MICRO-48

Efficient Persist Barriers for Multicores

Arpit Joshi, Vijay Nagarajan, Marcelo Cintra, Stratis Viglas

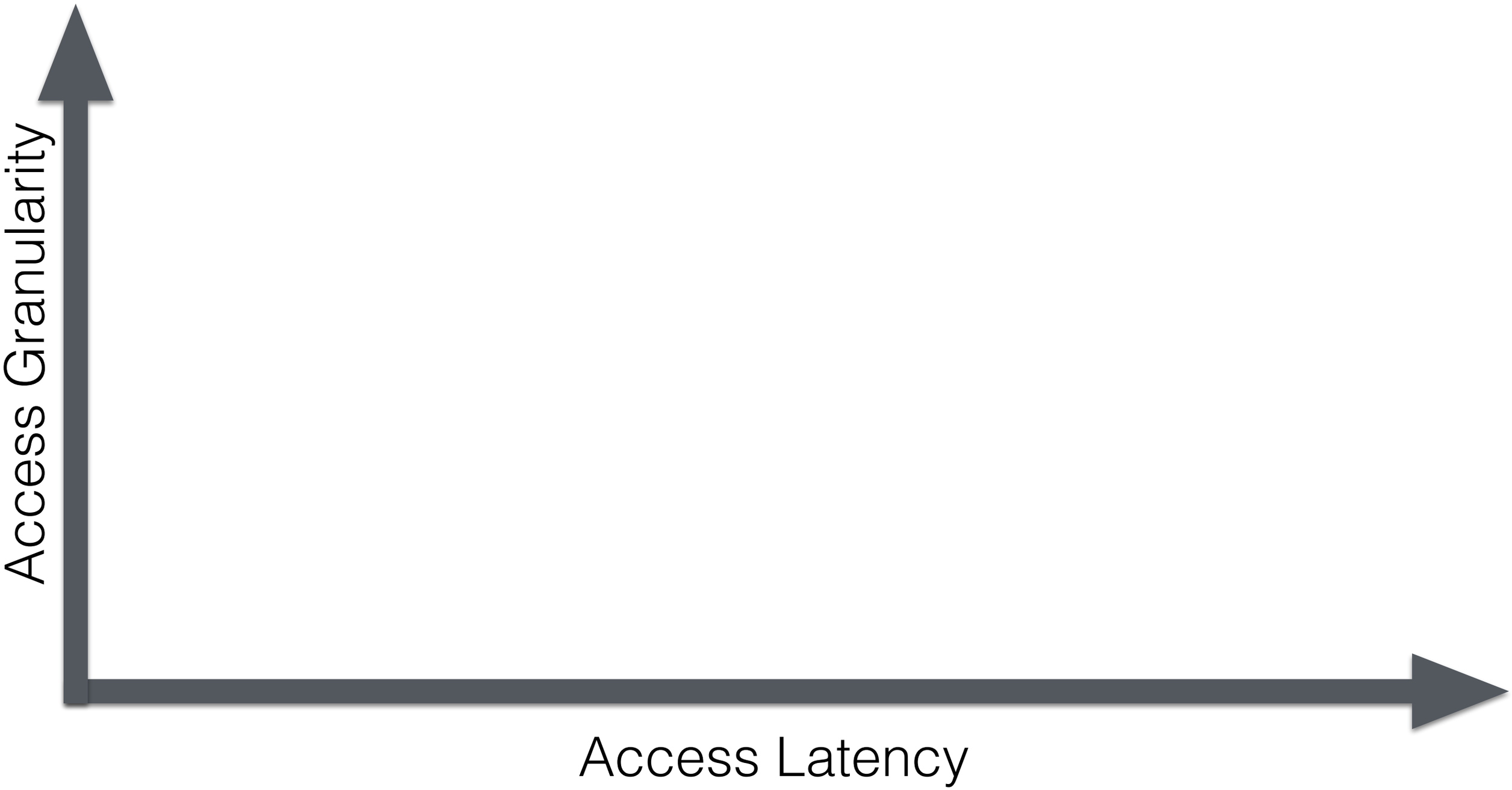
THE UNIVERSITY of EDINBURGH

intel

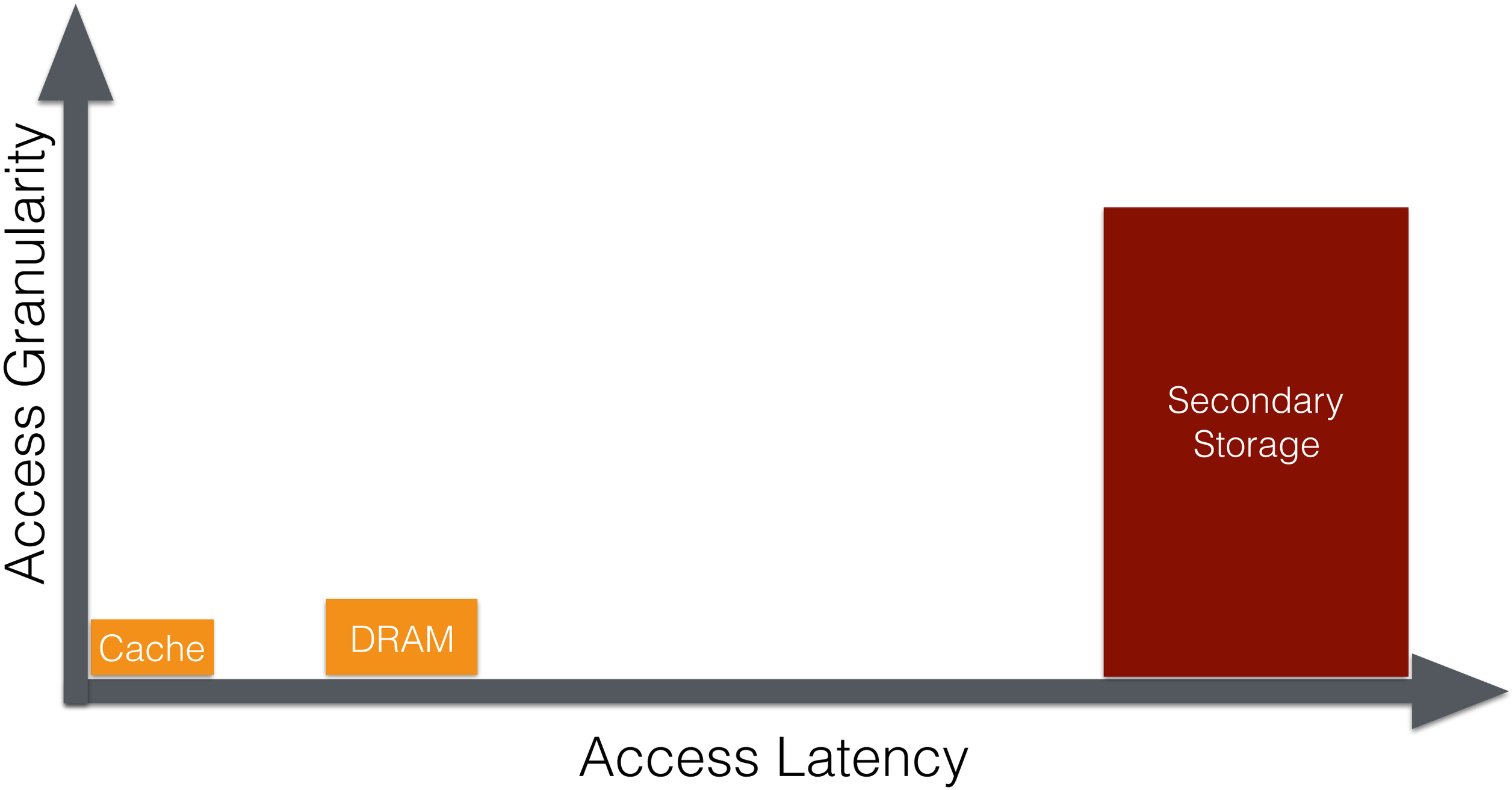
Summary

- Efficient persist barrier
- Used to implement persistency models
  - Persistency = when stores become durable
    (Consistency = when stores become visible)
- Evaluated
  - Buffered Epoch Persistency
  - Bulk Strict Persistency

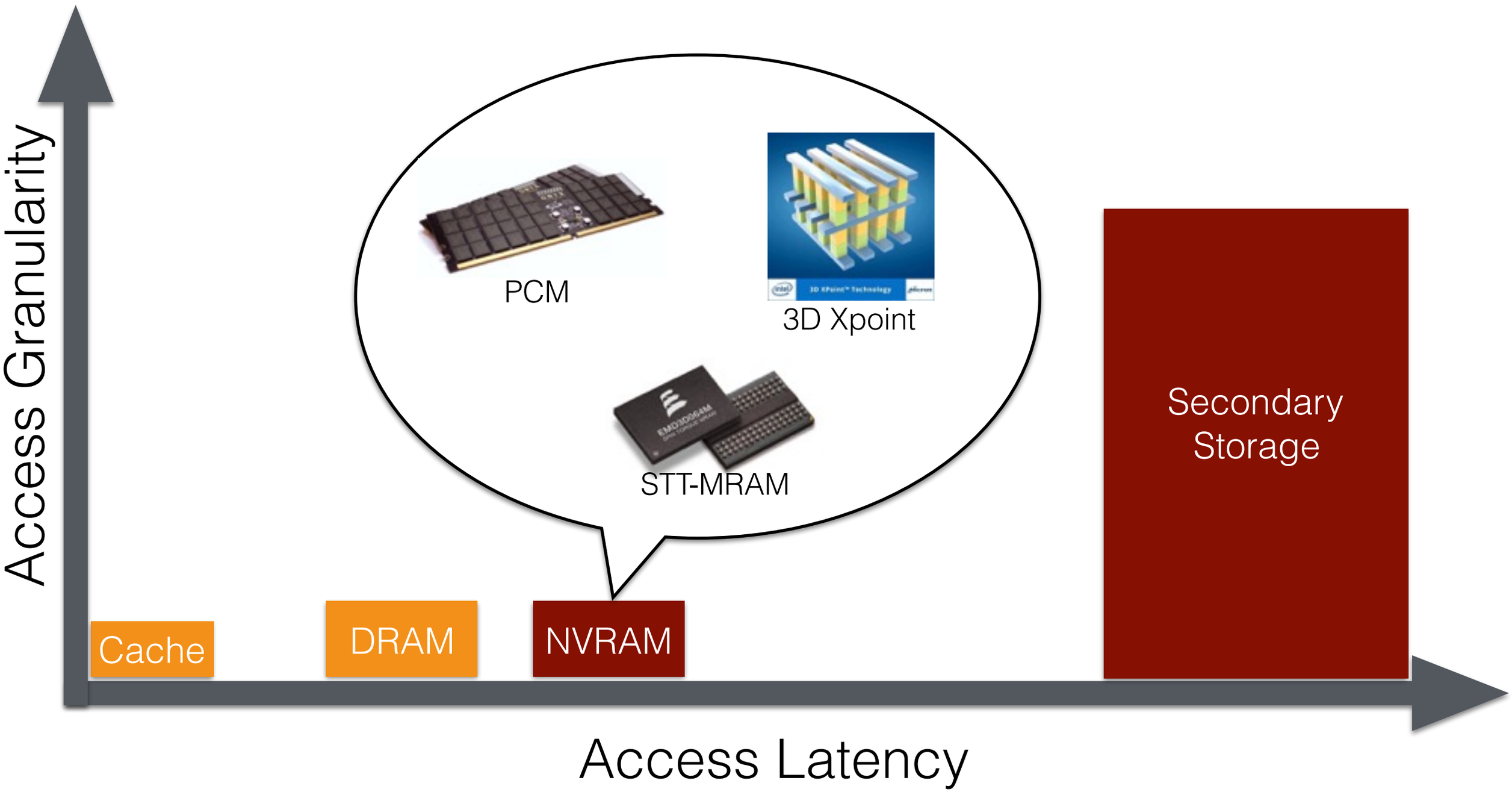
Persistent Memory



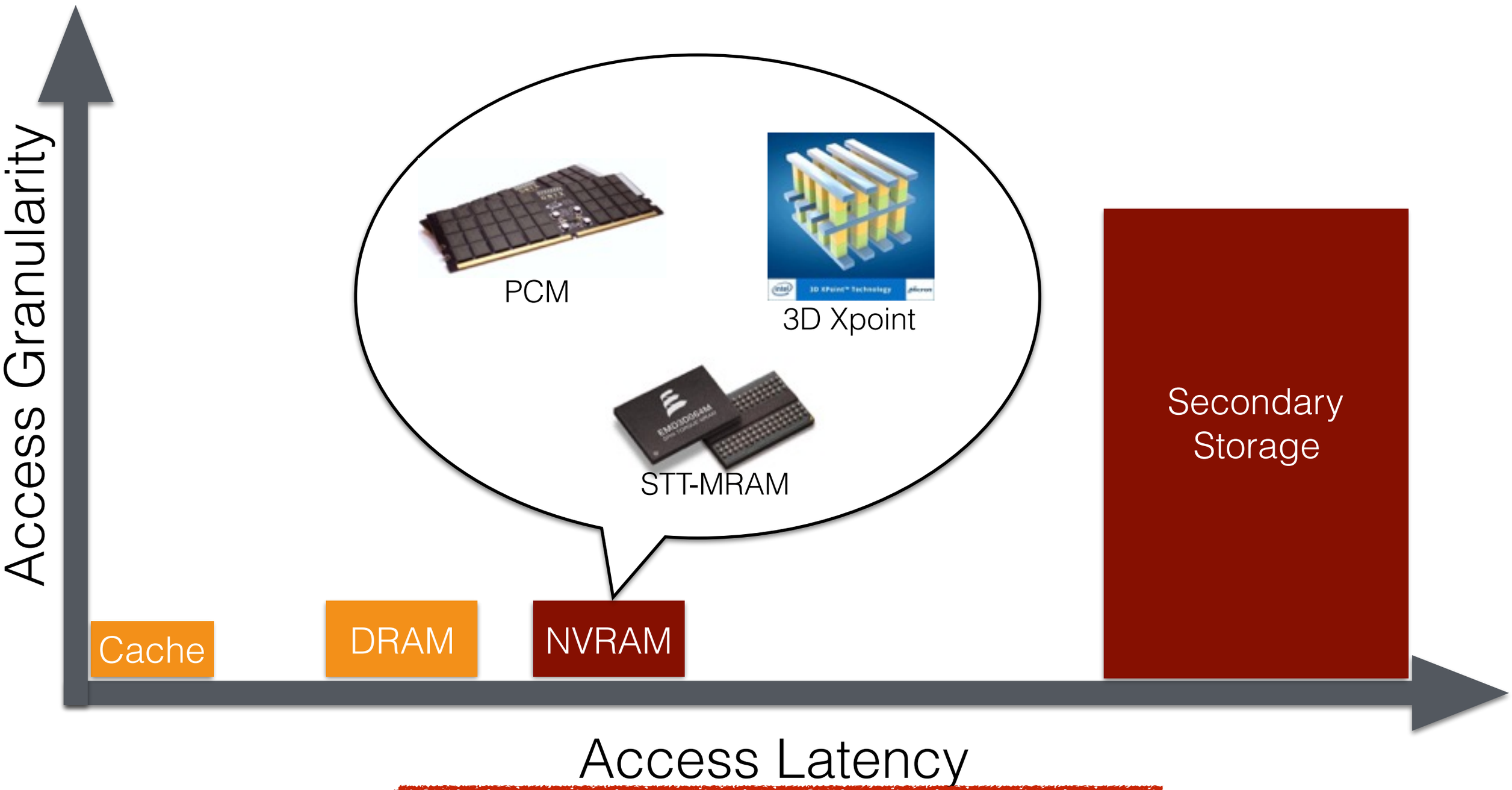
Persistent Memory



Persistent Memory



Persistent Memory



Fast, fine grained persistence.

Persistent Memory

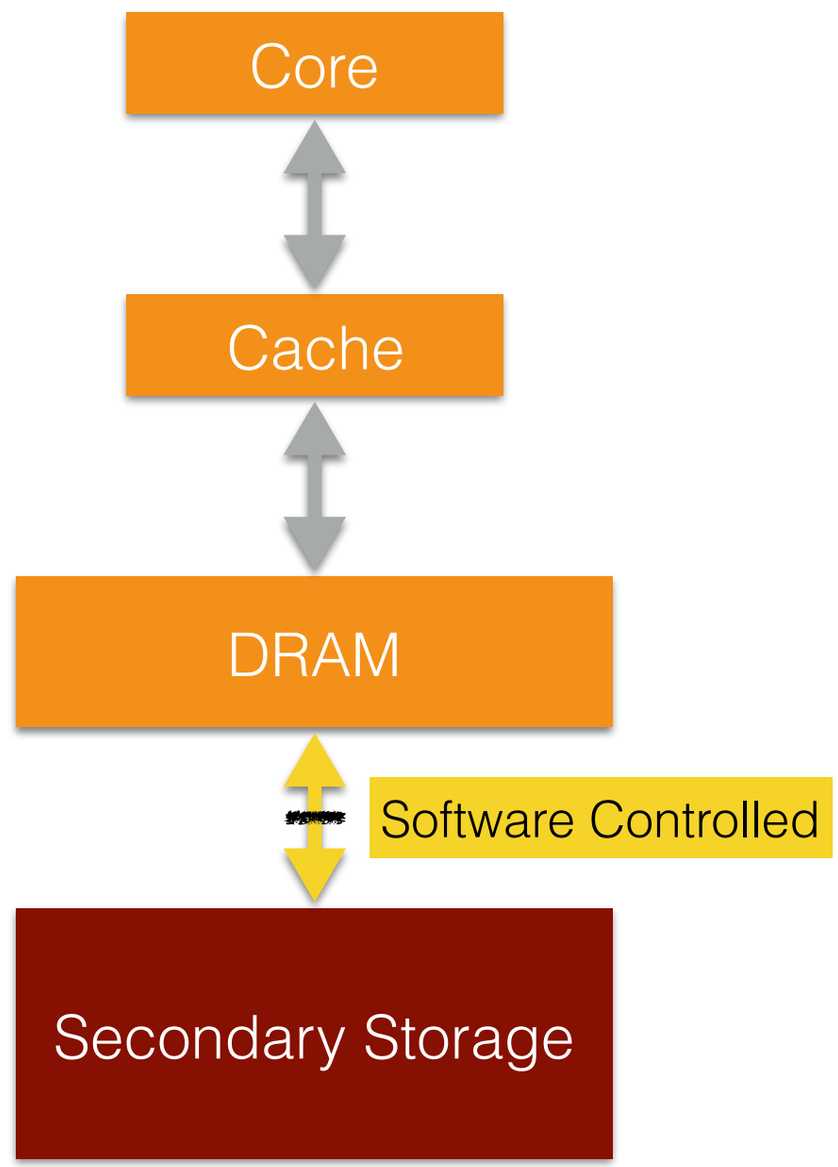
Advantages:

- Unify memory and storage
- Access to persistent data through processor load/store interface

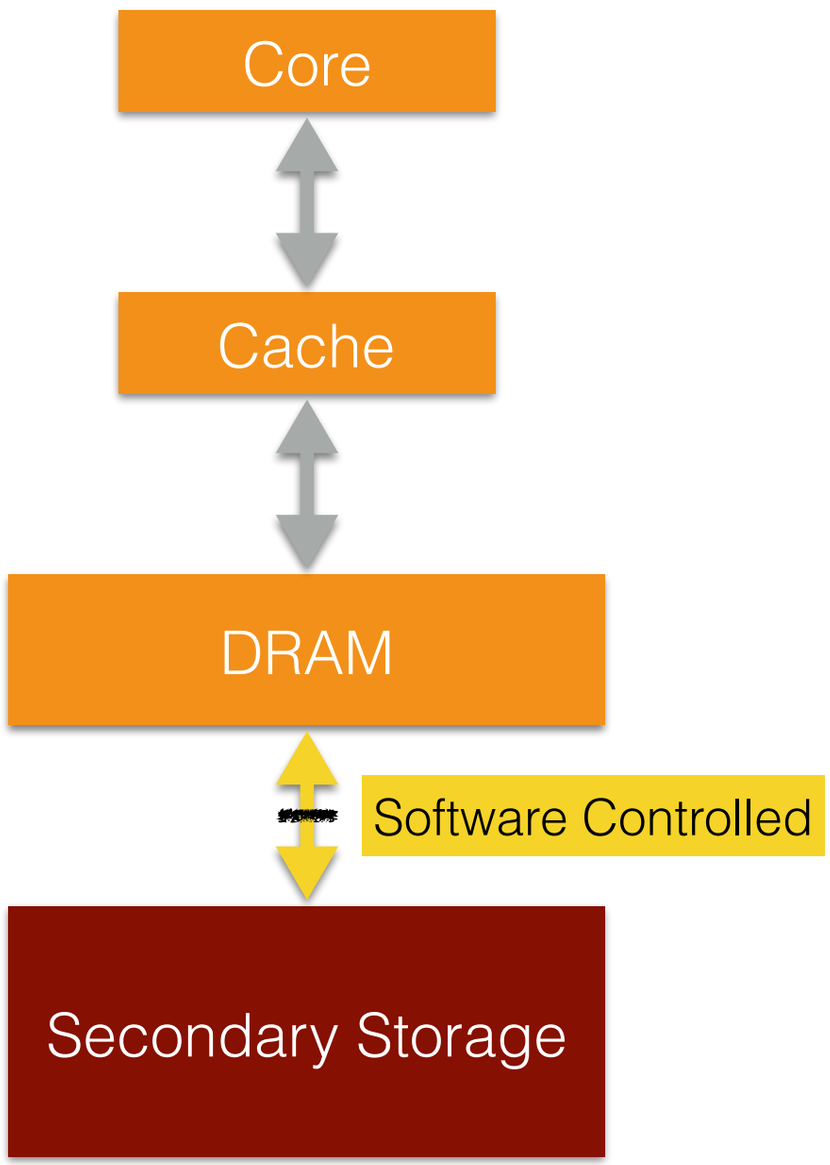
Challenge:

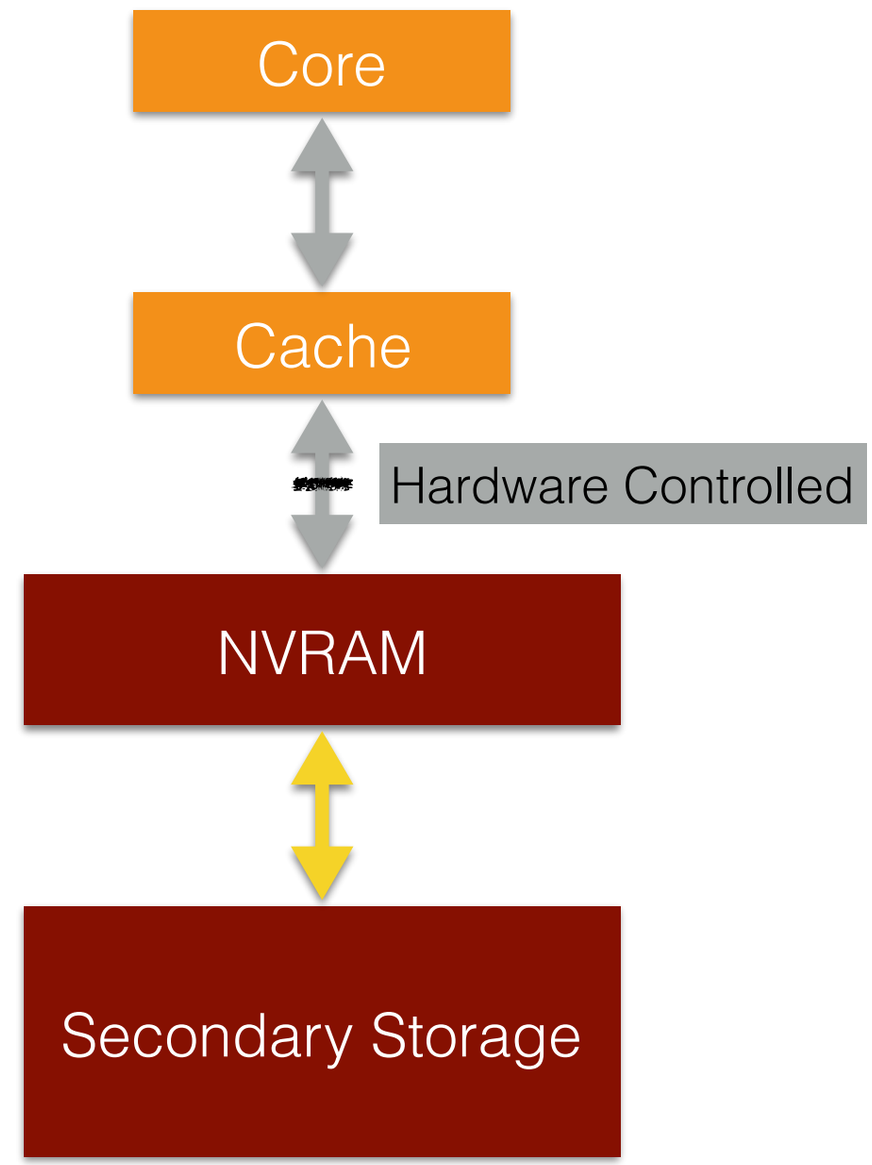
- Maintaining consistency of data structures in memory

Consistency Challenge



Consistency Challenge





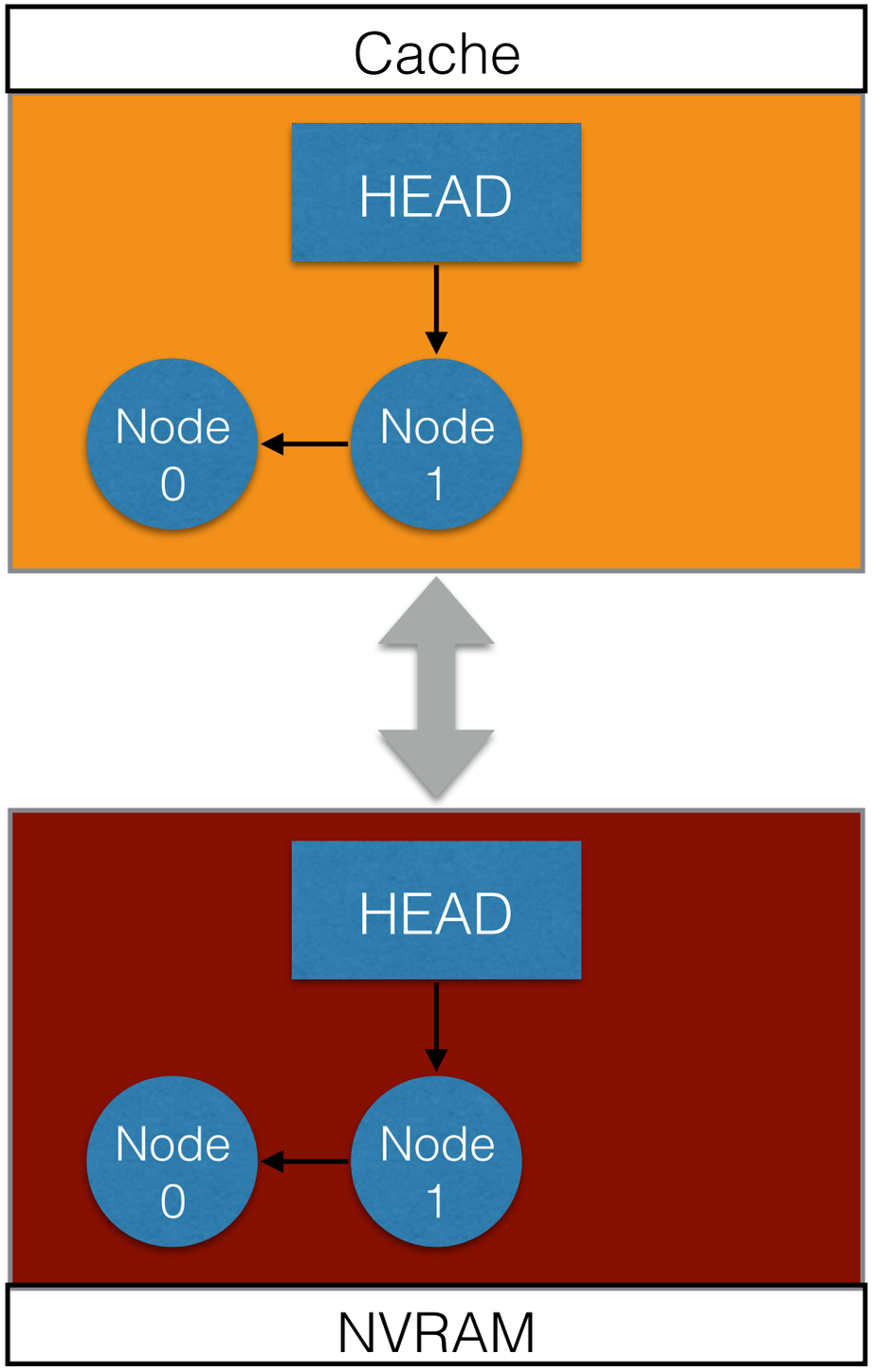
Linked List - Naïve

Pseudo-code

1. Create Node

2. Update Node Pointer

3. Update Head Pointer



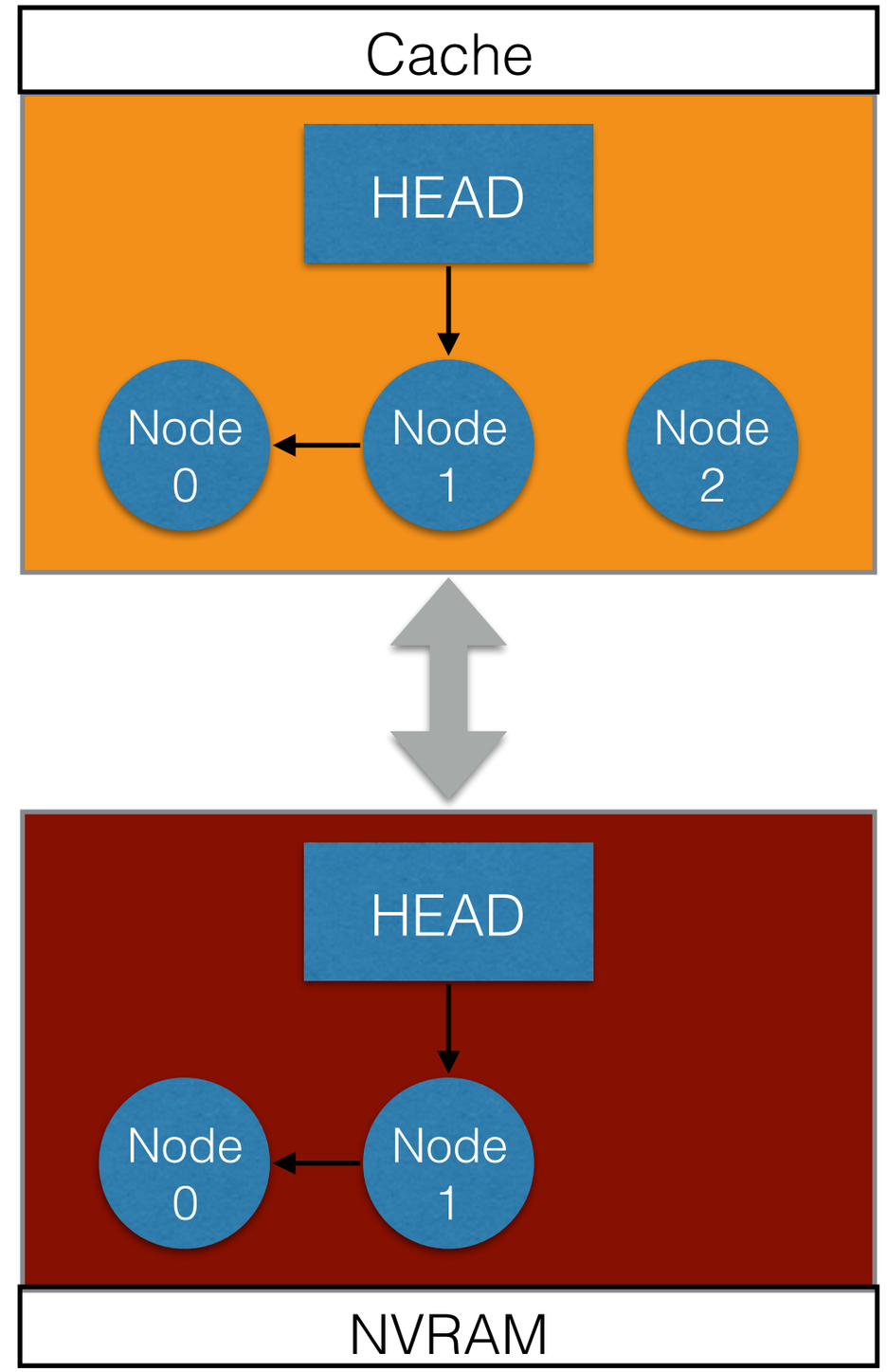
Linked List - Naïve

Pseudo-code

1. Create Node

2. Update Node Pointer

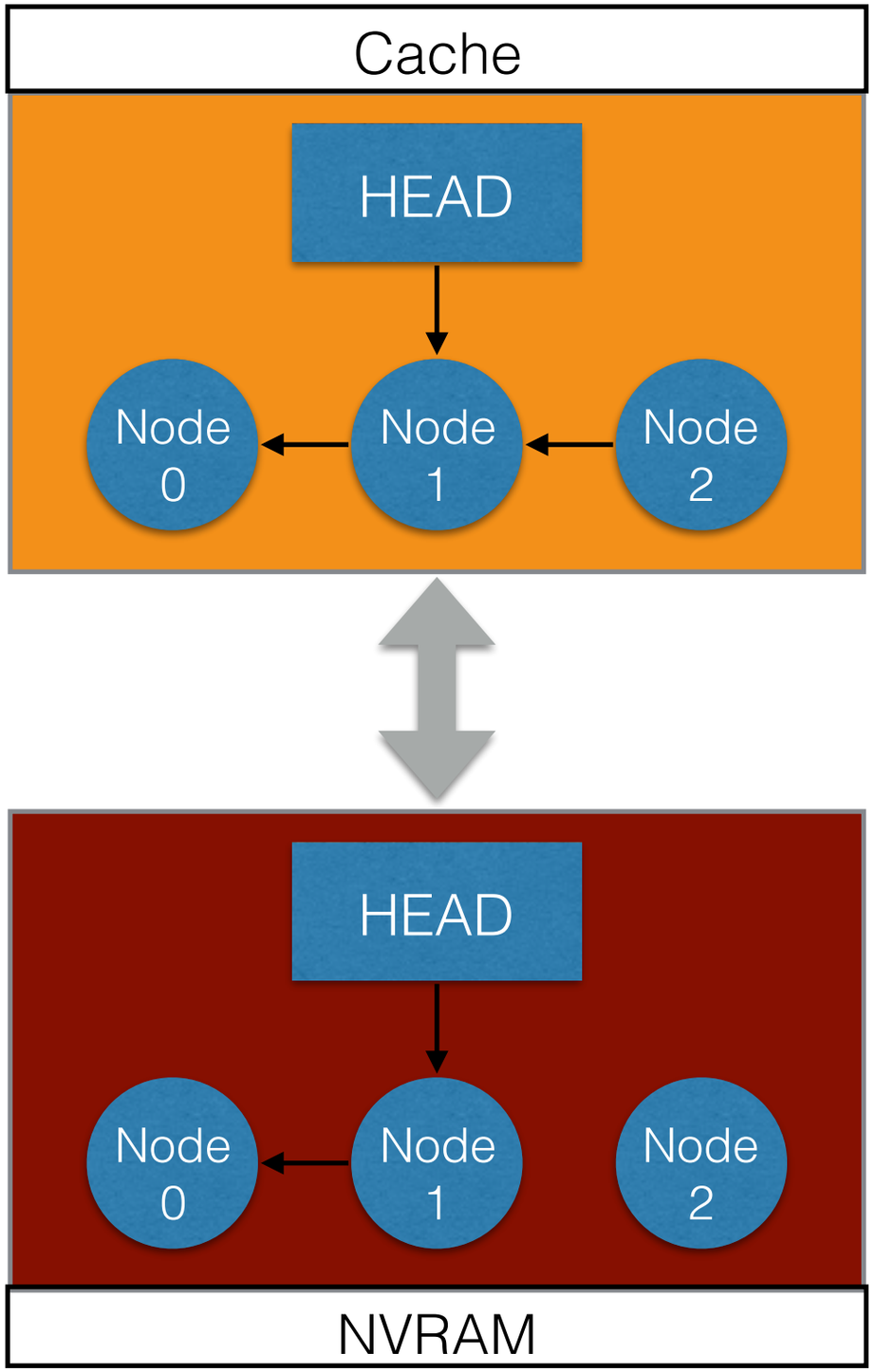
3. Update Head Pointer



Linked List - Naïve

Pseudo-code

1. Create Node
2. Update Node Pointer
3. Update Head Pointer



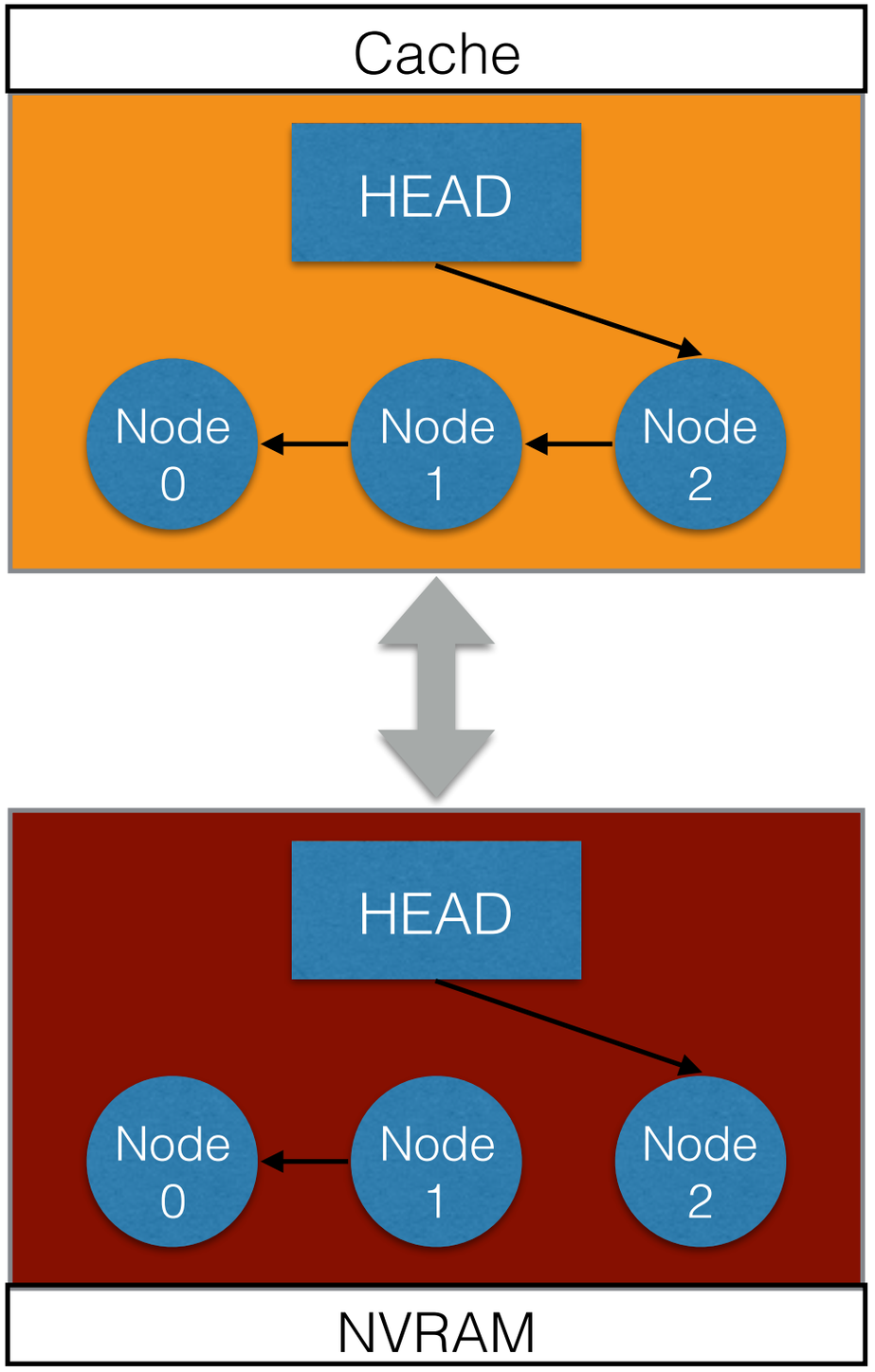
Linked List - Naïve

Pseudo-code

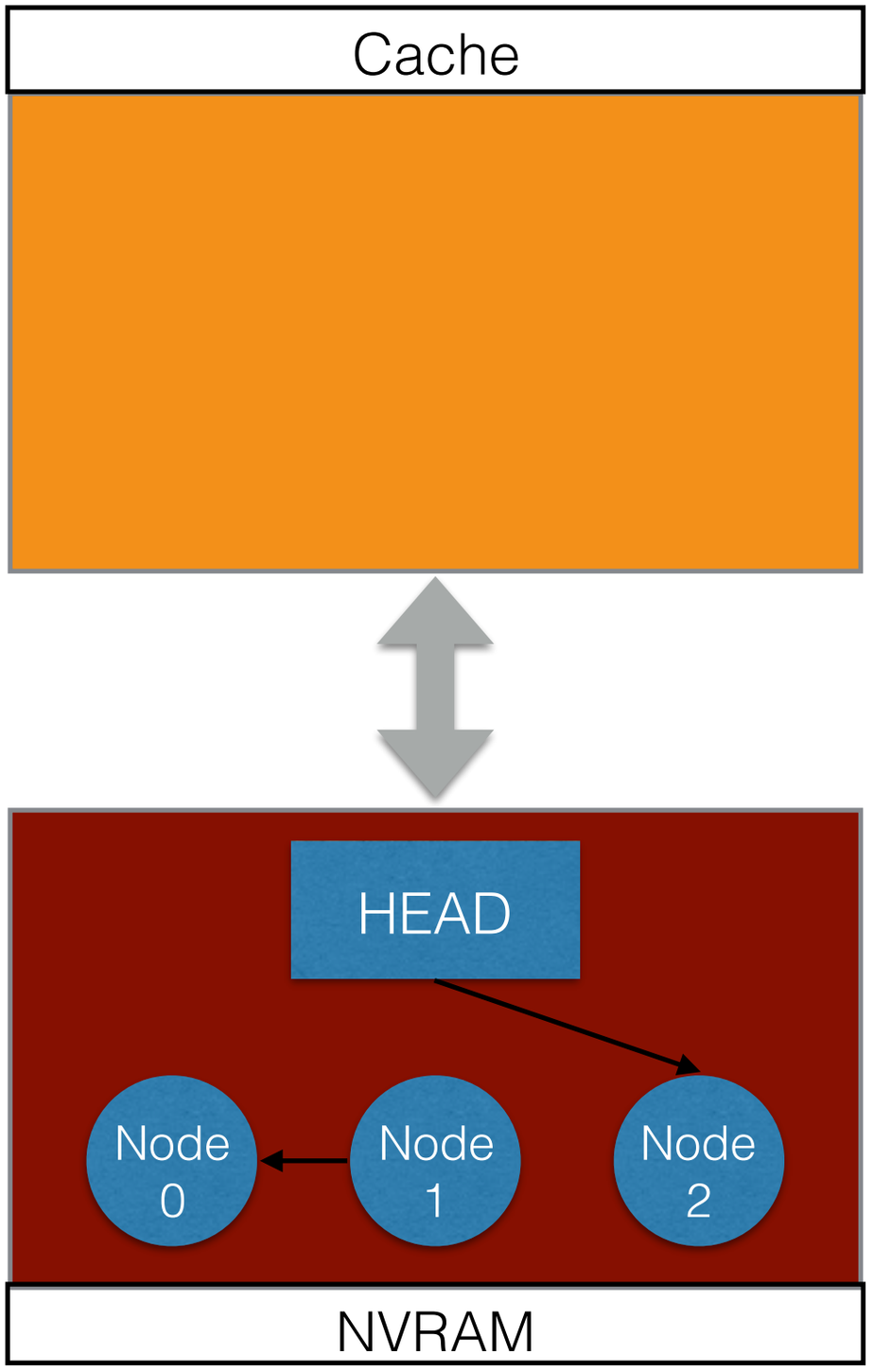
1. Create Node

2. Update Node Pointer

3. Update Head Pointer



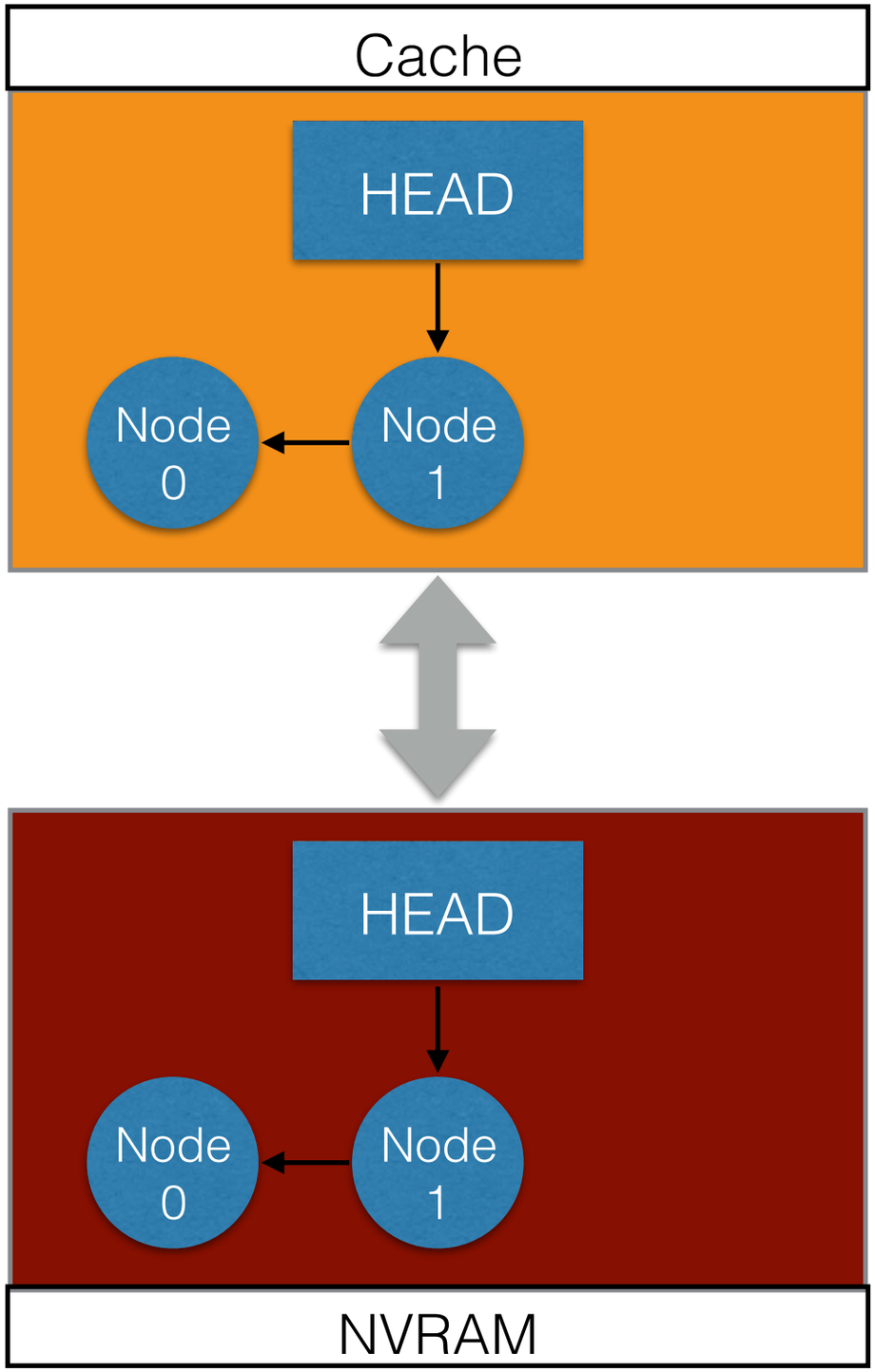
Linked List - Naïve



Linked List - Failsafe

Pseudo-code

1. Create Node
2. Update Node Pointer
3. Persist Barrier
4. Update Head Pointer



Linked List - Failsafe

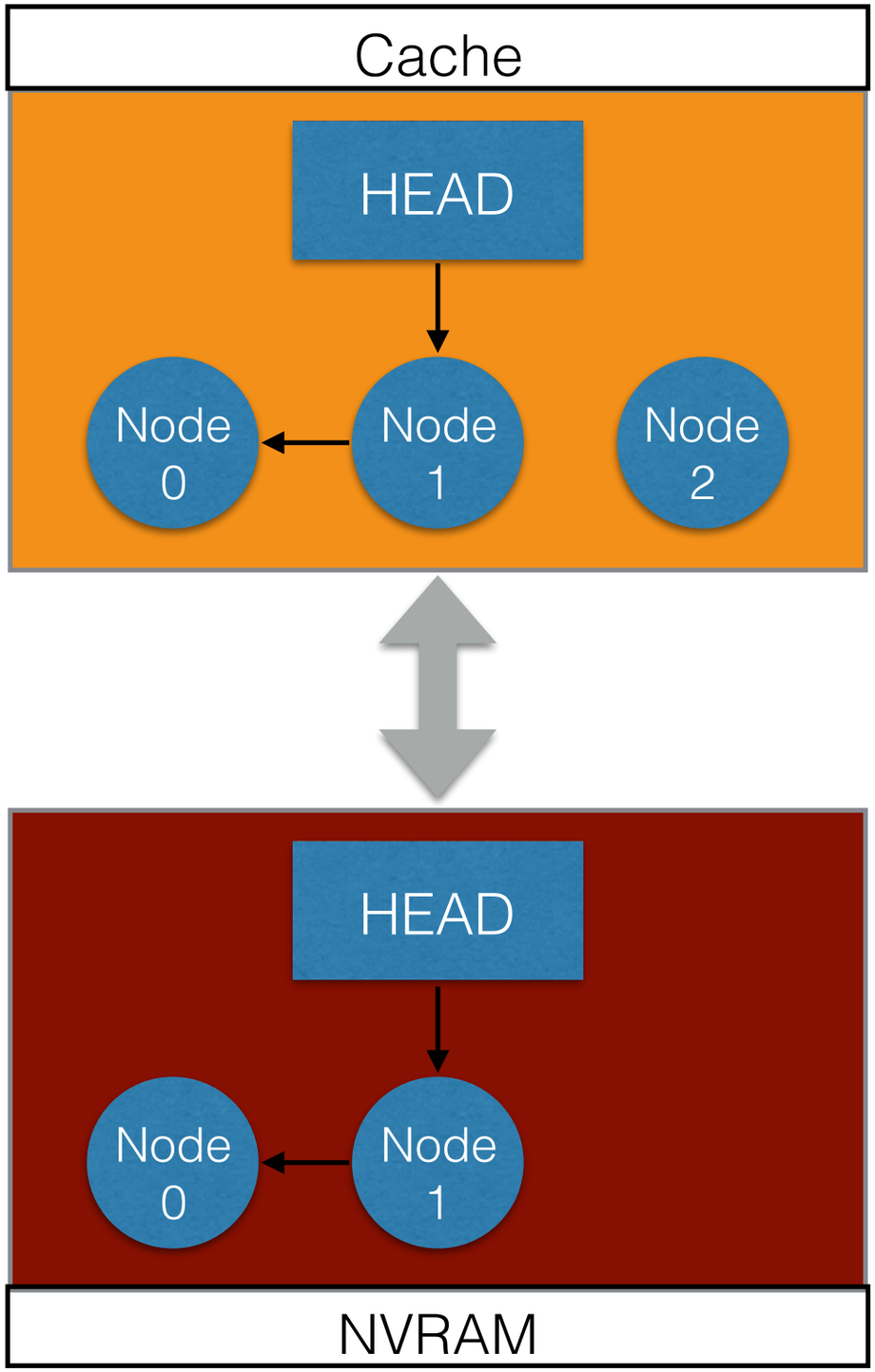
Pseudo-code

1. Create Node

2. Update Node Pointer

3. Persist Barrier

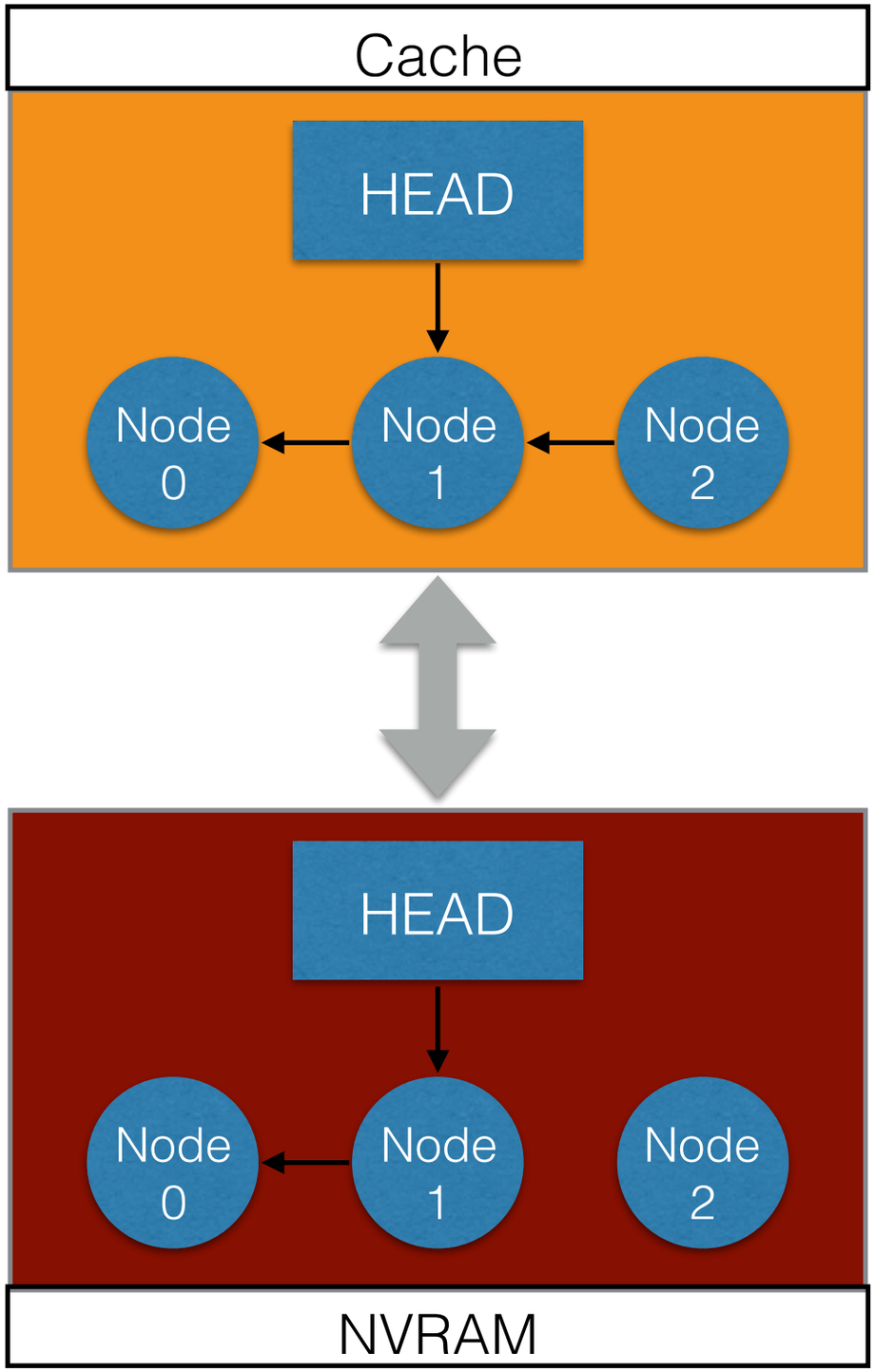
4. Update Head Pointer



Linked List - Failsafe

Pseudo-code

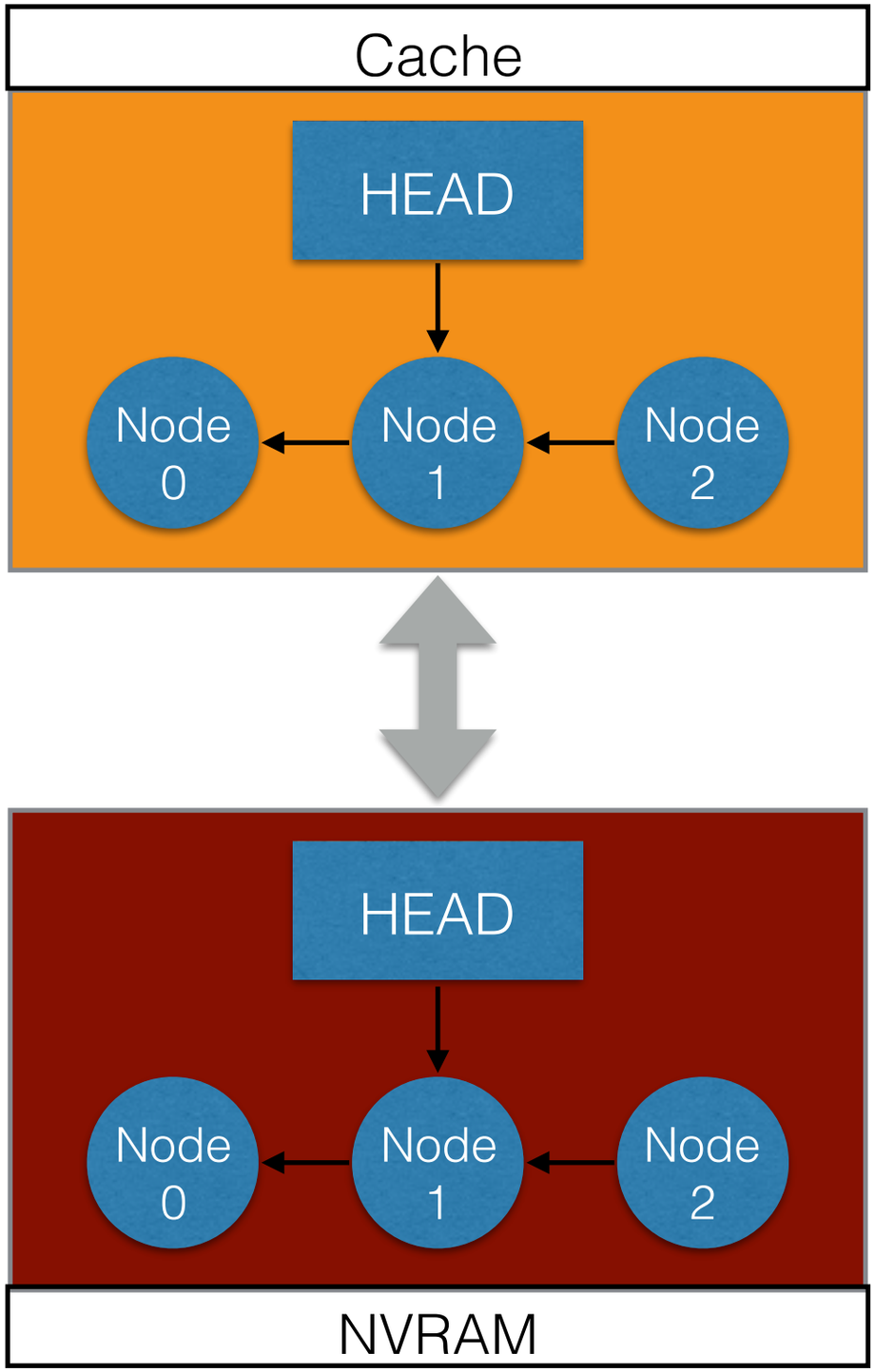
1. Create Node
2. Update Node Pointer
3. Persist Barrier
4. Update Head Pointer



Linked List - Failsafe

Pseudo-code

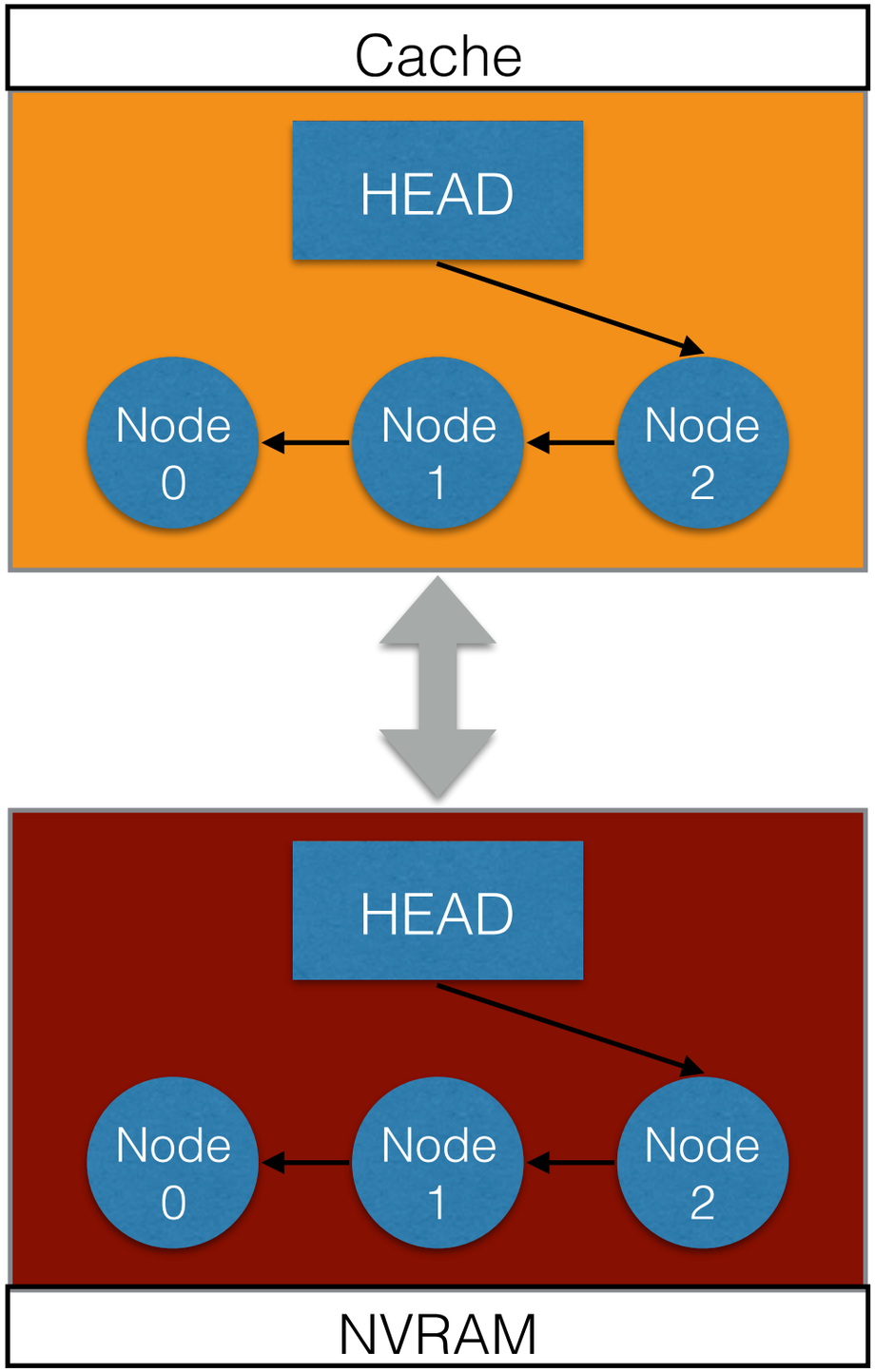
1. Create Node
2. Update Node Pointer
3. Persist Barrier
4. Update Head Pointer



Linked List - Failsafe

Pseudo-code

1. Create Node
2. Update Node Pointer
3. Persist Barrier
4. Update Head Pointer



Linked List - Failsafe

Pseudo-code

1. Create Node

2. Update Node Pointer

3. Persist Barrier

4. Update Head Pointer

Epoch A

Epoch B

Linked List - Failsafe

Pseudo-code

1. Create Node
2. Update Node Pointer
3. Persist Barrier
4. Update Head Pointer

Programmer Introduced

Linked List - Failsafe

Pseudo-code

1. Create Node
2. Update Node Pointer
3. Persist Barrier
4. Update Head Pointer

Programmer Introduced

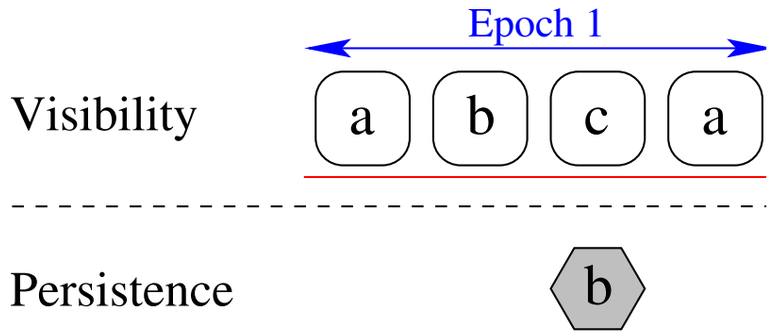
Divide program execution into epochs through programmer inserted persist barriers = Epoch Persistence

Epoch Persistence*

* Pelley et. al., “Memory Persistency”, in ISCA-2014.

Epoch Persistence*

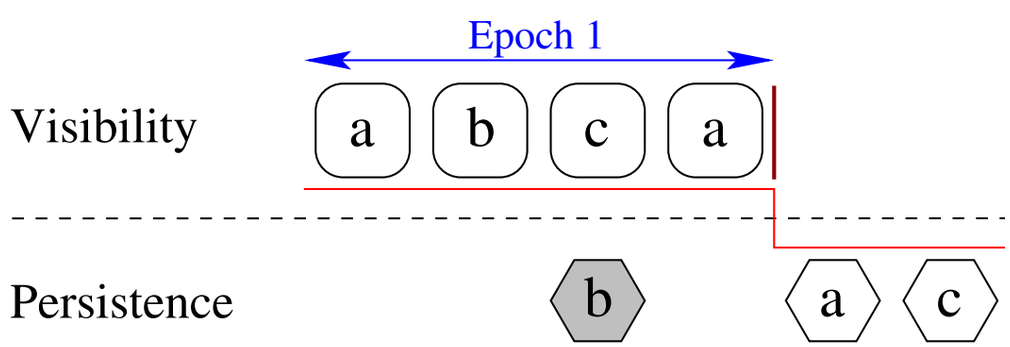
St a
St b
St c
St a



* Pelley et. al., “Memory Persistency”, in ISCA-2014.

Epoch Persistence*

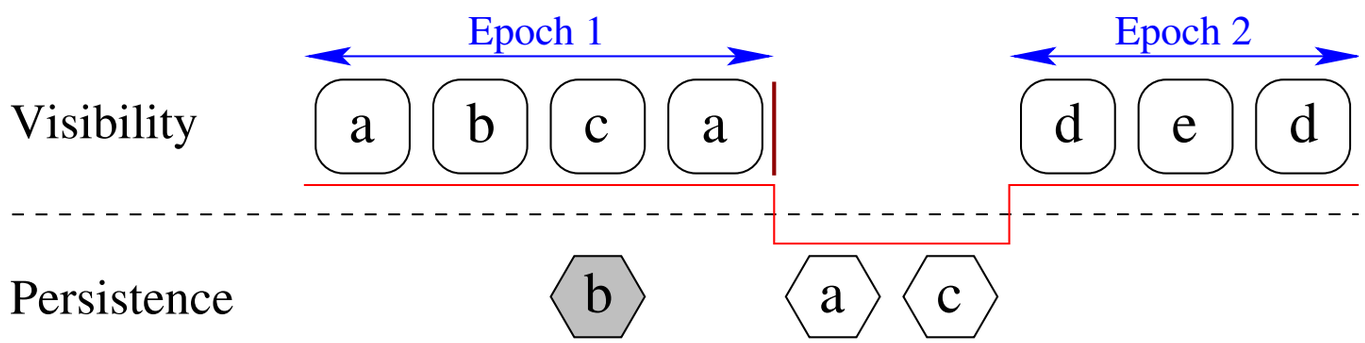
St a
St b
St c
St a
Persist Barrier



* Pelley et. al., “Memory Persistency”, in ISCA-2014.

Epoch Persistence*

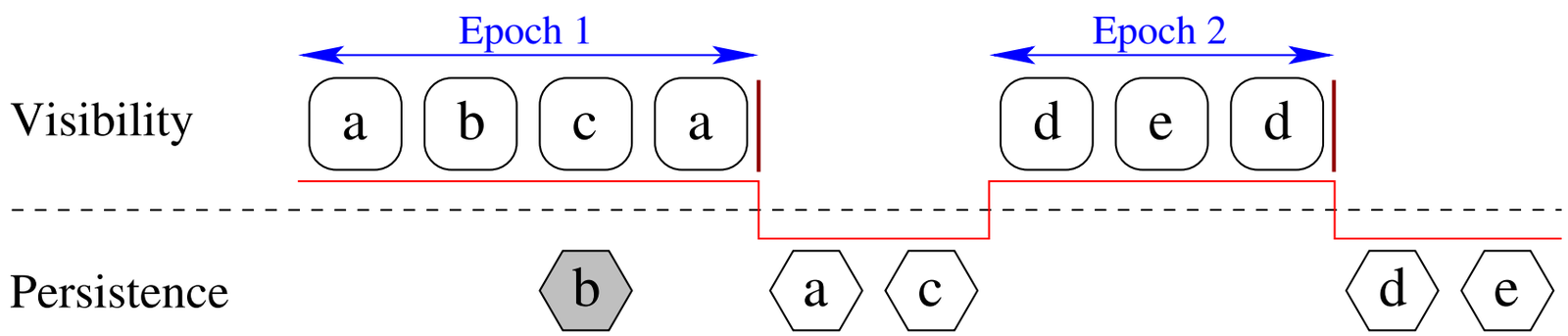
St a
St b
St c
St a
Persist Barrier
St d
St e
St d



* Pelley et. al., “Memory Persistency”, in ISCA-2014.

Epoch Persistence*

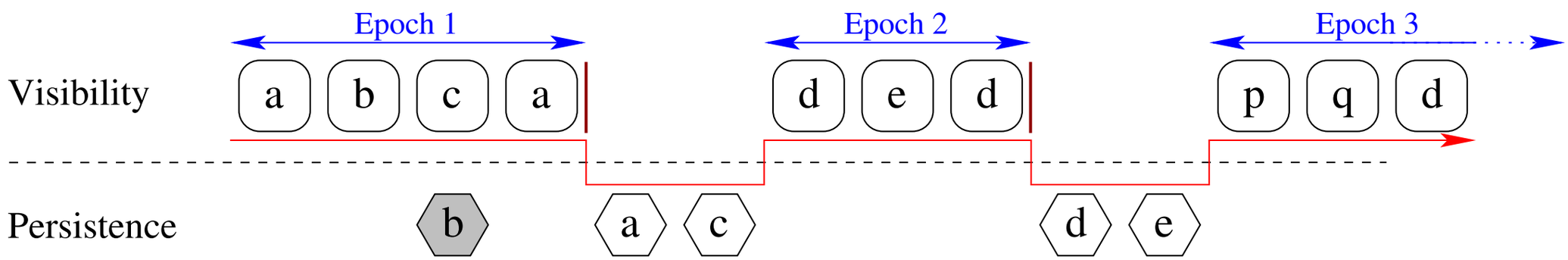
St a
St b
St c
St a
Persist Barrier
St d
St e
St d
Persist Barrier



* Pelley et. al., “Memory Persistency”, in ISCA-2014.

Epoch Persistence*

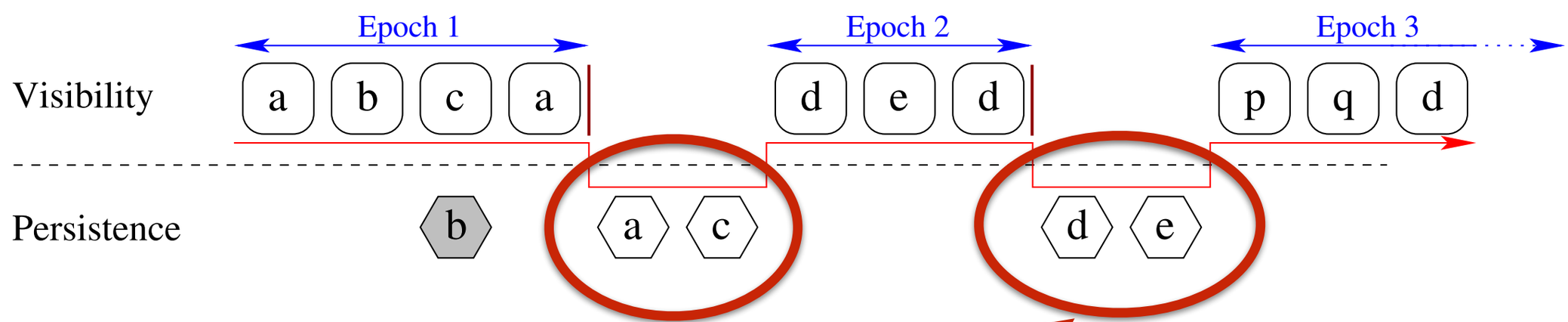
St a
St b
St c
St a
Persist Barrier
St d
St e
St d
Persist Barrier
St p
St q
St d
…



* Pelley et. al., “Memory Persistency”, in ISCA-2014.

Epoch Persistence*

St a
St b
St c
St a
Persist Barrier
St d
St e
St d
Persist Barrier
St p
St q
St d
…



Persist operations happen in the critical path of execution.

* P

Buffered Epoch Persistence* through Lazy Barrier (LB)

- Implementation of Epoch Persistence
- Durability lags visibility
  - To allow performing persist operations out of critical path

* Pelley et. al., "Memory Persistency", in ISCA-2014.
* Condit et. al., "Better I/O through byte-addressable, persistent memory", in SOSP-2009.

Buffered Epoch Persistence* through Lazy Barrier (LB)

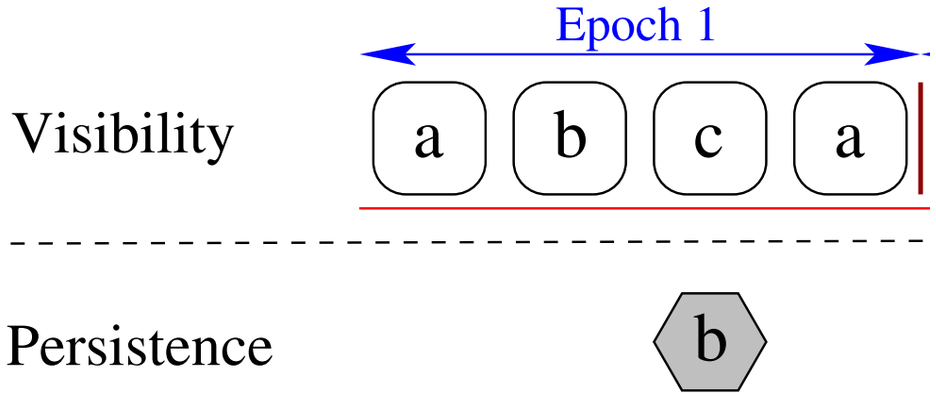
Visibility

- - - - - - - - - - - -

Persistence

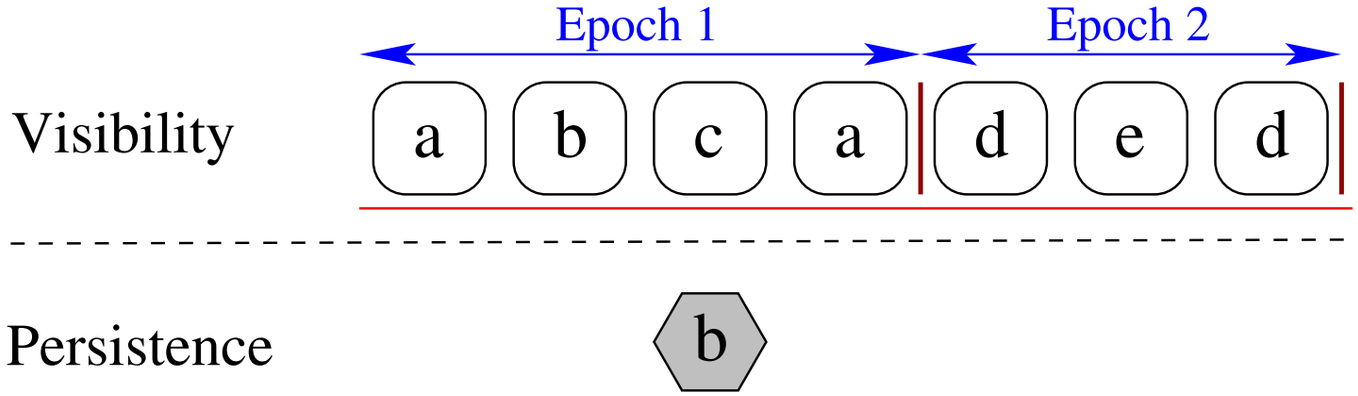
* Pelley et. al., “Memory Persistency”, in ISCA-2014.
* Condit et. al., “Better I/O through byte-addressable, persistent memory”, in SOSP-2009.

Buffered Epoch Persistence* through Lazy Barrier (LB)



* Pelley et. al., “Memory Persistency”, in ISCA-2014.
* Condit et. al., “Better I/O through byte-addressable, persistent memory”, in SOSP-2009.

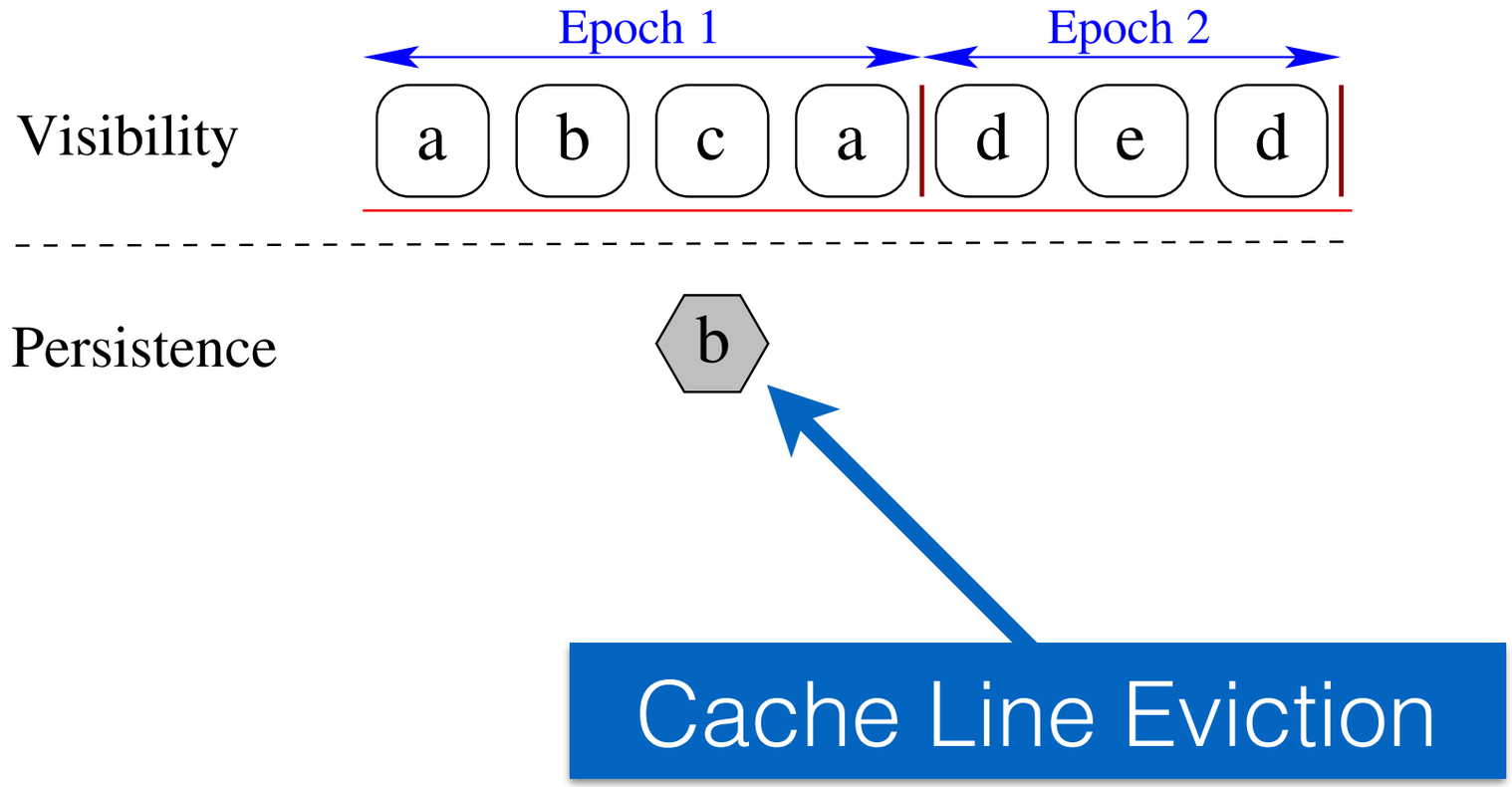
Buffered Epoch Persistence* through Lazy Barrier (LB)



* Pelley et. al., “Memory Persistency”, in ISCA-2014.

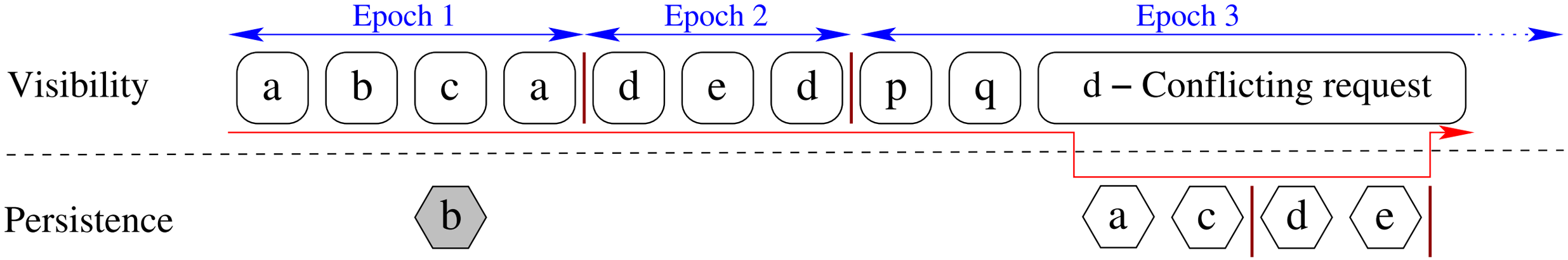
* Condit et. al., “Better I/O through byte-addressable, persistent memory”, in SOSP-2009.

Buffered Epoch Persistence* through Lazy Barrier (LB)



* Pelley et. al., "Memory Persistency", in ISCA-2014.
* Condit et. al., "Better I/O through byte-addressable, persistent memory", in SOSP-2009.

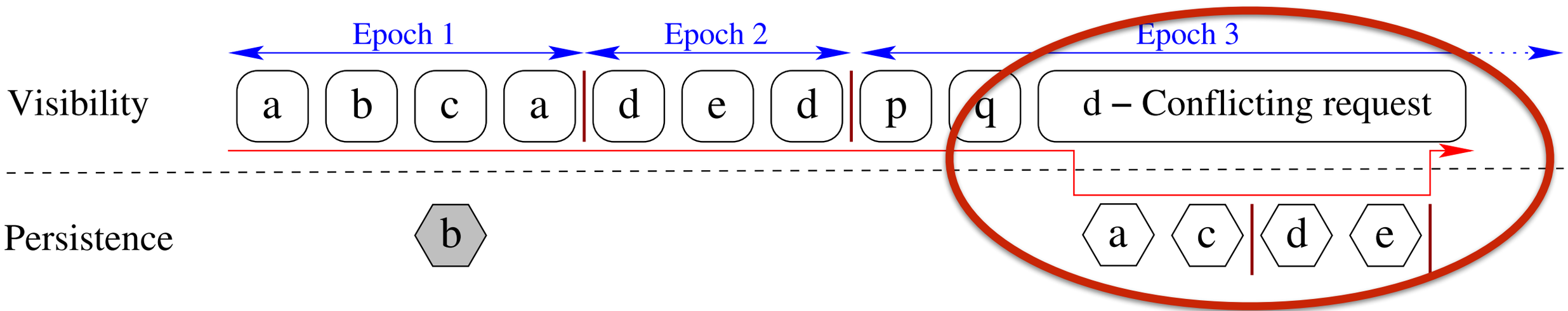
Buffered Epoch Persistence* through Lazy Barrier (LB)



* Pelley et. al., “Memory Persistency”, in ISCA-2014.

* Condit et. al., “Better I/O through byte-addressable, persistent memory”, in SOSP-2009.

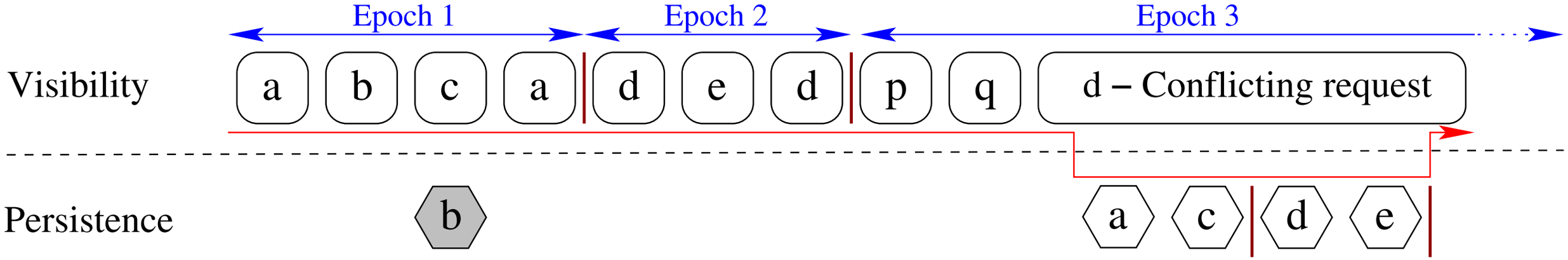
Buffered Epoch Persistence* through Lazy Barrier (LB)



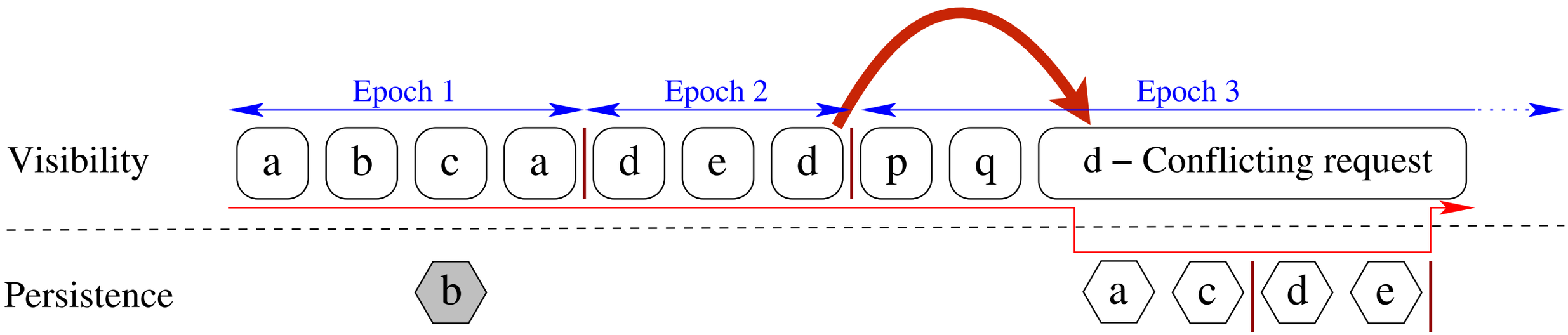
Conflicts bring persist operations back in the critical path.

*

Intra-thread Conflict



Intra-thread Conflict



Proactive Flush (PF)

- Persist Triggers in Lazy Barrier
  - Passive Trigger: cache line eviction
  - Reactive Trigger: flush on (intra/inter)-thread conflicts

Proactive Flush (PF)

- Persist Triggers in Lazy Barrier
  - Passive Trigger: cache line eviction
  - Reactive Trigger: flush on (intra/inter)-thread conflicts
- Persist Trigger with Proactive Flush
  - Proactive Trigger: proactively flush on epoch completion

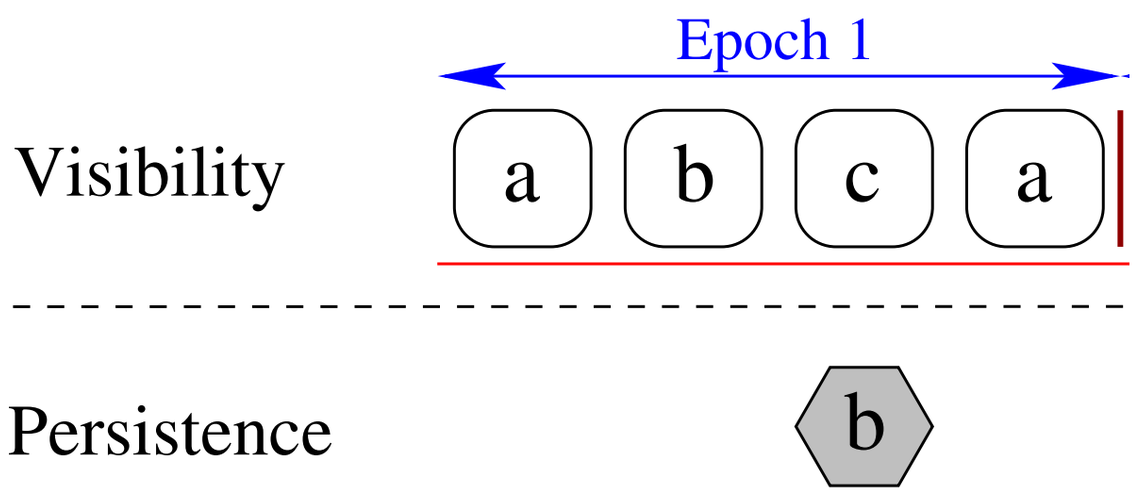
Proactive Flush (PF)

Visibility

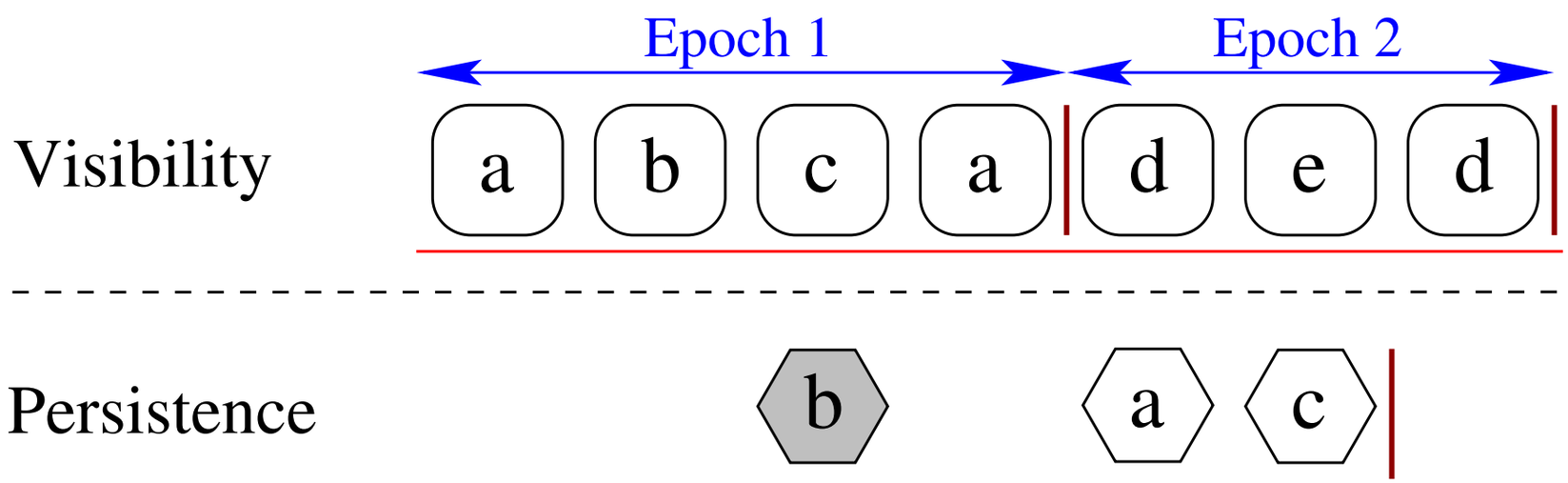
- - - - - - - - - - - -

Persistence

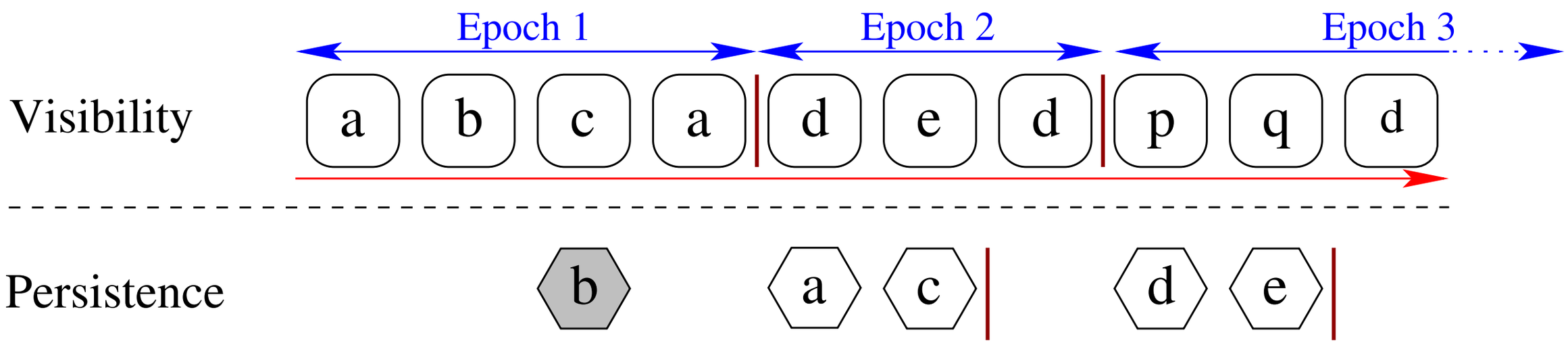
Proactive Flush (PF)



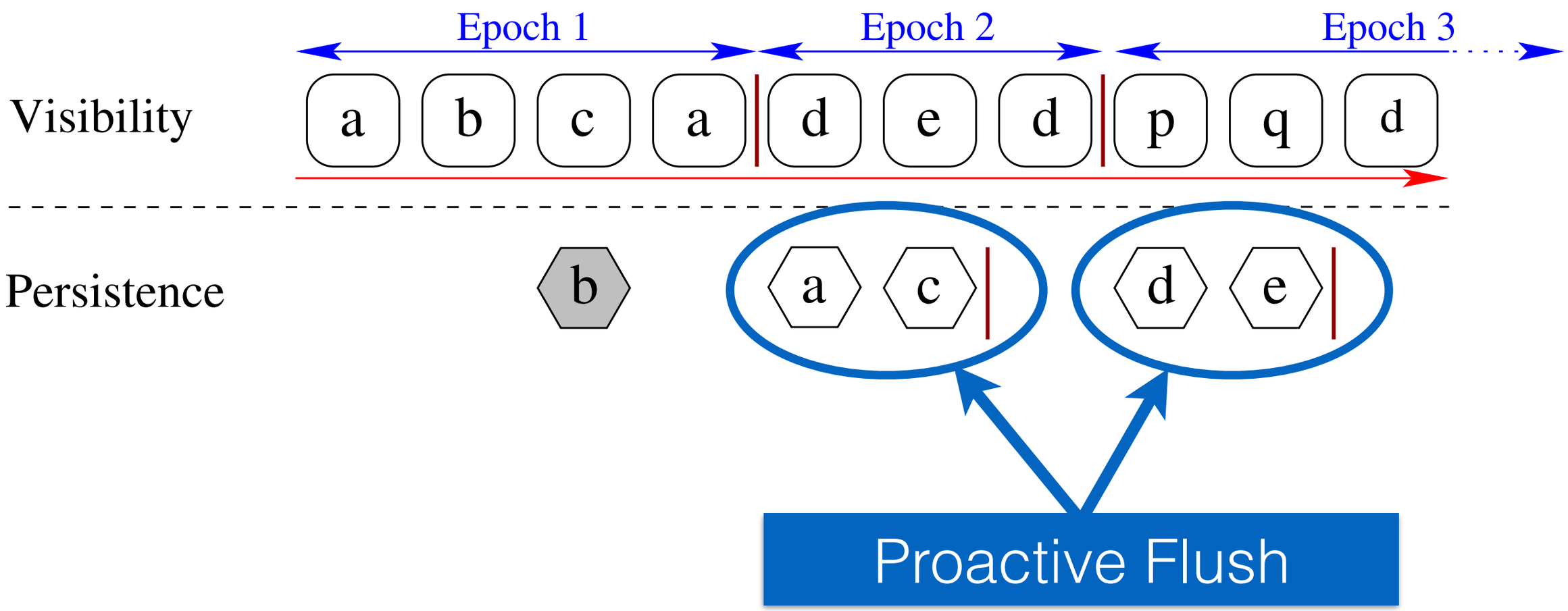
Proactive Flush (PF)



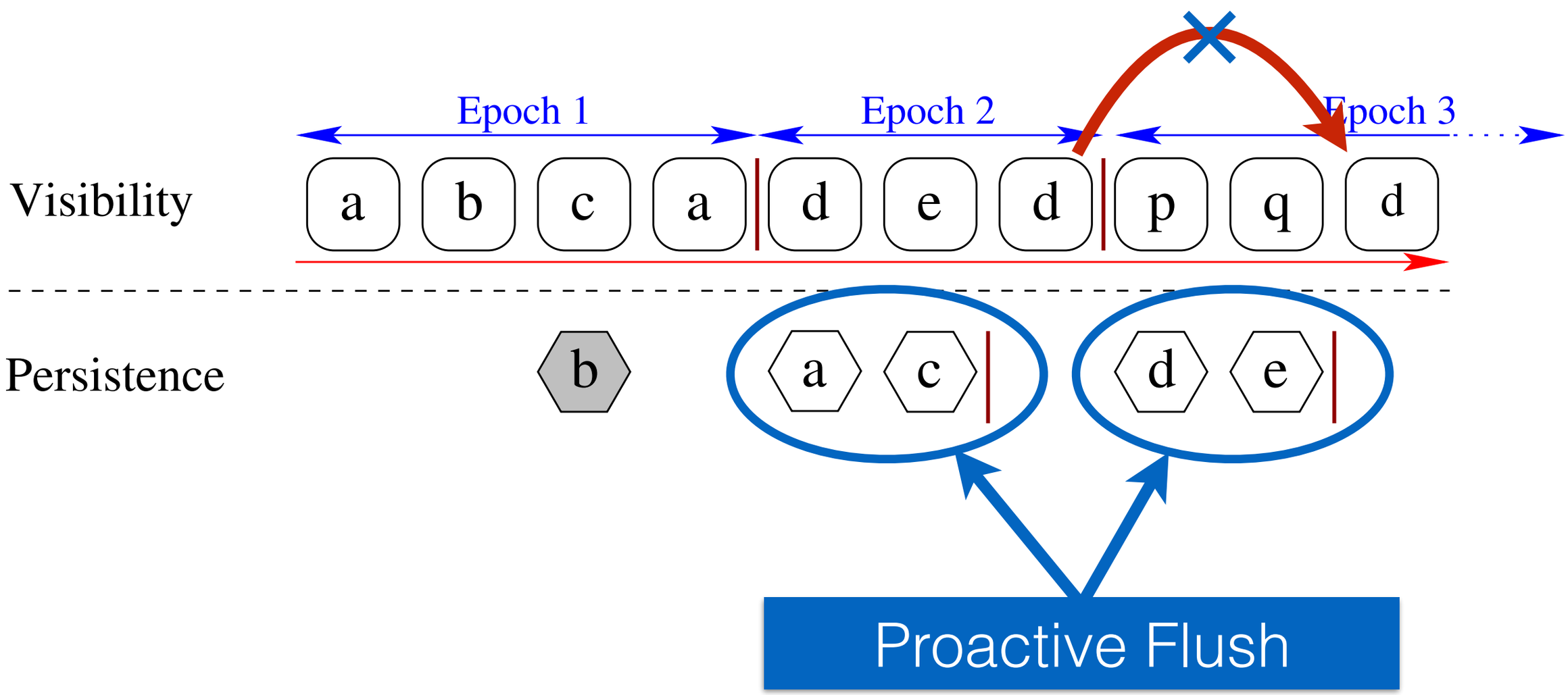
Proactive Flush (PF)



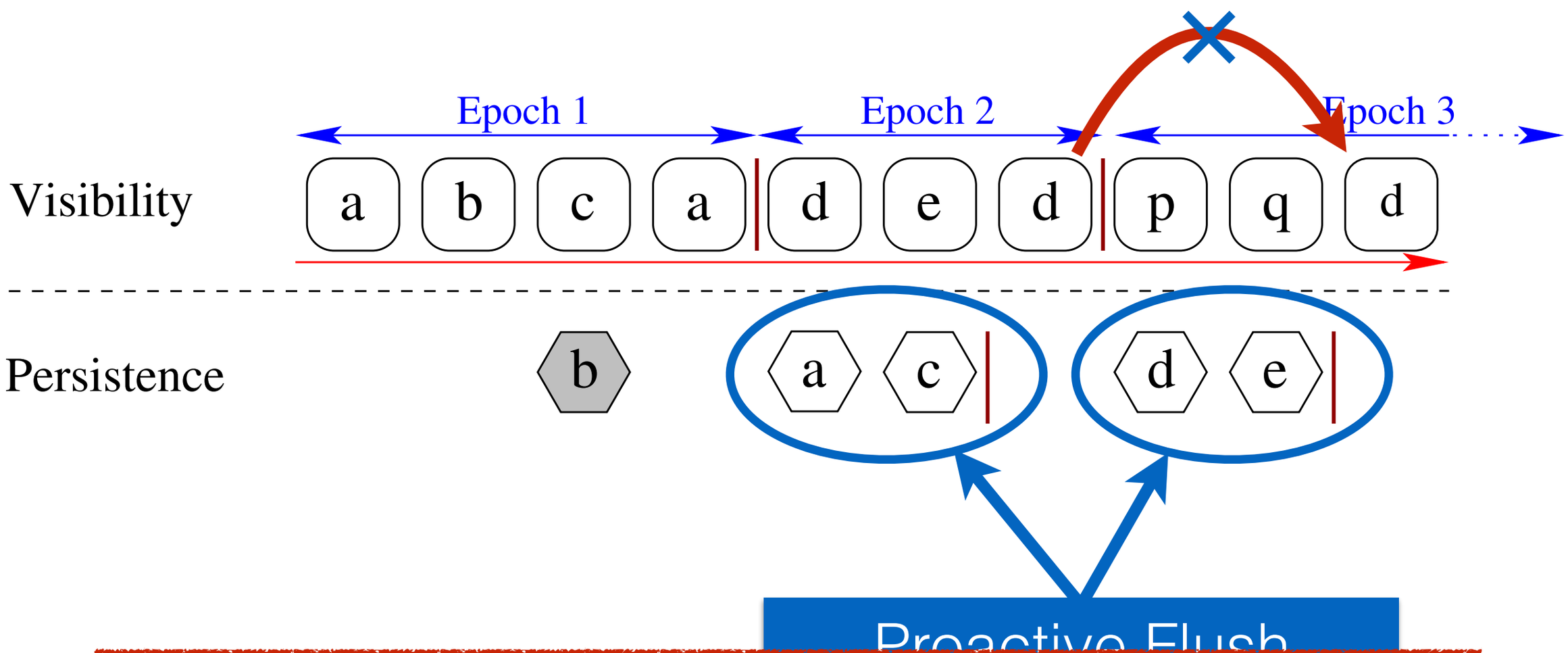
Proactive Flush (PF)



Proactive Flush (PF)



Proactive Flush (PF)



Reduces the probability of encountering conflicts.

Inter-thread Conflict

Thread T_0

Visibility

- - - - - - - - - - - - - -

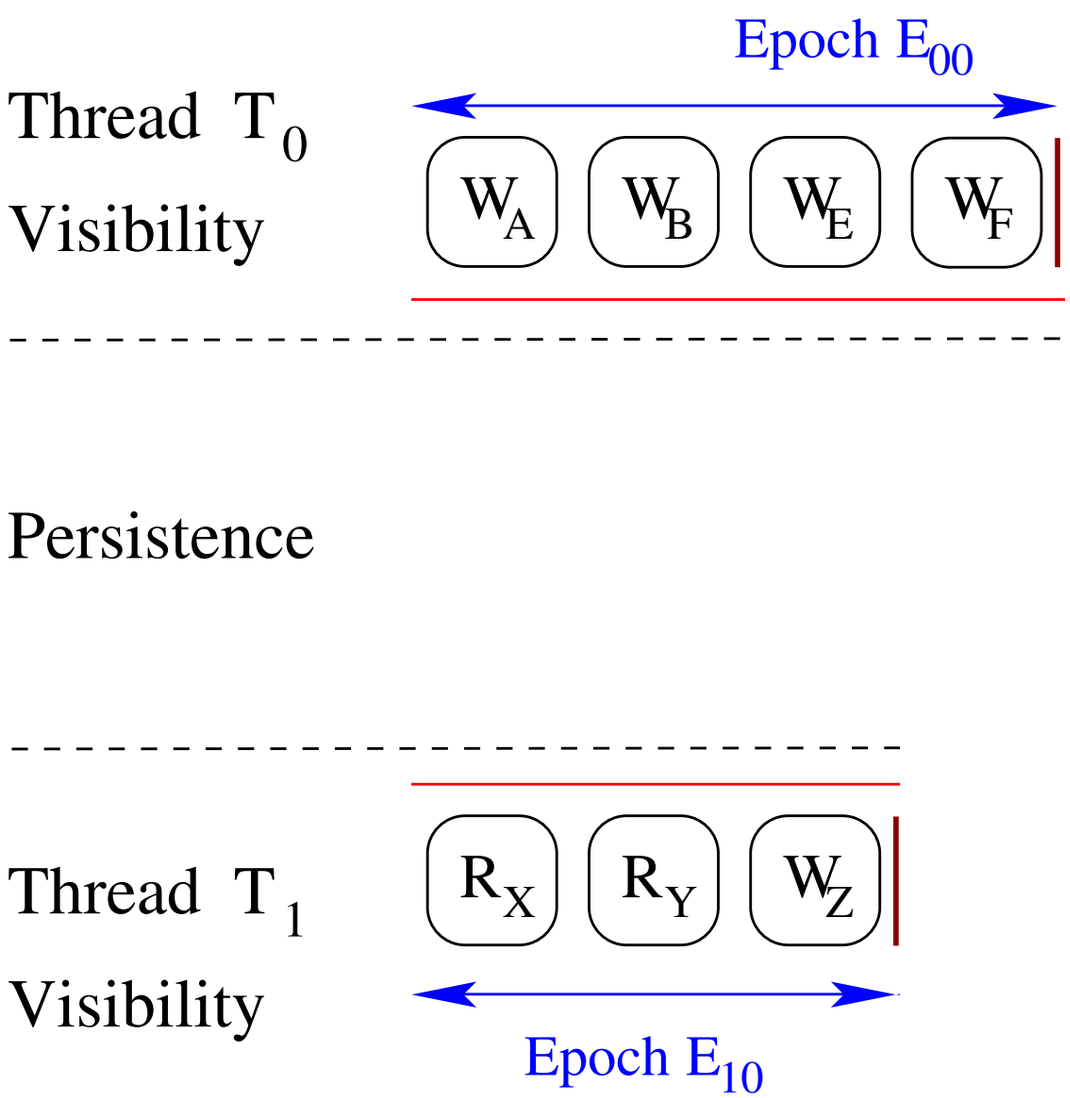
Persistence

- - - - - - - - - - - -

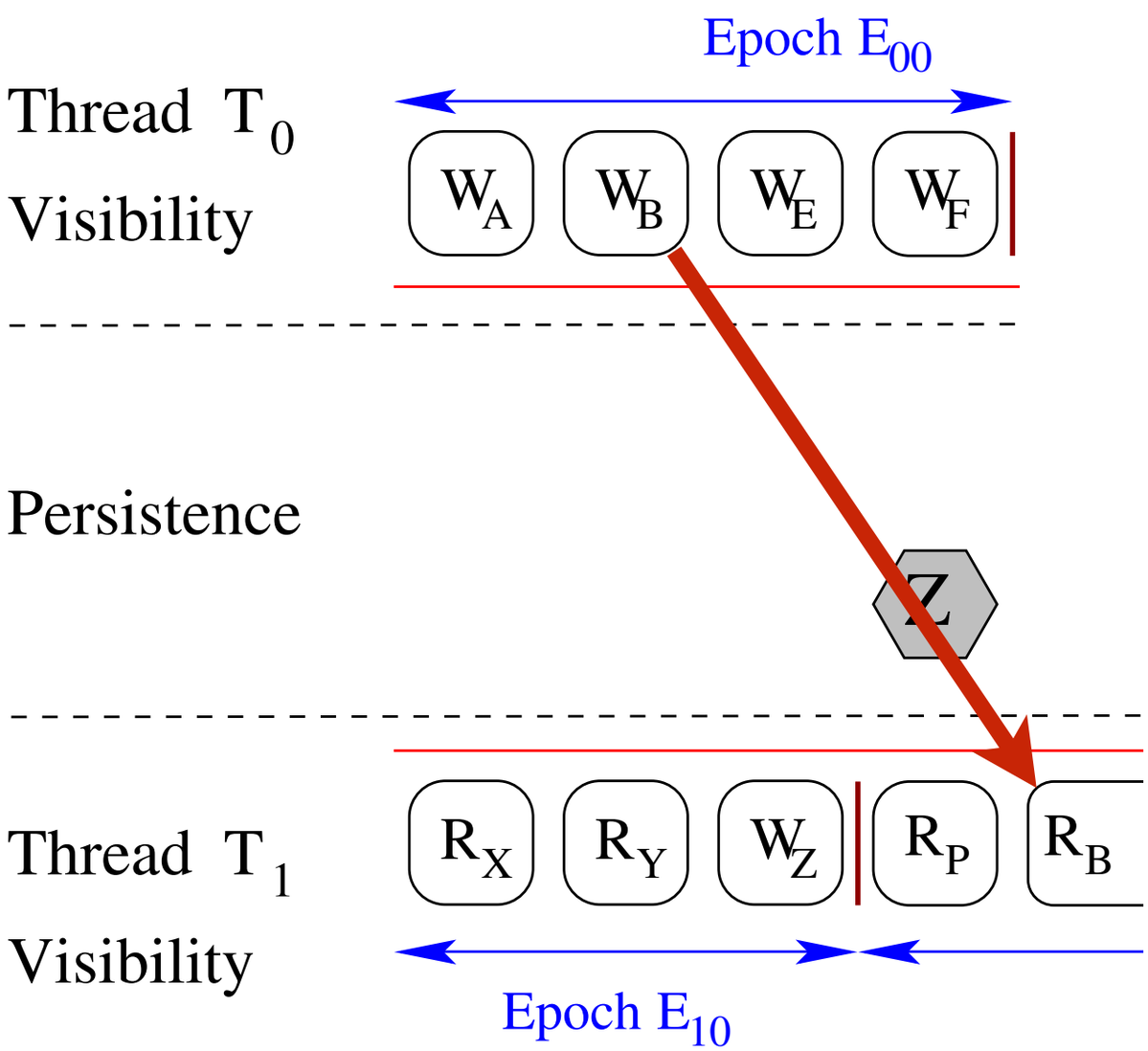
Thread T_1

Visibility

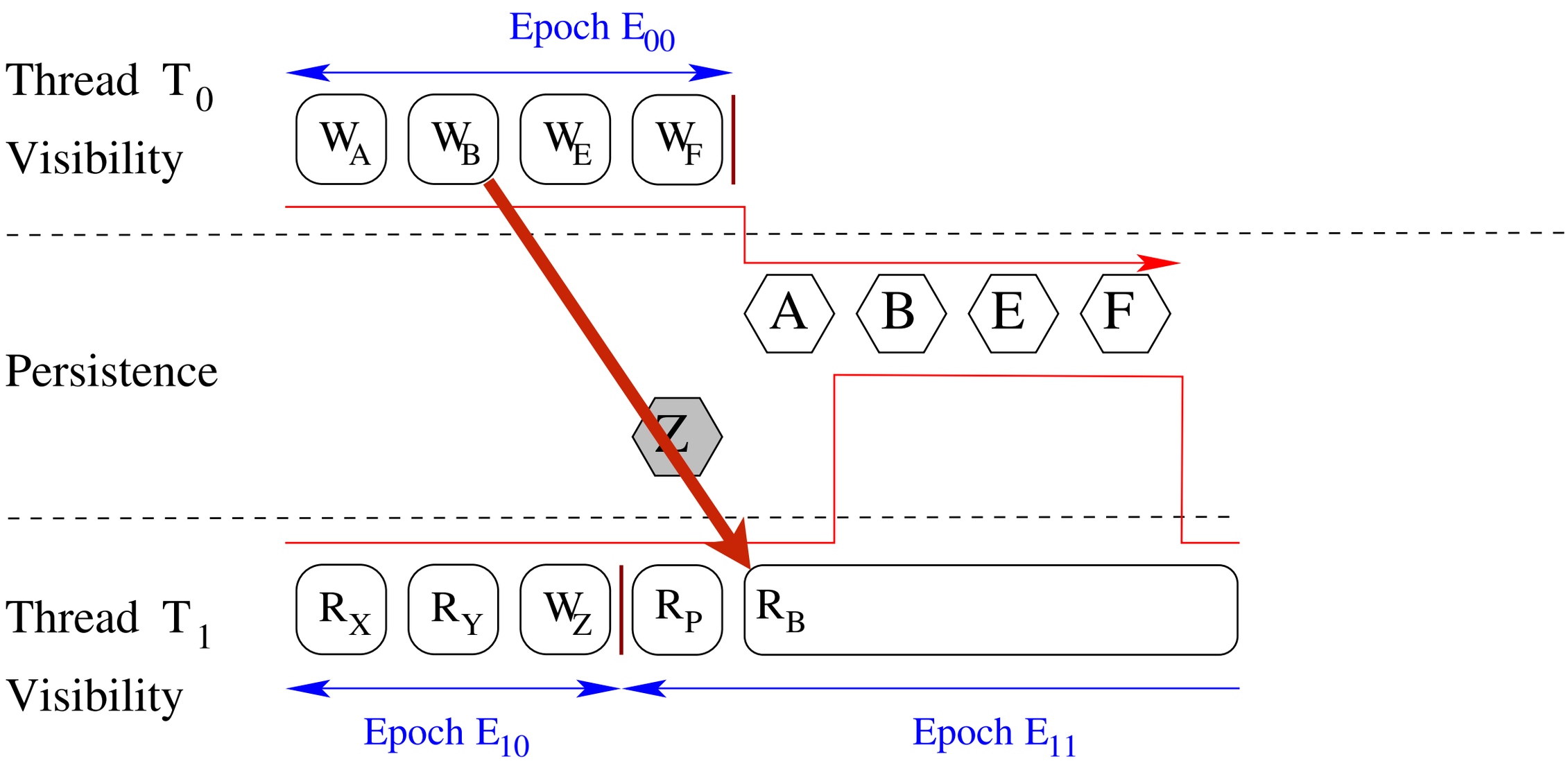
Inter-thread Conflict



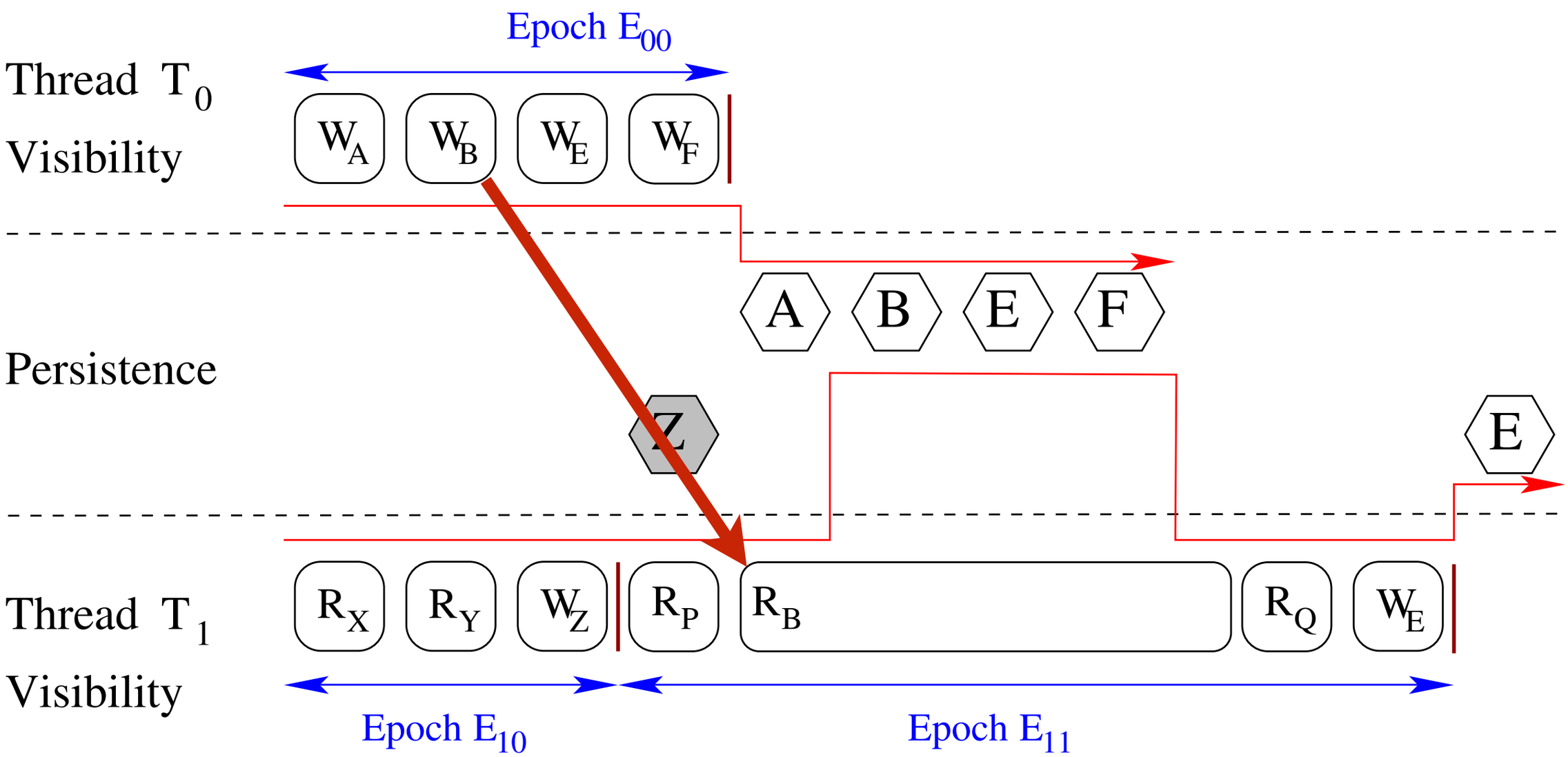
Inter-thread Conflict



Inter-thread Conflict



Inter-thread Conflict



Inter-thread Dependency Tracking (IDT)

- Lazy barrier
  - No tracking of inter-thread dependencies
  - Need to enforce dependencies online

Inter-thread Dependency Tracking (IDT)

- Lazy barrier
  - No tracking of inter-thread dependencies
  - Need to enforce dependencies online
- Inter-thread Dependency Tracking
  - Add inter-thread dependence tracking registers
  - Track dependencies to enforce offline

Inter-thread Dependency Tracking (IDT)

IDT Table	
Source	Dependent

Thread T_0

Visibility

- - - - - - - - - - - -

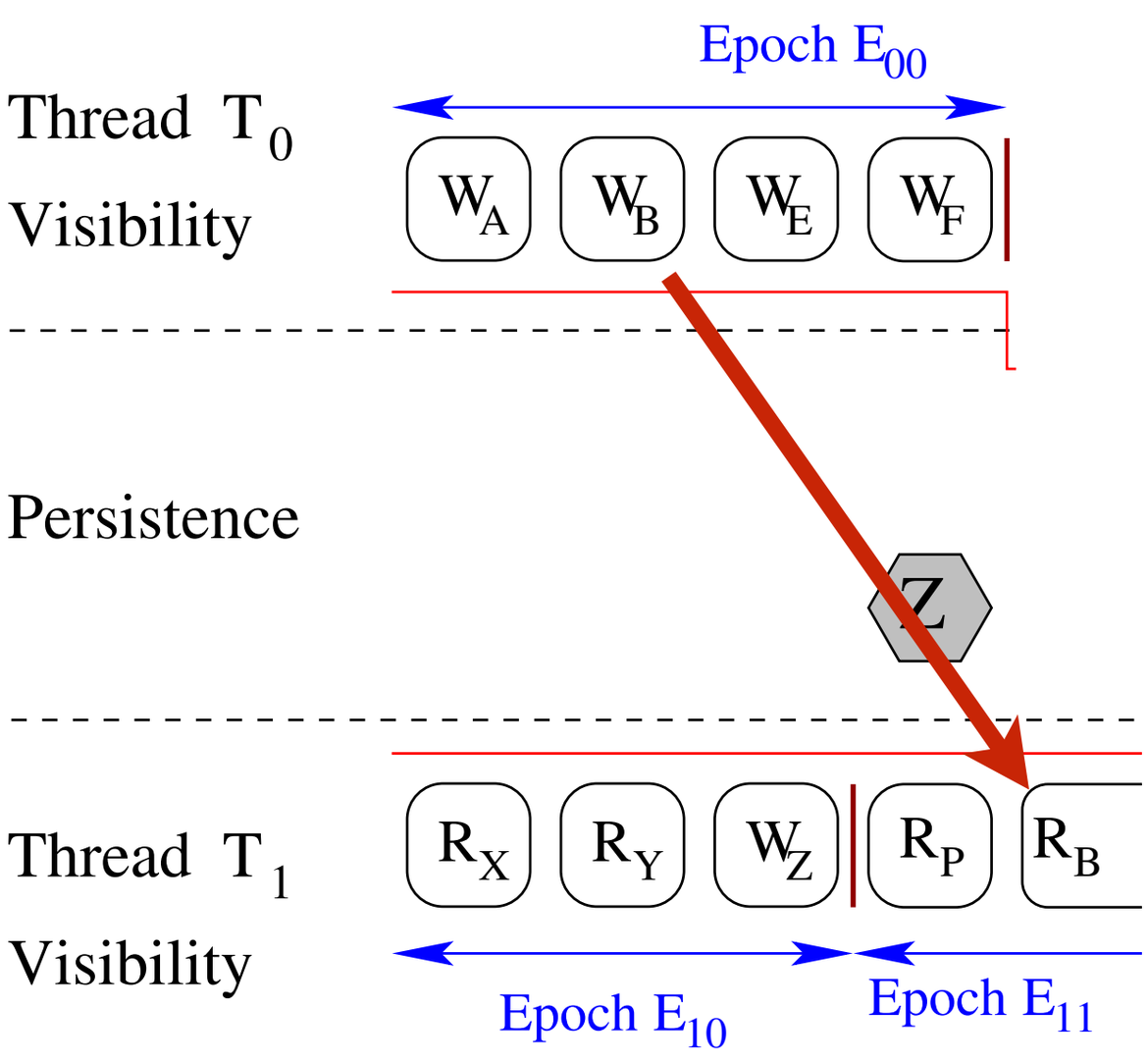
Persistence

- - - - - - - - - - -

Thread T_1

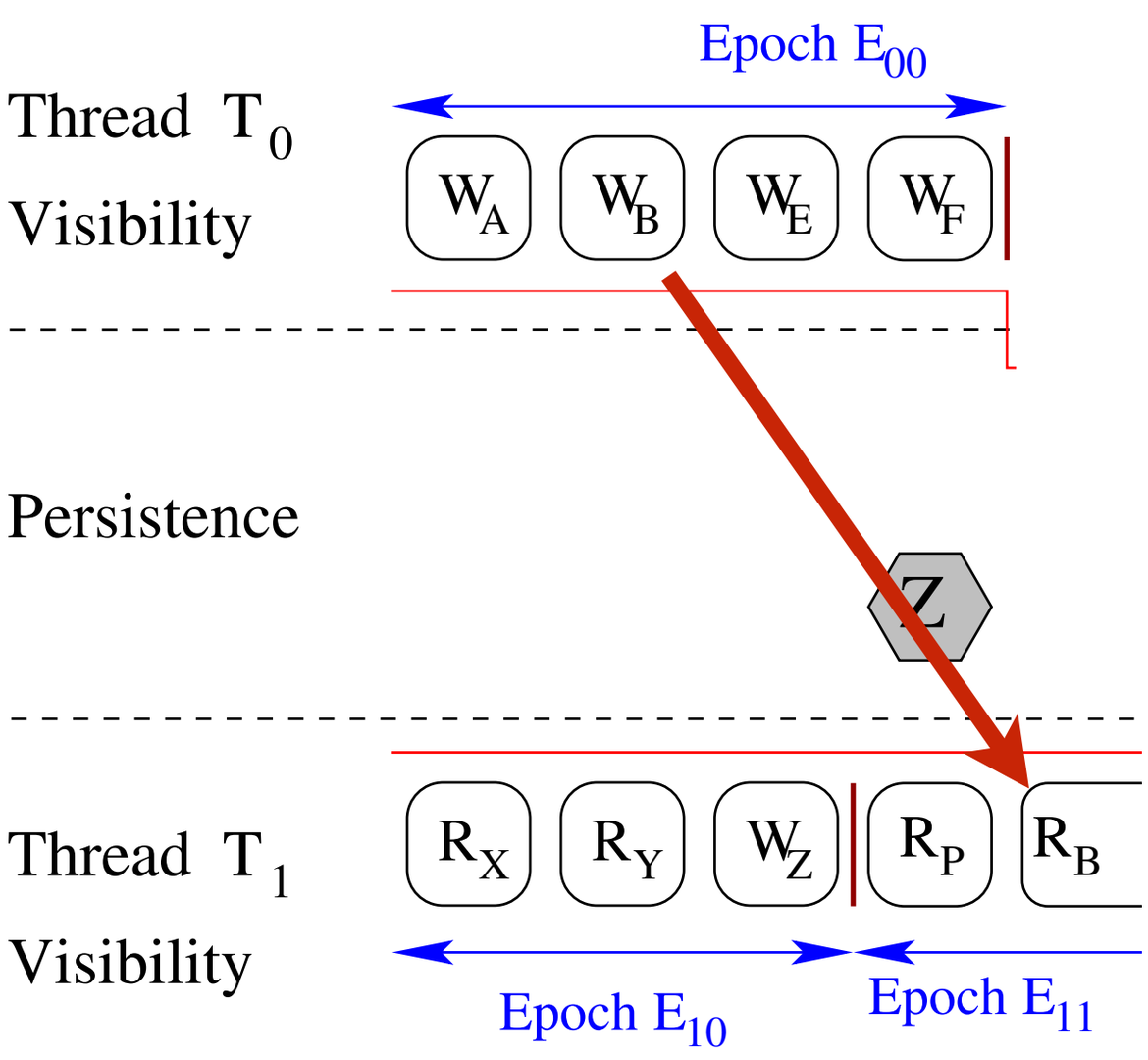
Visibility

Inter-thread Dependency Tracking (IDT)



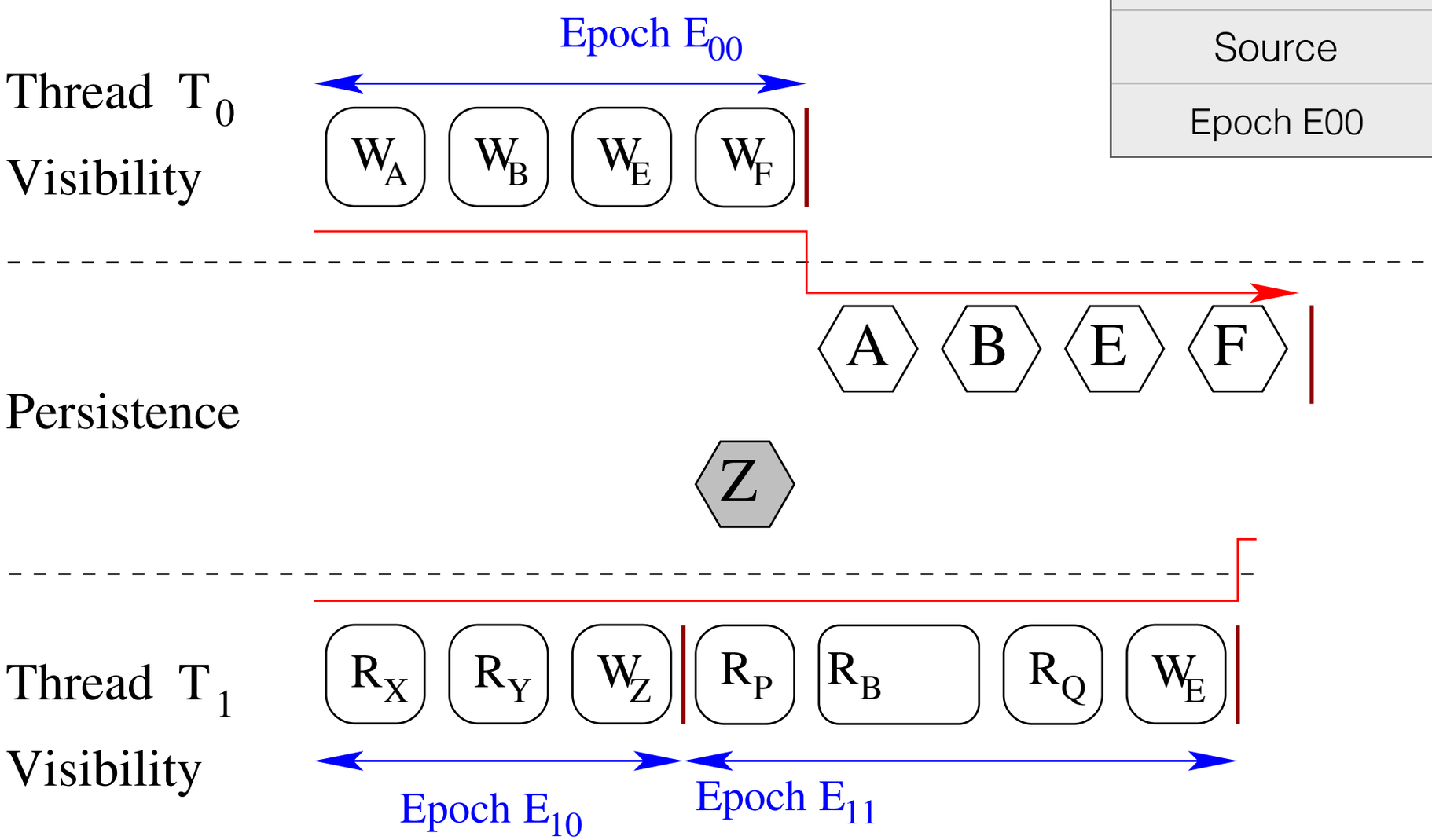
IDT Table	
Source	Dependent

Inter-thread Dependency Tracking (IDT)



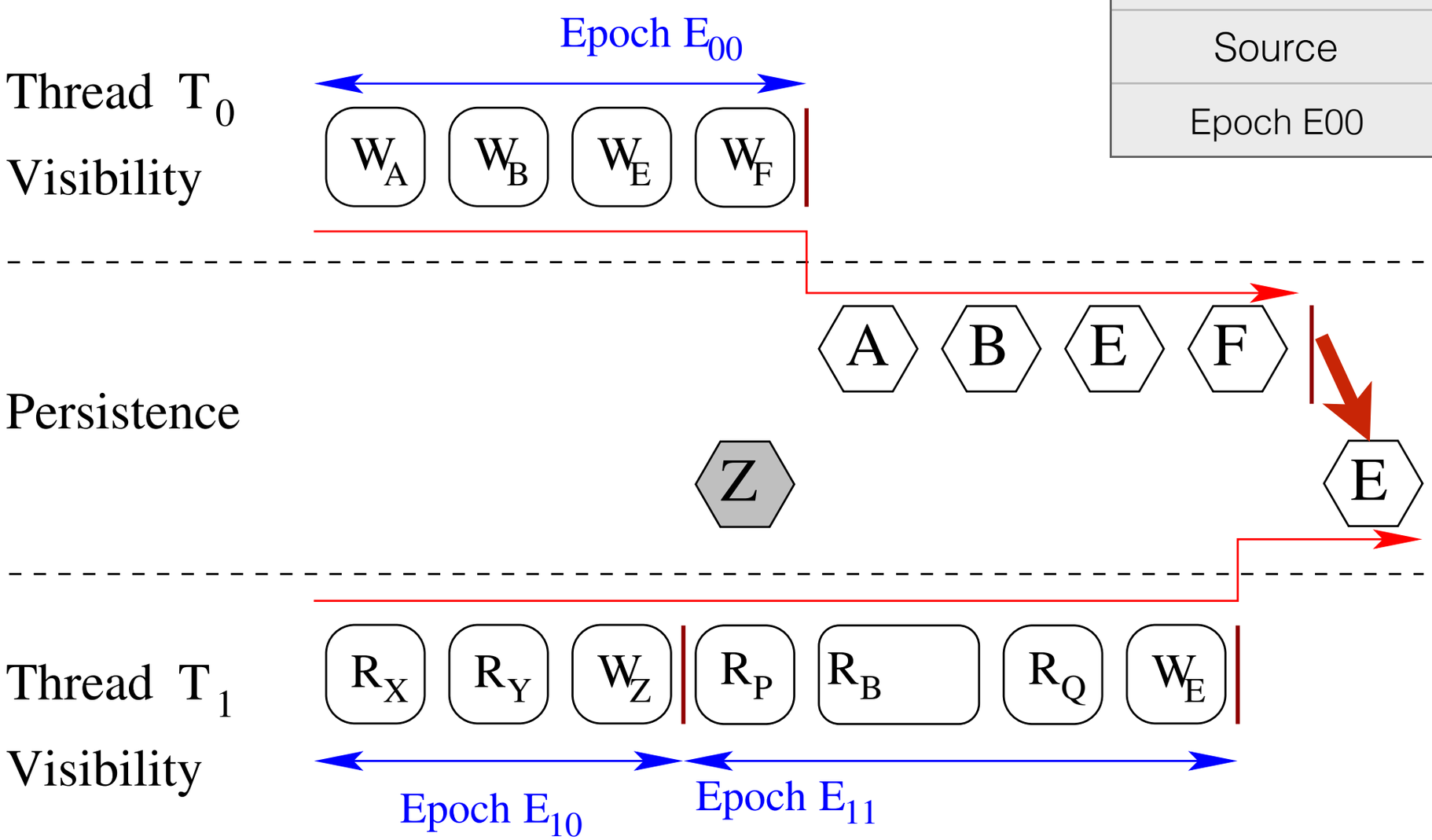
IDT Table	
Source	Dependent
Epoch E00	Epoch E11

Inter-thread Dependency Tracking (IDT)



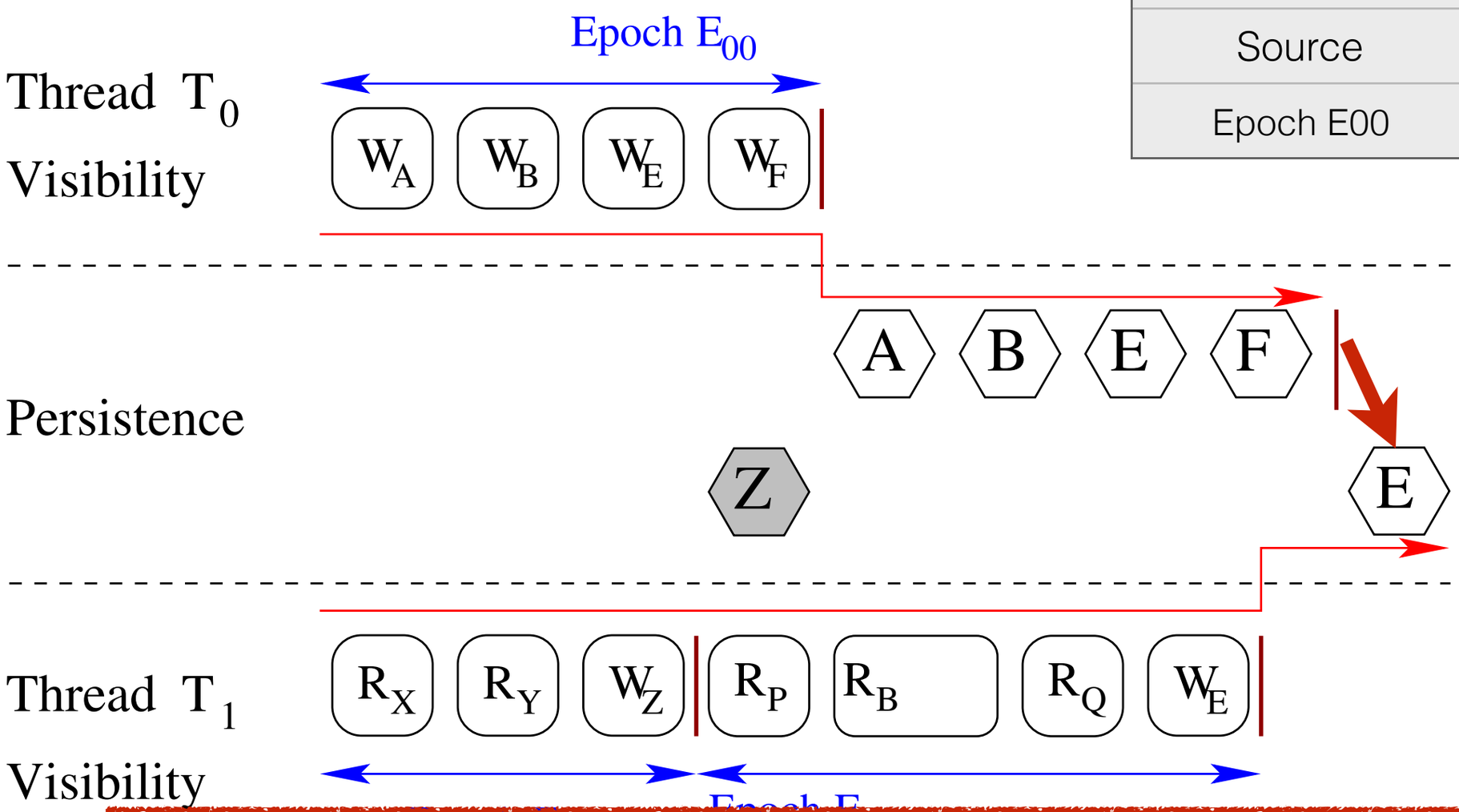
IDT Table	
Source	Dependent
Epoch E00	Epoch E11

Inter-thread Dependency Tracking (IDT)



IDT Table	
Source	Dependent
Epoch E00	Epoch E11

Inter-thread Dependency Tracking (IDT)



IDT Table	
Source	Dependent
Epoch E00	Epoch E11

Reduces the latency of conflicting requests.

Evaluation

LB	Lazy barrier
LB+IDT	Lazy barrier with inter-thread dependence tracking (IDT)
LB+PF	Lazy barrier with proactive flush (PF)
LB++	Lazy barrier with both IDT and PF

Persist Barrier Designs

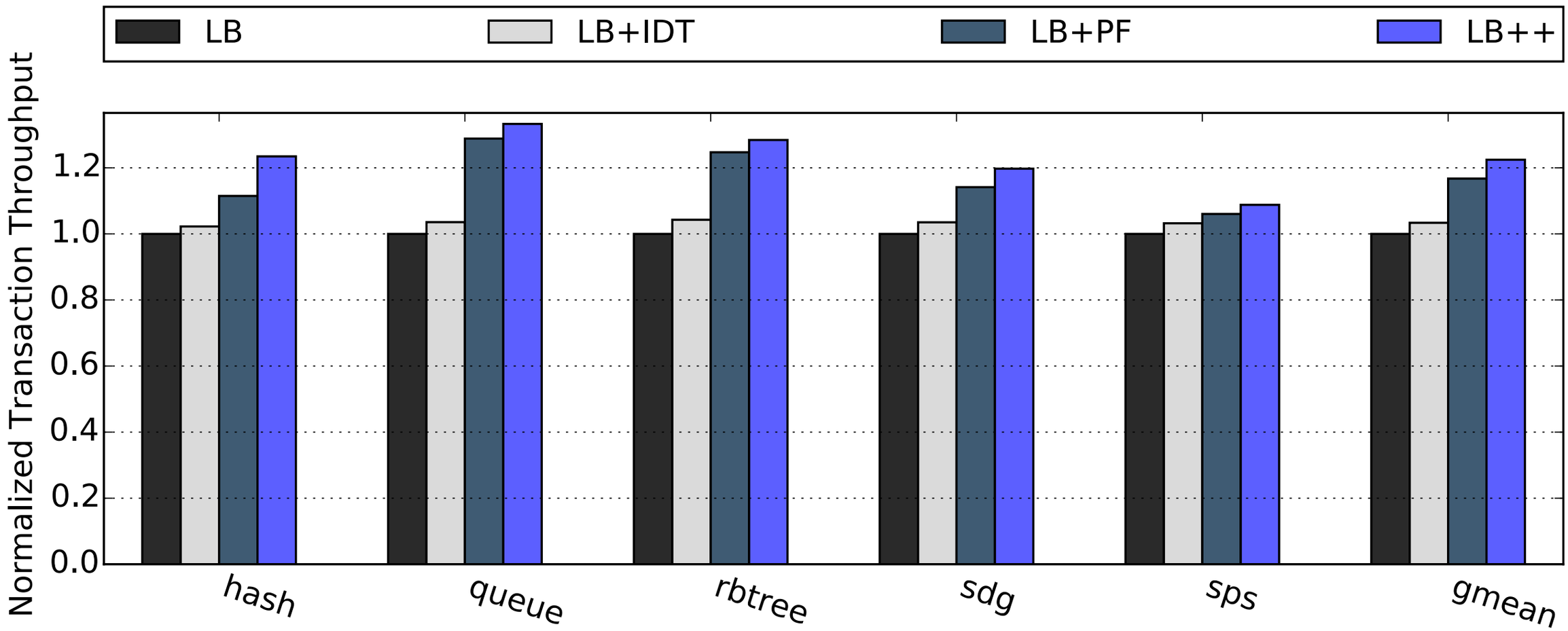
- Persistency Models
  - Buffered Epoch Persistency (BEP)
    - maintaining in-memory persistent data structures
  - Bulk Strict Persistency (BSP) = BEP + atomicity
    - provide stronger persistency model (strict persistency) — similar to doing sequential consistency in bulk mode*

* Ceze et. al., “BulkSC: Bulk enforcement of sequential consistency”, in ISCA-2007.

System Configuration

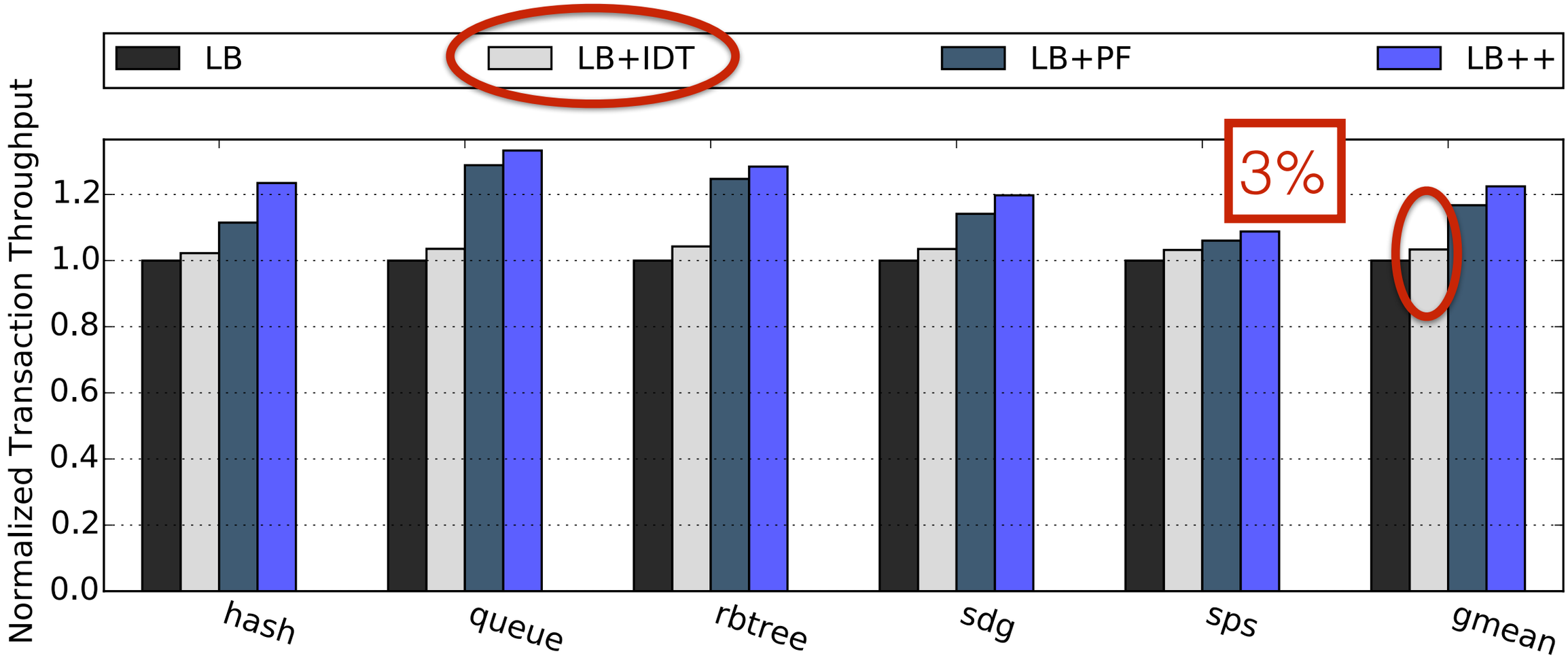
- We evaluate proposed design using GEM5 full-system simulation mode
- 32 Core CMP with 32x1MB LLC cache banks and 4 memory controllers
  - More details on implementation of persist barrier for such a system are in the paper.

BEP
Transaction Throughput



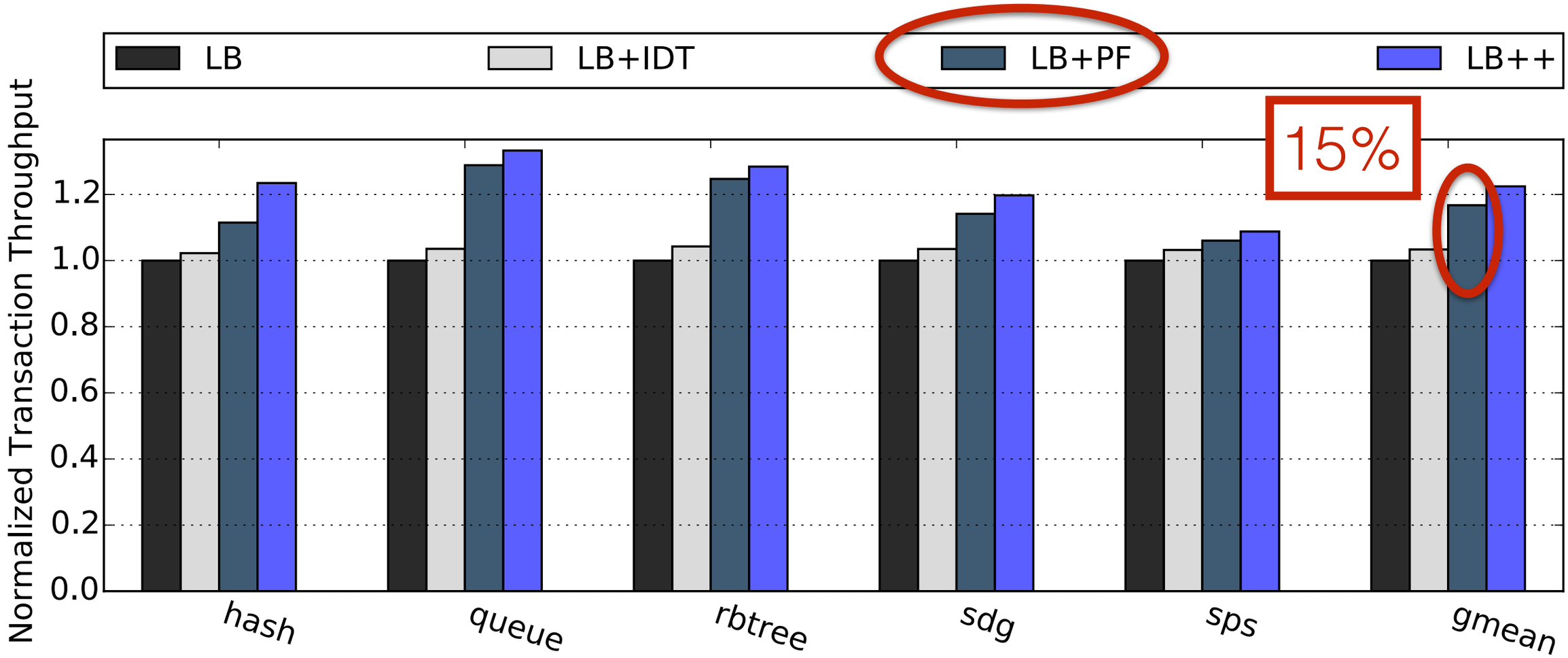
Higher is Better

BEP
Transaction Throughput



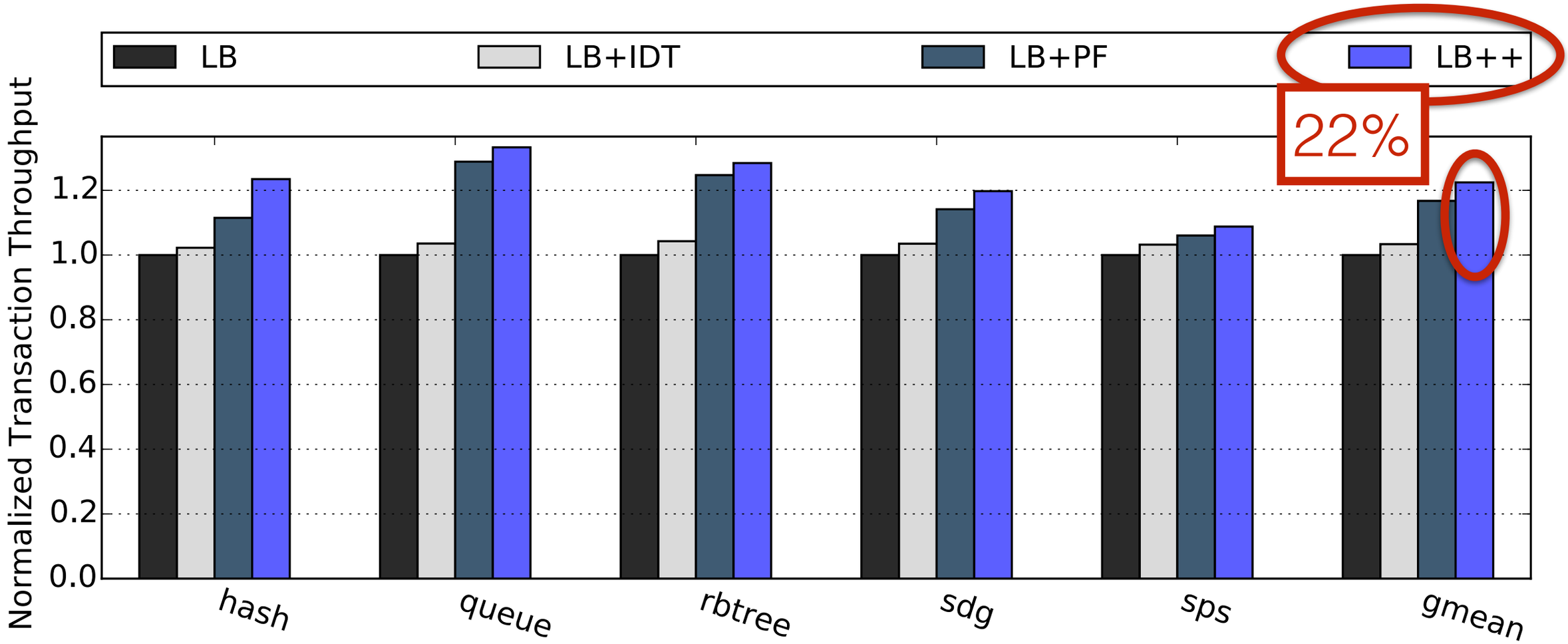
Higher is Better

BEP
Transaction Throughput



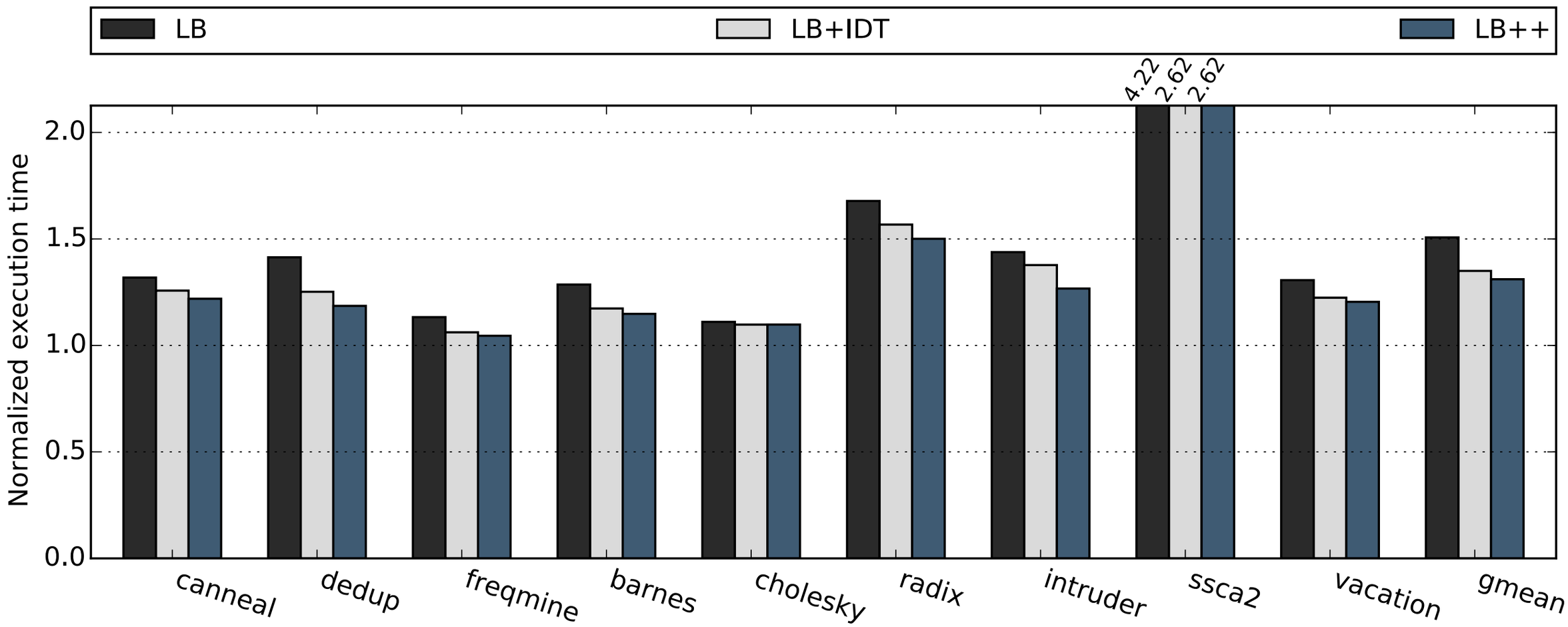
Higher is Better

BEP
Transaction Throughput



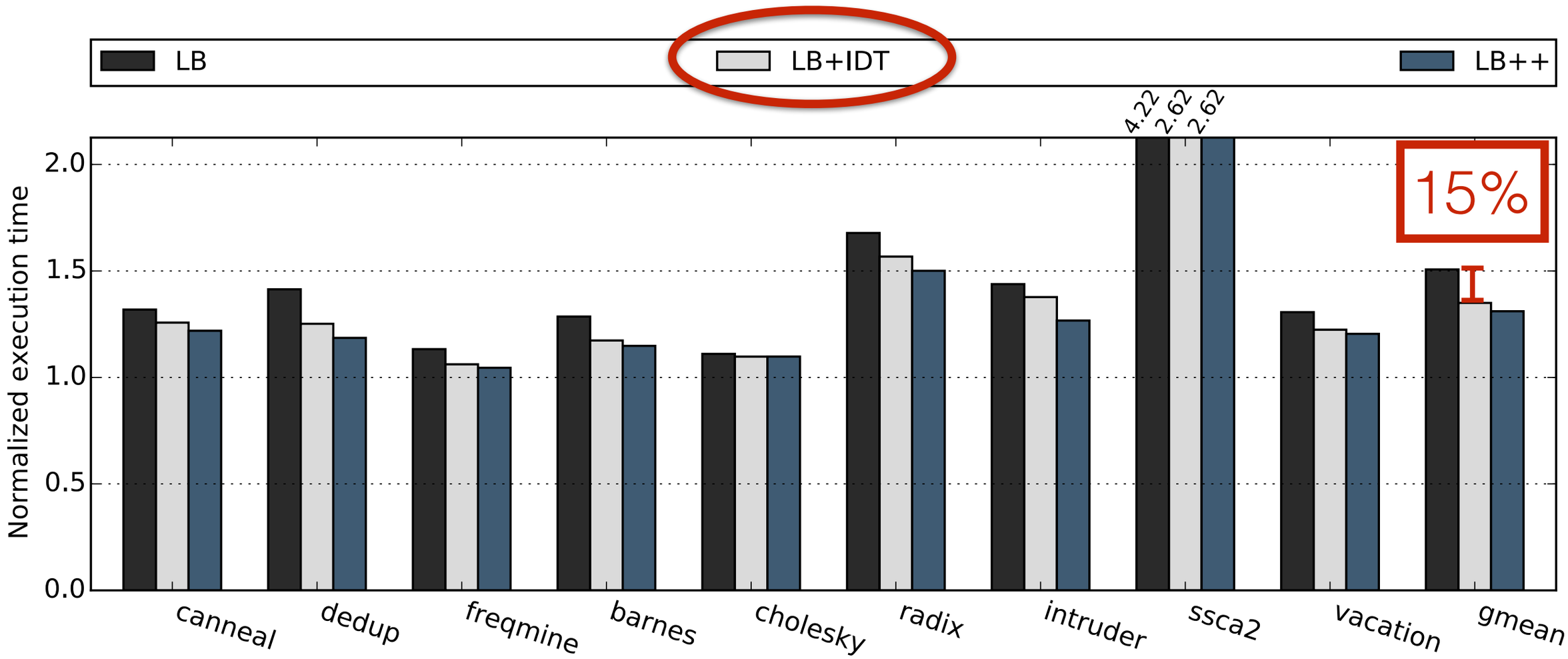
Higher is Better

BSP
Execution Time



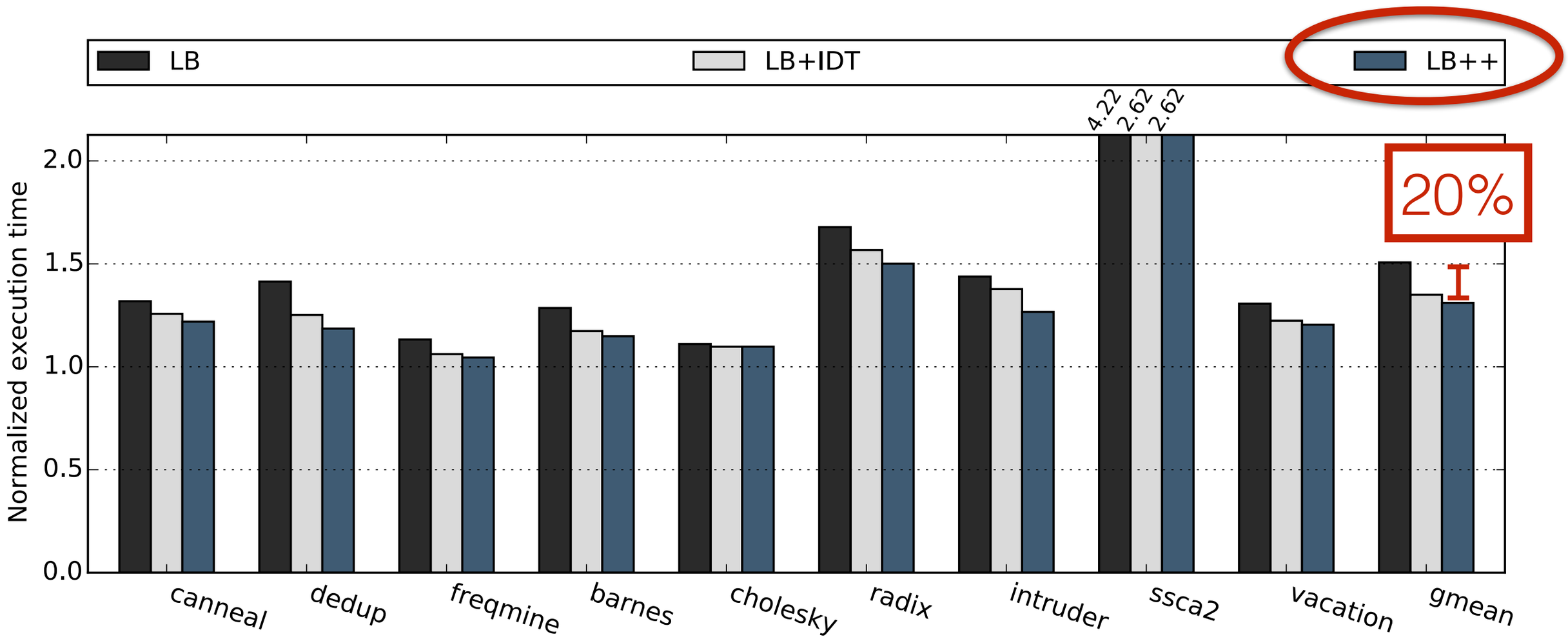
Lower is Better

BSP
Execution Time



Lower is Better

BSP
Execution Time



Lower is Better

Conclusion

- We propose an efficient implementation of a persist barrier primitive
  - Buffered implementation, to move persists out of critical path
  - We highlight how conflicts bring them back into critical path
- We propose and implement two optimizations
  - Proactive Flush: Reduce the percentage of conflicting epochs
  - Inter-thread Dependence Tracking: Reduce the penalty of inter-thread conflicts
- We demonstrate the efficacy by implementing two persistence models, namely BEP and BSP efficiently

MICRO-48

Efficient Persist Barriers for Multicores

Arpit Joshi, Vijay Nagarajan, Marcelo Cintra, Stratis Viglas

THE UNIVERSITY of EDINBURGH

intel