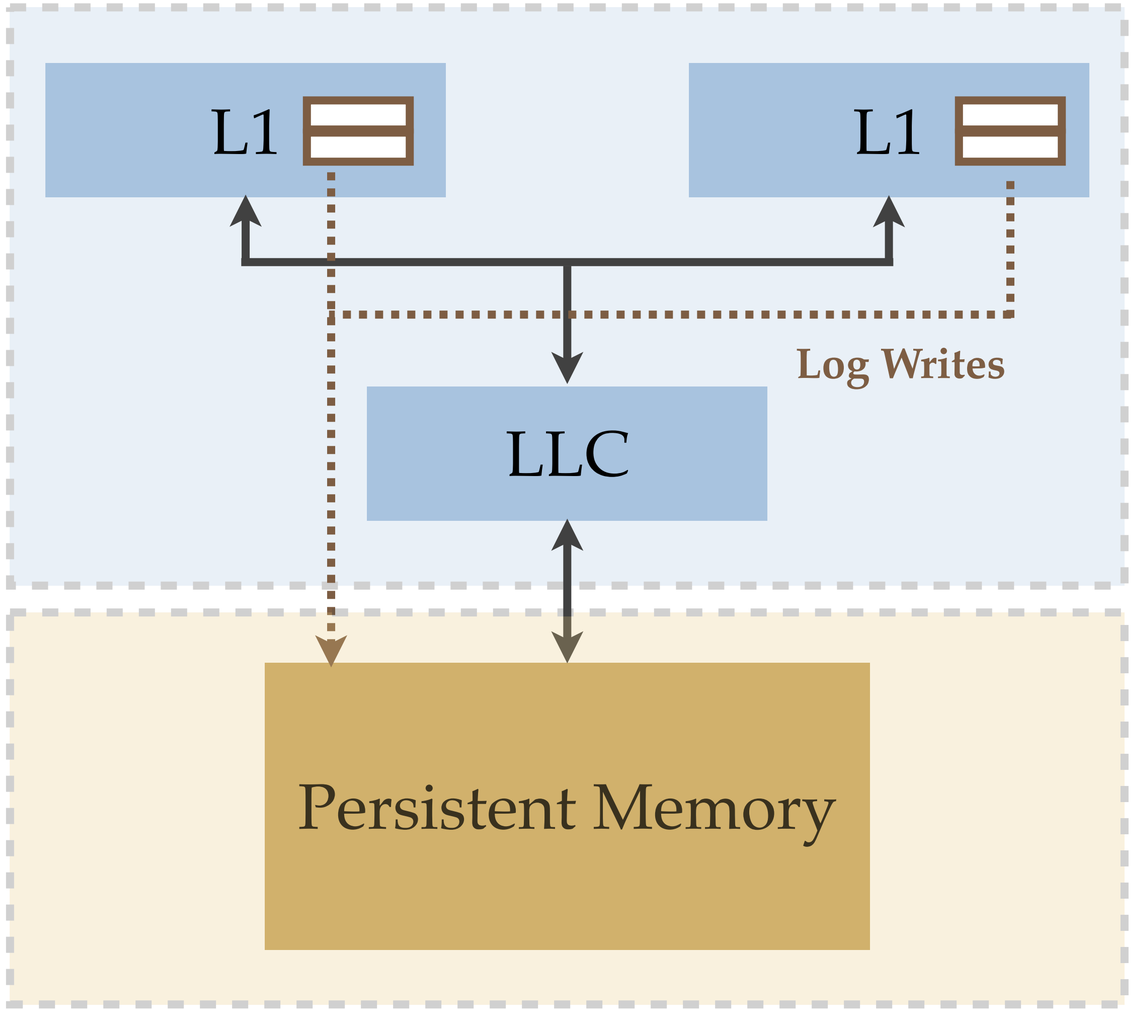
Commercial HTM + **Hardware Redo Log**

- 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

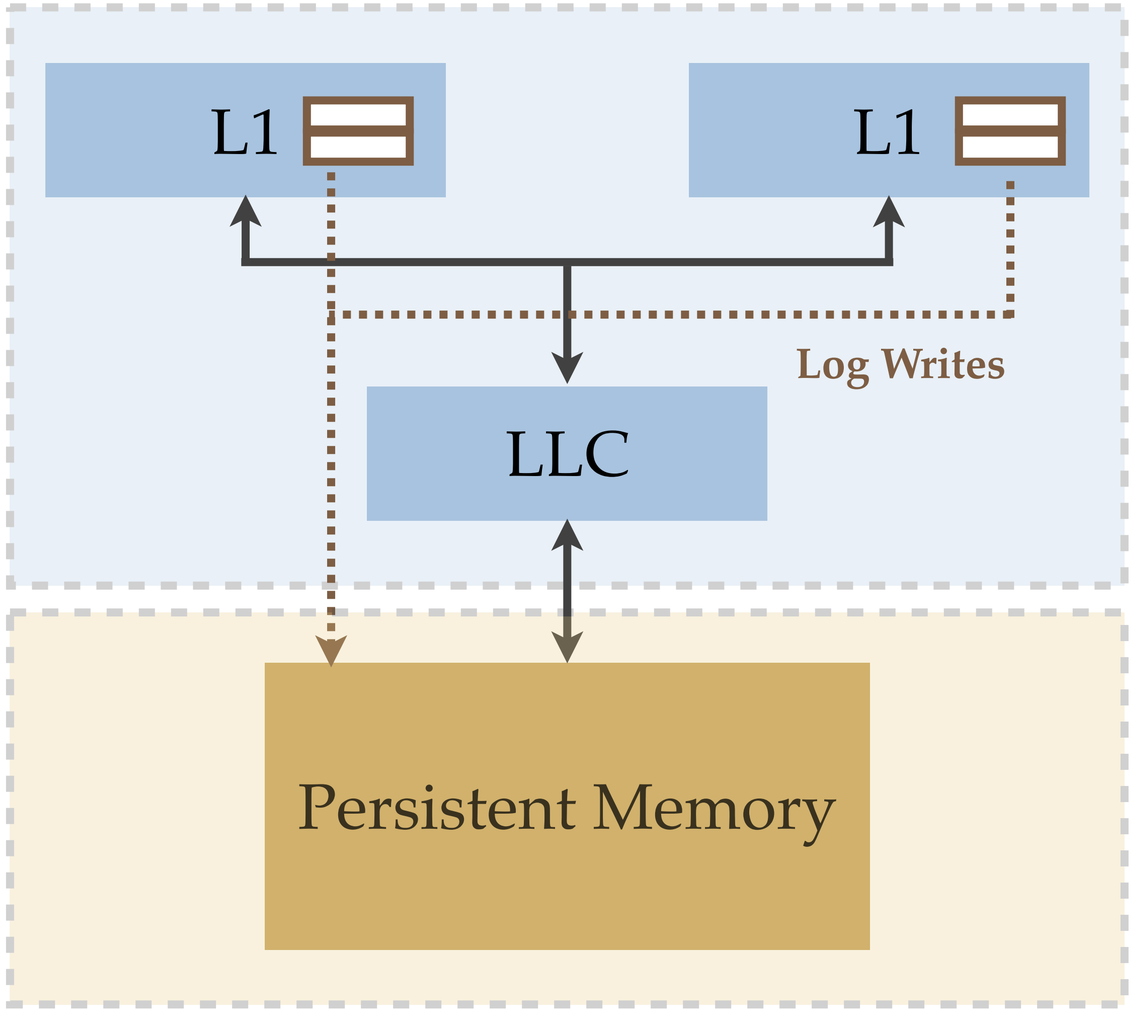


DHTM: Log Buffer



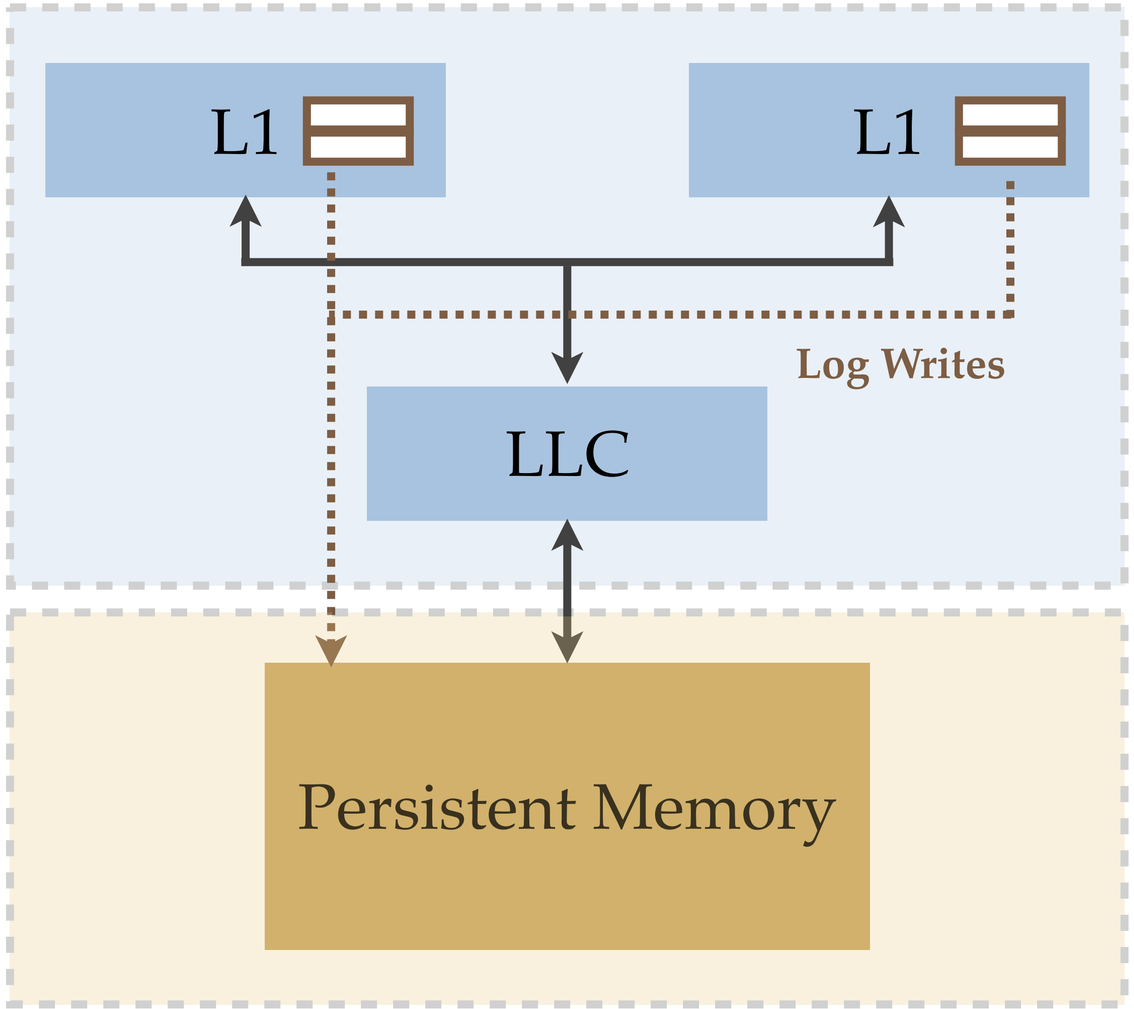
- **Redo Log Bandwidth Problem**

DHTM: Log Buffer



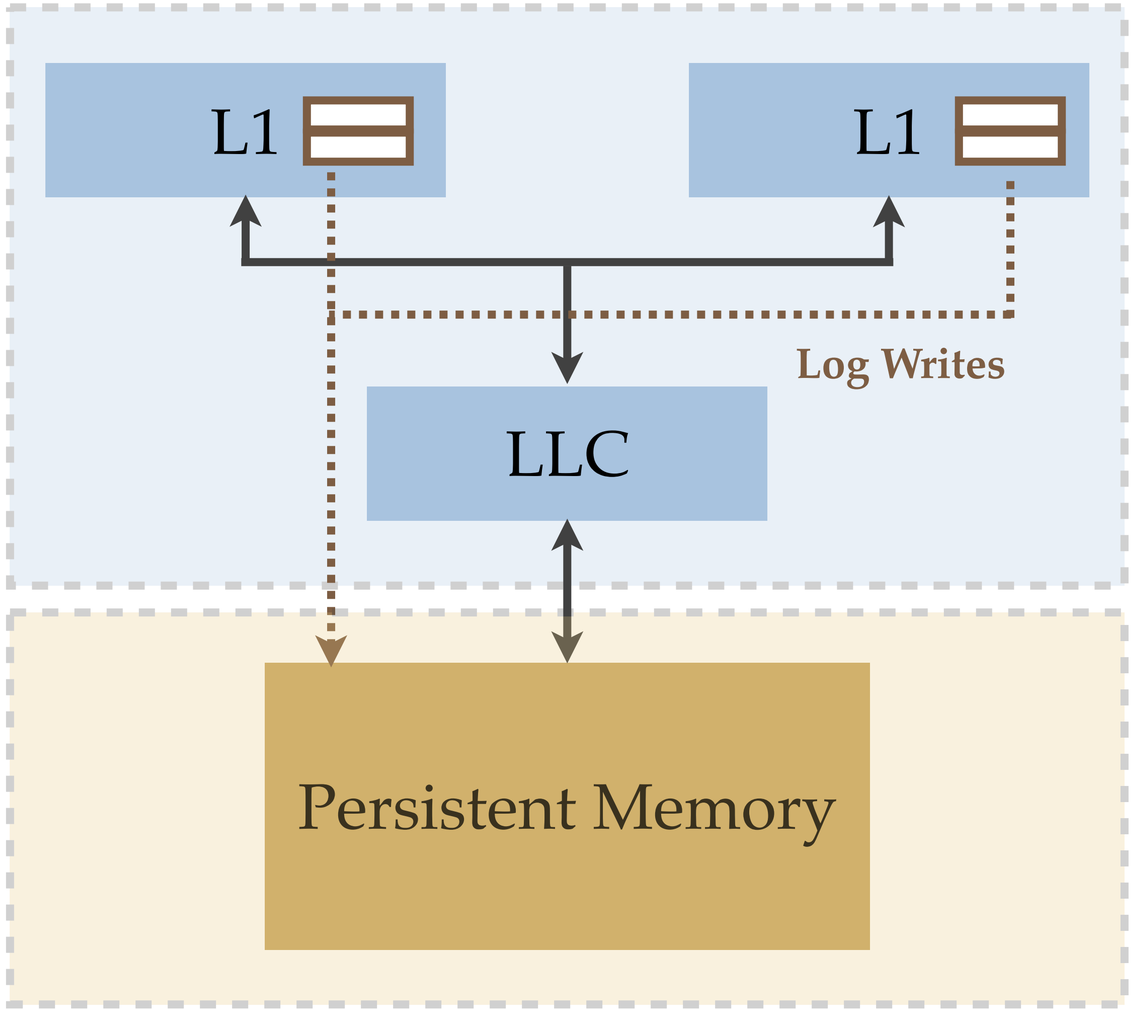
- **Redo Log Bandwidth Problem**
  - write a log entry for every store

DHTM: Log Buffer



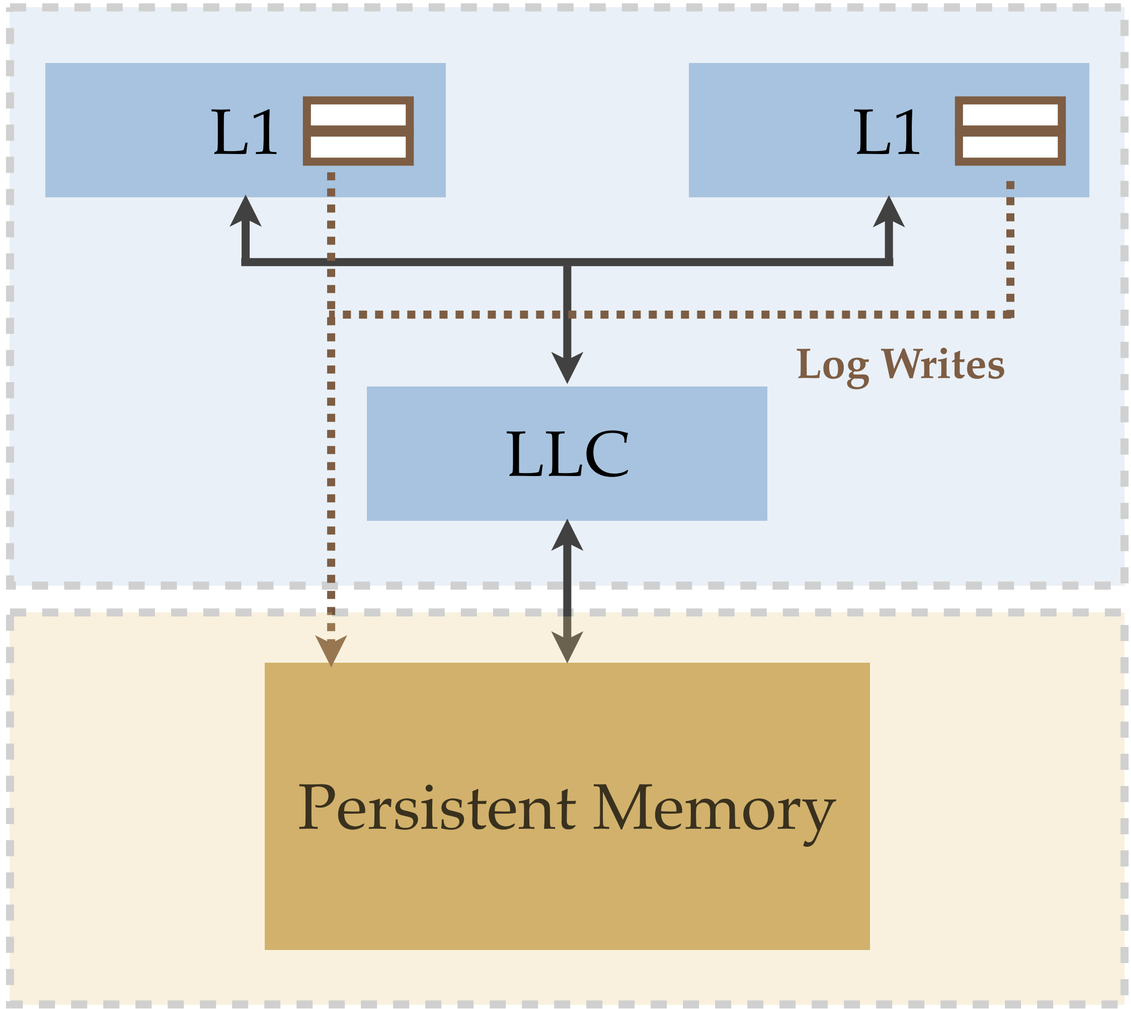
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries

DHTM: Log Buffer



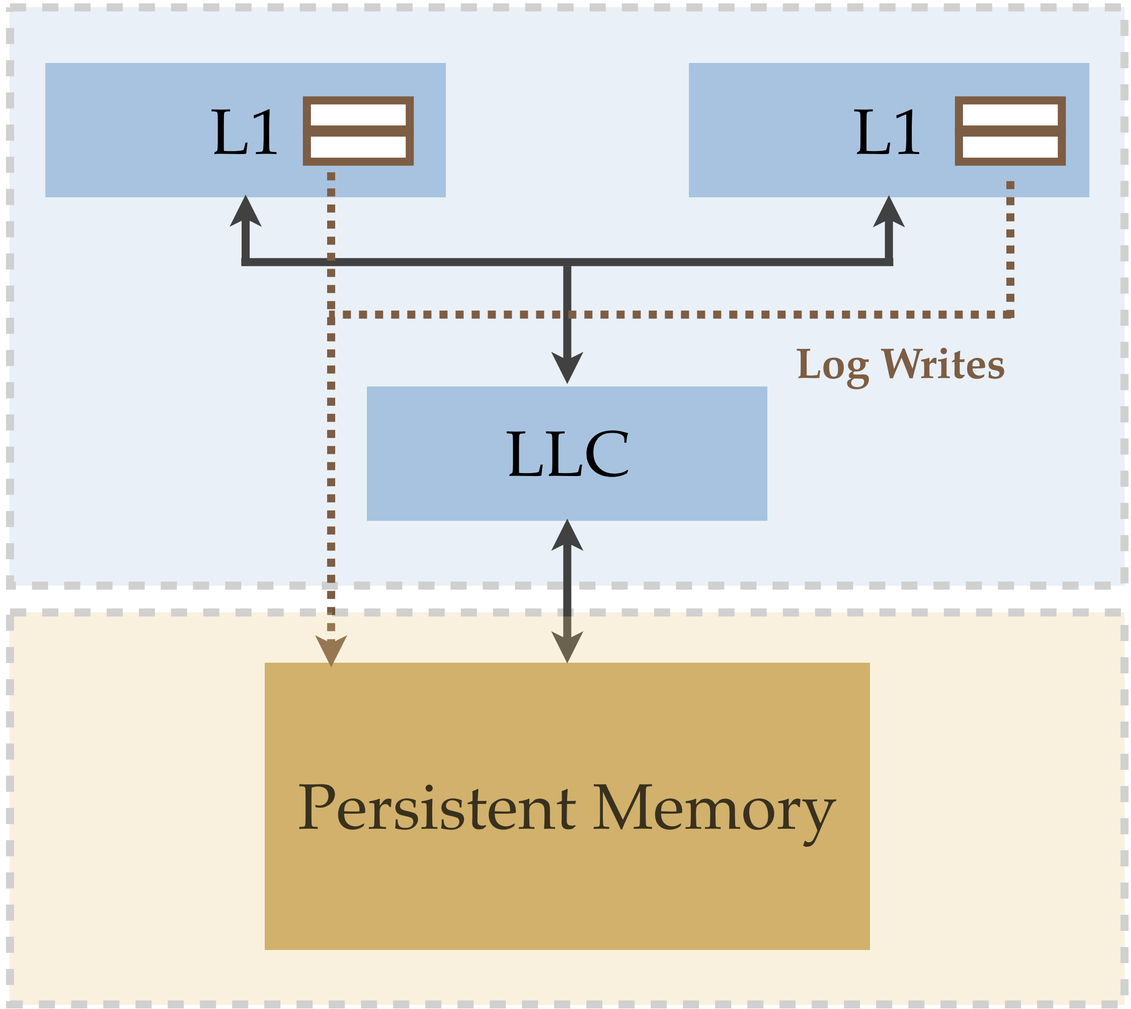
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries
- **Solution: Log Buffer**

DHTM: Log Buffer



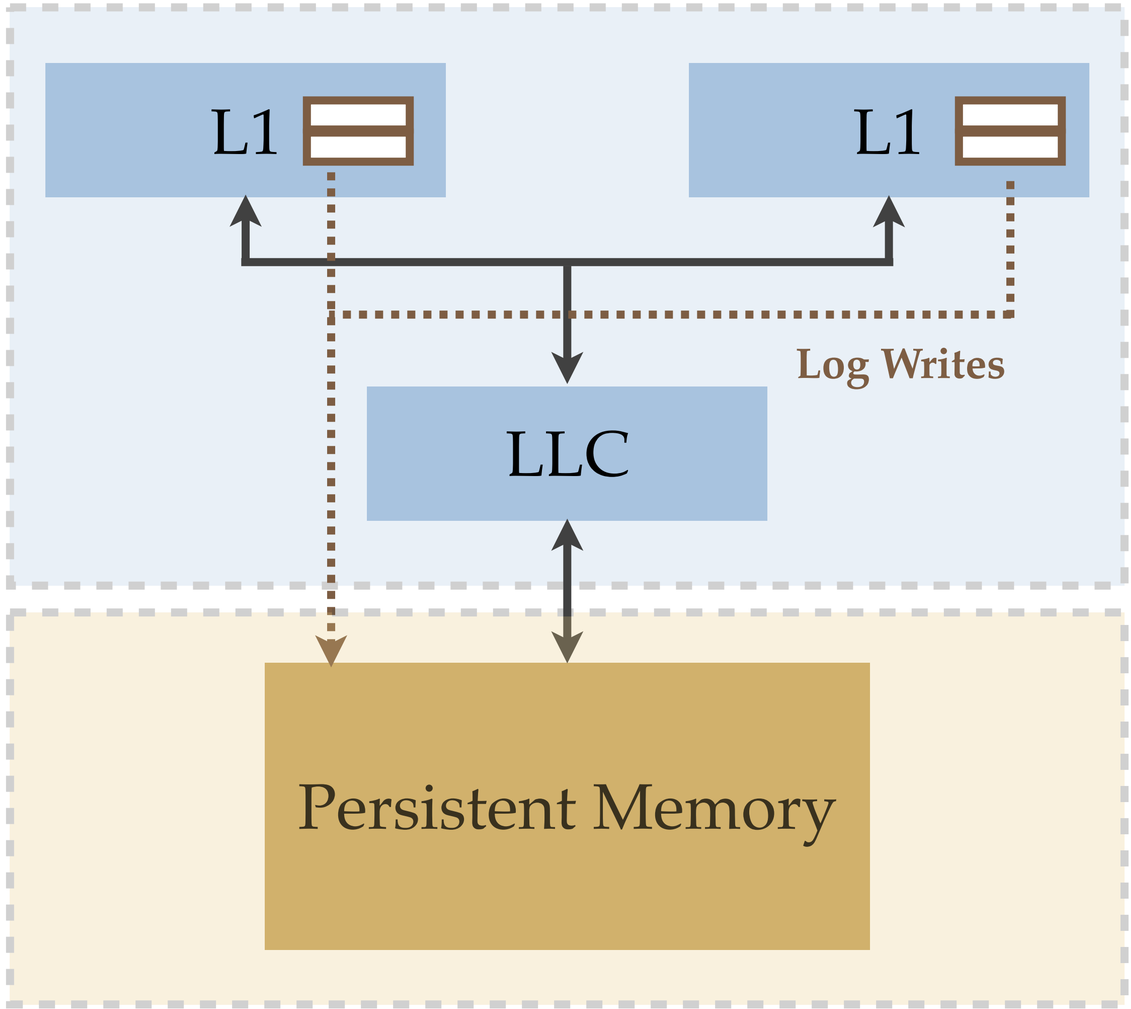
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries
- **Solution: Log Buffer**
  - track cache lines being modified

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

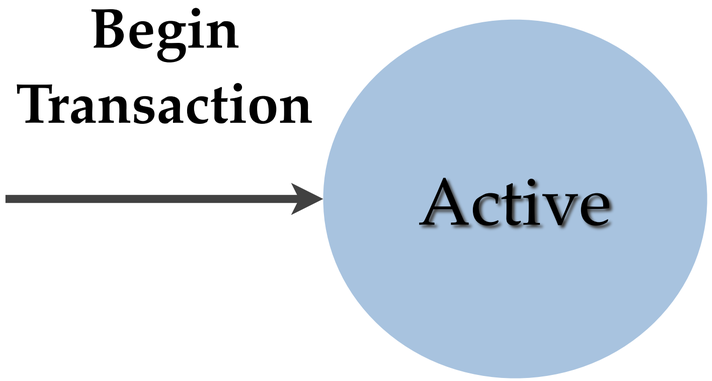
DHTM: Log Buffer



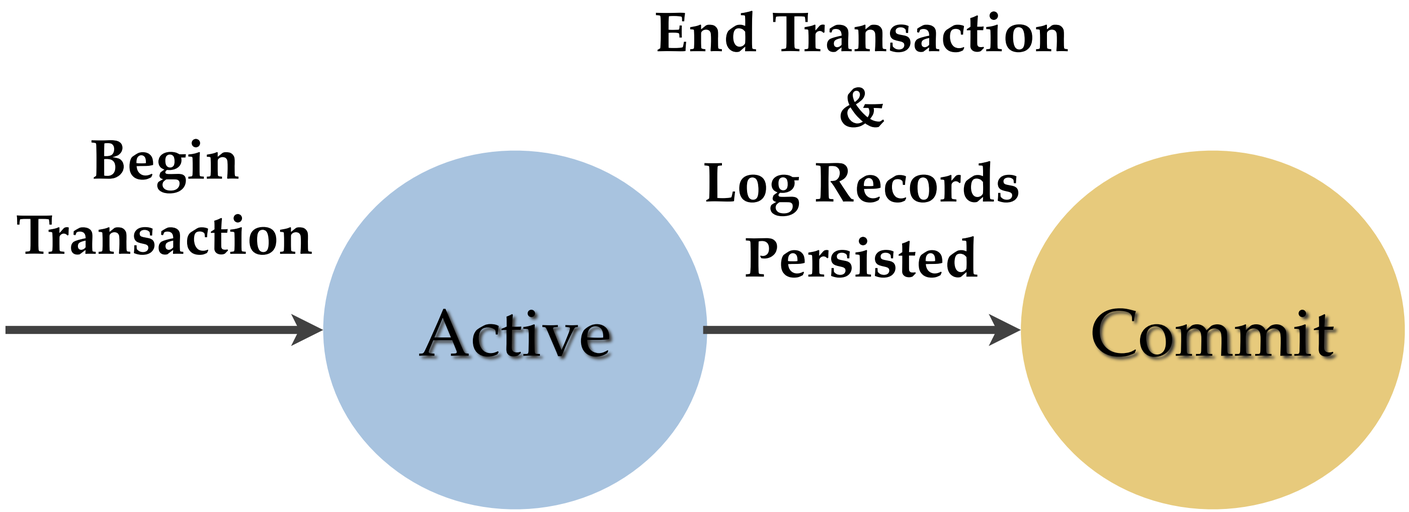
- **Redo Log Bandwidth Problem**
  - write a log entry for every store
  - multiple stores create multiple log entries
- **Solution: Log Buffer**
  - track cache lines being modified
  - multiple writes coalesced in a log entry
  - log entry written to persistent memory on eviction from log buffer

DHTM: Transaction States

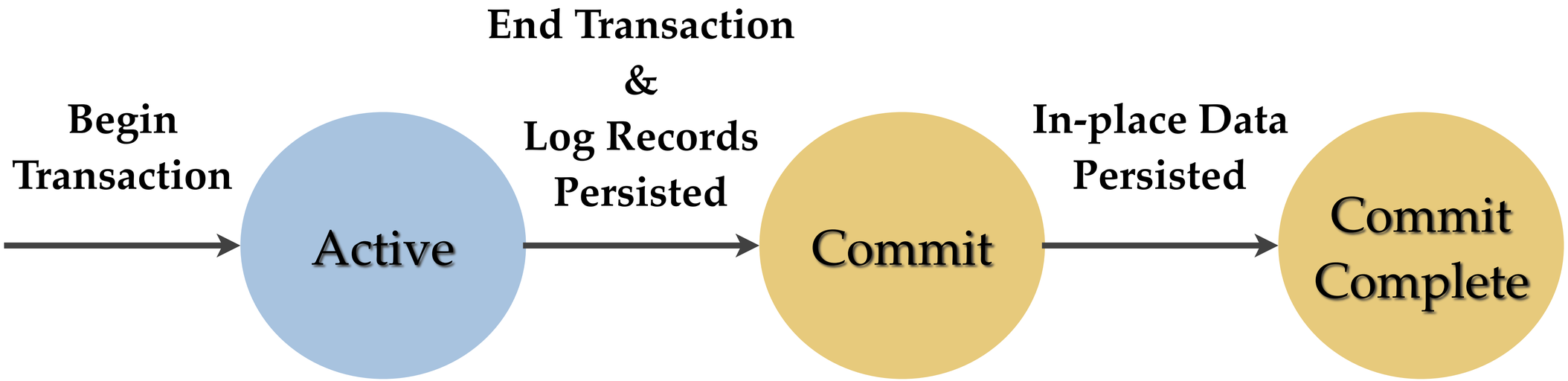
DHTM: Transaction States



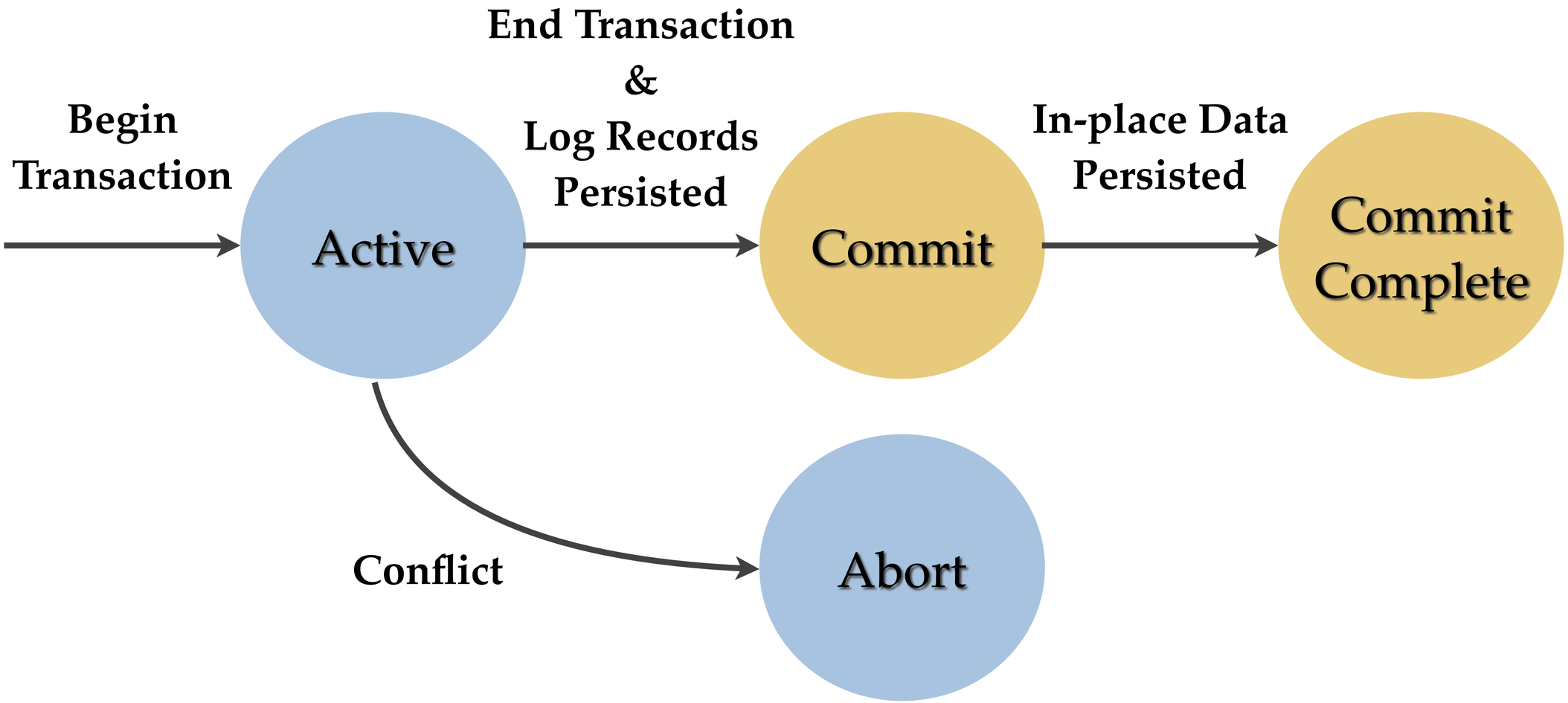
DHTM: Transaction States



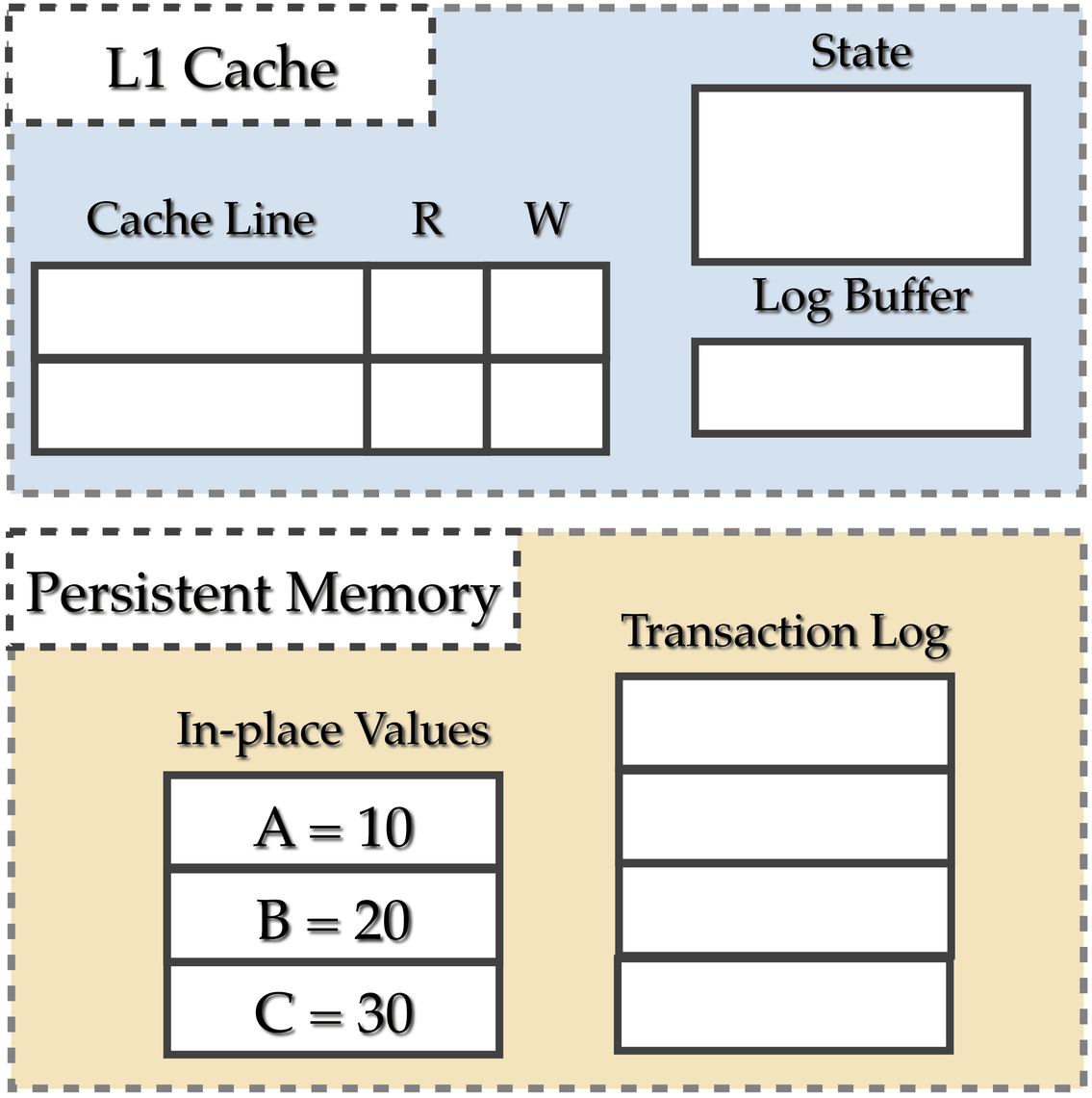
DHTM: Transaction States



DHTM: Transaction States

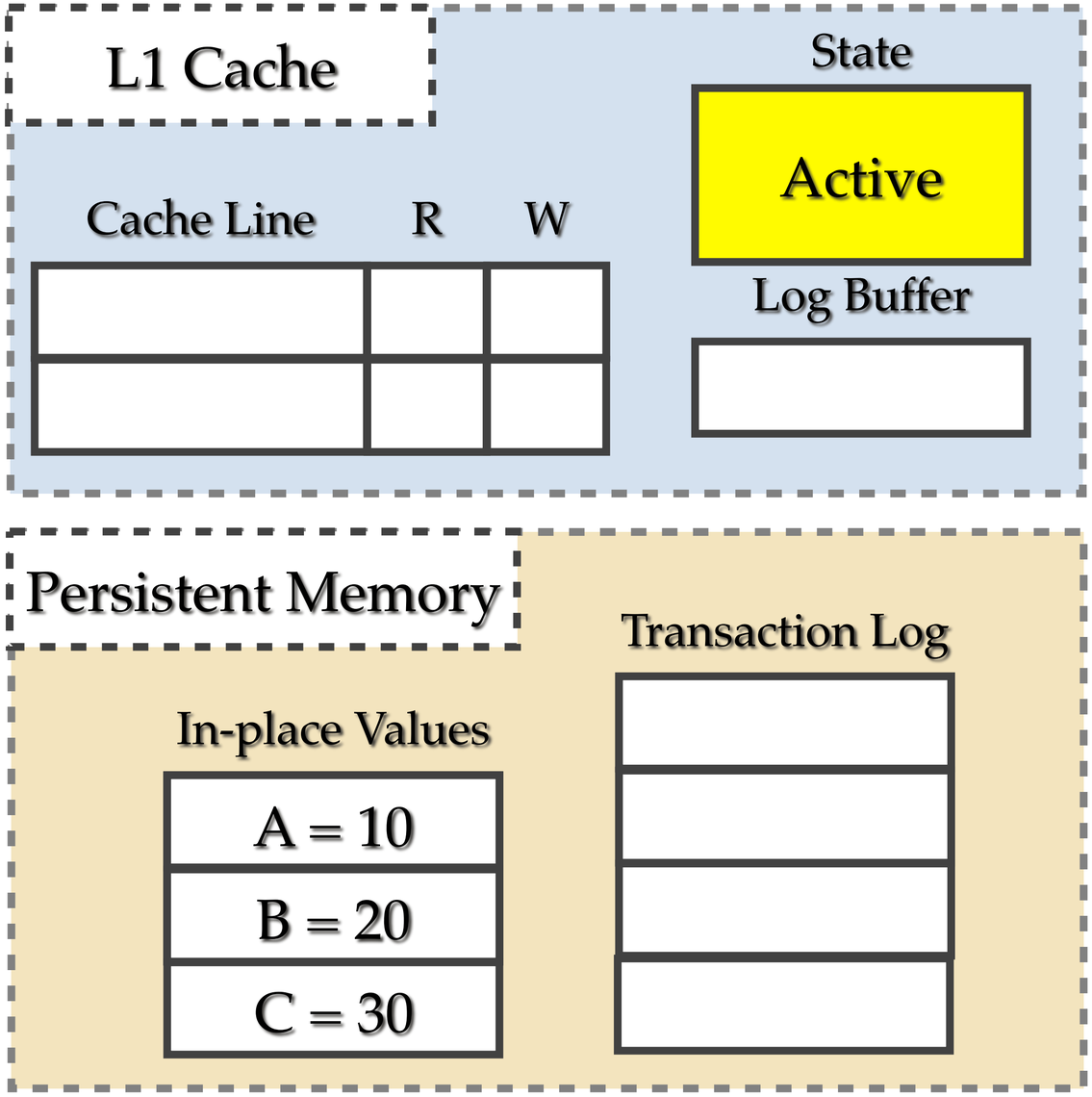


DHTM: Commit Example



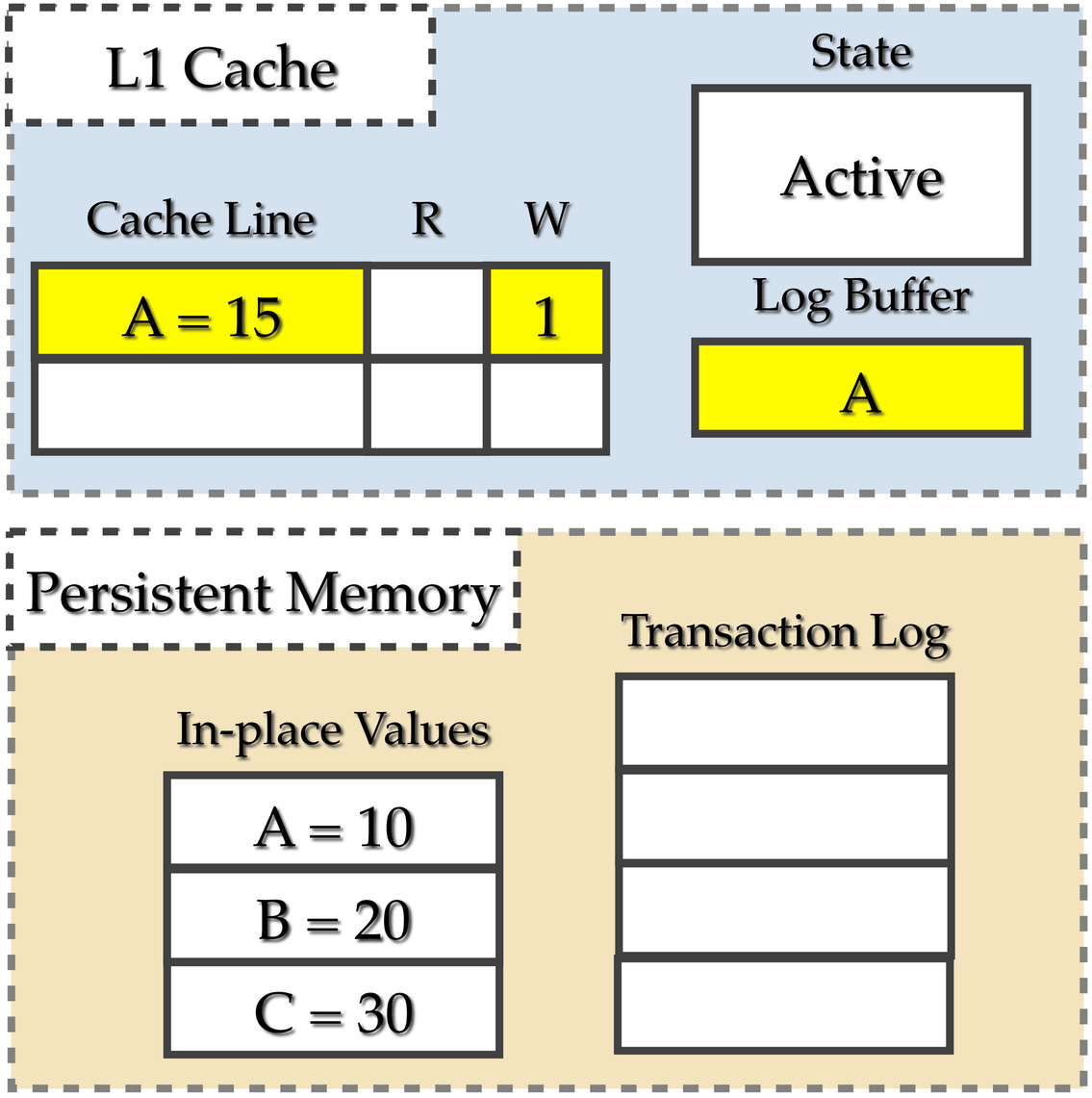

```
Begin_Transaction
      Write (A=15)
      Read  (B)
      Write (B=25)
End_Transaction
```

DHTM: Commit Example



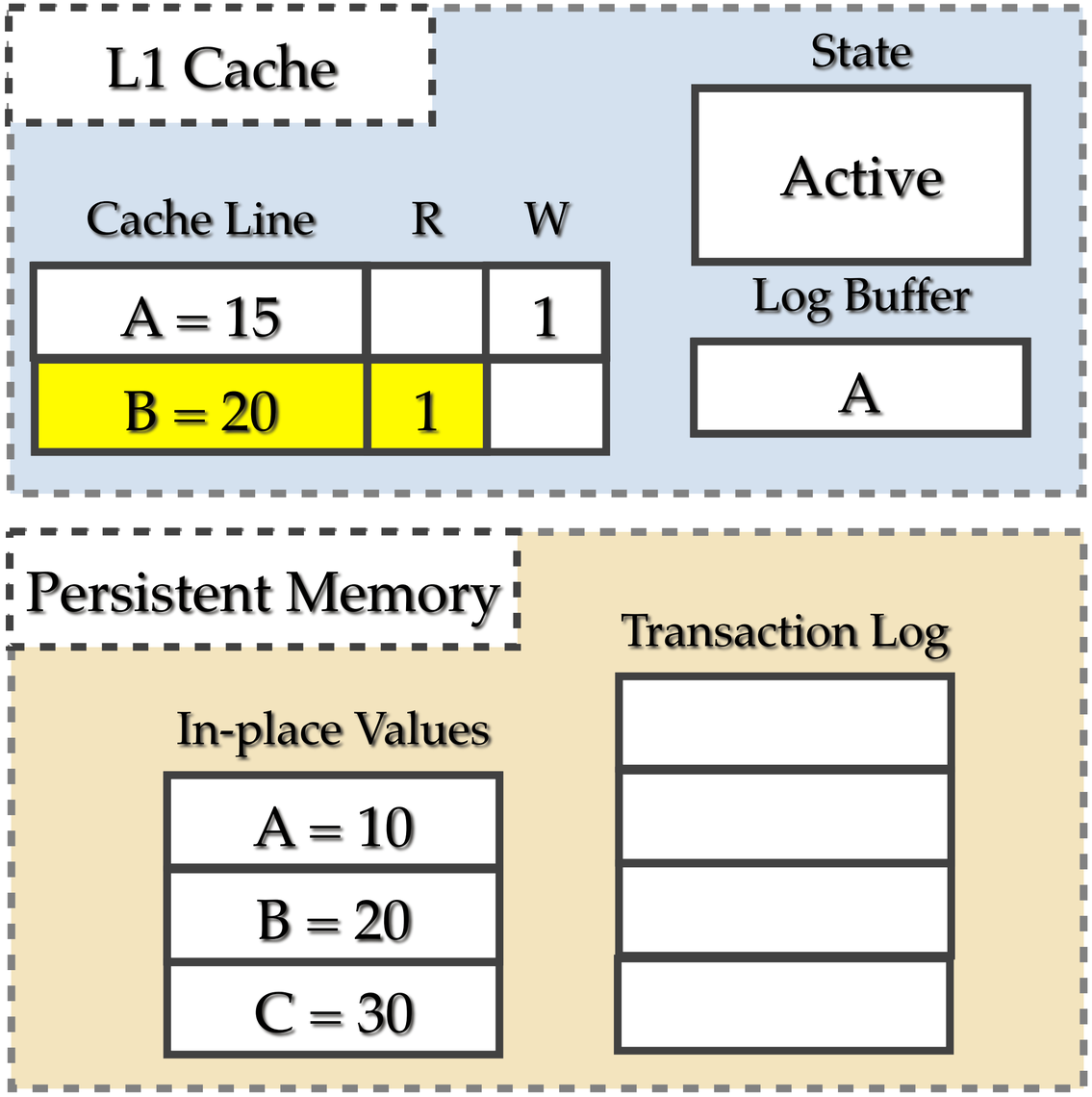

```
Begin_Transaction
      Write (A=15)
      Read  (B)
      Write (B=25)
End_Transaction
```

DHTM: Commit Example



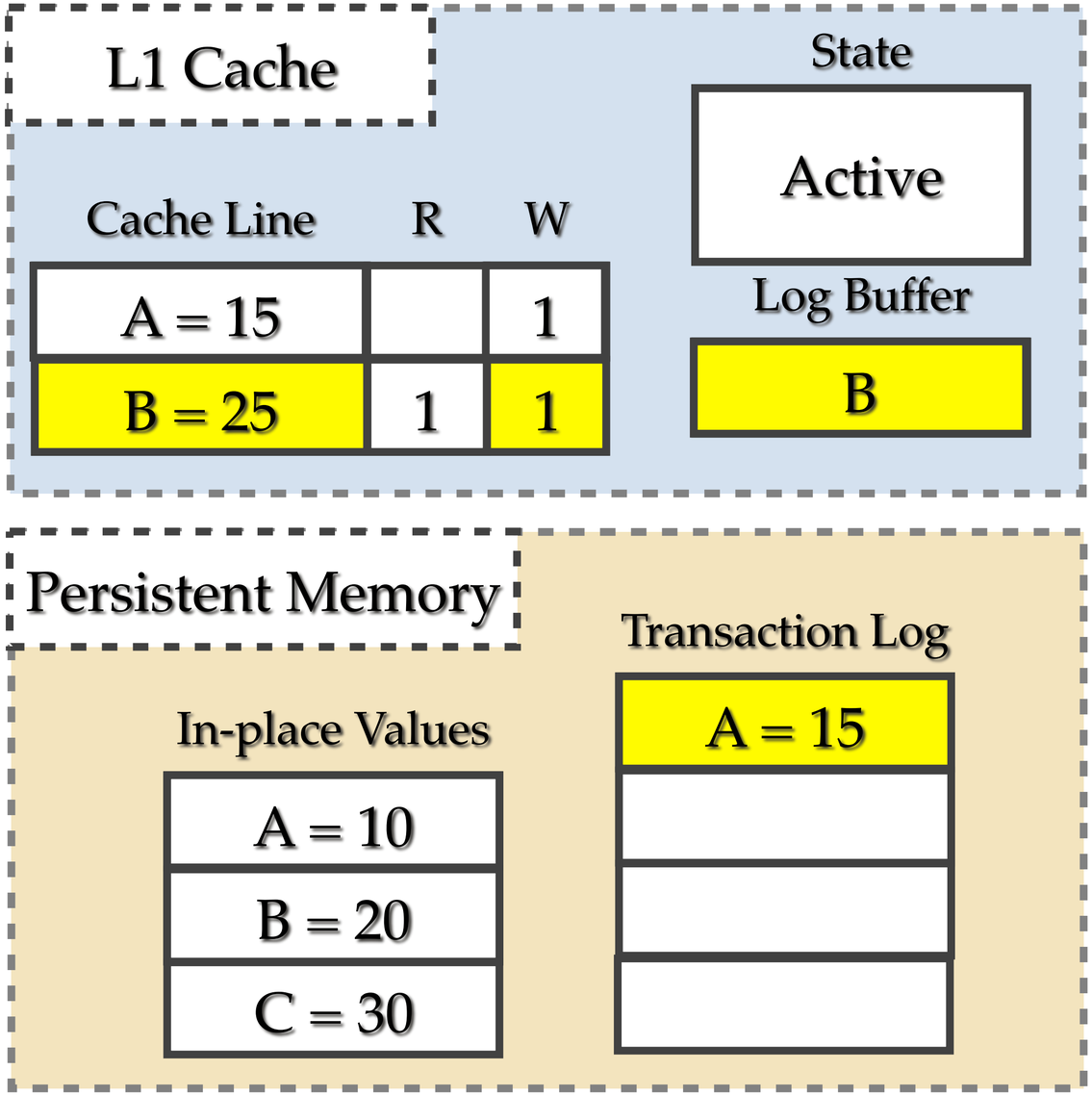

```
Begin_Transaction
      Write (A=15)
      Read  (B)
      Write (B=25)
End_Transaction
```

DHTM: Commit Example



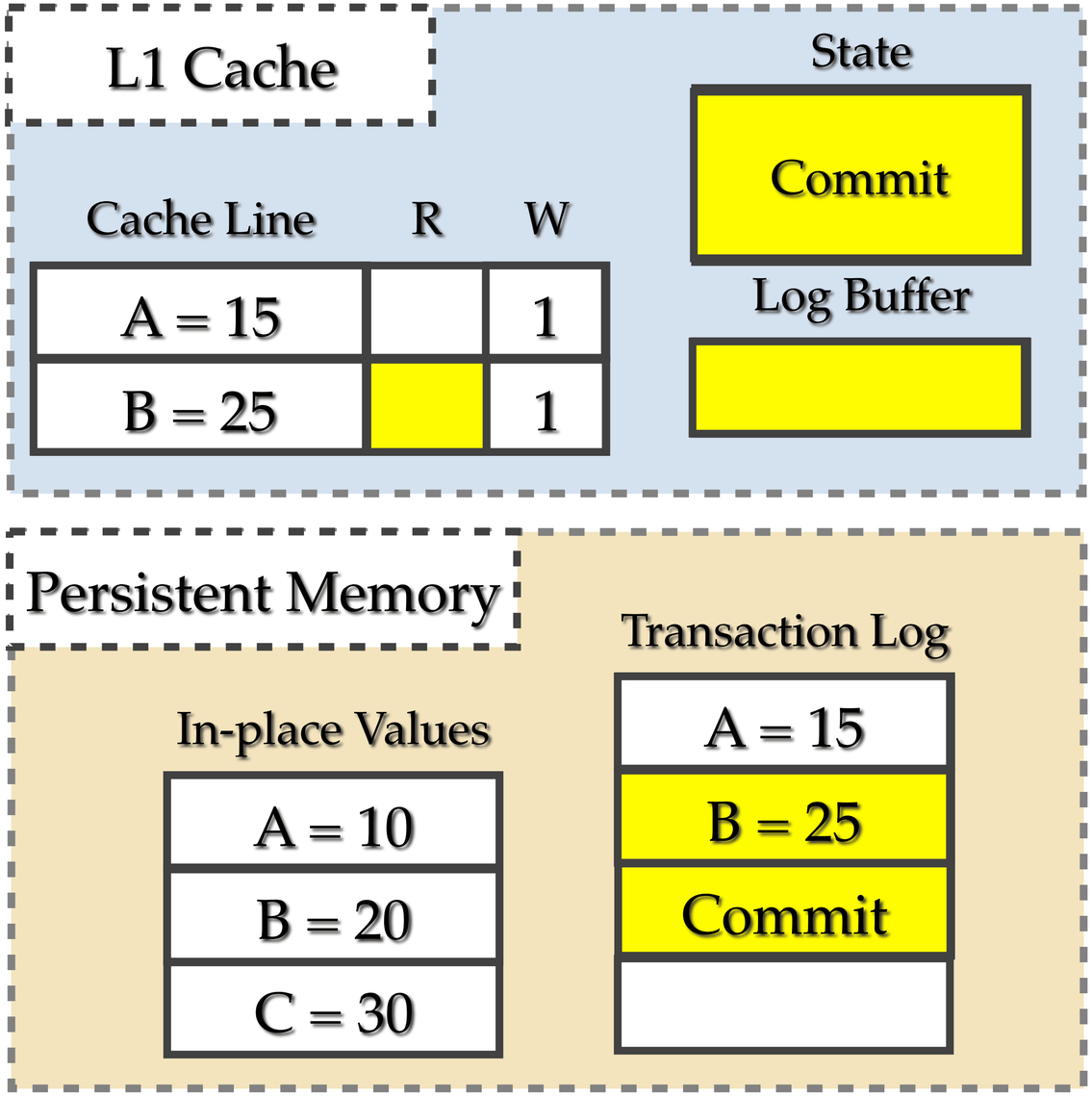

```
Begin_Transaction
      Write (A=15)
      Read  (B)
      Write (B=25)
End_Transaction
```

DHTM: Commit Example



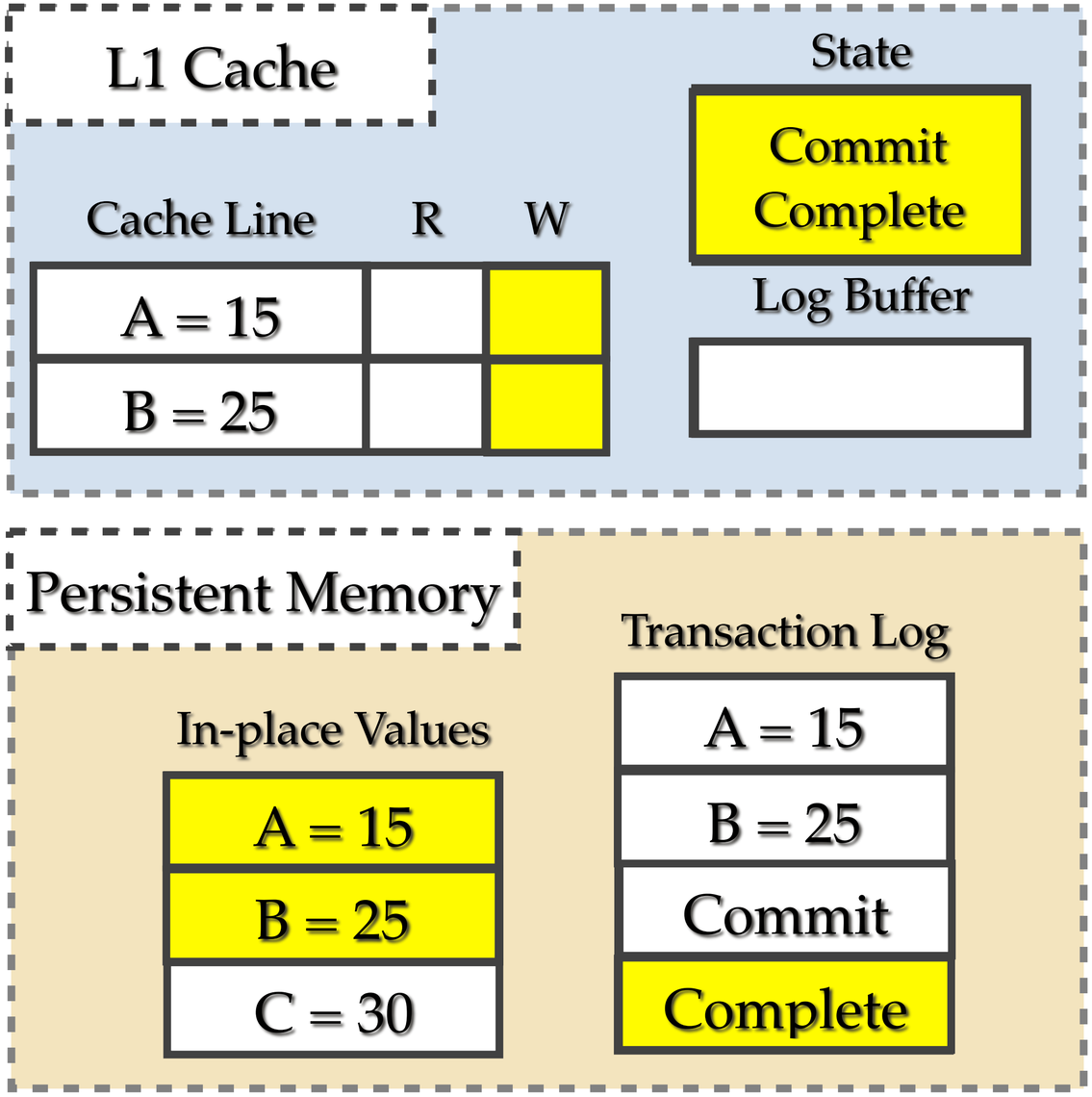

```
Begin_Transaction
      Write (A=15)
      Read  (B)
      Write (B=25)
End_Transaction
```

DHTM: Commit Example




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

DHTM: Commit Example




```
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

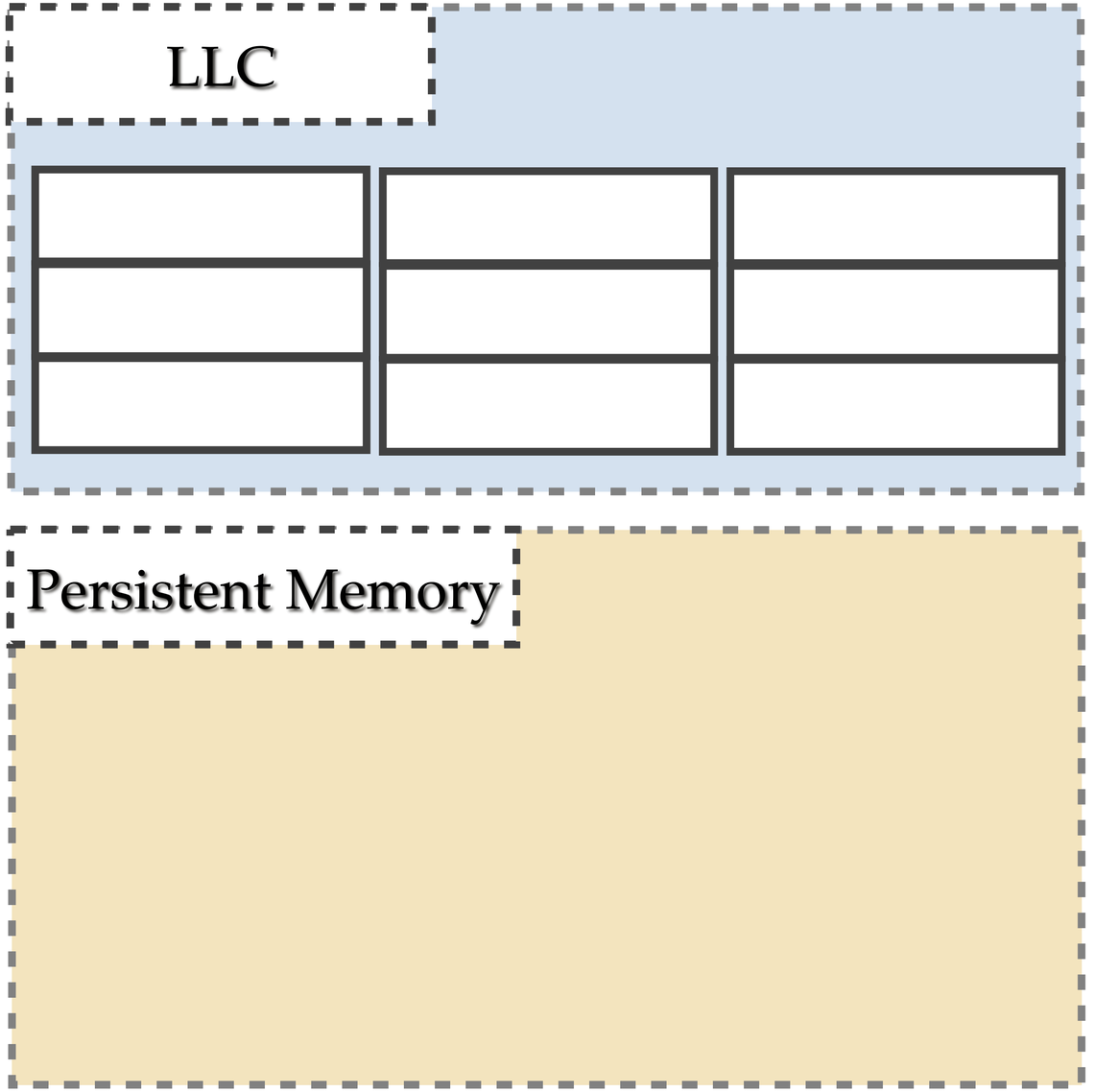
DHTM: Supporting Overflow

DHTM: Supporting Overflow

- Solution

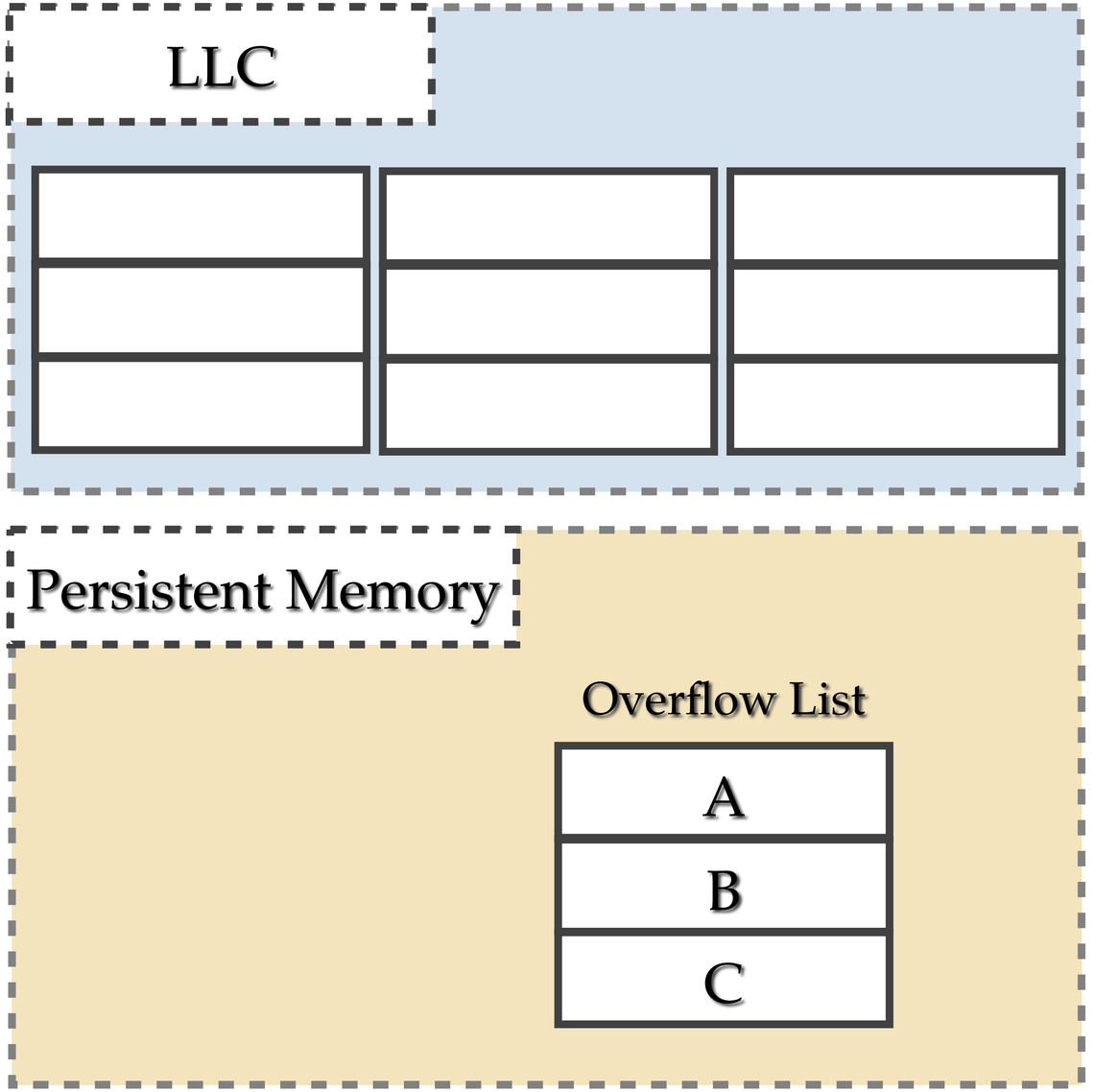
DHTM: Supporting Overflow

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



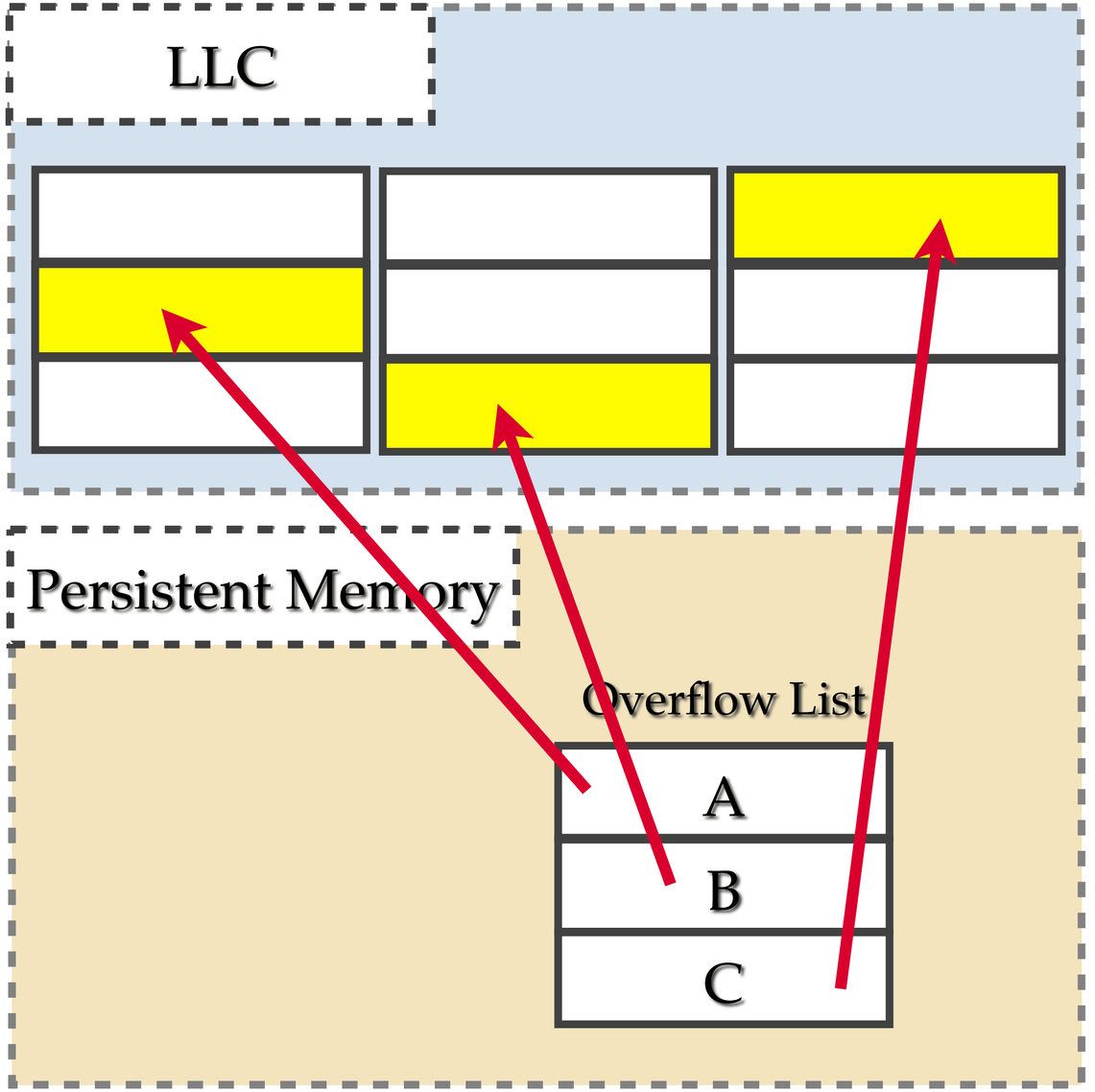
DHTM: Supporting Overflow

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



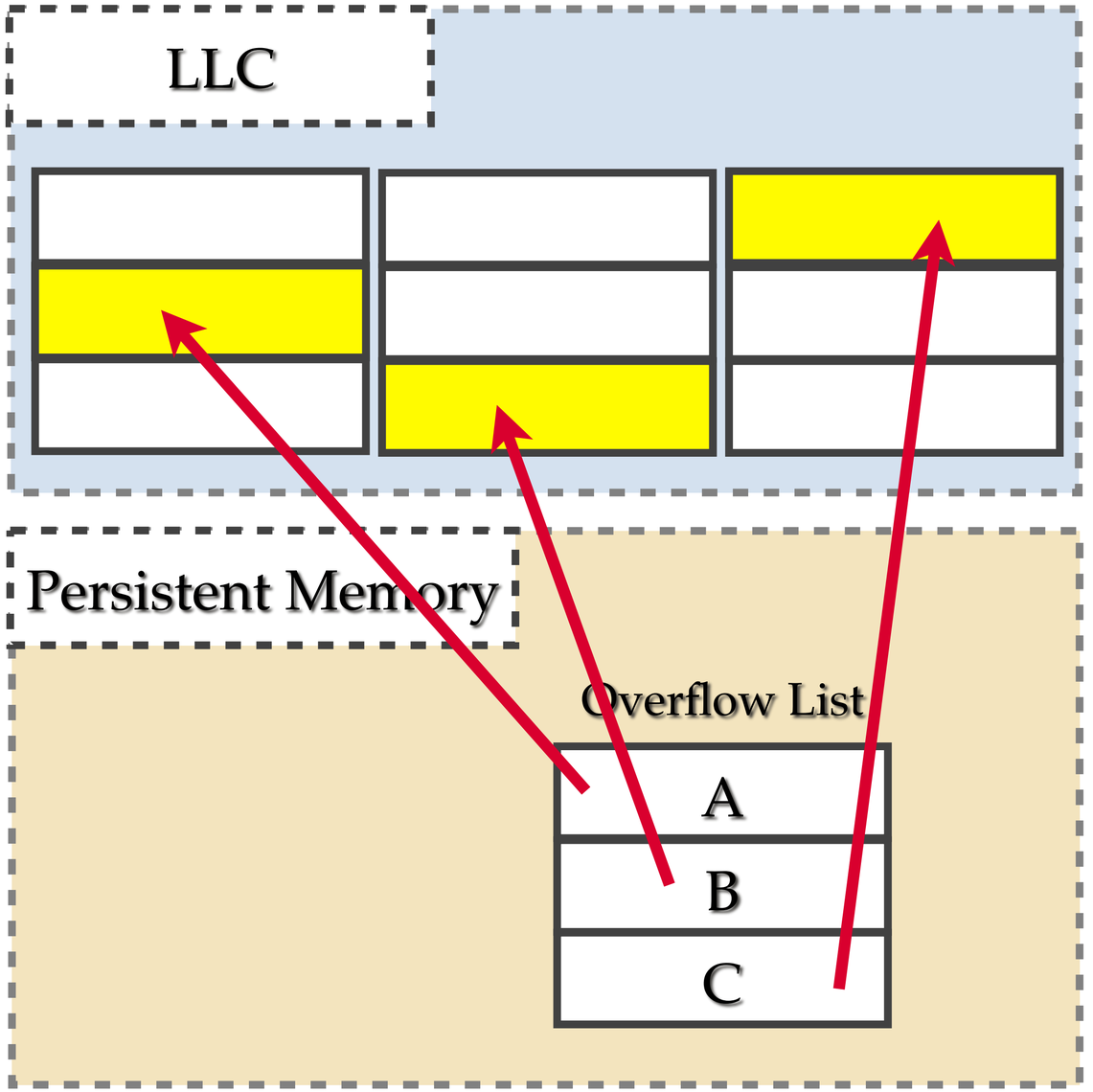
DHTM: Supporting Overflow

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



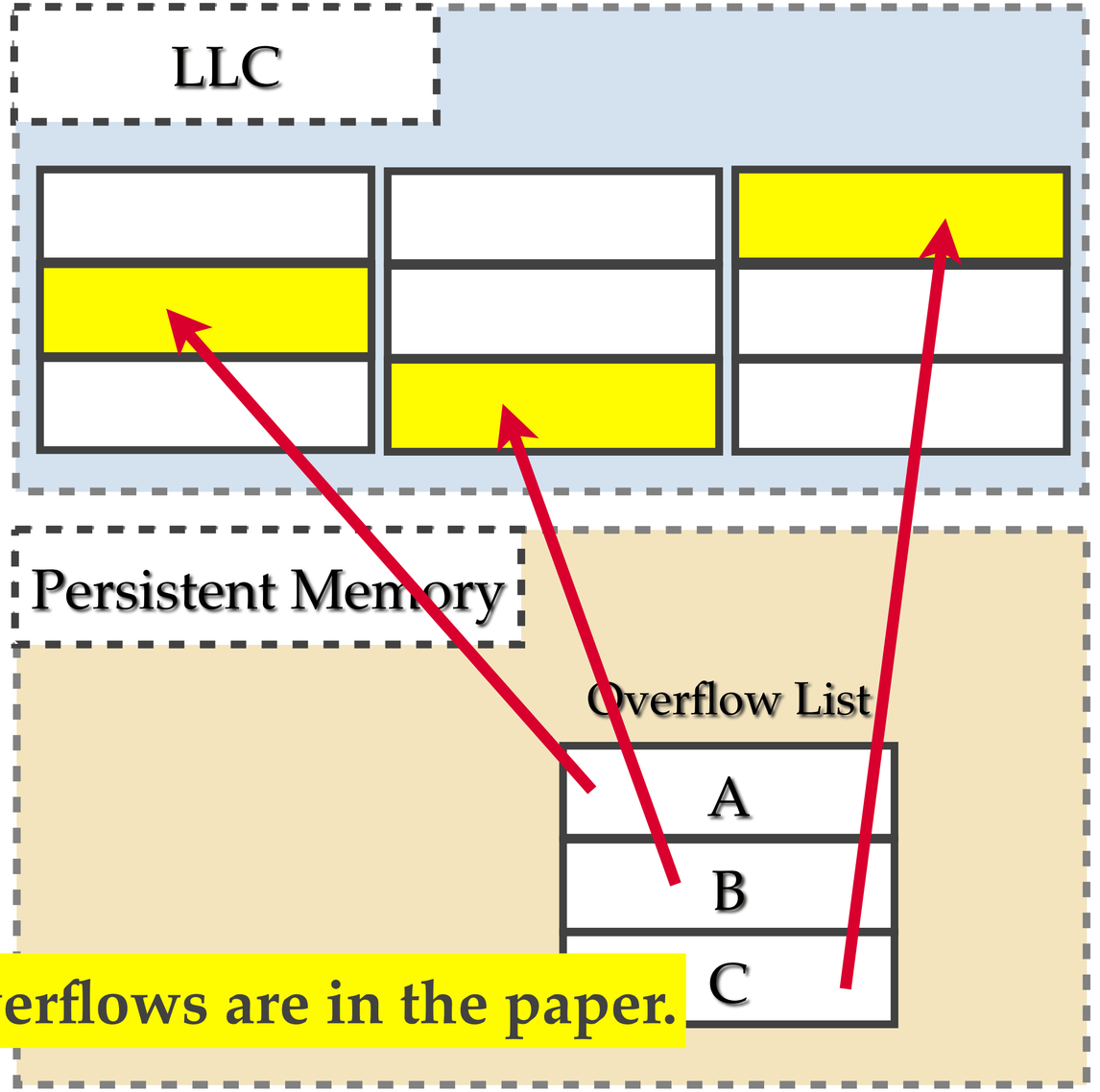
DHTM: Supporting Overflow

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



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



Further details on supporting overflows are in the paper.

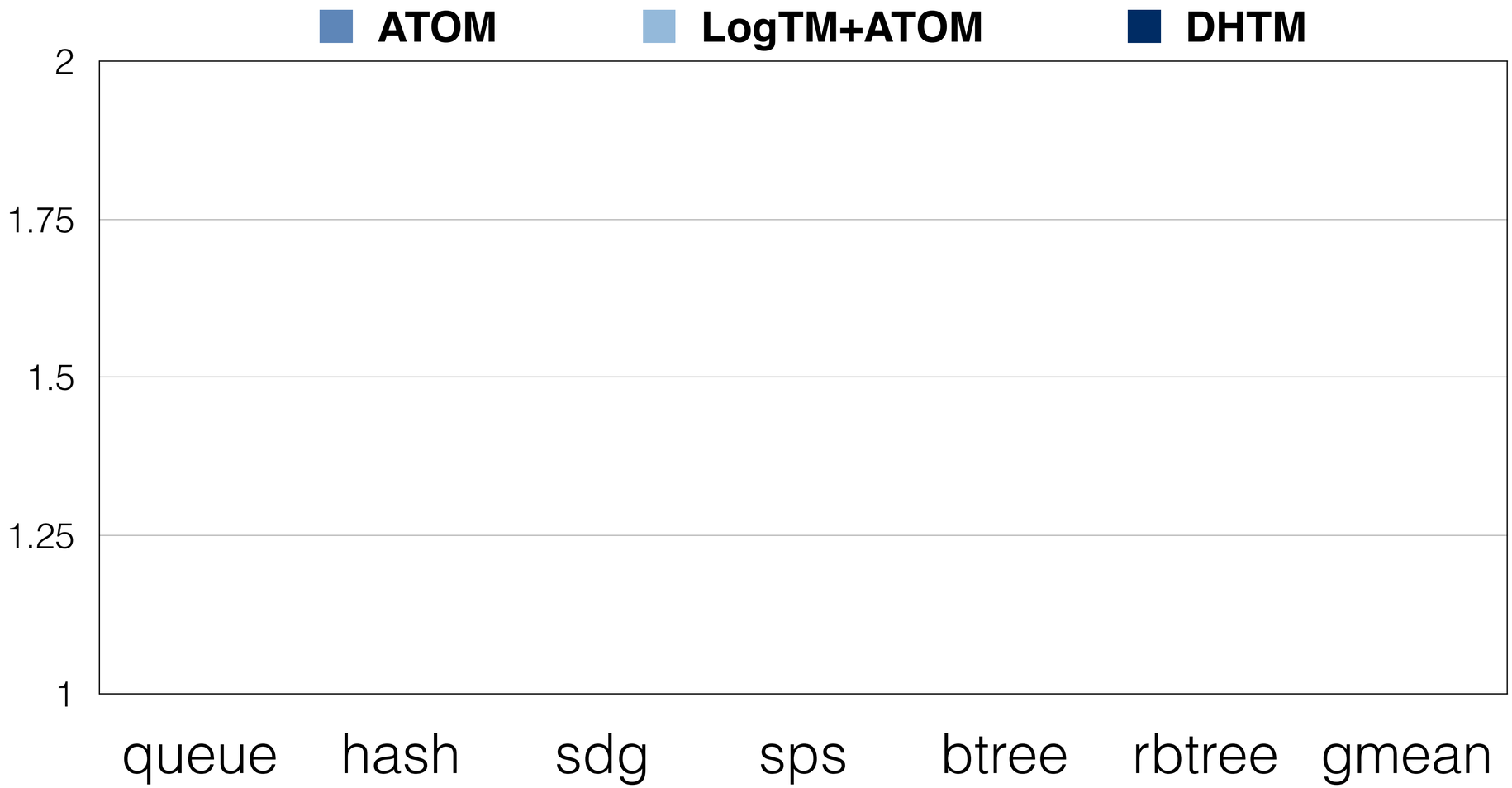
Evaluation

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

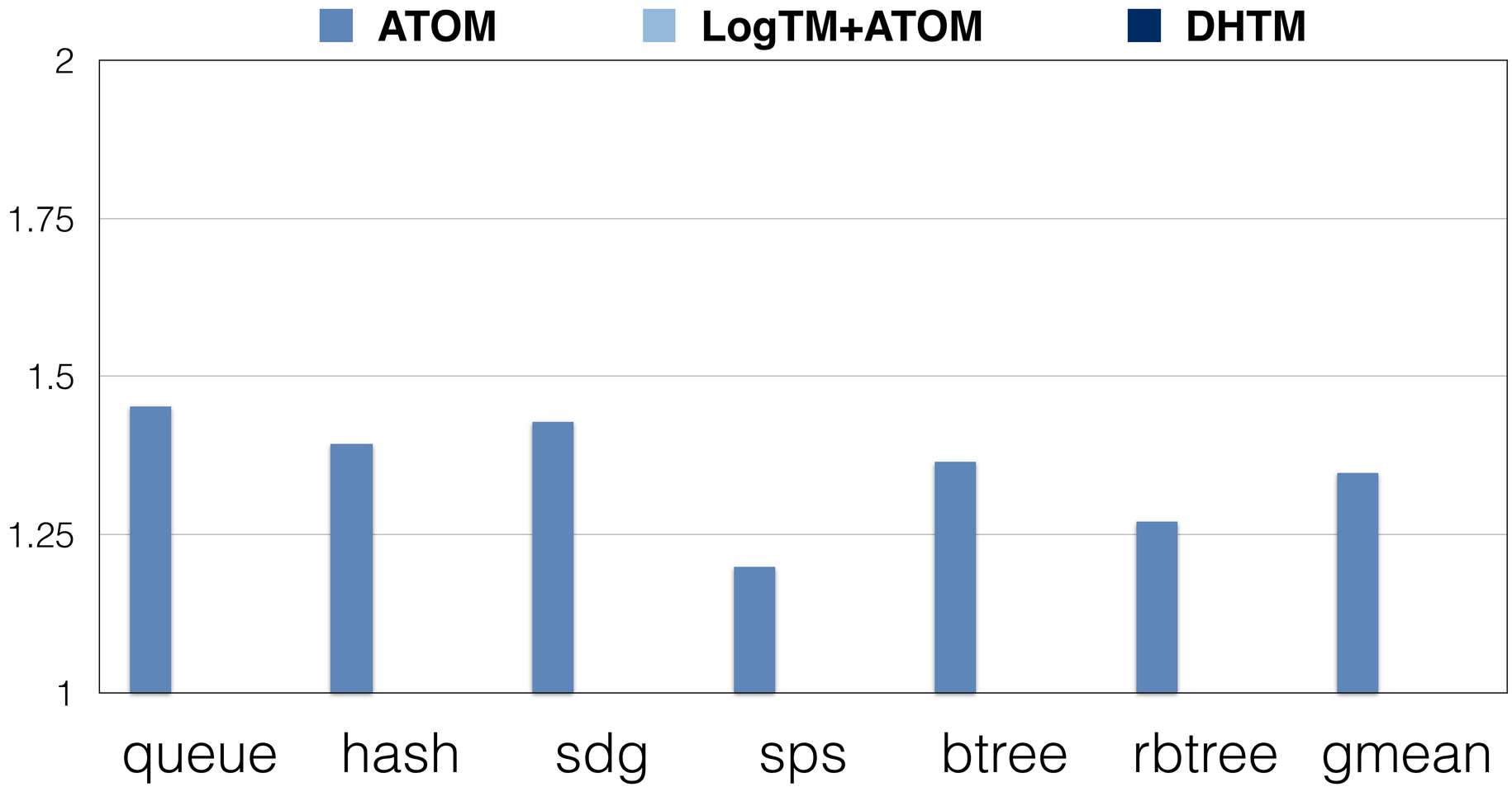
- **System Configuration**
  - We evaluate an 8-core machine with a 2-level cache hierarchy
  - HTM's implement (first) writer wins conflict resolution policy

Evaluation

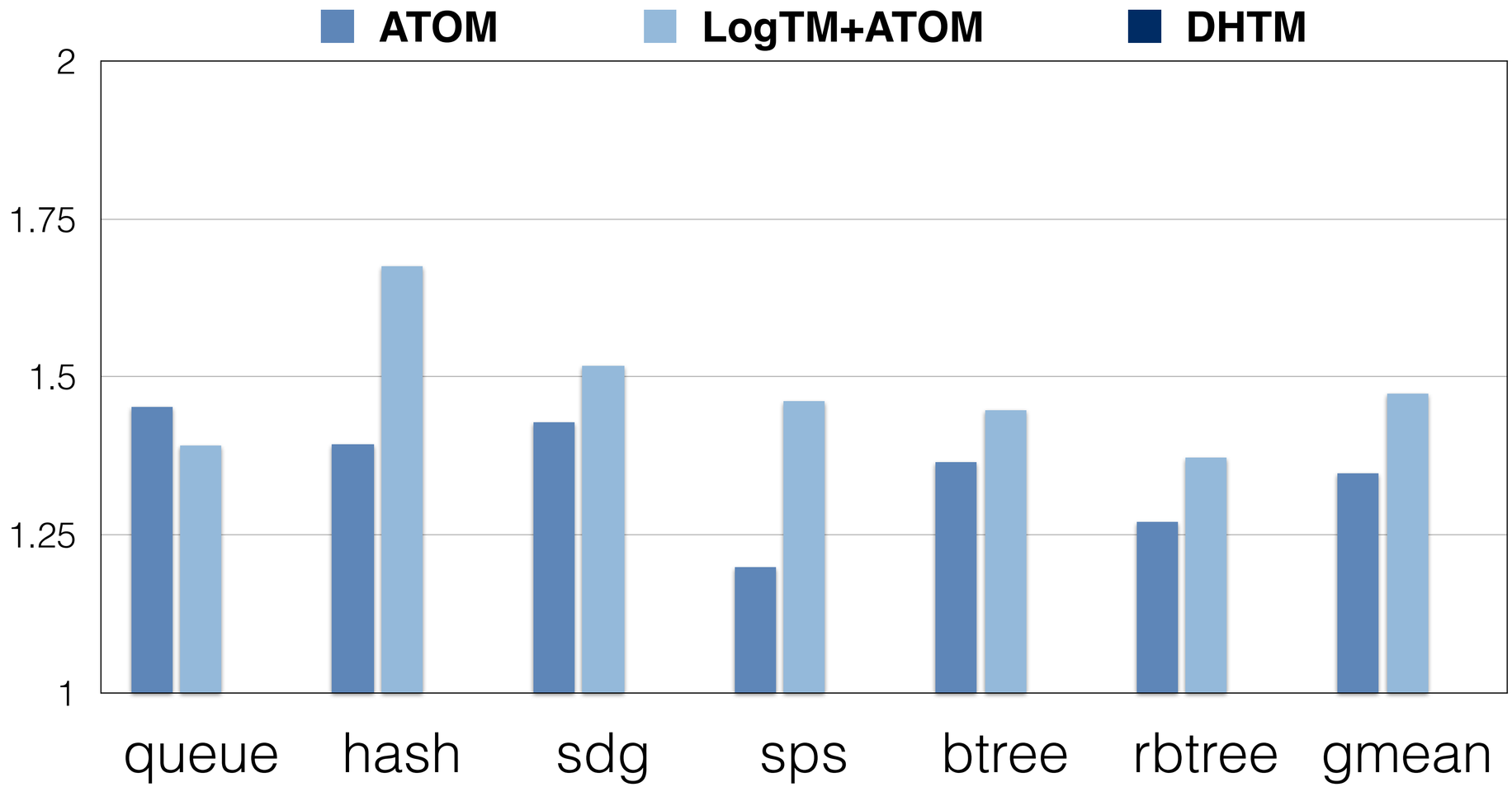
Evaluation


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

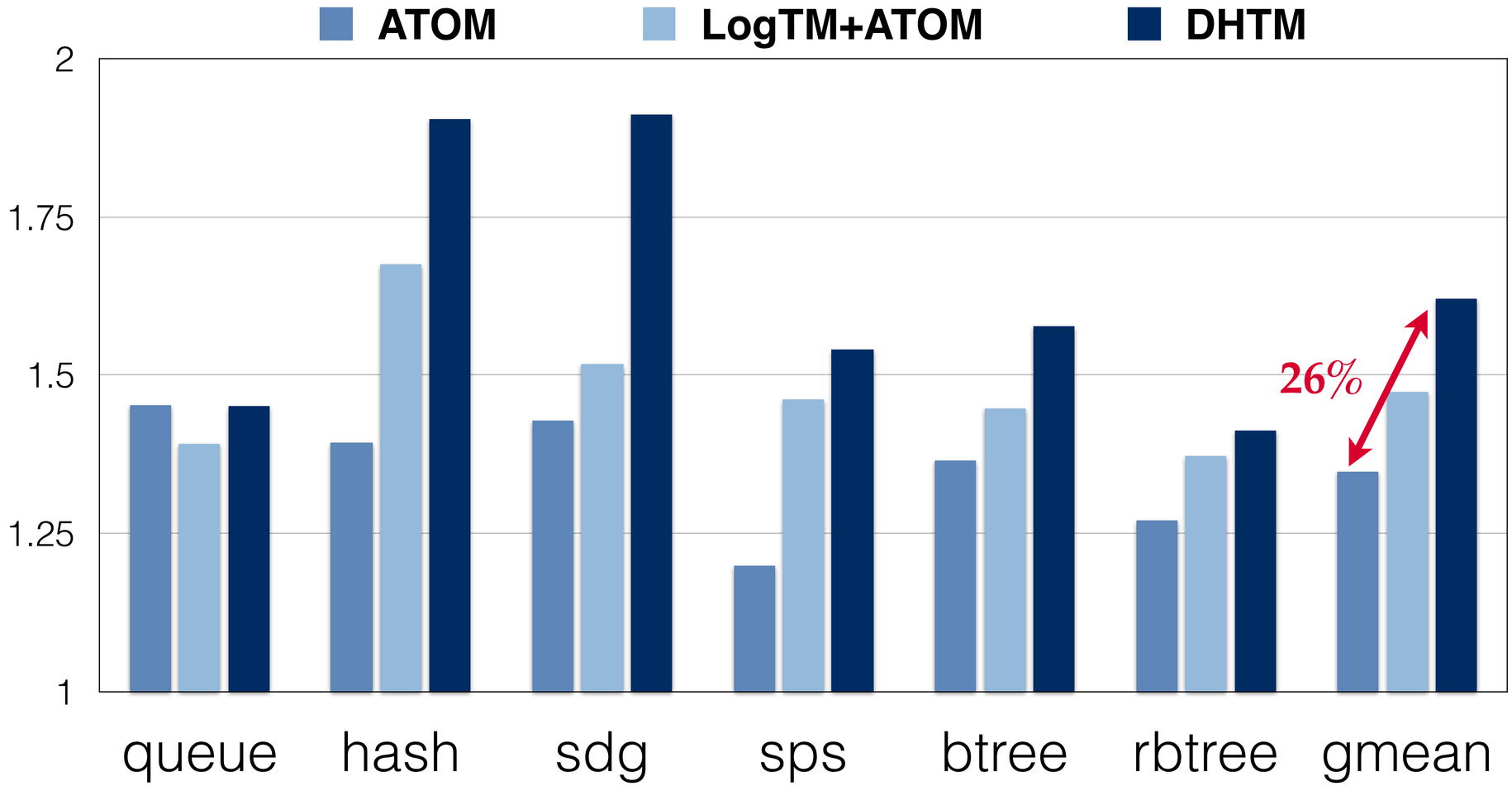
Evaluation


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

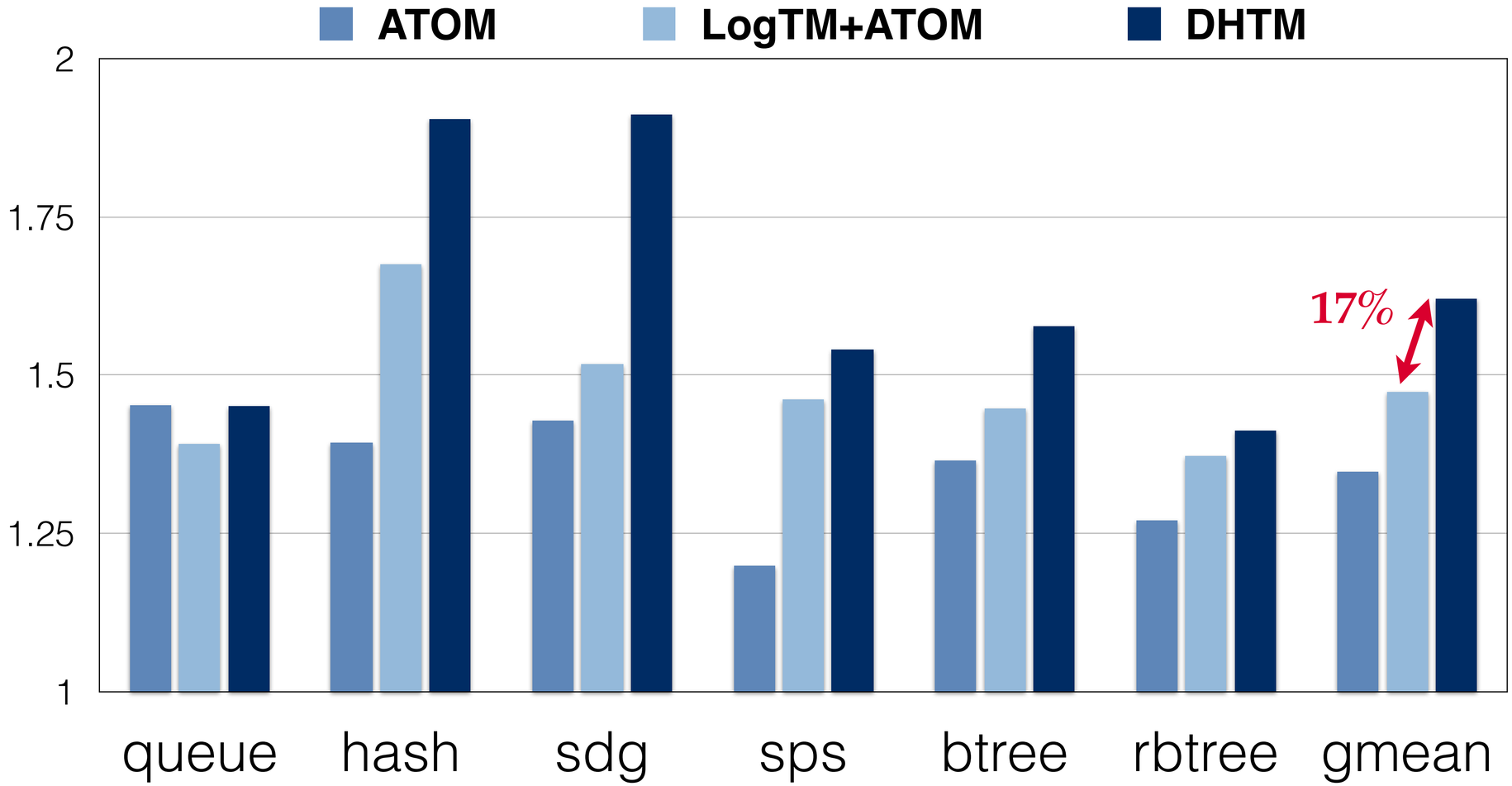
Evaluation


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
26%
queue
hash
sdg
sps
btree
rbtree
gmean

Evaluation


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

Conclusion

- Persistent memory systems require crash consistency
- ACID Transactions: widely understood 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

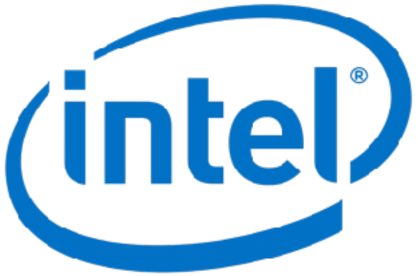
DHTM: Durable Hardware Transactional Memory

Arpit Joshi, *Vijay Nagarajan, Marcelo Cintra, Stratis Viglas*

ISCA 2018

THE UNIVERSITY *of* EDINBURGH
informatics

icsa | Institute for Computing Systems Architecture



Google