Relaxed Persist Ordering Using Strand Persistency

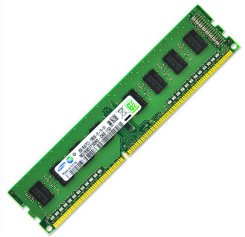
Vaibhav Gogte, William Wang[$], Stephan Diestelhorst[$],
Peter M. Chen, Satish Narayanasamy, Thomas F. Wenisch

ISCA 2020

arm[$]

Promise of persistent memory (PM)

Performance



+

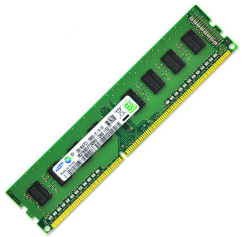
Density

+

Non-volatility

Promise of persistent memory (PM)

Performance



+

Density

+

Non-volatility

Intel Announces New Optane DC Persistent Memory *

By Joel Hruska on May 31, 2018 at 8:15 am | 1 Comment

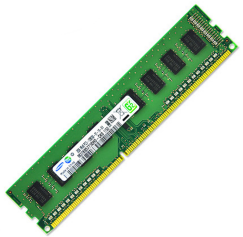
"Optane DC Persistent Memory will be offered in packages of up to 512GB per stick."

"… expanding memory per CPU socket to as much as 3TB."

* Source: www.extremetech.com

Promise of persistent memory (PM)

Performance



+

Density

+

Non-volatility

Intel Announces New Optane DC Persistent Memory *

By Joel Hruska on May 31, 2018 at 8:15 am | 1 Comment

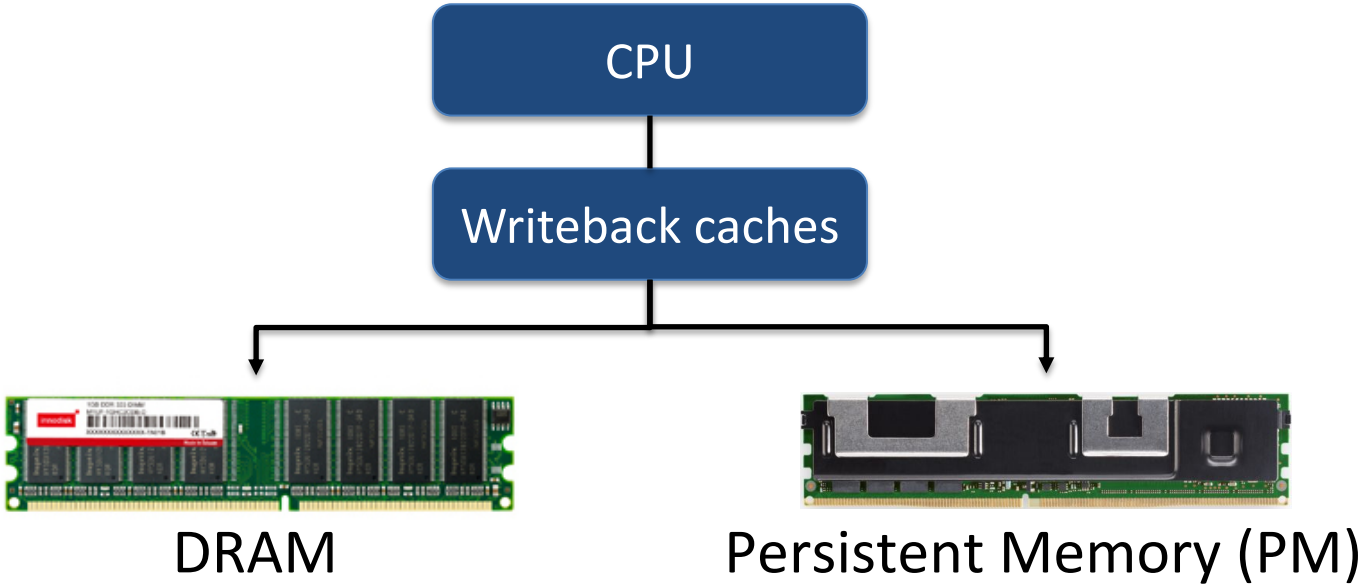
"Optane DC Persistent Memory will be offered in packages of up to 512GB per stick."

"… expanding memory per CPU socket to as much as 3TB."

* Source: www.extremetech.com

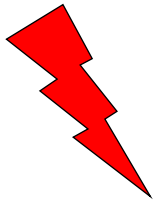
Byte-addressable, load-store interface to durable storage

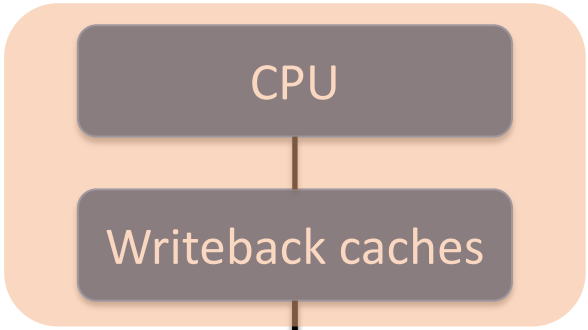
Persistent memory system



Persistent memory system

Failure



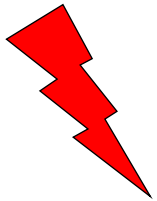


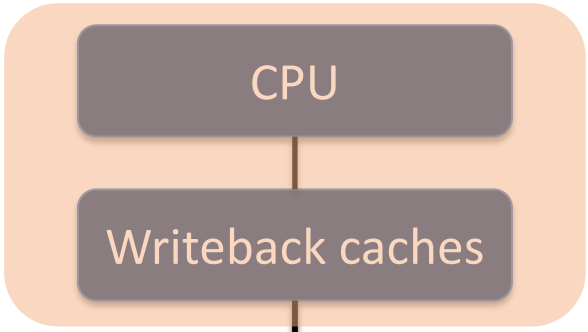
DRAM

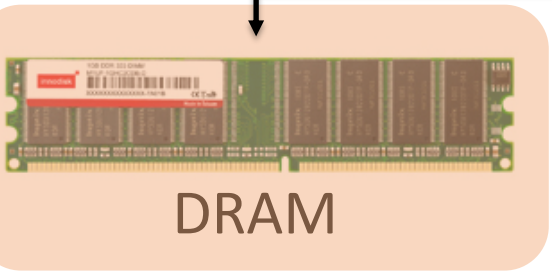
Persistent Memory (PM)

Persistent memory system

Failure







Persistent Memory (PM)

Recovery

Recovery can inspect PM data-structures to restore system to a consistent state

Recovery requires PM access ordering

Intel x86 primitives

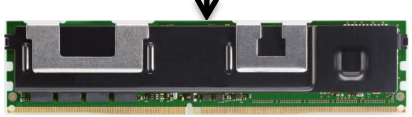
St a = x

for recovery

St b = y

CPU

Writeback caches



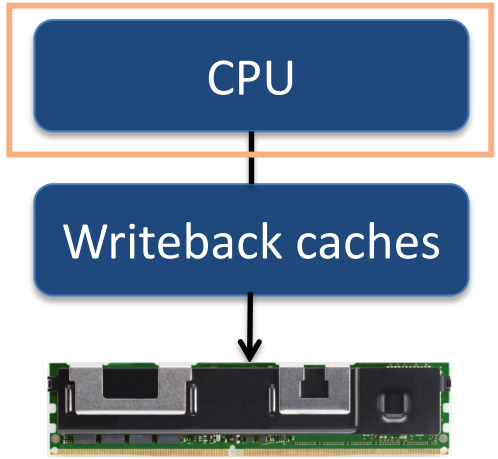
PM

Recovery requires PM access ordering

St a = x

for recovery

St b = y



Consistency model

Intel x86 primitives

St a = x

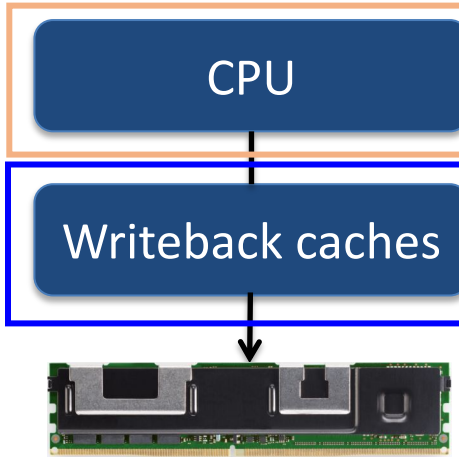
St b = y

Recovery requires PM access ordering

St a = x

for recovery

St b = y



Consistency model

Persistency model

Intel x86 primitives

St a = x

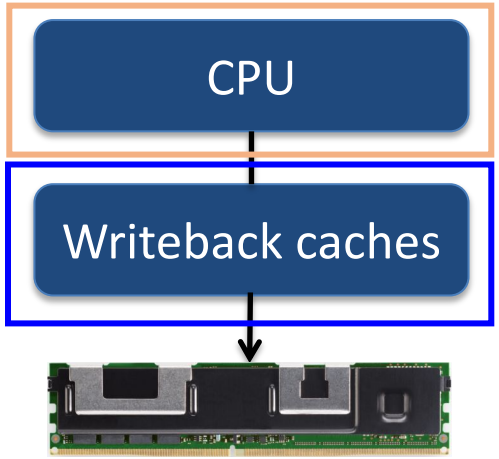
St b = y

Recovery requires PM access ordering

St a = x

for recovery

St b = y



Consistency model

Persistency model

PM

Intel x86 primitives

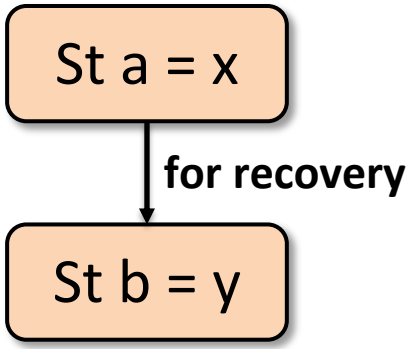
St a = x

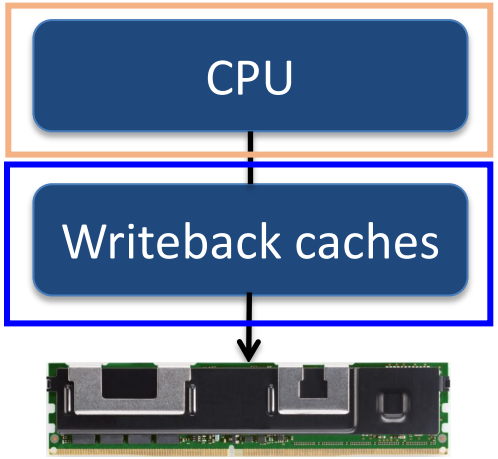
CLWB(a)

St b = y

CLWB(b)

Recovery requires PM access ordering





Consistency model

Persistency model

PM

Intel x86 primitives

St a = x

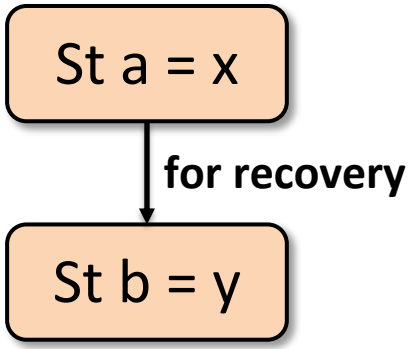
CLWB(a)

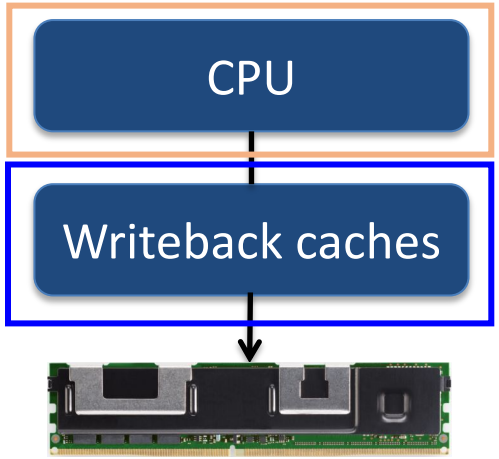
SFENCE

St b = y

CLWB(b)

Recovery requires PM access ordering





Consistency model

Persistency model

Intel x86 primitives

St a = x

CLWB(a)

SFENCE

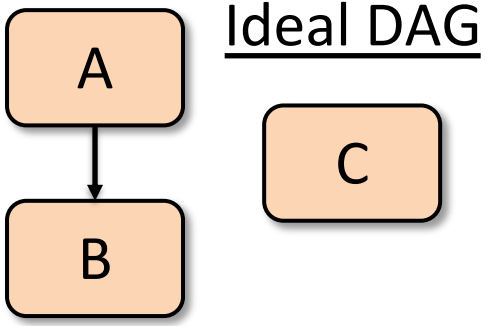
St b = y

CLWB(b)

Hardware systems provide primitives to express *persist* order to PM

Hardware imposes overly strict constraints

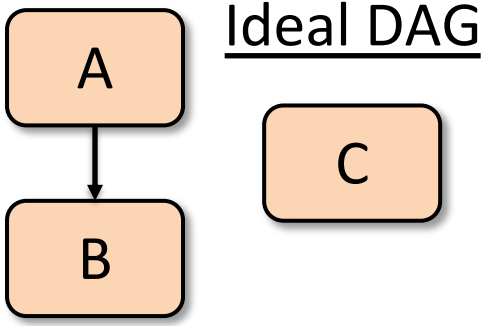
St A = 1; CLWB (A)
St B = 2; CLWB (B)
St C = 3; CLWB (C)



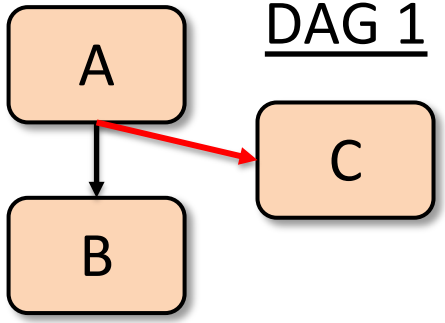
Ideal DAG

Hardware imposes overly strict constraints

St A = 1; CLWB (A)
St B = 2; CLWB (B)
St C = 3; CLWB (C)

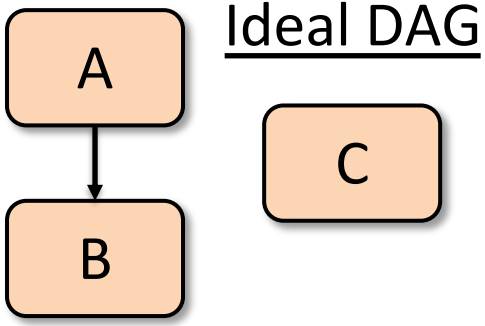


St A = 1; CLWB (A)
SFENCE
St B = 2; CLWB (B)
St C = 3; CLWB (C)

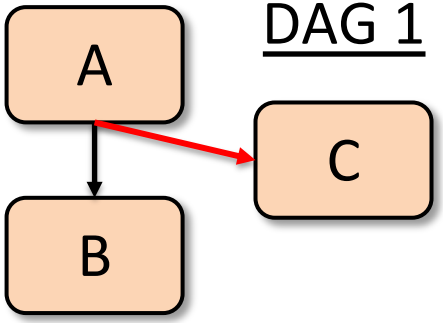


Hardware imposes overly strict constraints

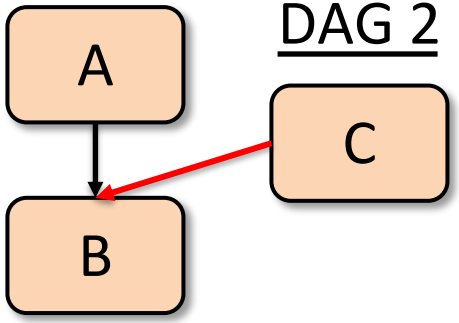
St A = 1; CLWB (A)
St B = 2; CLWB (B)
St C = 3; CLWB (C)



St A = 1; CLWB (A)
SFENCE
St B = 2; CLWB (B)
St C = 3; CLWB (C)

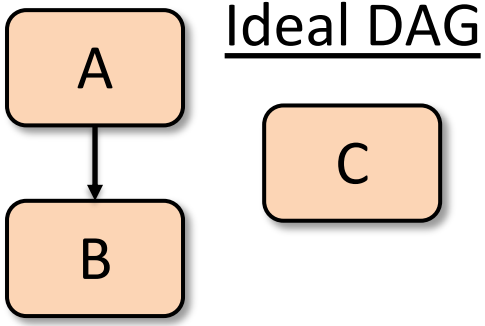


St A = 1 ; CLWB (A)
St C = 3; CLWB (C)
SFENCE
St B = 2; CLWB (B)

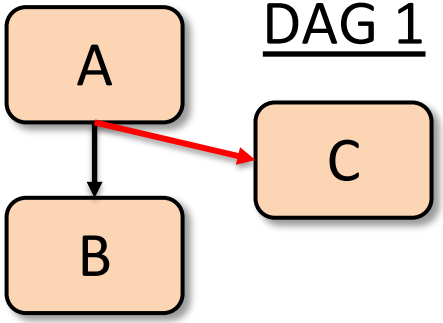


Hardware imposes overly strict constraints

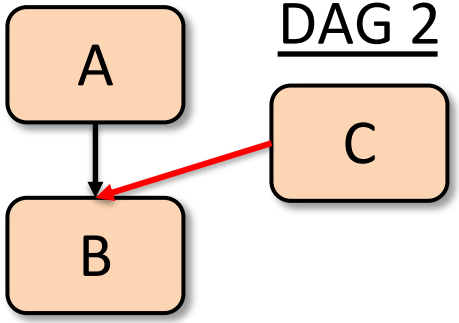
St A = 1; CLWB (A)
St B = 2; CLWB (B)
St C = 3; CLWB (C)



St A = 1; CLWB (A)
SFENCE
St B = 2; CLWB (B)
St C = 3; CLWB (C)



St A = 1 ; CLWB (A)
St C = 3; CLWB (C)
SFENCE
St B = 2; CLWB (B)



Primitives in existing hardware systems overconstrain PM accesses

Contributions

- Our proposal: StrandWeaver
  - Builds strand persistency model in hardware
  - Specifies precise persist ordering constraints
- Comprises primitives: **PersistBarrier**, **NewStrand**, and **JoinStrand**
  - Can encode an arbitrary DAG
- Map language-level persistency models to ISA level primitives
  - Leverage hw primitives to build persistency models efficiently

Contributions

- Our proposal: StrandWeaver
  - Builds strand persistency model in hardware
  - Specifies precise persist ordering constraints
- Comprises primitives: **PersistBarrier**, **NewStrand**, and **JoinStrand**
  - Can encode an arbitrary DAG
- Map language-level persistency models to ISA level primitives
  - Leverage hw primitives to build persistency models efficiently

StrandWeaver results in 1.45x (avg.) speedup over Intel x86

Outline

- Contributions
- Example: Failure atomicity
- Existing hardware vs. strand persistency model
- Our proposal: StrandWeaver
- Evaluation

Example: Failure atomicity

Failure atomicity:

Which group of stores persist atomically?

Failure-atomic region

```
atomic_begin()
    x = 100;
    y = 200;
atomic_end()
```

Example: Failure atomicity

Failure atomicity:

Which group of stores persist atomically?

Failure-atomic region

atomic_begin()

x = 100;

y = 200;

atomic_end()

Failure atomicity limits state that recovery can observe after failure

Undo logging for failure atomicity

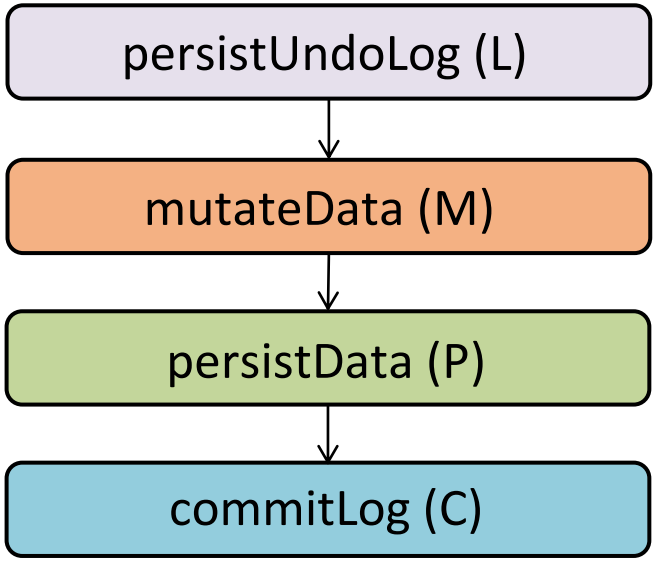
Init: x = 0; y = 0

atomic_begin()

x = 1;

y = 2;

atomic_end()



Undo logging for failure atomicity

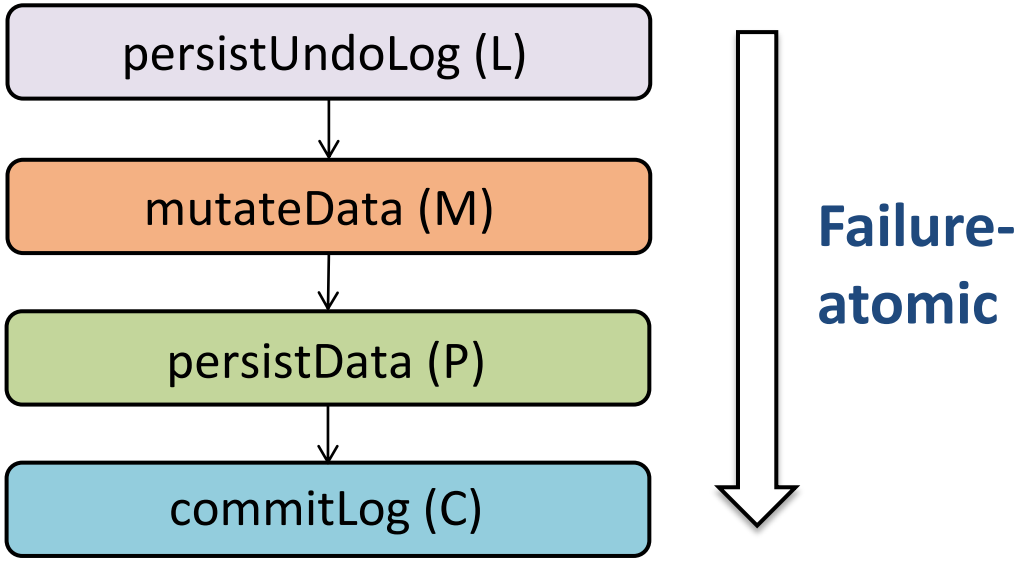
Init: x = 0; y = 0

atomic_begin()

x = 1;

y = 2;

atomic_end()



Undo logging steps ordered to ensure failure atomicity

Undo logging for failure atomicity

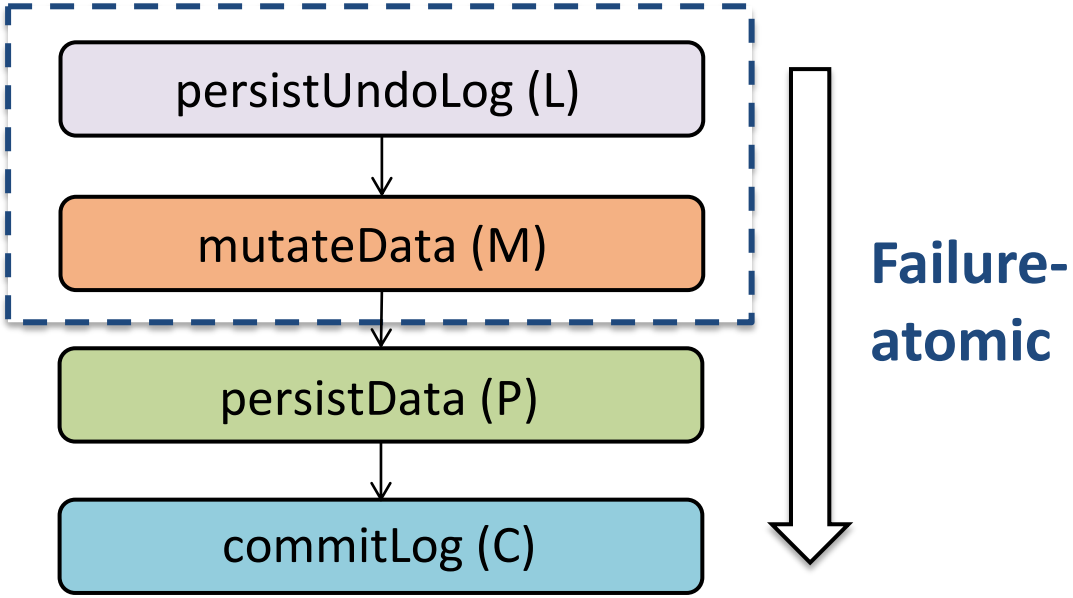
Init: x = 0; y = 0

atomic_begin()

x = 1;

y = 2;

atomic_end()

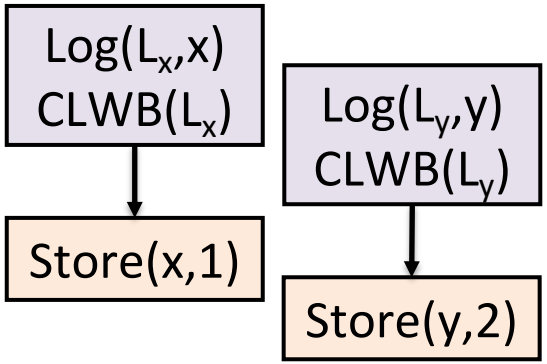


Undo logging steps ordered to ensure failure atomicity

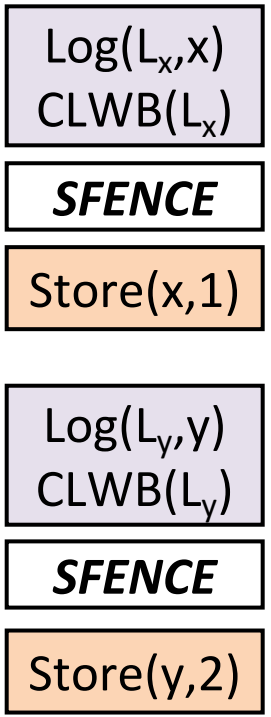
Hardware imposes stricter constraints

```
atomic_begin()
    x = 1;
    y = 2;
atomic_end()
```

Ideal ordering



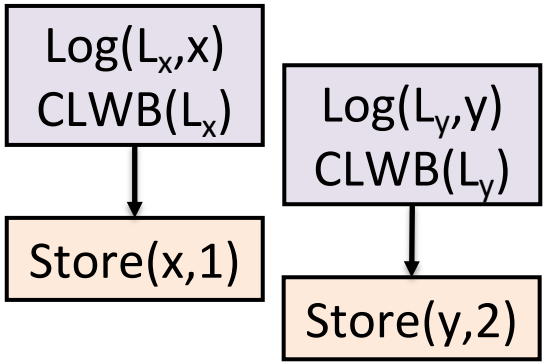
SFENCE ordering



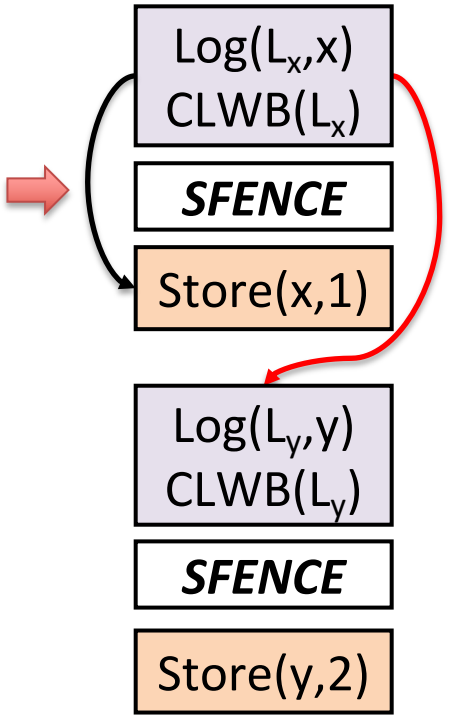
Hardware imposes stricter constraints

```
atomic_begin()
    x = 1;
    y = 2;
atomic_end()
```

Ideal ordering



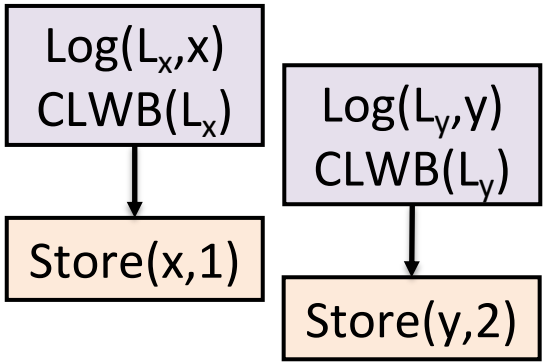
SFENCE ordering



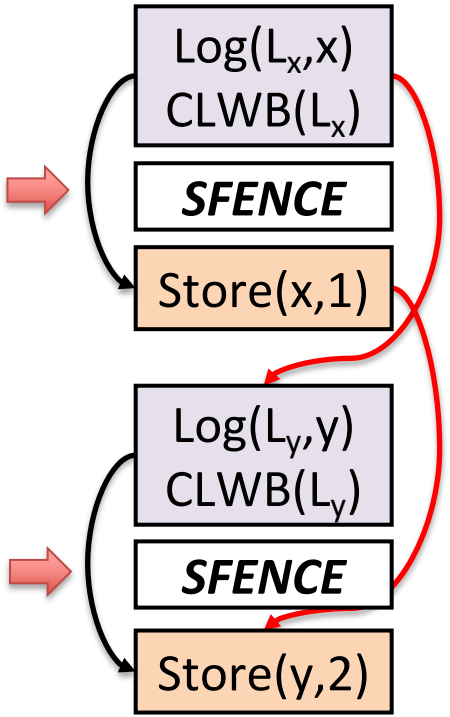
Hardware imposes stricter constraints

```
atomic_begin()
    x = 1;
    y = 2;
atomic_end()
```

Ideal ordering



SFENCE ordering



StrandWeaver: Hardware Strand Persistency Model

High-level languages	Failure atomicity for language-level persistency models
Compiler	Logging impl. that map to hardware primitives
Hardware ISA	ISA primitives: PersistBarrier, NewStrand, JoinStrand

StrandWeaver: Hardware Strand Persistency Model

High-level languages — Failure atomicity for language-level persistency models

Compiler — Logging impl. that map to hardware primitives

Hardware ISA — ISA primitives: PersistBarrier, NewStrand, JoinStrand

StrandWeaver enables persist concurrency

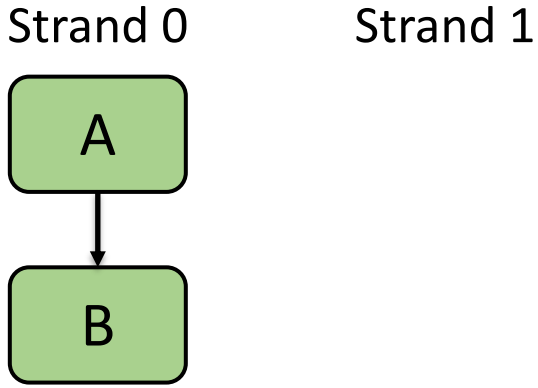
- Provides primitives to express precise persist order

Orders persists within a thread ←

Persist A

PersistBarrier

Persist B



StrandWeaver enables persist concurrency

- Provides primitives to express precise persist order

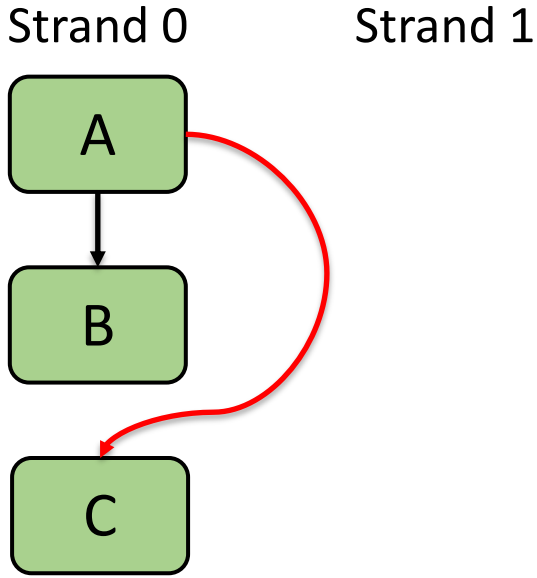
Orders persists within a thread ←

Persist A

PersistBarrier

Persist B

Persist C



StrandWeaver enables persist concurrency

- Provides primitives to express precise persist order

strand

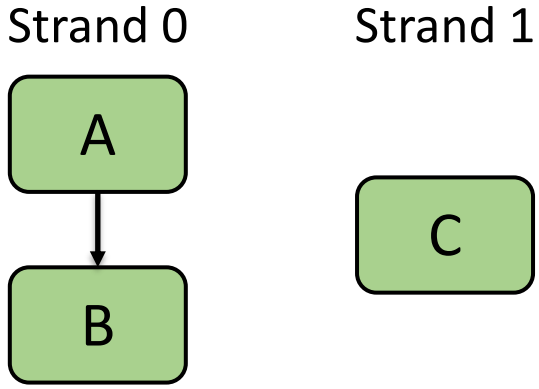
Orders persists within a ~~thread~~ ← *PersistBarrier*

Initiates new stream of persists ← *NewStrand*

Persist A

Persist B

Persist C



StrandWeaver enables persist concurrency

- Provides primitives to express precise persist order

strand

Orders persists within a ~~thread~~ ← *PersistBarrier*

Initiates new stream of persists ← *NewStrand*

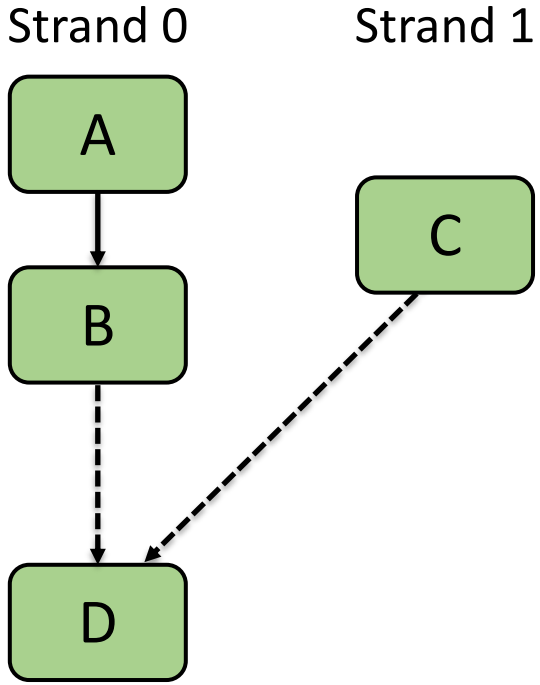
Merges prior initiated strands ← *JoinStrand*

Persist A

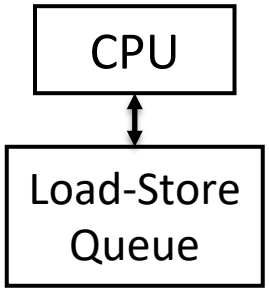
Persist B

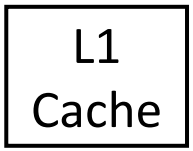
Persist C

Persist D

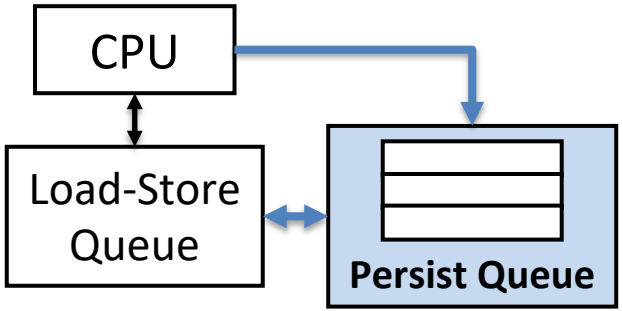


StrandWeaver architecture





StrandWeaver architecture

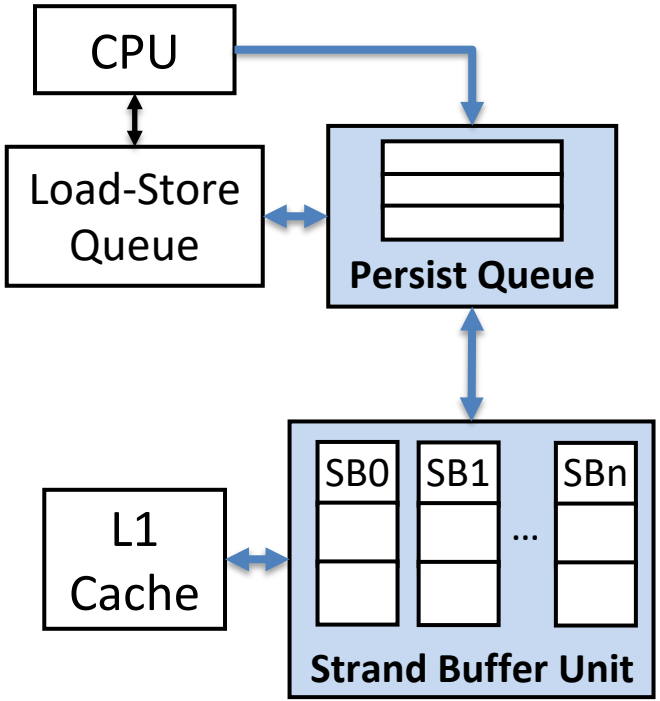


Persist queue

- Manages ongoing StrandWeaver primitives
- Orders CLWBs separated by *JoinStrand*

L1 Cache

StrandWeaver architecture



Persist queue

- Manages ongoing StrandWeaver primitives
- Orders CLWBs separated by *JoinStrand*

Strand Buffer Unit

- Issues CLWBs and flushes dirty cache lines
- Ensures CLWBs on diff. strands are concurrent
- Monitors coherence reqs. for inter-thread order

Running example

Example code

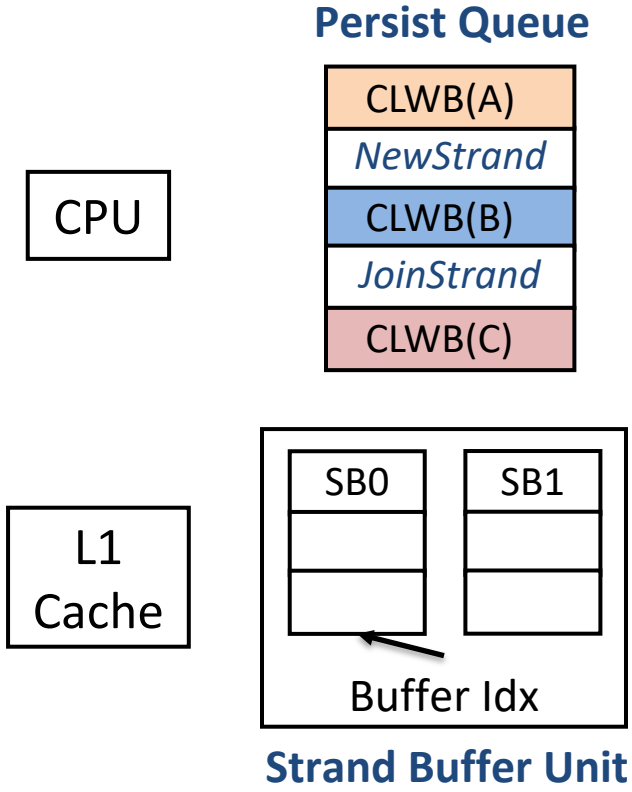
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

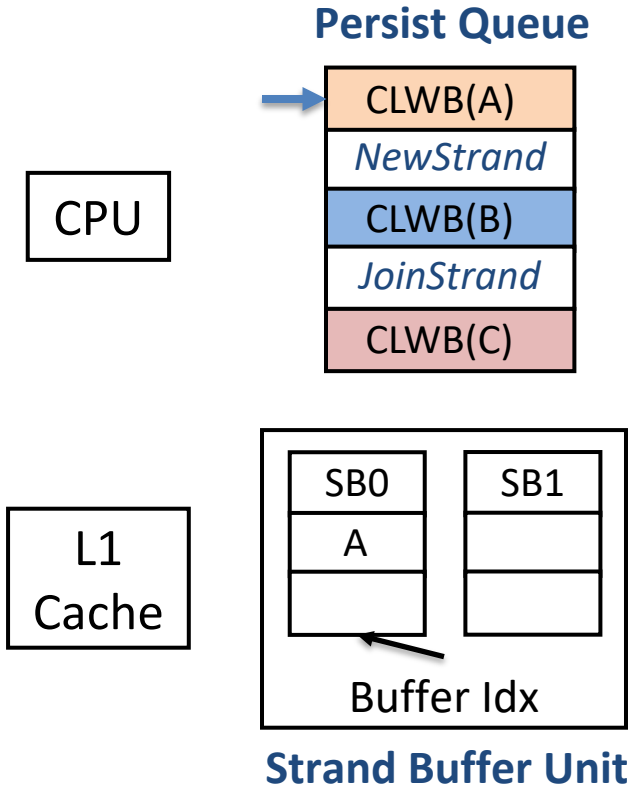
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

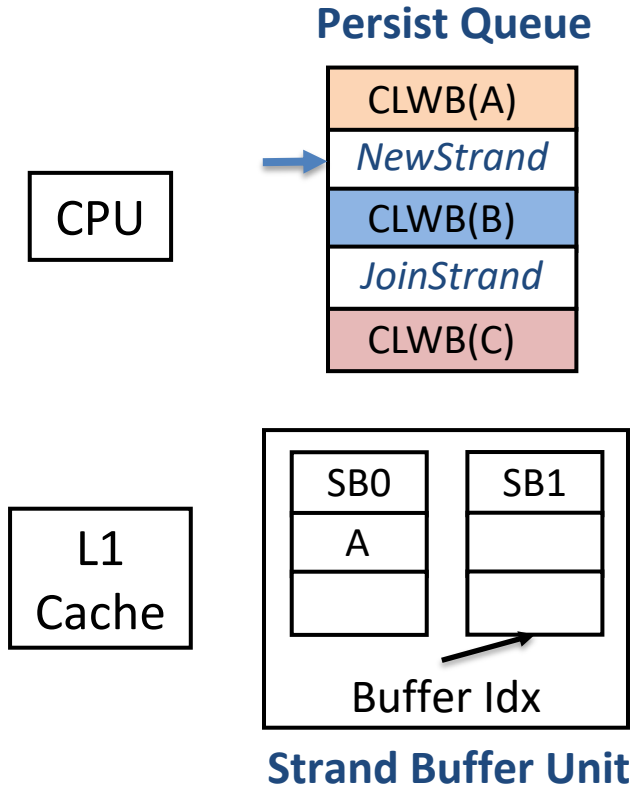
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

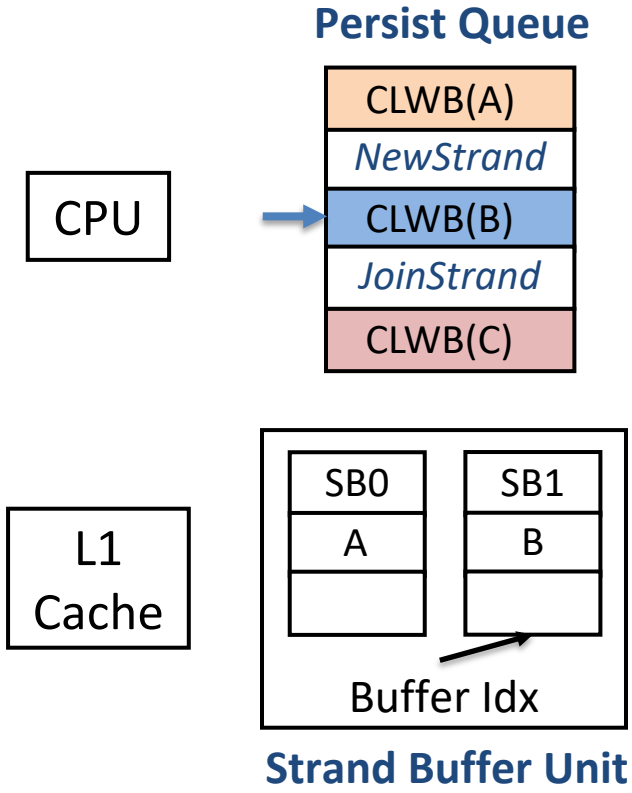
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

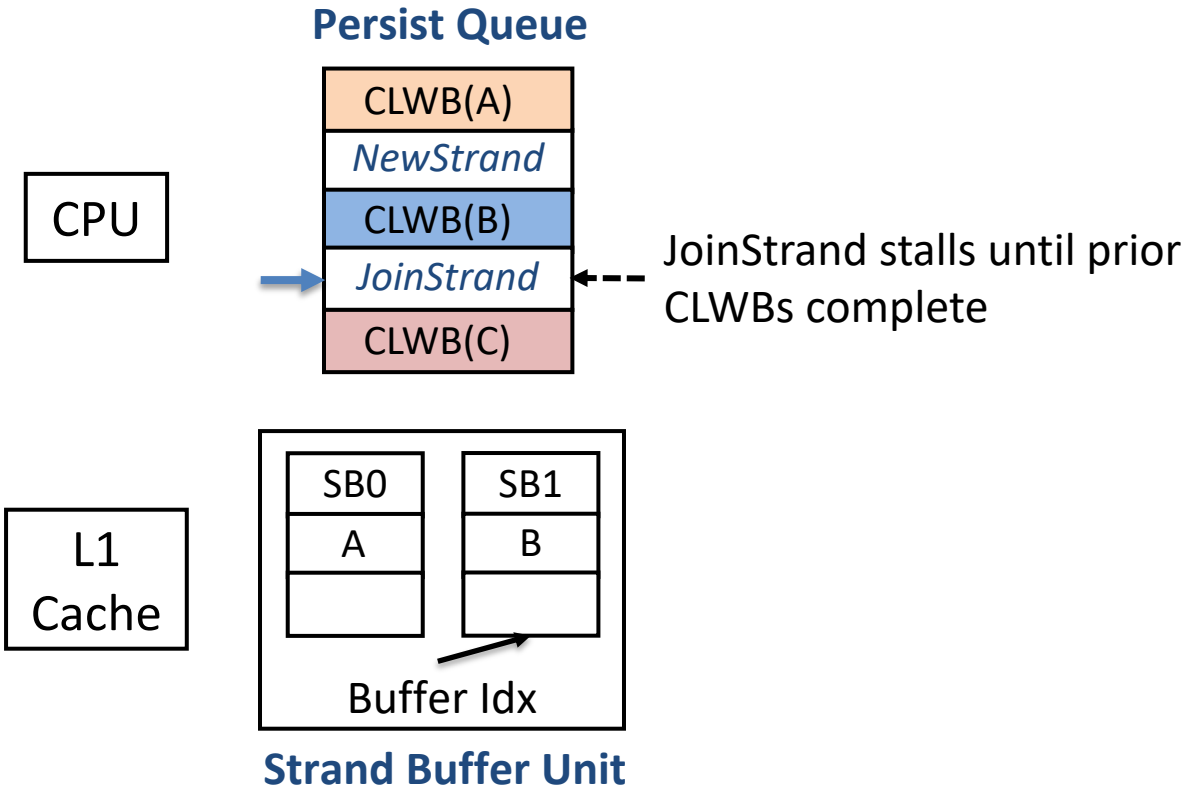
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

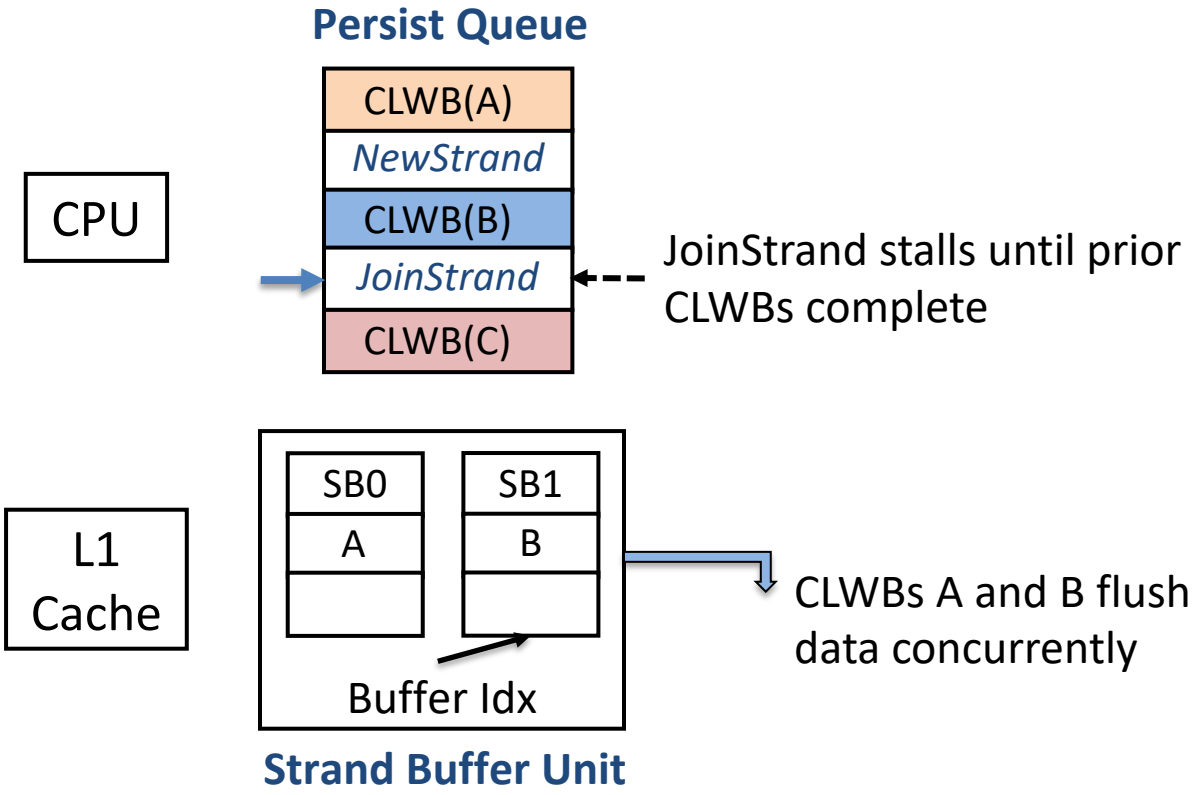
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

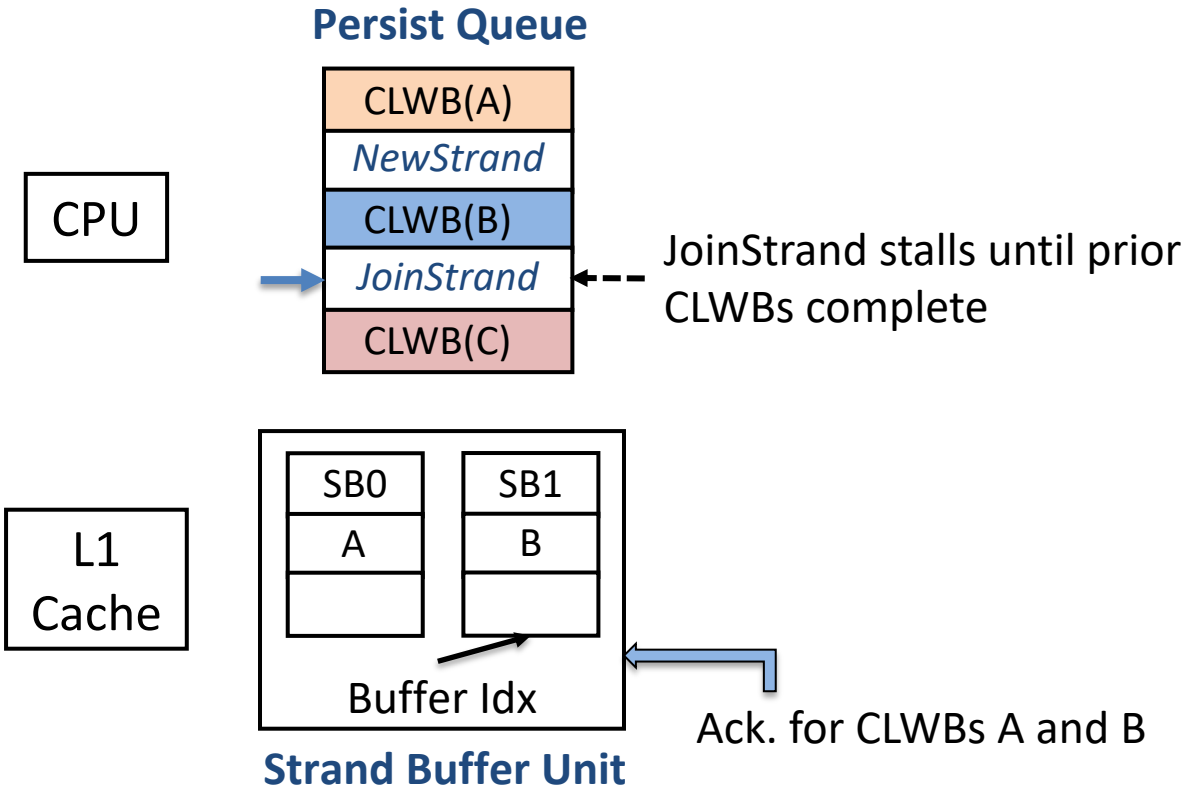
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

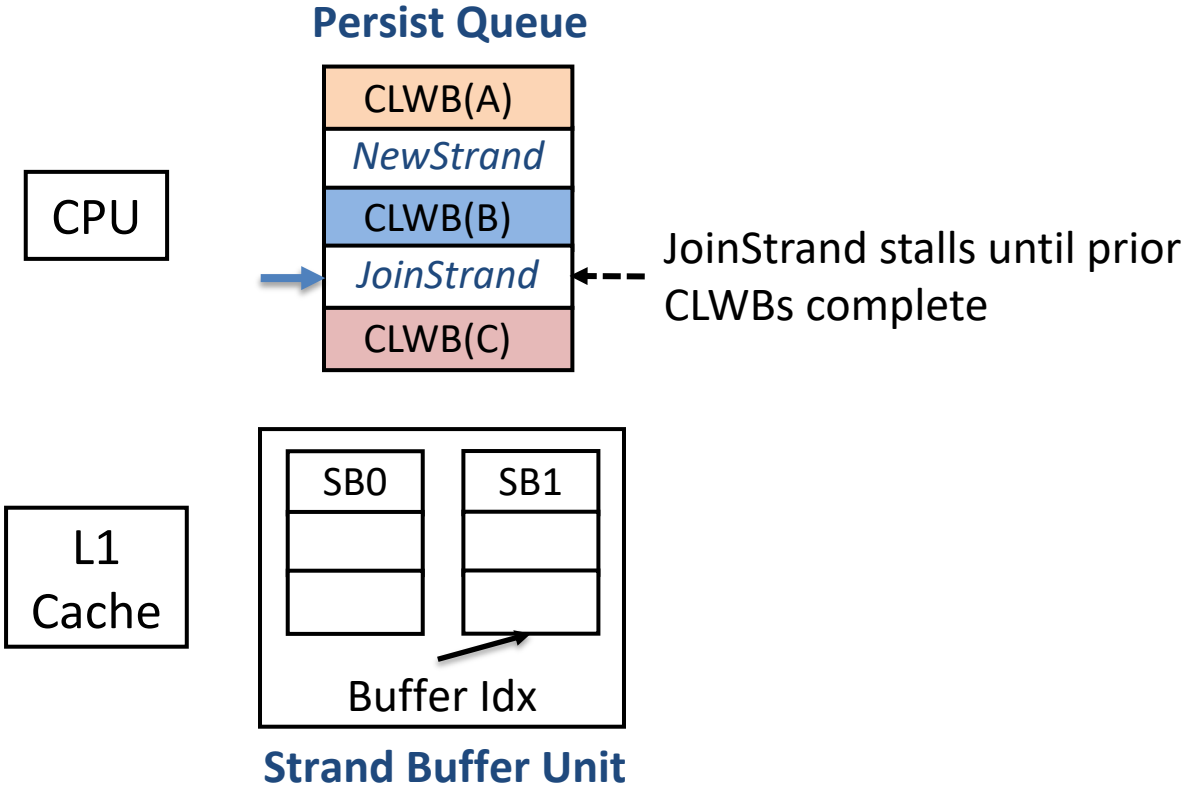
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



Running example

Example code

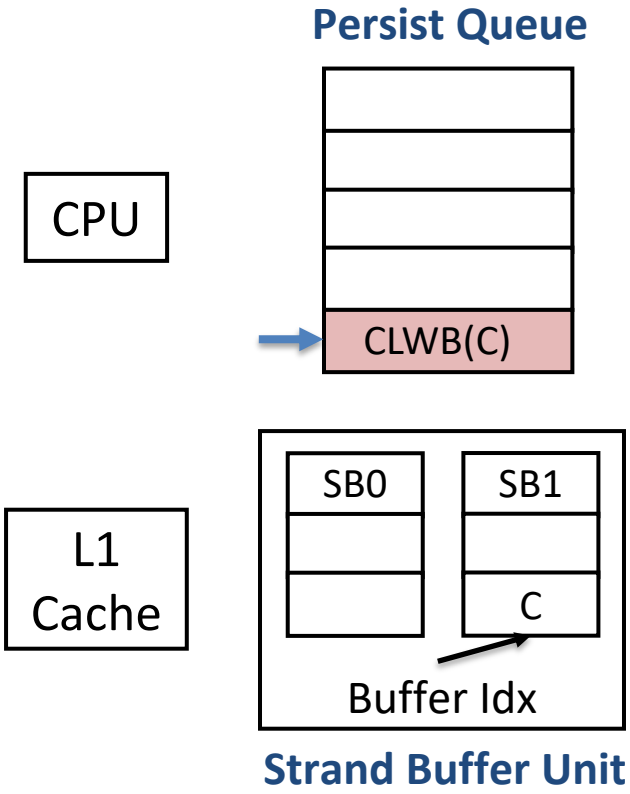
CLWB(A)

NewStrand

CLWB(B)

JoinStrand

CLWB(C)



StrandWeaver: From ISA to high-level language

High-level languages — Failure atomicity for language-level persistency models

Compiler — Logging impl. that map to hardware primitives

Hardware ISA — ISA primitives: PersistBarrier, NewStrand, JoinStrand

Logging using StrandWeaver primitives

```
atomic_begin()
    x = 1;
    y = 2;
atomic_end()
```

Log(L_x,x)
CLWB(L_x)

PersistBarrier

Store(x,1)

NewStrand

Log(L_y,y)
CLWB(L_y)

PersistBarrier

Store(y,2)

JoinStrand

Logging using StrandWeaver primitives

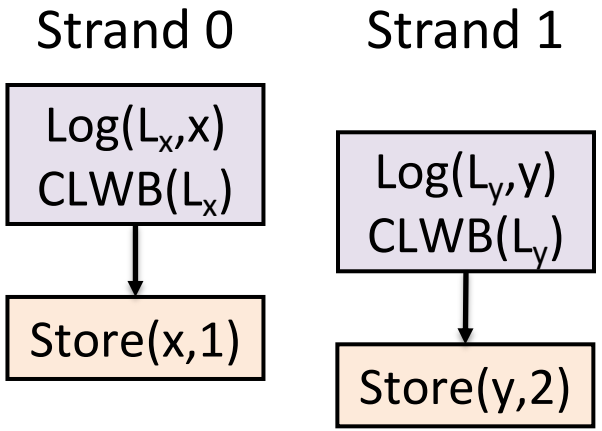
```
atomic_begin()
    x = 1;
    y = 2;
atomic_end()
```

Log(L_x,x)
CLWB(L_x)
PersistBarrier
Store(x,1)

NewStrand

Log(L_y,y)
CLWB(L_y)
PersistBarrier
Store(y,2)

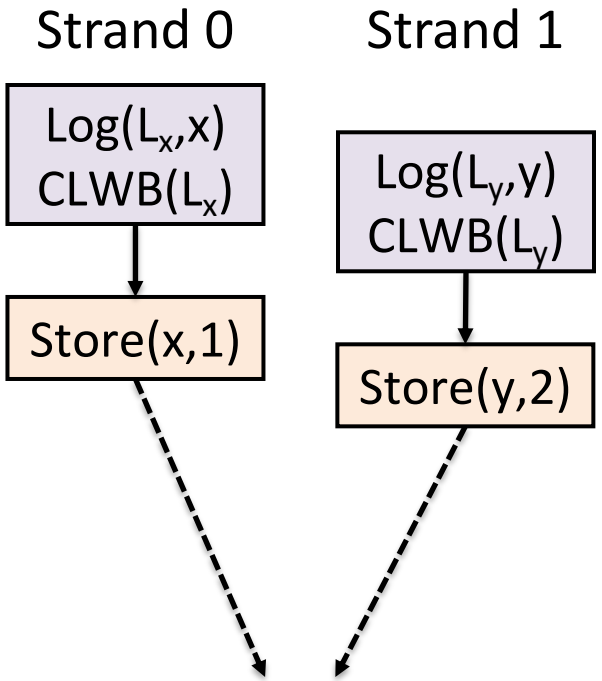
JoinStrand



Logging using StrandWeaver primitives

```
atomic_begin()
    x = 1;
    y = 2;
atomic_end()
```

Log(L_x,x)
CLWB(L_x)
PersistBarrier
Store(x,1)
NewStrand
Log(L_y,y)
CLWB(L_y)
PersistBarrier
Store(y,2)
JoinStrand



StrandWeaver: From ISA to high-level language

High-level languages — Failure atomicity for language-level persistency models

Compiler — Logging impl. that map to hardware primitives

Hardware ISA — ISA primitives: PersistBarrier, NewStrand, JoinStrand

High-level language implementations

```
L1.lock();
      x -= 100;
      y += 100;
      L2.lock();
            a -= 100;
            b += 100;
      L2.unlock();
L1.unlock();
```

ATLAS [Chakrabarti14]

- Failure-atomic outermost critical sections

High-level language implementations

```
L1.lock();
        x -= 100;
        y += 100;
        L2.lock();
                a -= 100;
                b += 100;
        L2.unlock();
L1.unlock();
```

ATLAS [Chakrabarti14]

- Failure-atomic outermost critical sections

Coupled-SFR [Gogte18]

- Failure-atomic synchronization-free regions

Decoupled-SFR [Gogte18]

- Failure-atomic synchronization-free regions

High-level language implementations

```
L1.lock();
        x -= 100;
        y += 100;
        L2.lock();
                a -= 100;
                b += 100;
        L2.unlock();
L1.unlock();
```

ATLAS [Chakrabarti14]

- Failure-atomic outermost critical sections

Coupled-SFR [Gogte18]

- Failure-atomic synchronization-free regions

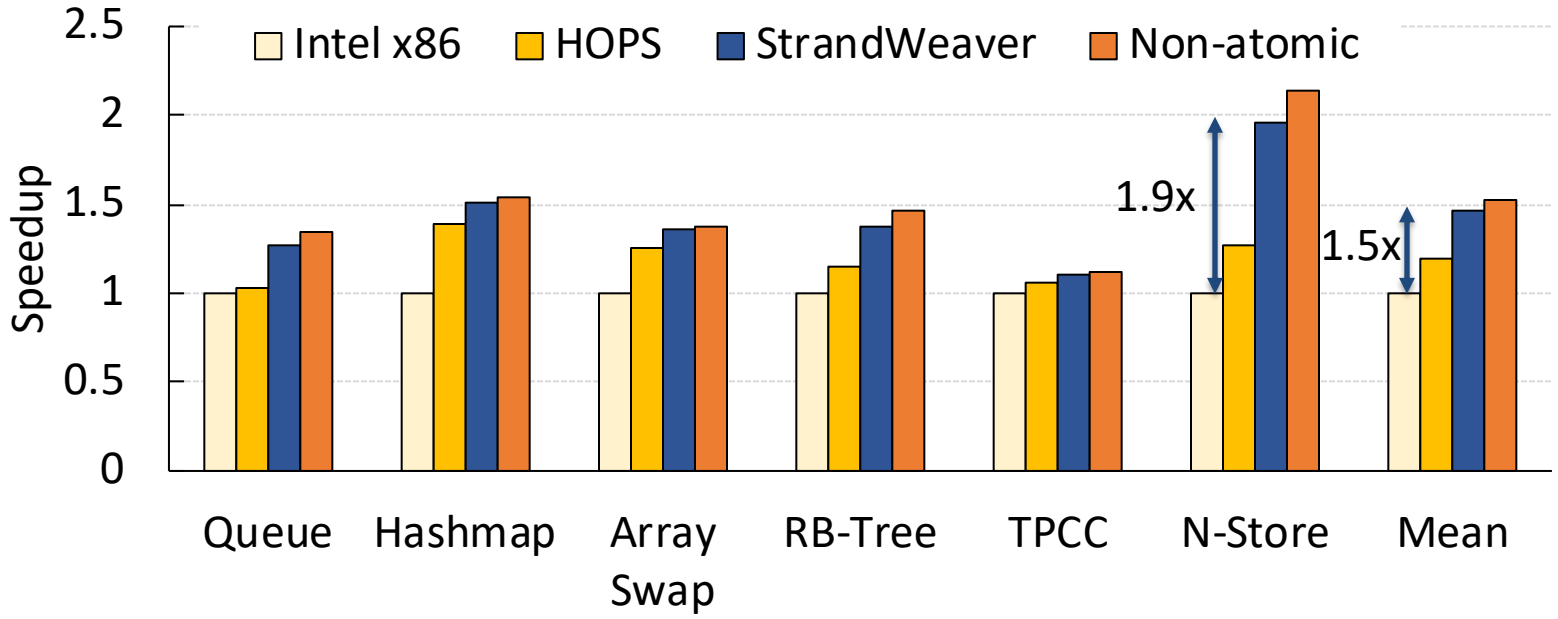
Decoupled-SFR [Gogte18]

- Failure-atomic synchronization-free regions

Methodology

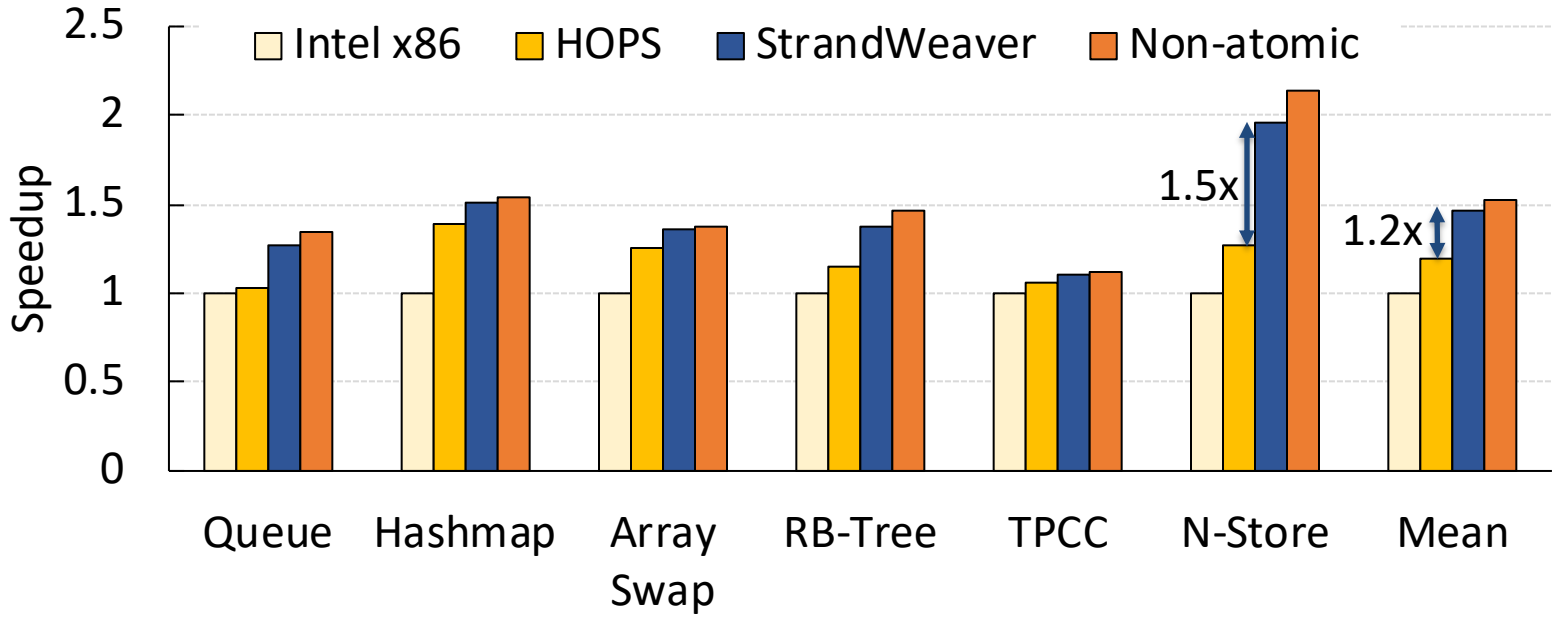
- Gem5 simulator
- Micro-benchmarks:
  - **Queue:** insert/delete entries in a queue
  - **Hashmap**: update values in persistent hash table
  - **Array swaps:** random swaps of array elements
  - **RBTree**: insert/delete entries in red-black tree
  - **TPCC:** new order transaction from TPCC
- Benchmarks:
  - **N-Store** [Arulraj15]: persistent KV-Store benchmark

Performance comparison with Intel x86



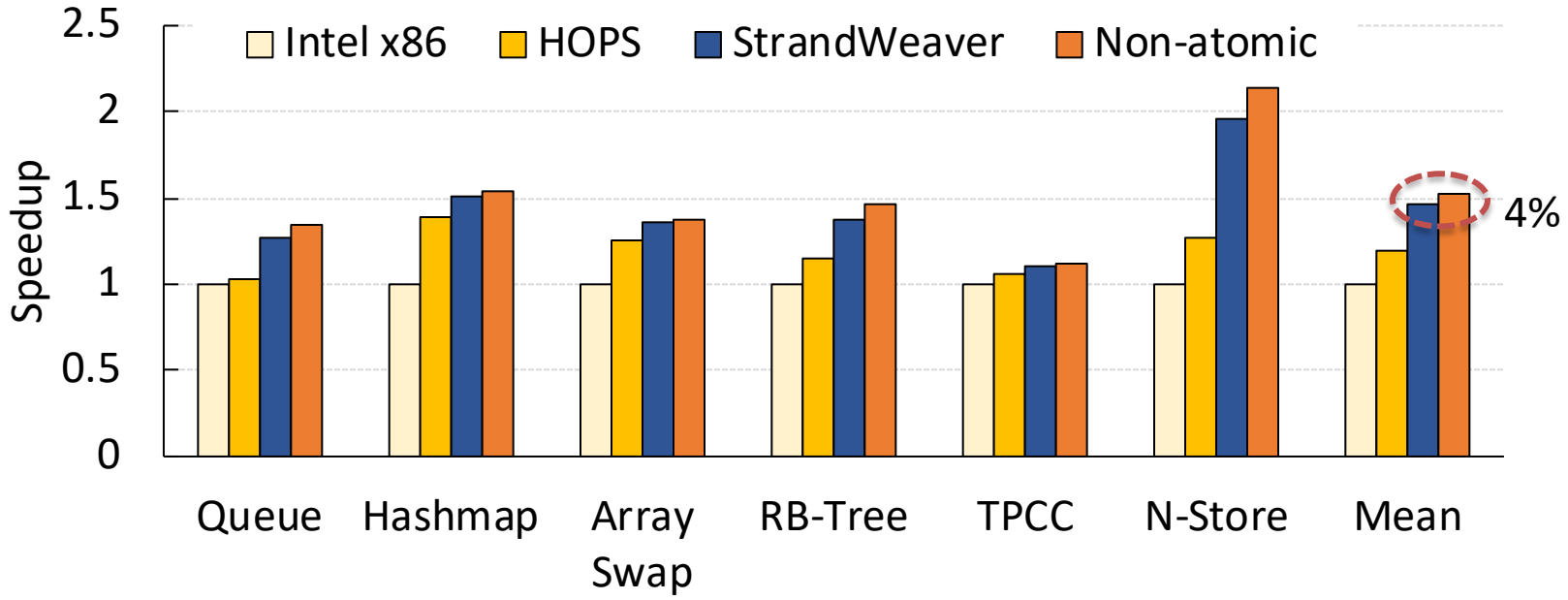
StrandWeaver achieves avg. speedup of 1.5x compared to the baseline

Performance comparison with Intel x86



StrandWeaver achieves avg. speedup of 1.2x over HOPS

Performance comparison with Intel x86



StrandWeaver performance is within 4% of non-atomic design

Conclusion

- Strand persistency to precisely order persists
- Three primitives: **PersistBarrier, NewStrand and JoinStrand**
  - Work together to relax ordering constraints in undo logging
- Evaluation using language-level persistency models
- Performance improvement of 1.45x average over Intel x86

Relaxed Persist Ordering Using Strand Persistency

Vaibhav Gogte, William Wang[$], Stephan Diestelhorst[$],
Peter M. Chen, Satish Narayanasamy, Thomas F. Wenisch

ISCA 2020

arm[$]