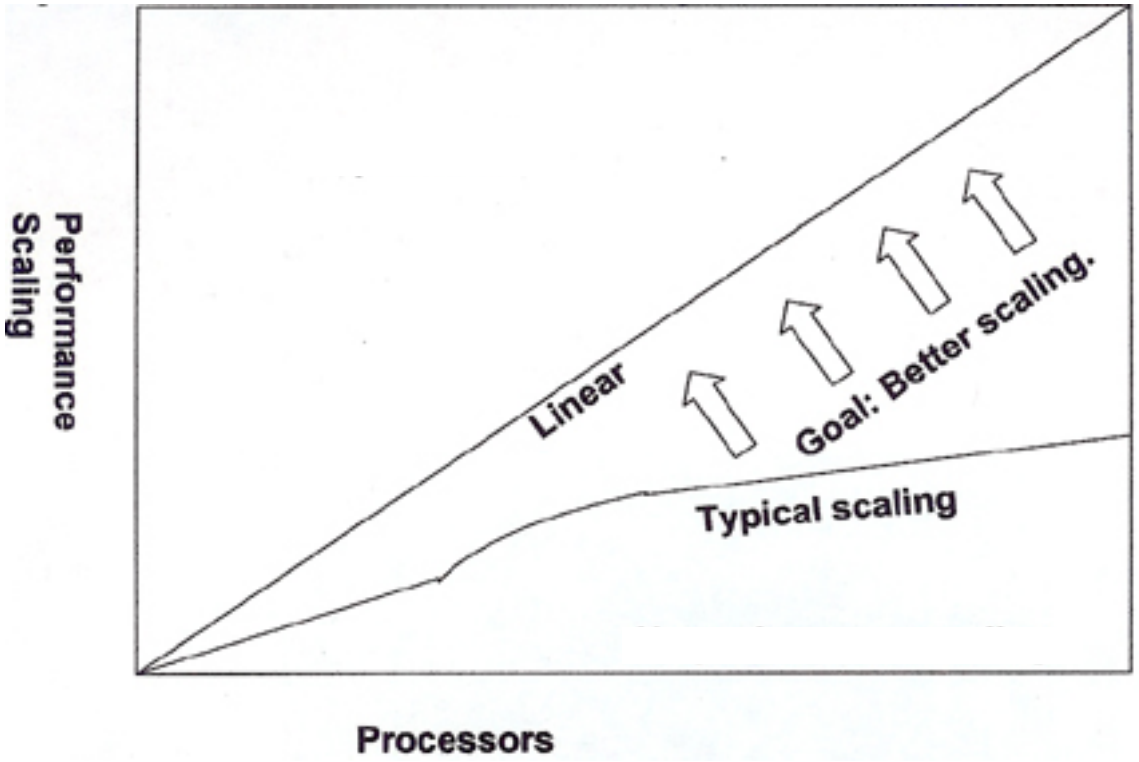
Invyswell: A HyTM for Haswell RTM

Irina Calciu, Justin Gottschlich, Tatiana Shpeisman,
Gilles Pokam, Maurice Herlihy

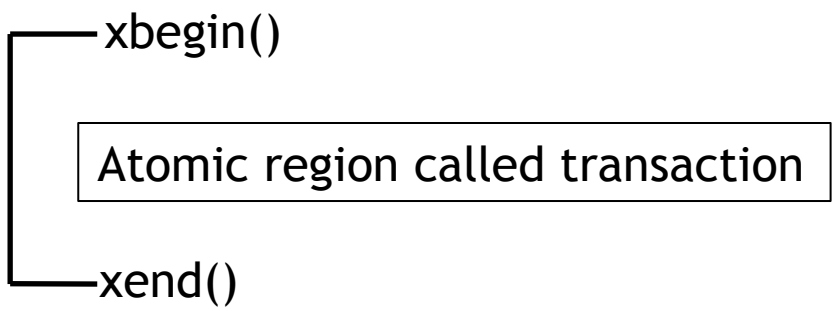
Multicore Performance Scaling



- Problem: Locking
- Solution: HTM?
  - IBM BG/Q, zEC12, POWER
  - Intel Haswell TSX

Source: embedded.com

Restricted Transactional Memory (RTM)



Execute optimistically, without any locks

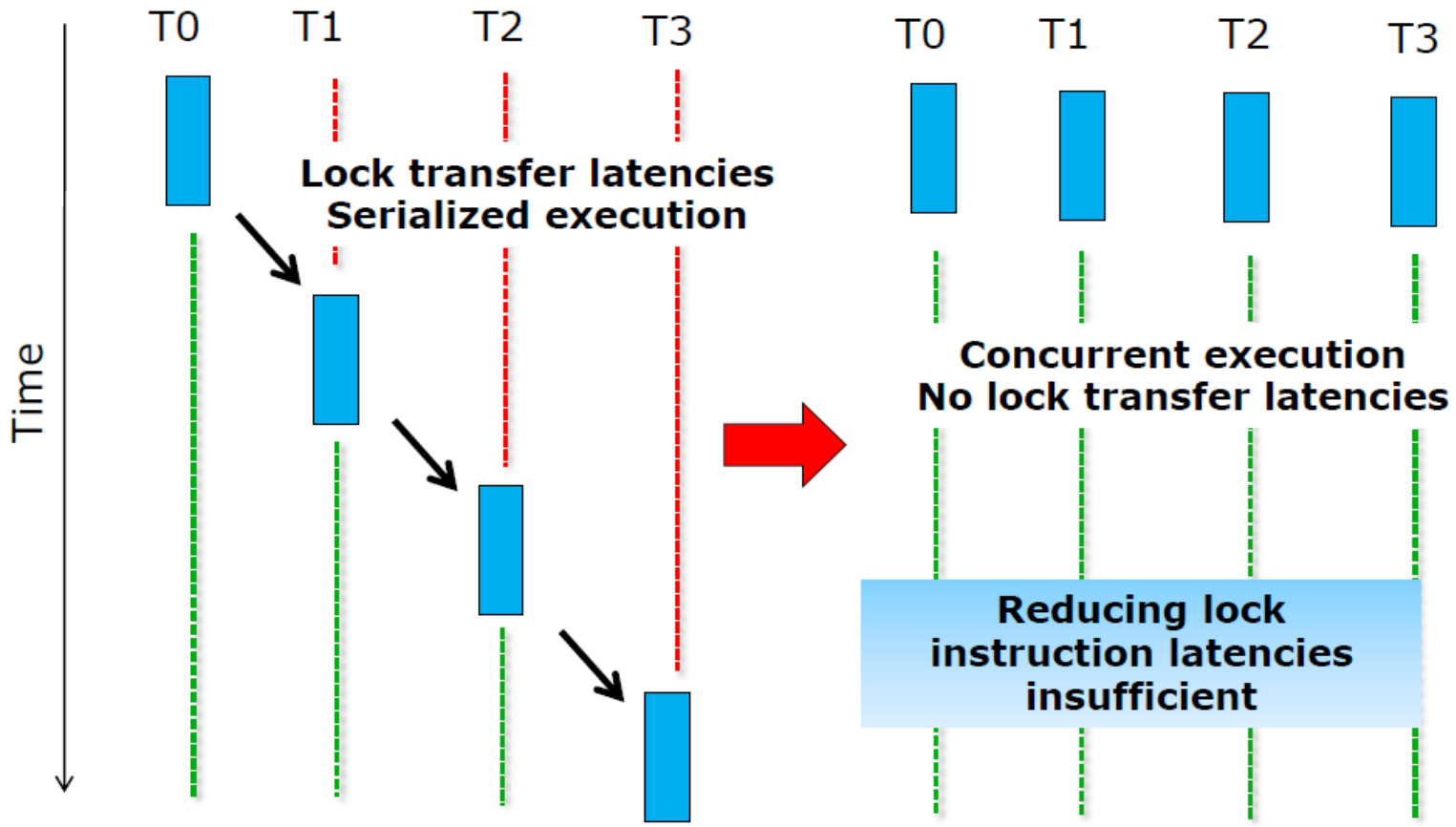
Read and Write Sets

Abort on memory conflict: programmer defined behavior

RTM Fallback: Global Lock

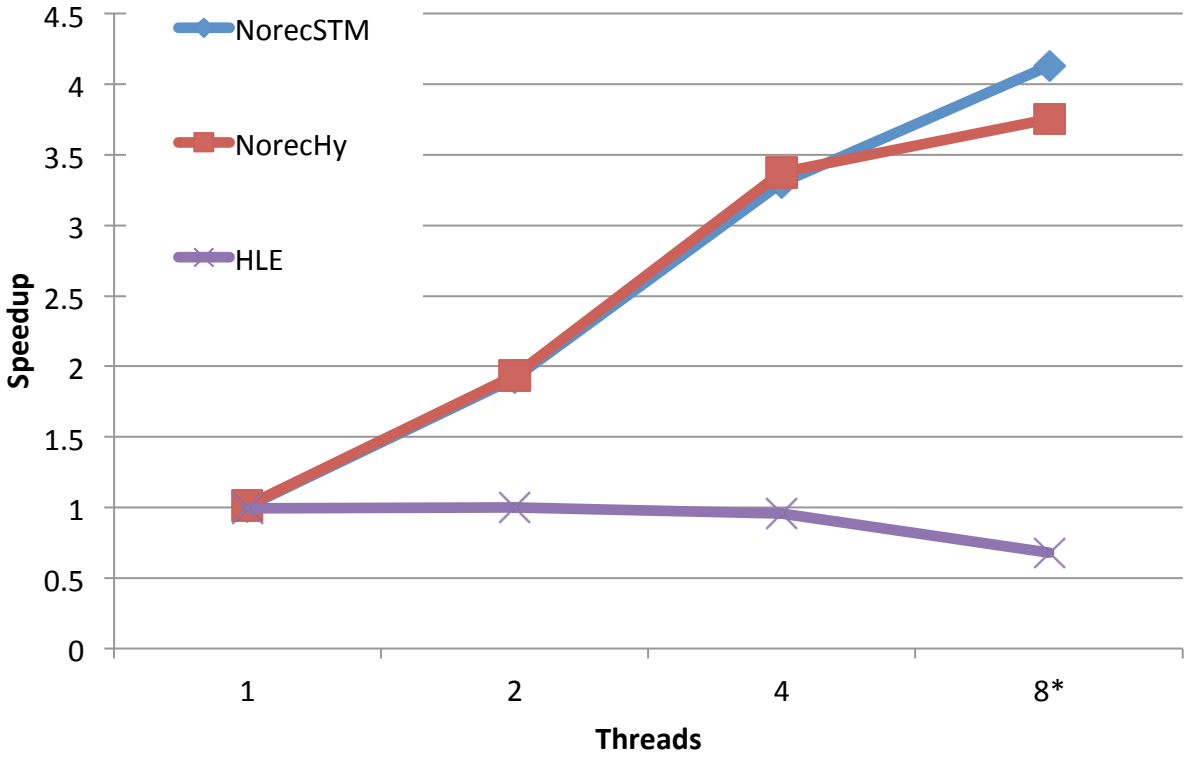
```
if (xbegin() == XBEGIN_STARTED)
        Execute Transaction
        xend()
else
        Execute Fallback Path
```

Lock Elision



Source: Anand Tech

Why Lock Elision Is Not Enough

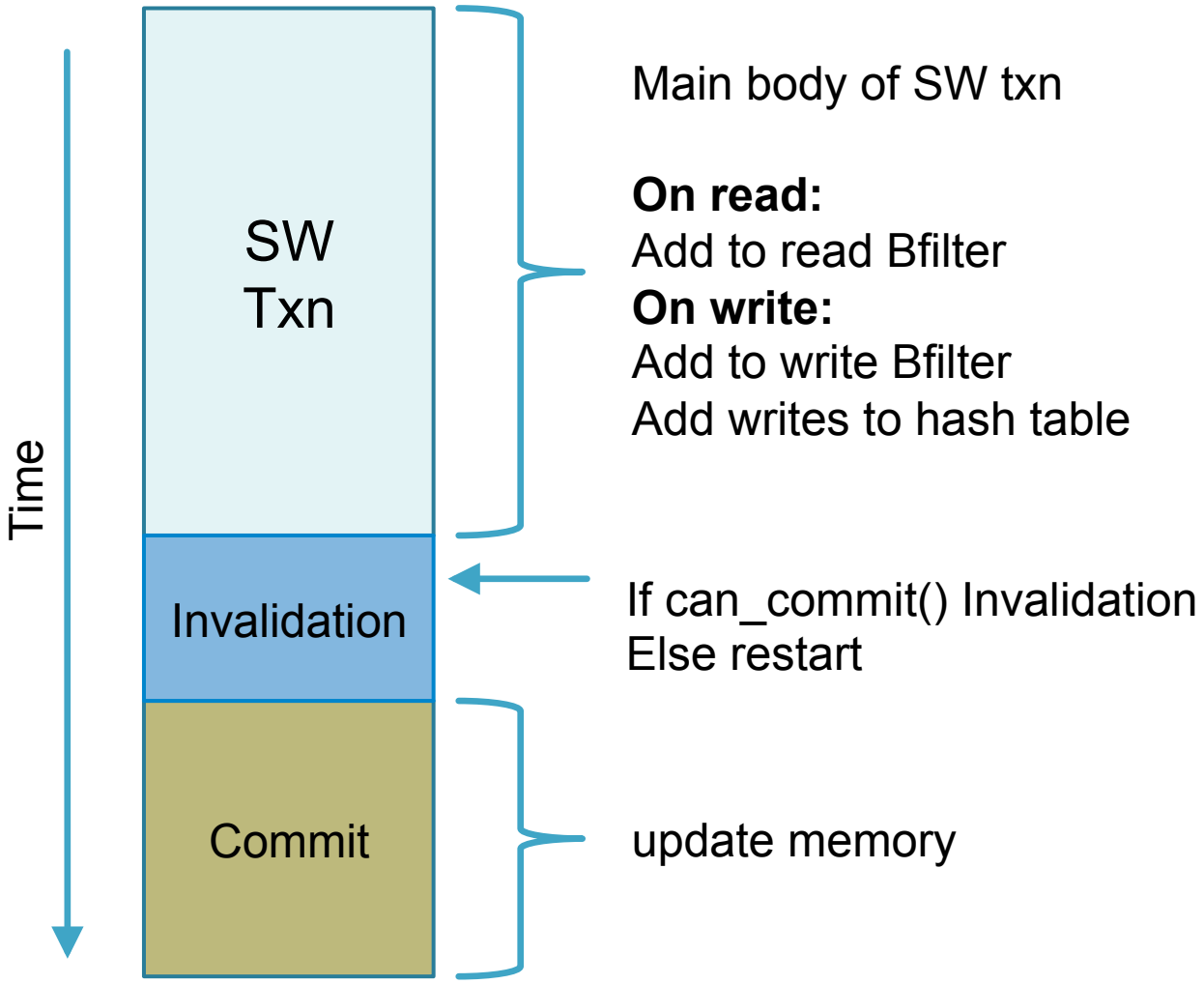


Labyrinth

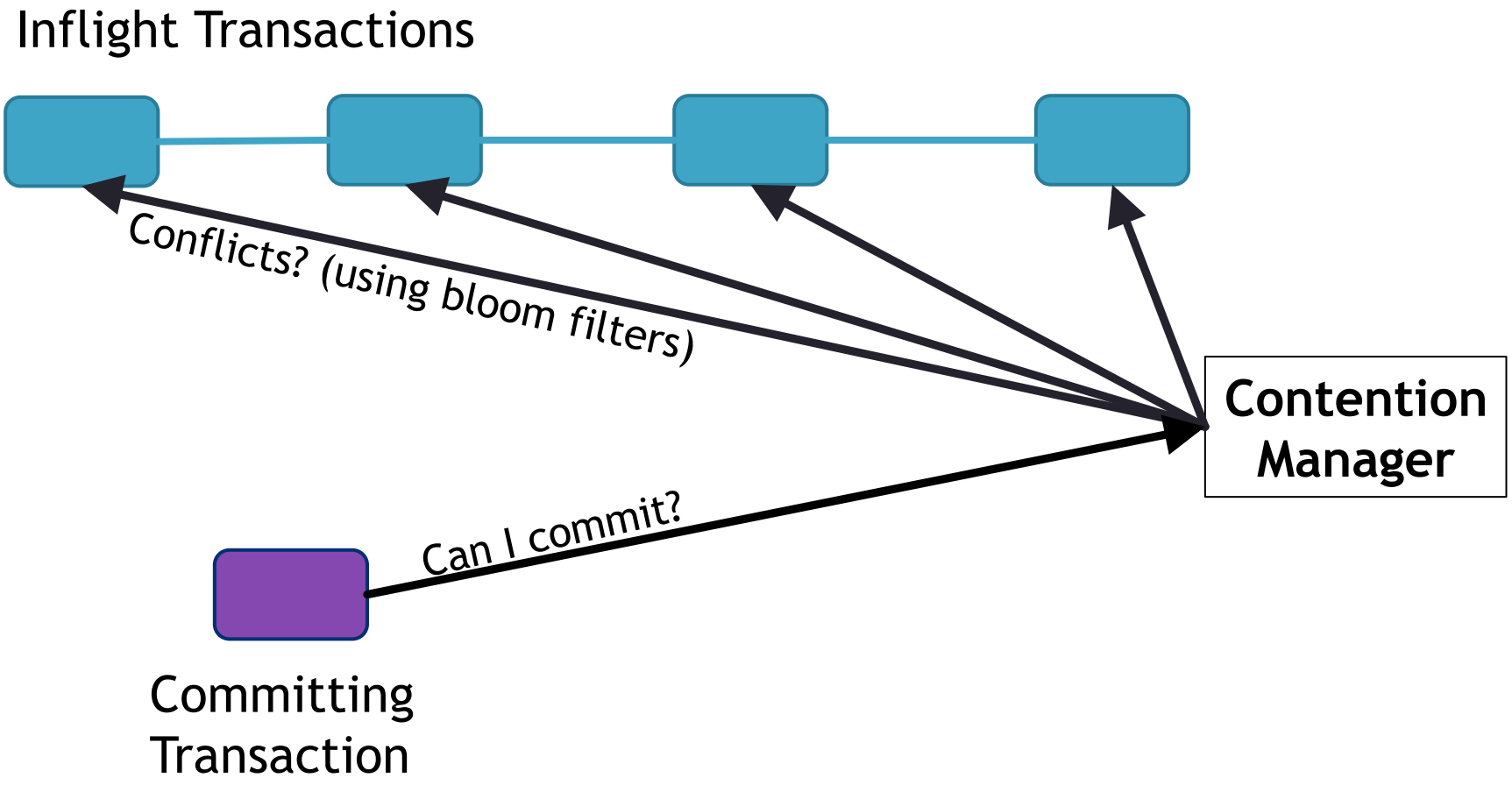
InvalSTM (prior work)

- [Gottschlich et al., CGO 2010]
- Scalable
- Good for large transactions
- Conflict detection using bloom filters

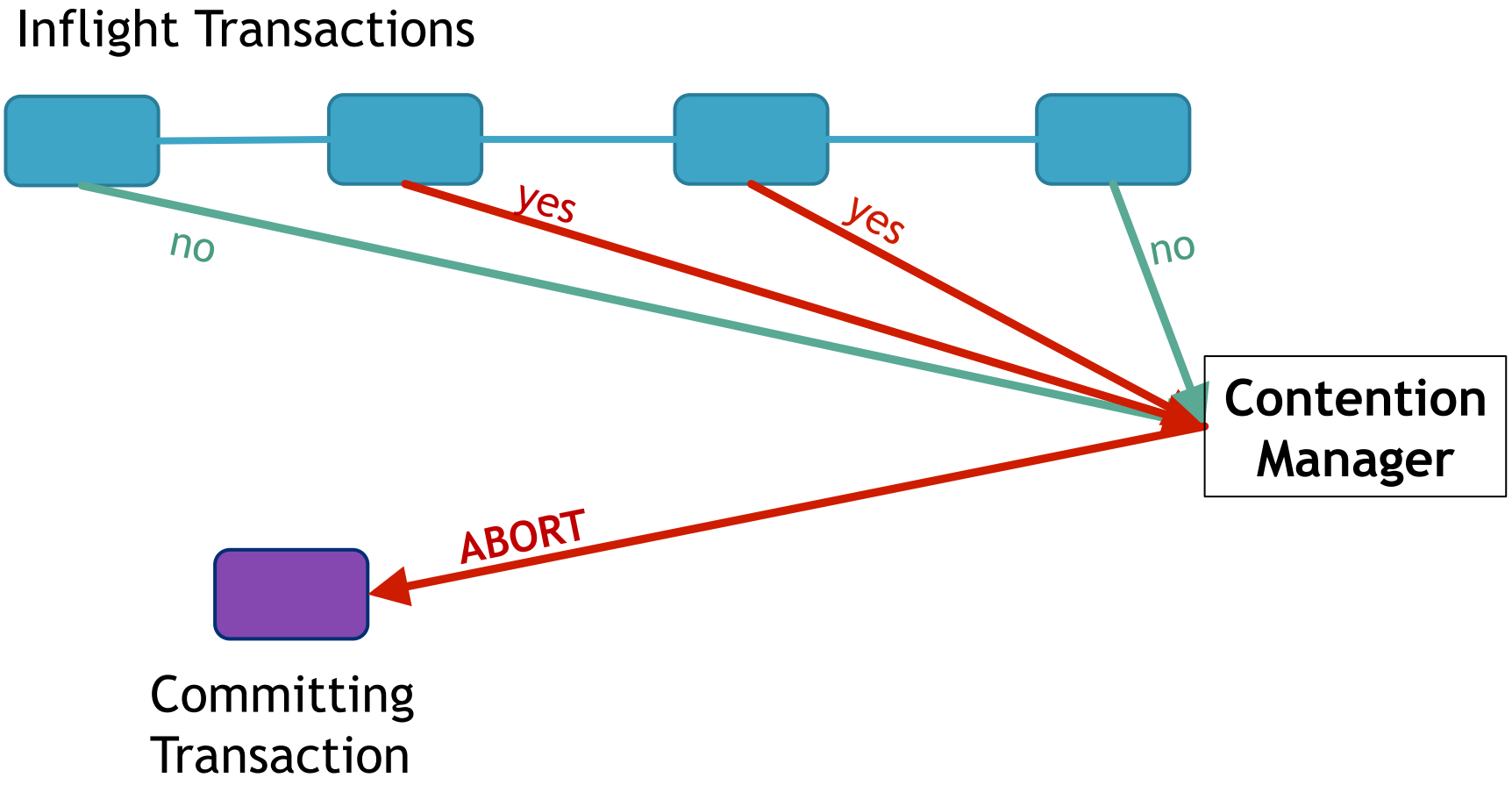
InvalSTM Software Transaction (prior work)



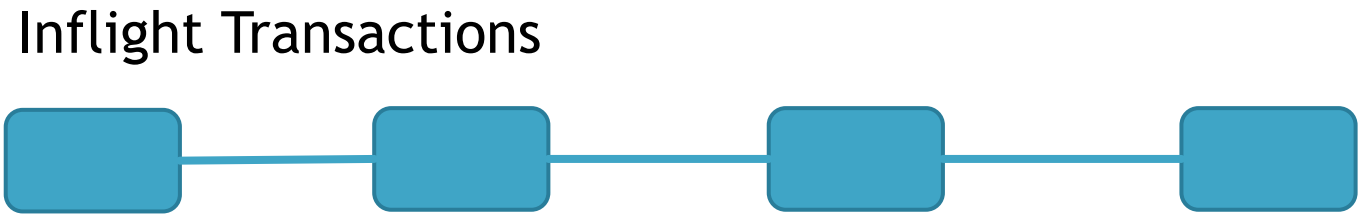
InvalSTM Invalidation (prior work)



InvalSTM Invalidation (prior work)



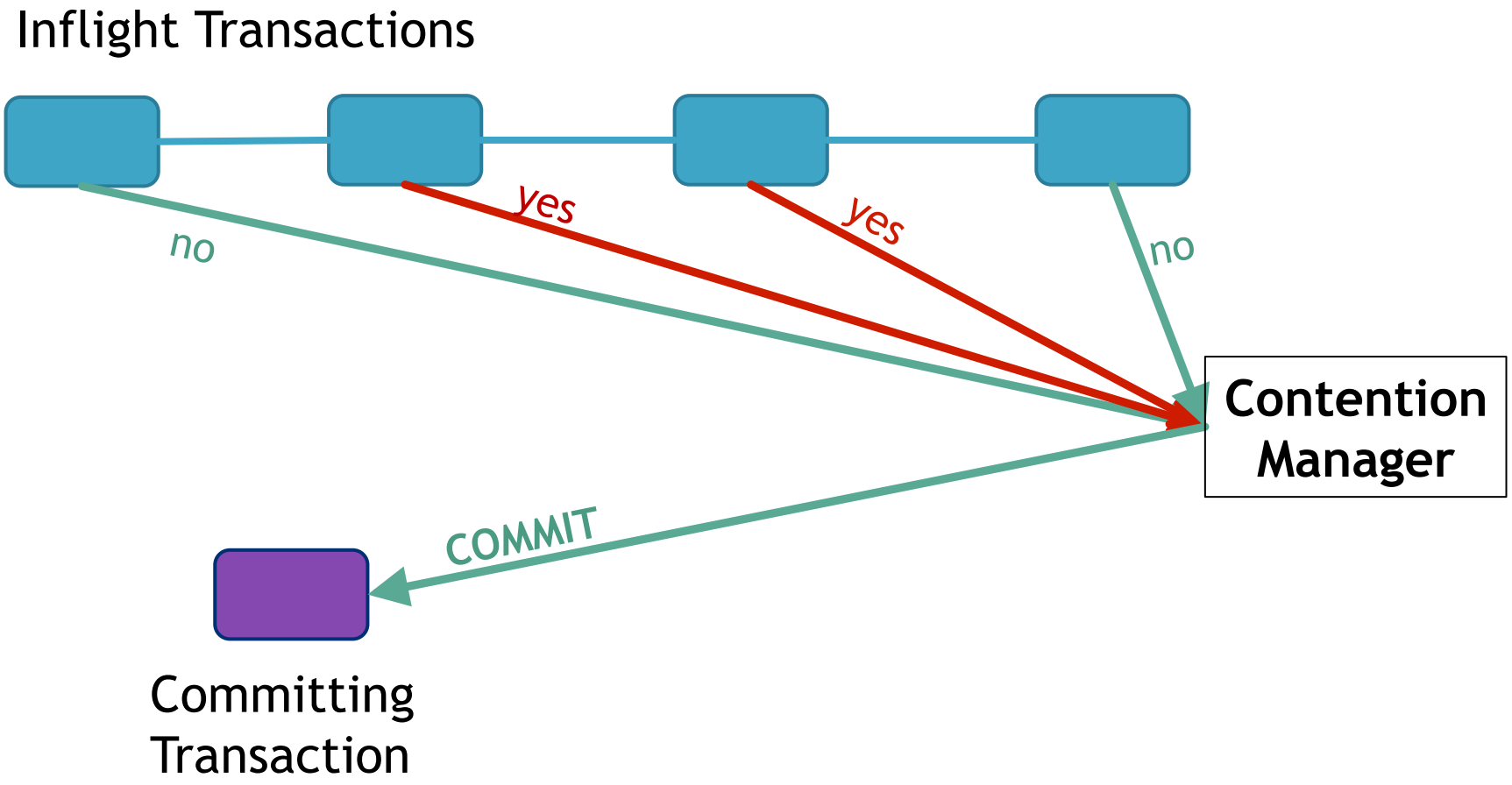
InvalSTM Invalidation (prior work)



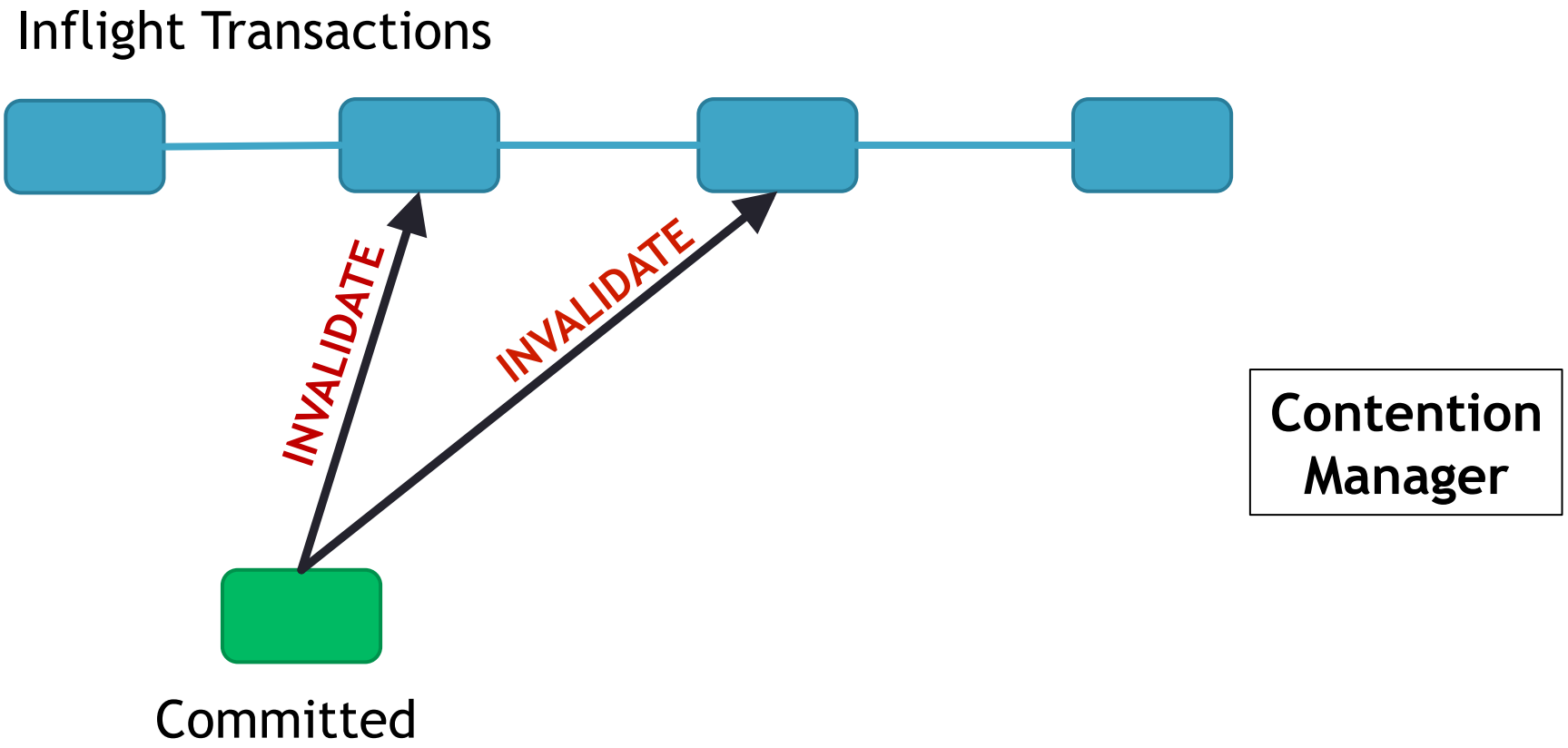
Contention Manager

Aborted

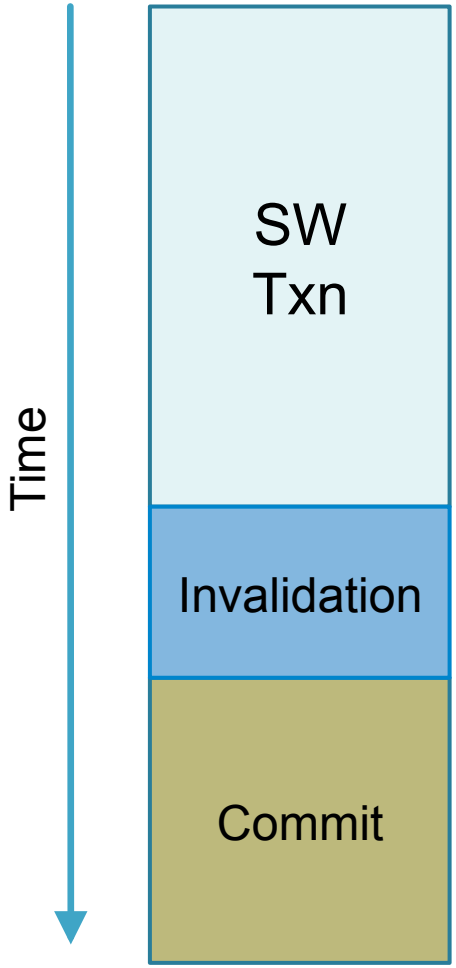
InvalSTM Invalidation (prior work)



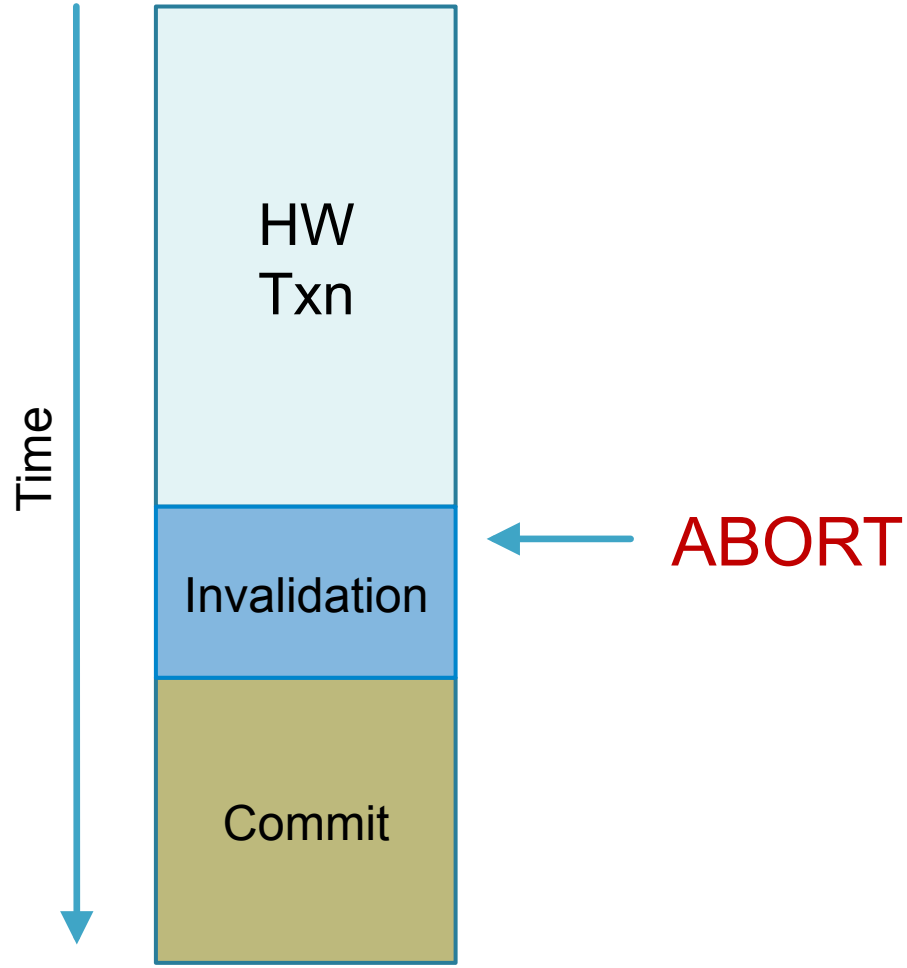
InvalSTM Invalidation (prior work)



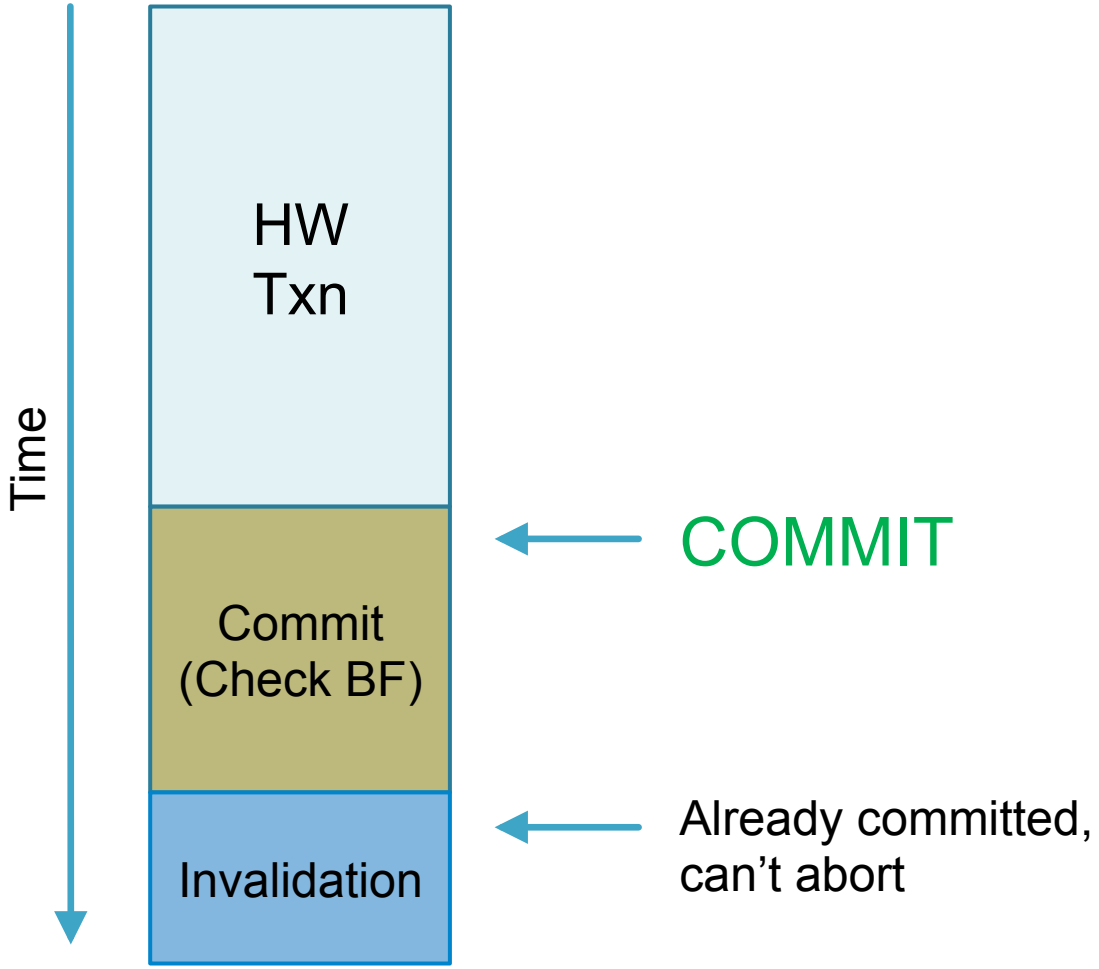
Software Transaction (InvalSTM)



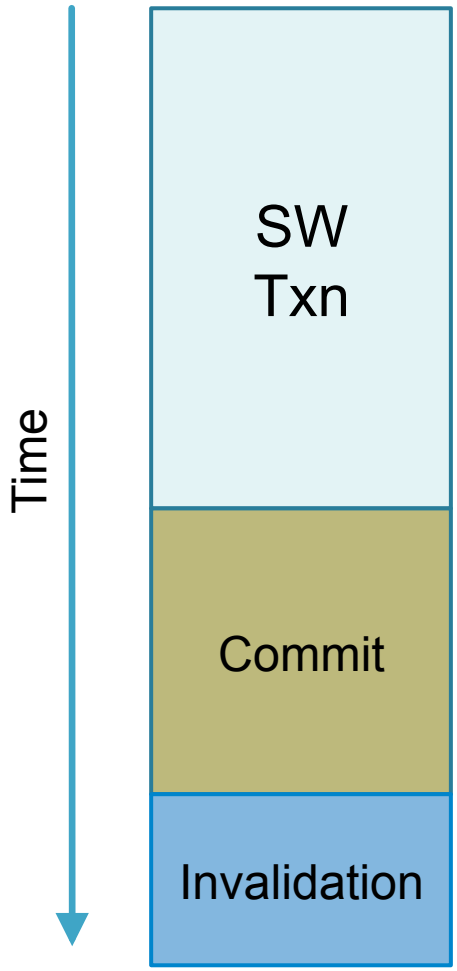
Hardware Transaction + Invalidation

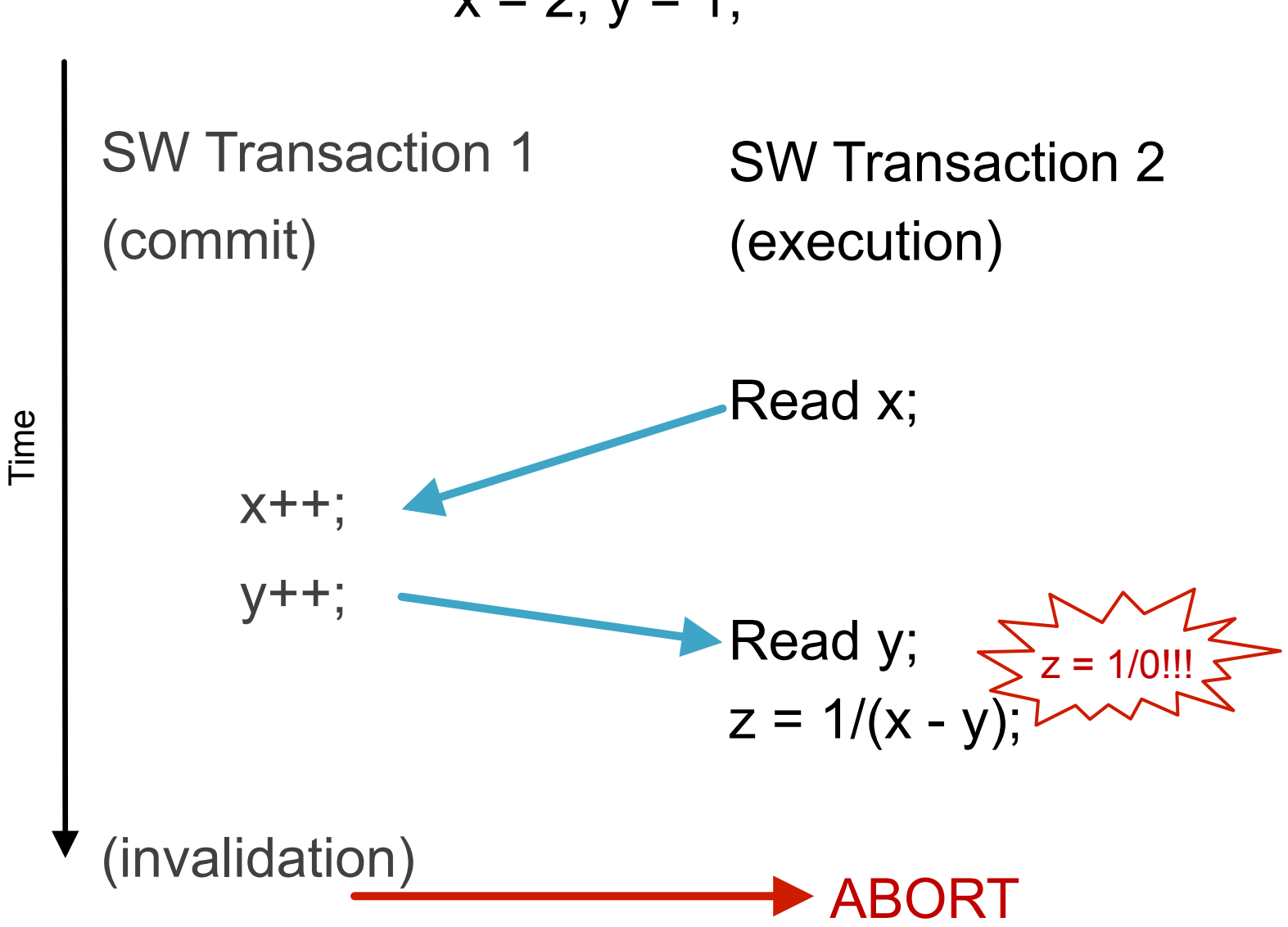


Hardware Transaction + Invalidation

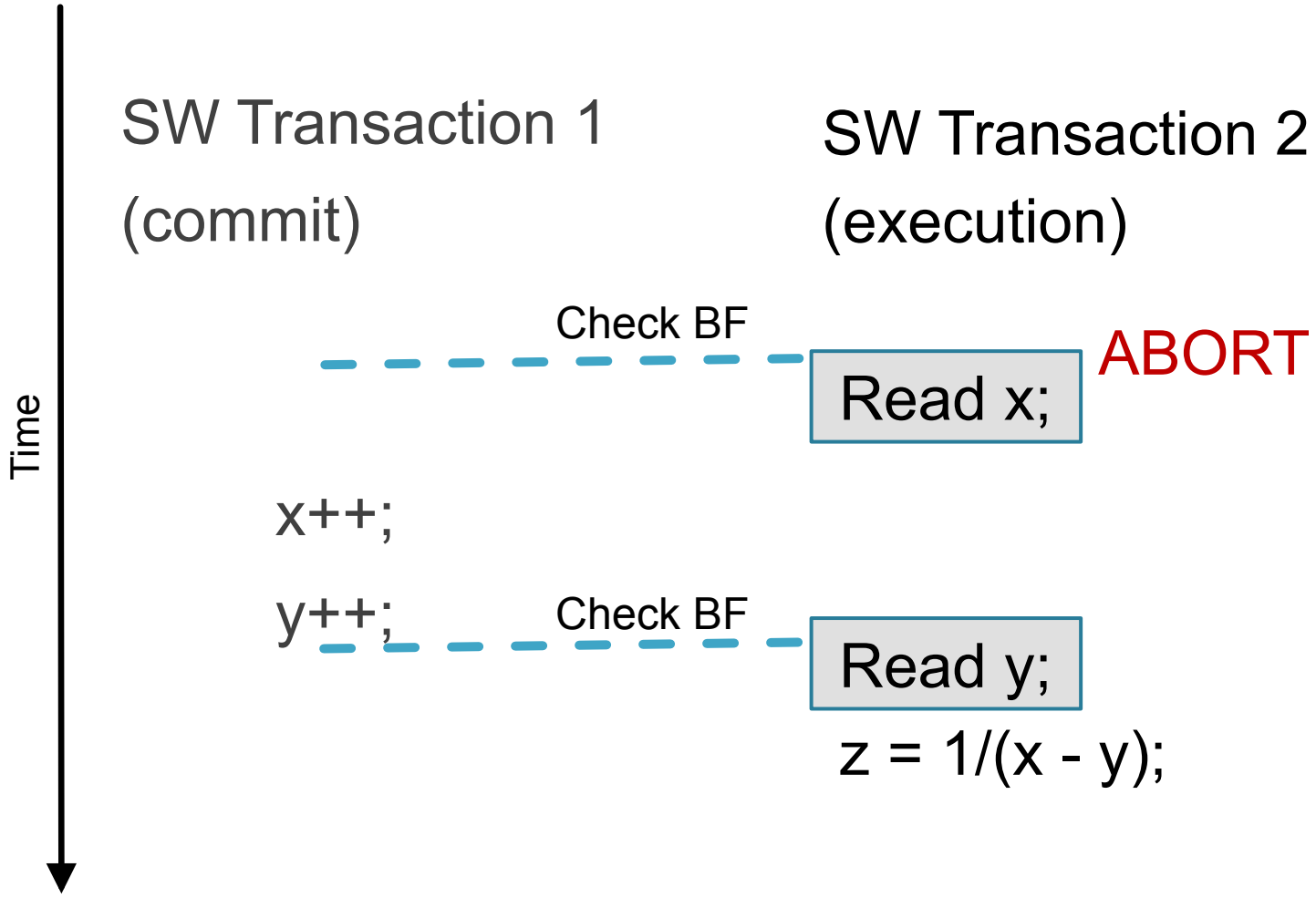


Software Transaction (Modified InvalSTM)

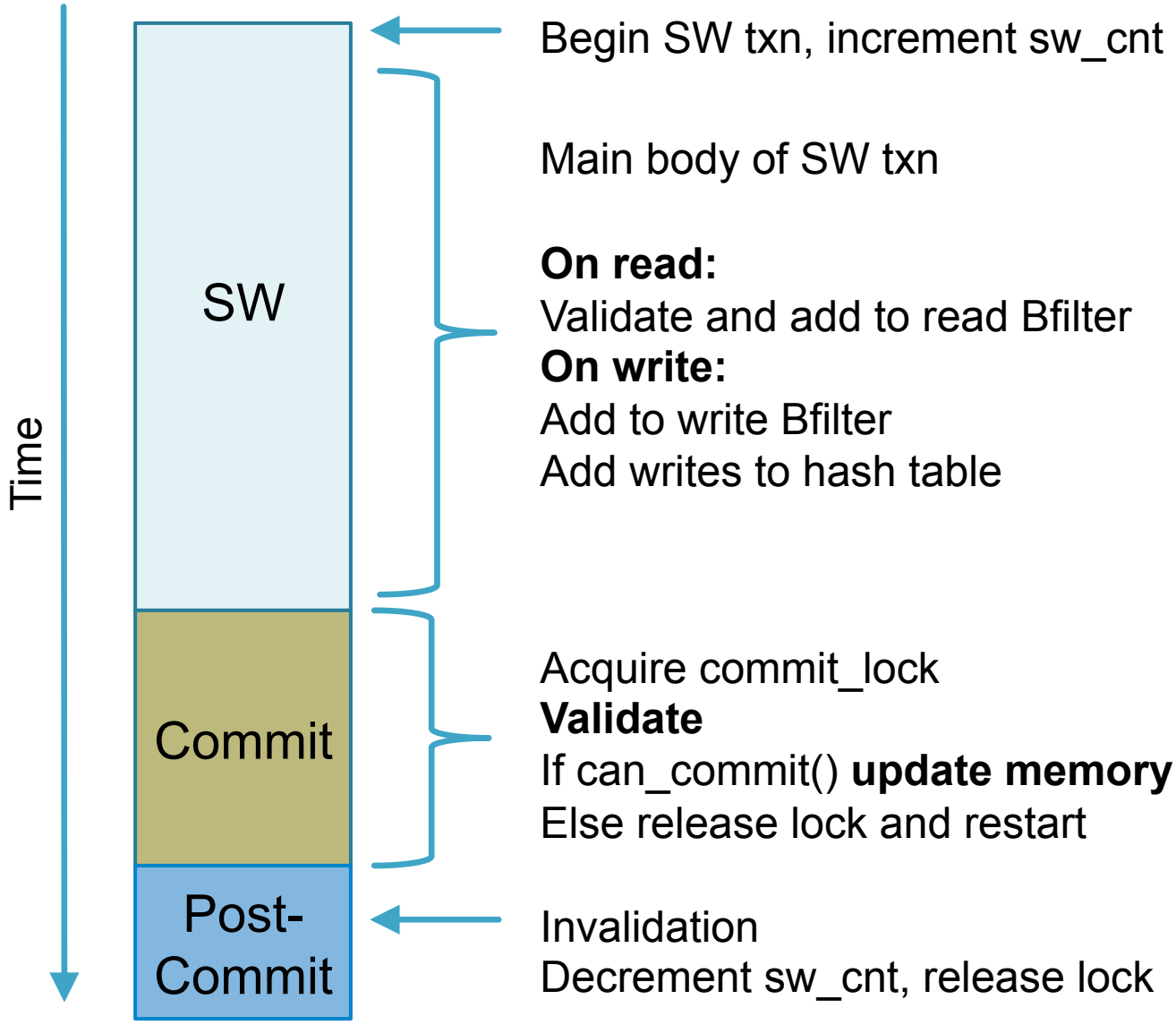



x = 2; y = 1;
Time
SW Transaction 1
(commit)
SW Transaction 2
(execution)
Read x;
x++;
y++;
Read y;
z = 1/(x - y);
z = 1/0!!!
(invalidation)
ABORT

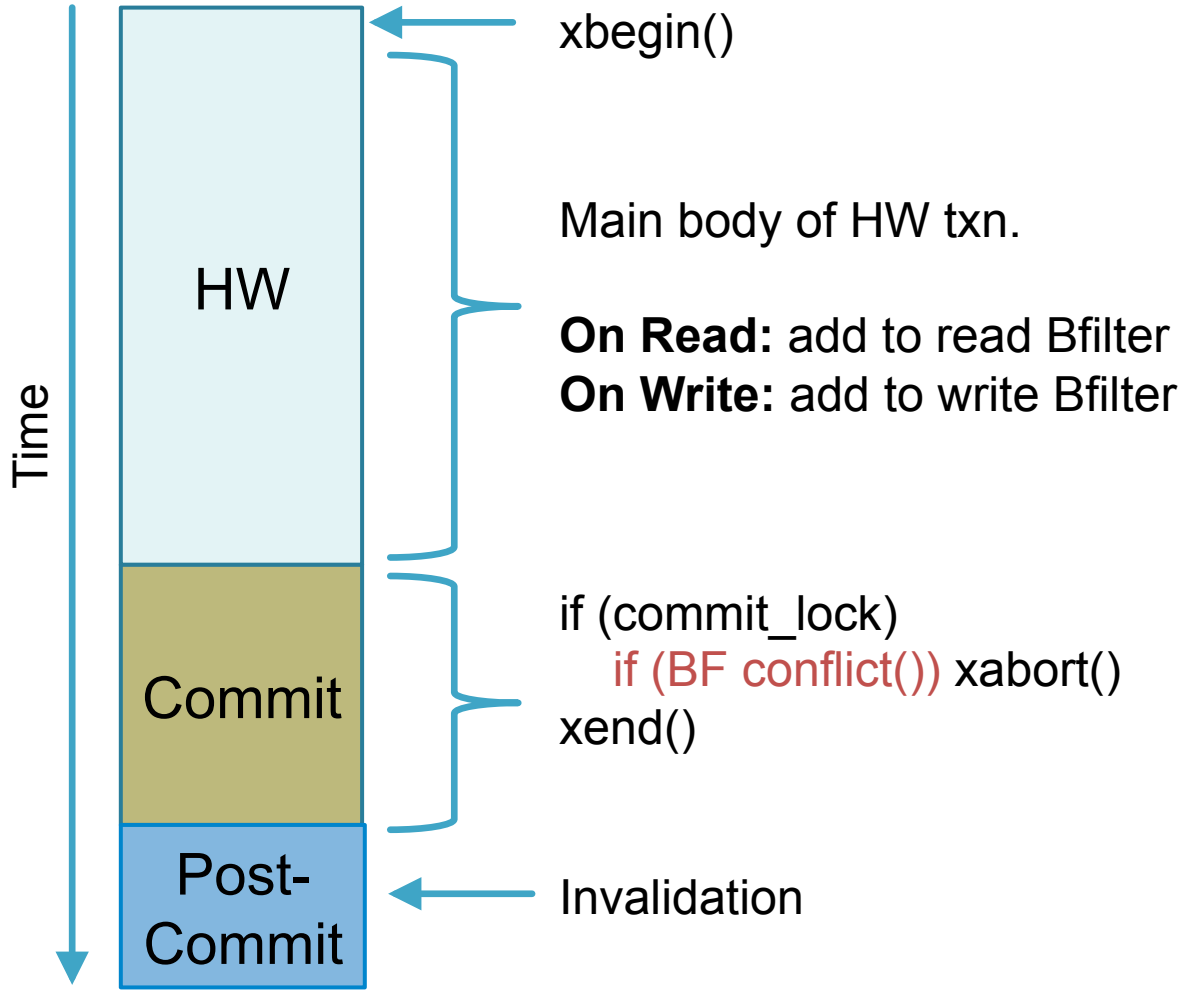
Read Validation

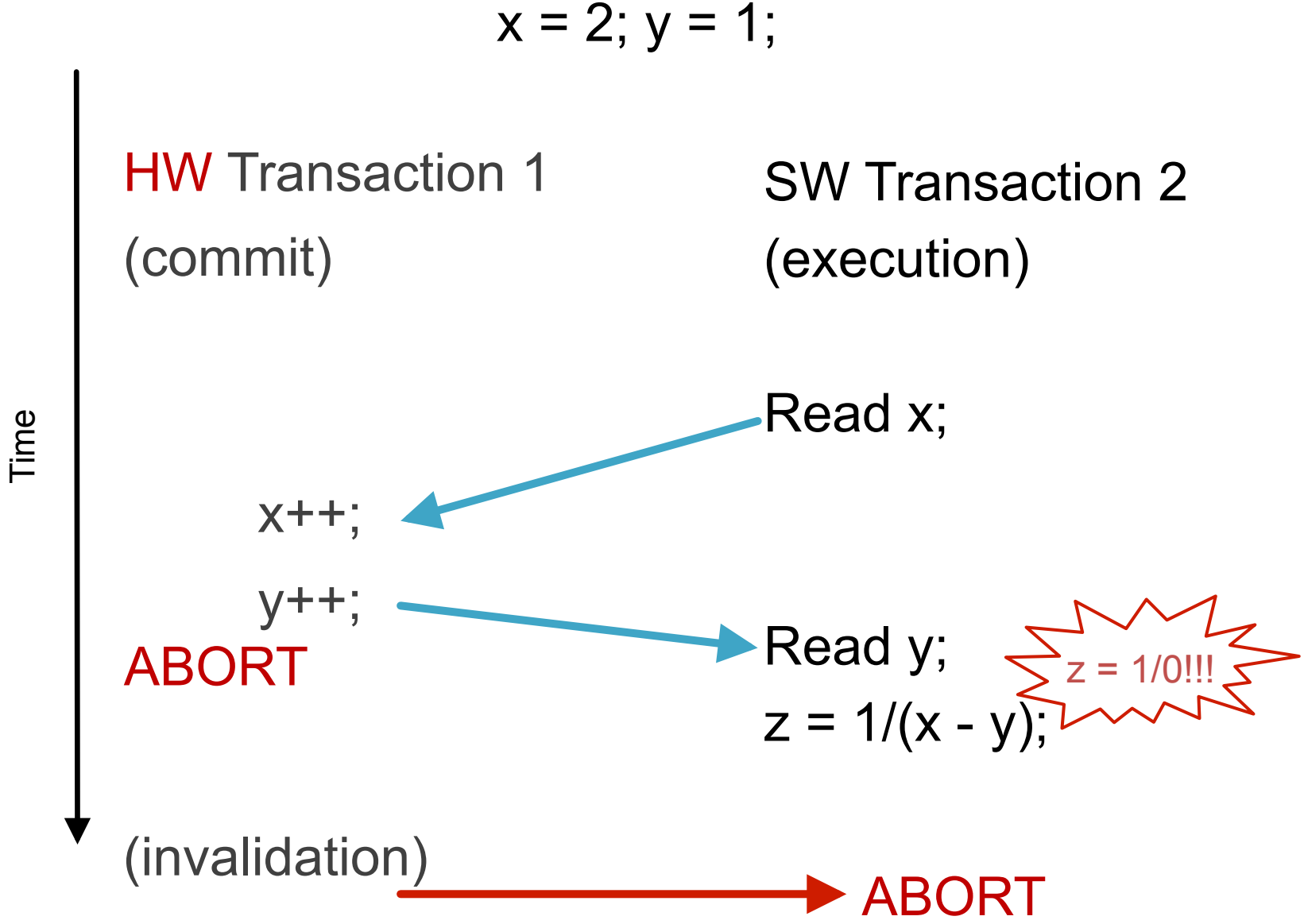


SPECSW (Speculative Software)

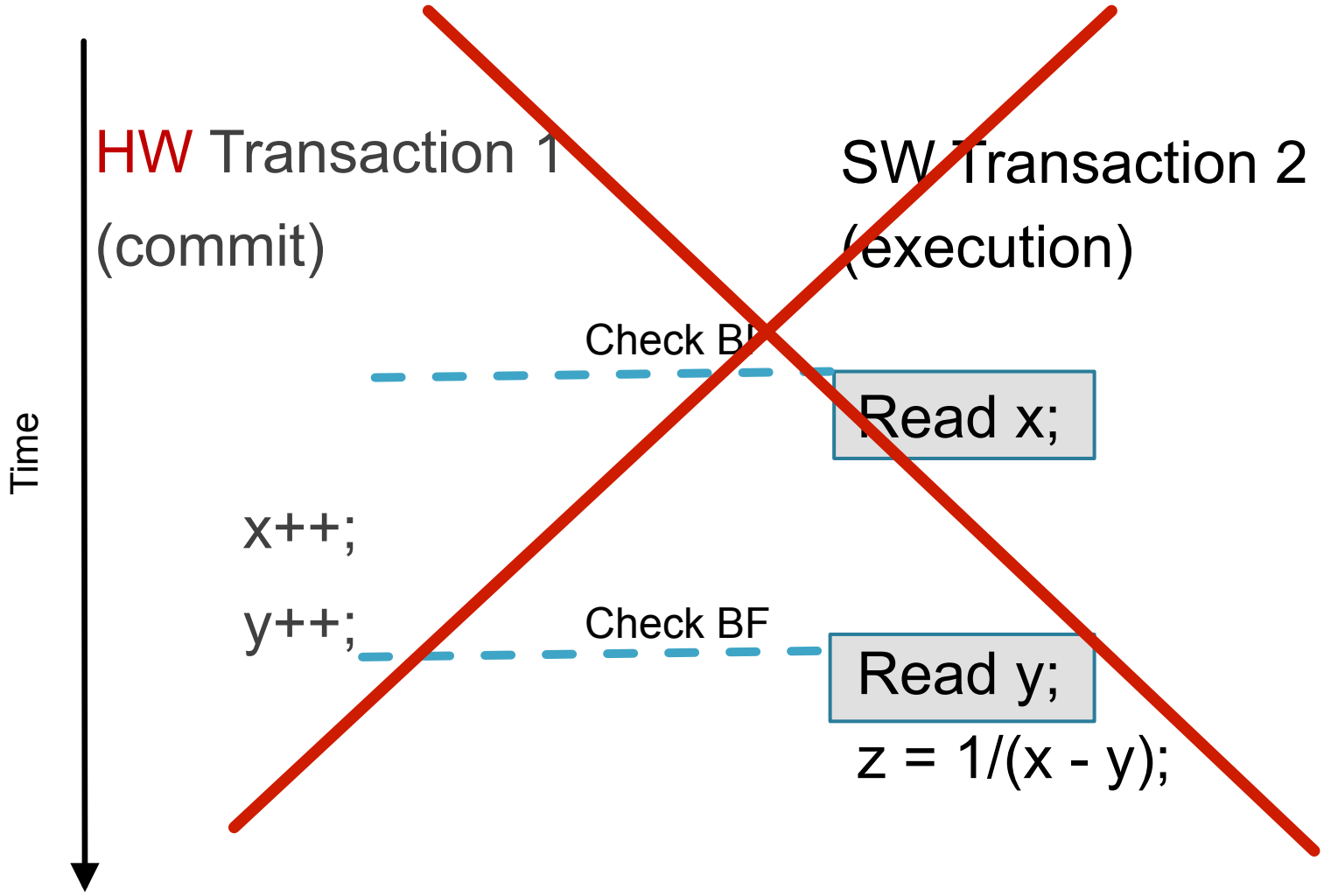


BFHW (Bloom Filters Hardware)

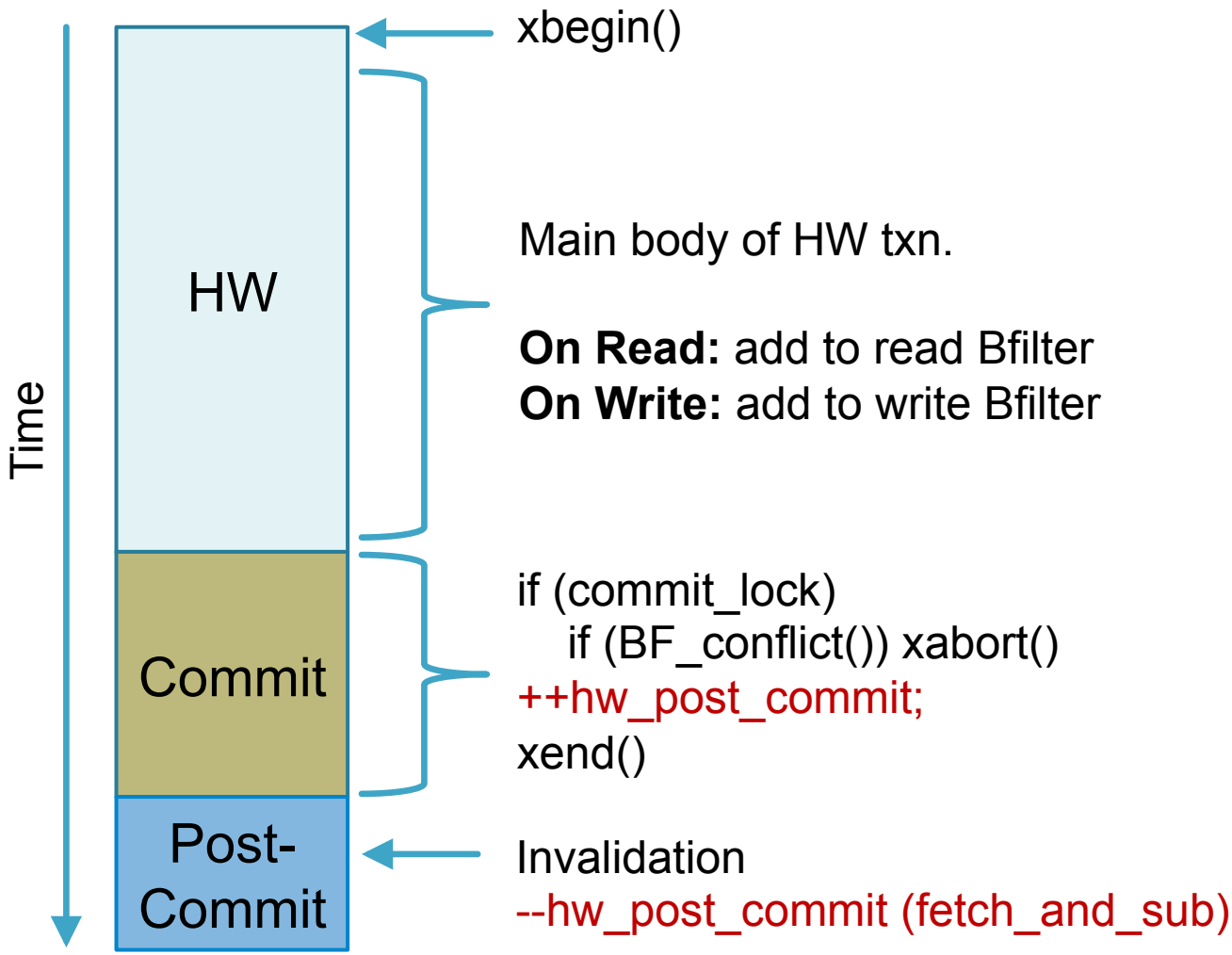



x = 2; y = 1;
Time
HW Transaction 1
(commit)
SW Transaction 2
(execution)
Read x;
x++;
y++;
ABORT
Read y;
z = 1/(x - y);
z = 1/0!!!
(invalidation)
ABORT

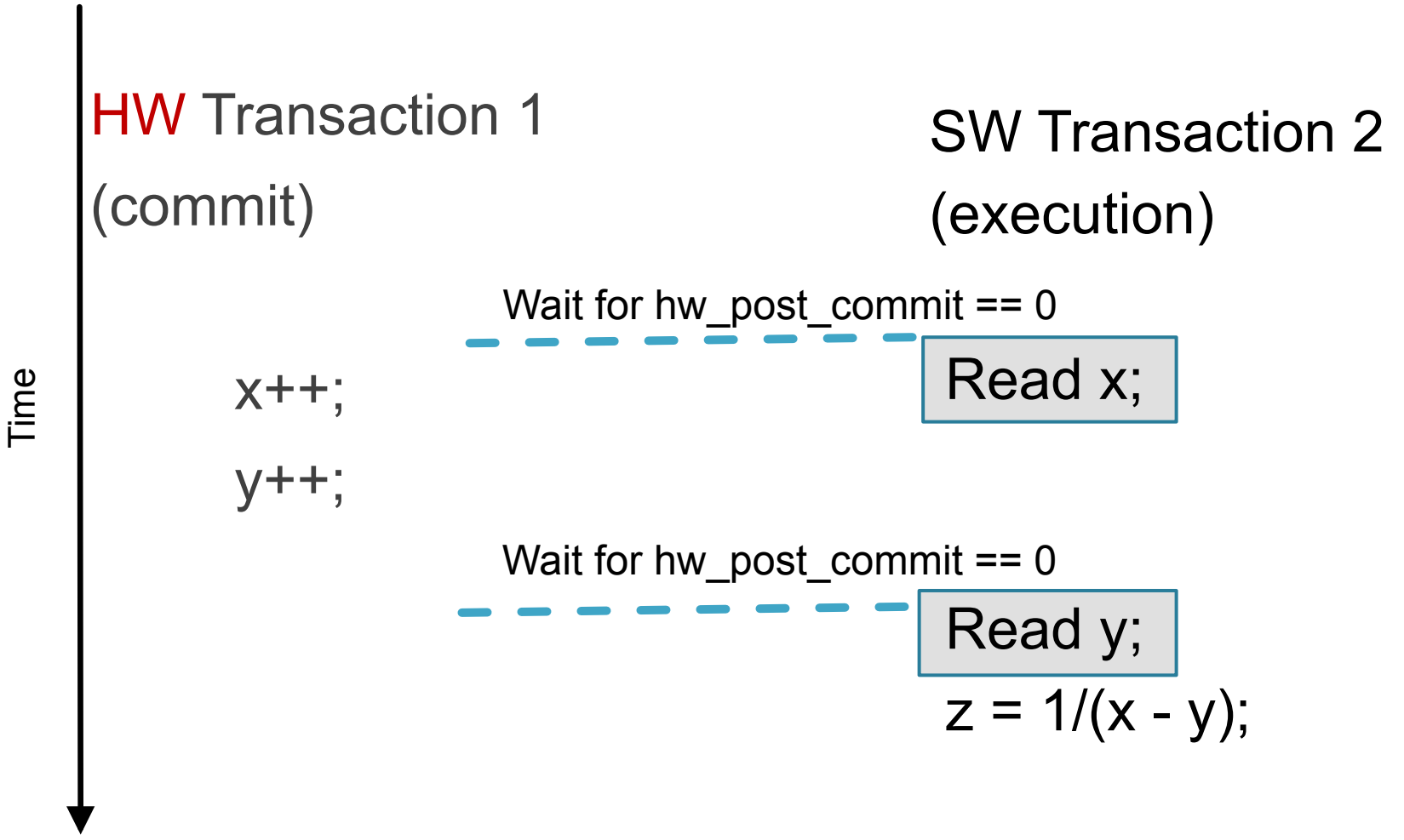
Read Validation

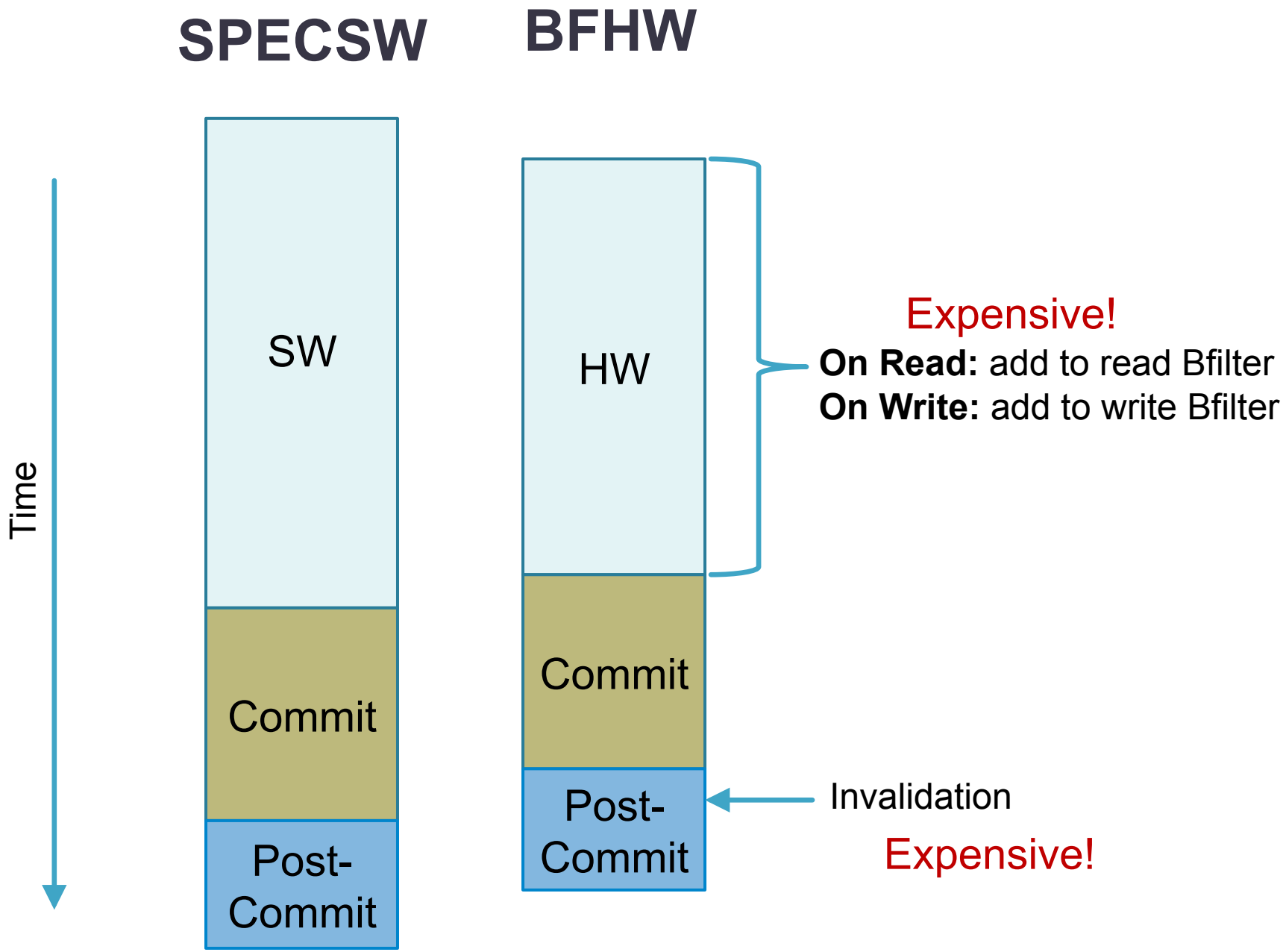


BFHW

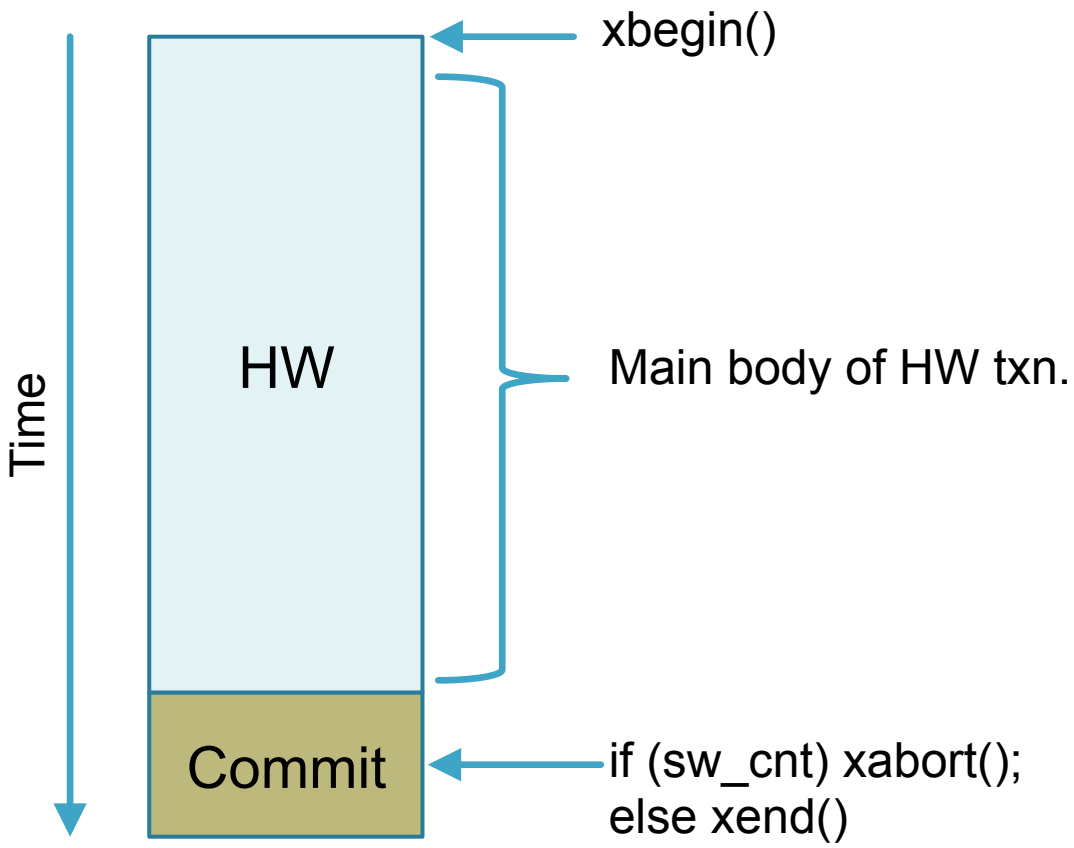

Time
HW
Commit
Post-
Commit
xbegin()
Main body of HW txn.
On Read: add to read Bfilter
On Write: add to write Bfilter
if (commit_lock)
if (BF_conflict()) xabort()
++hw_post_commit;
xend()
Invalidation
--hw_post_commit (fetch_and_sub)

Read Validation

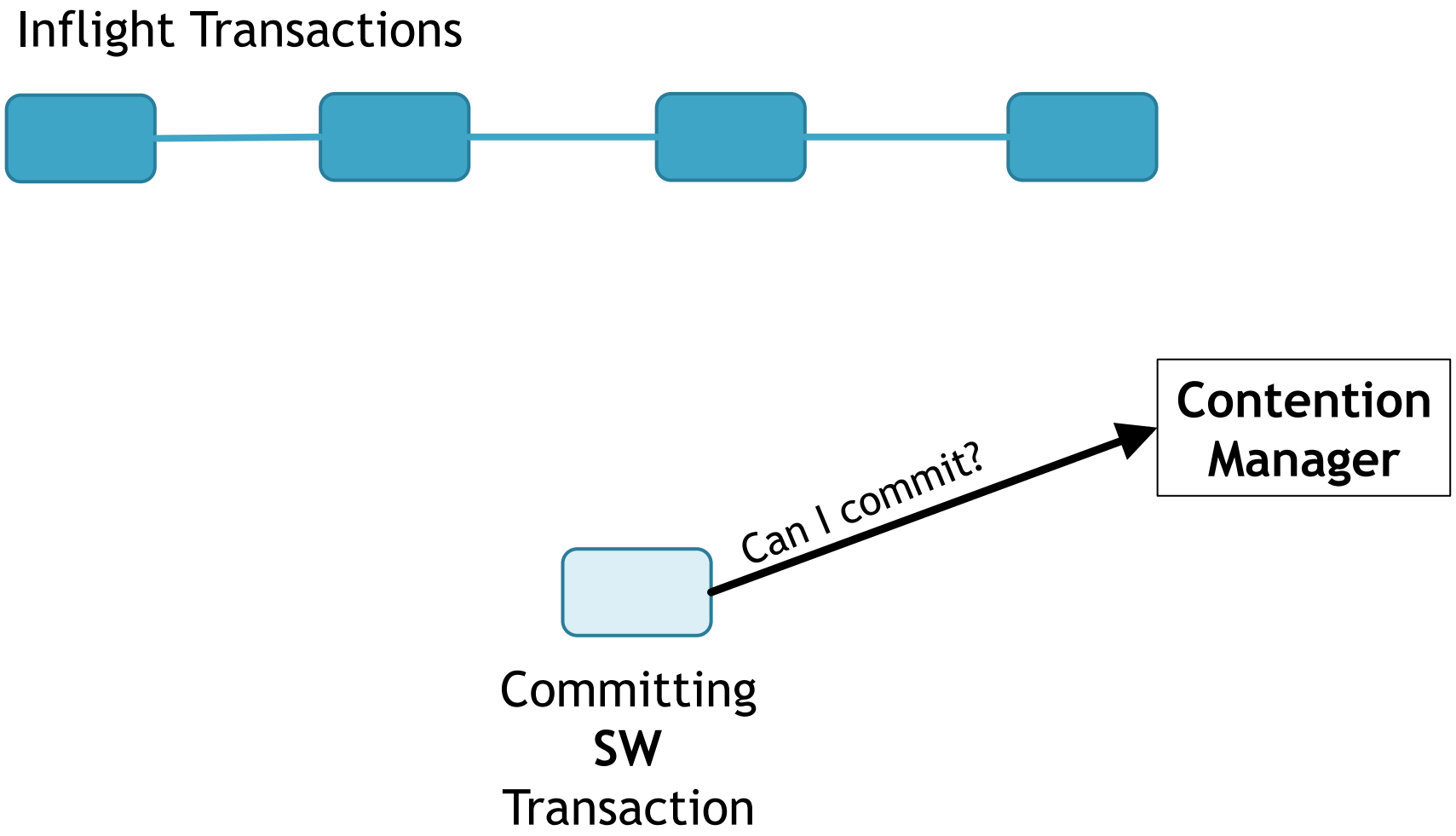



SPECSW
BFHW
Time
SW
Commit
Post-
Commit
HW
Commit
Post-
Commit
Expensive!
On Read: add to read Bfilter
On Write: add to write Bfilter
Invalidation
Expensive!

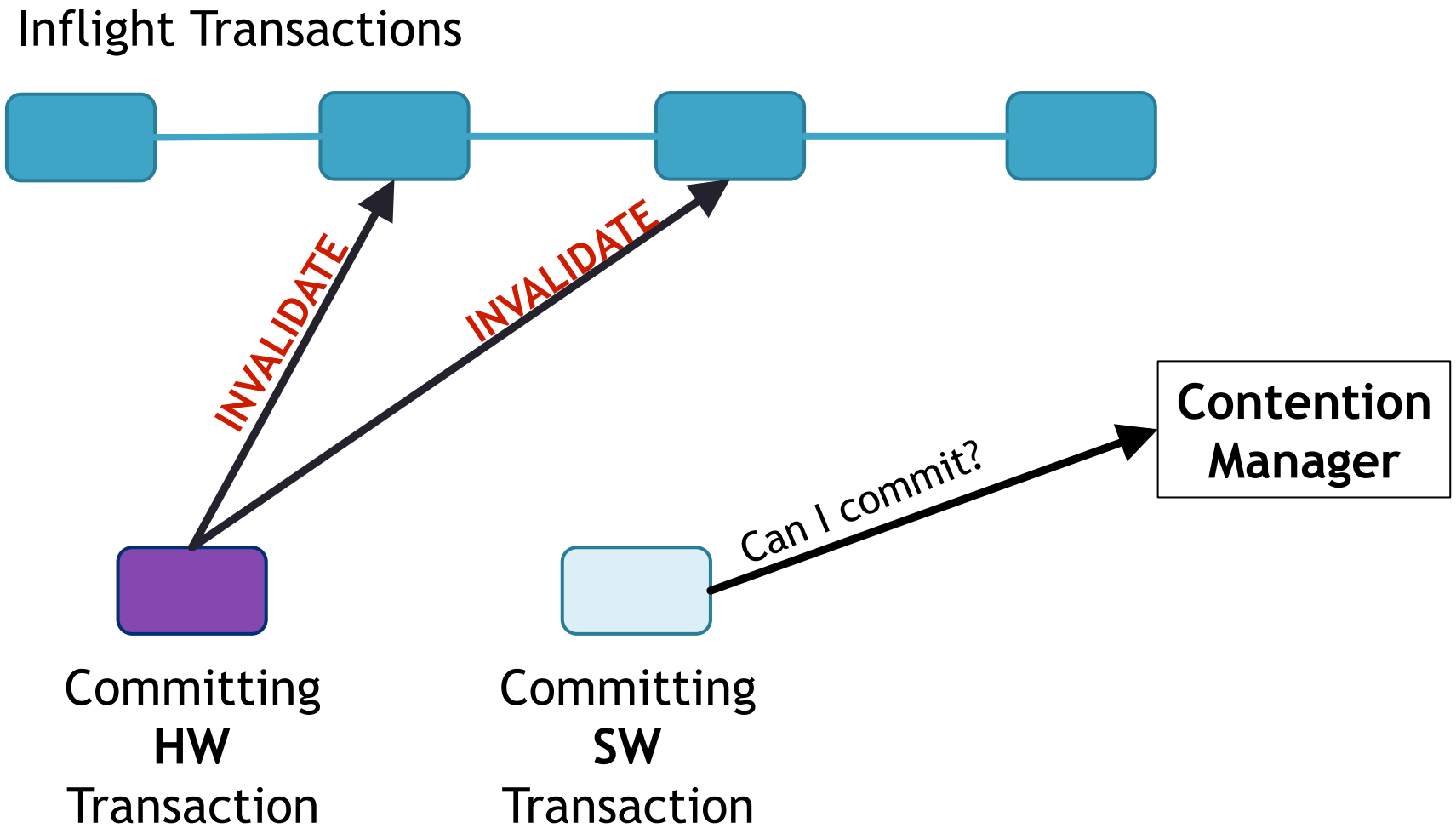
LITEHW (Light Hardware)



Ensuring Progress

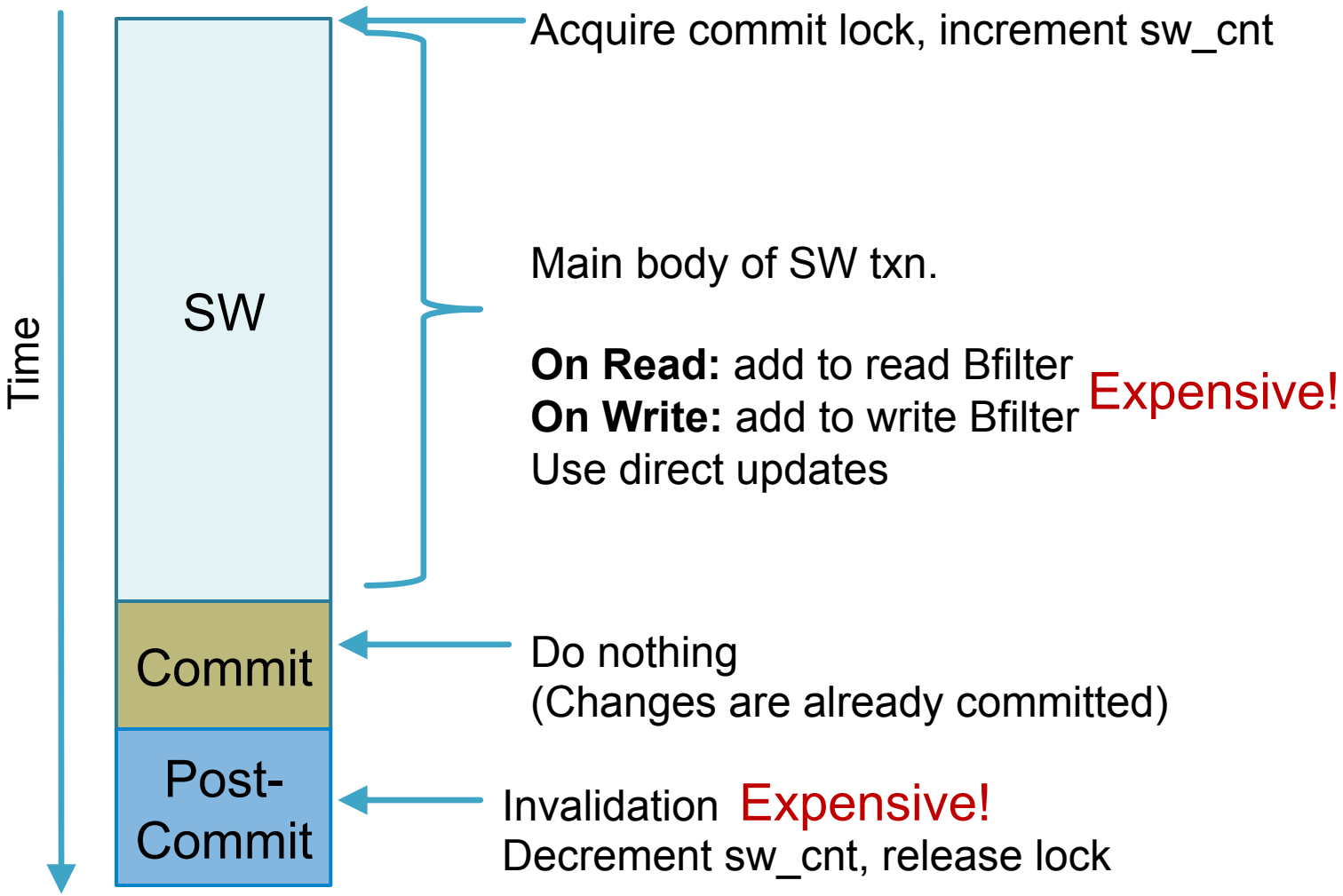


Ensuring Progress

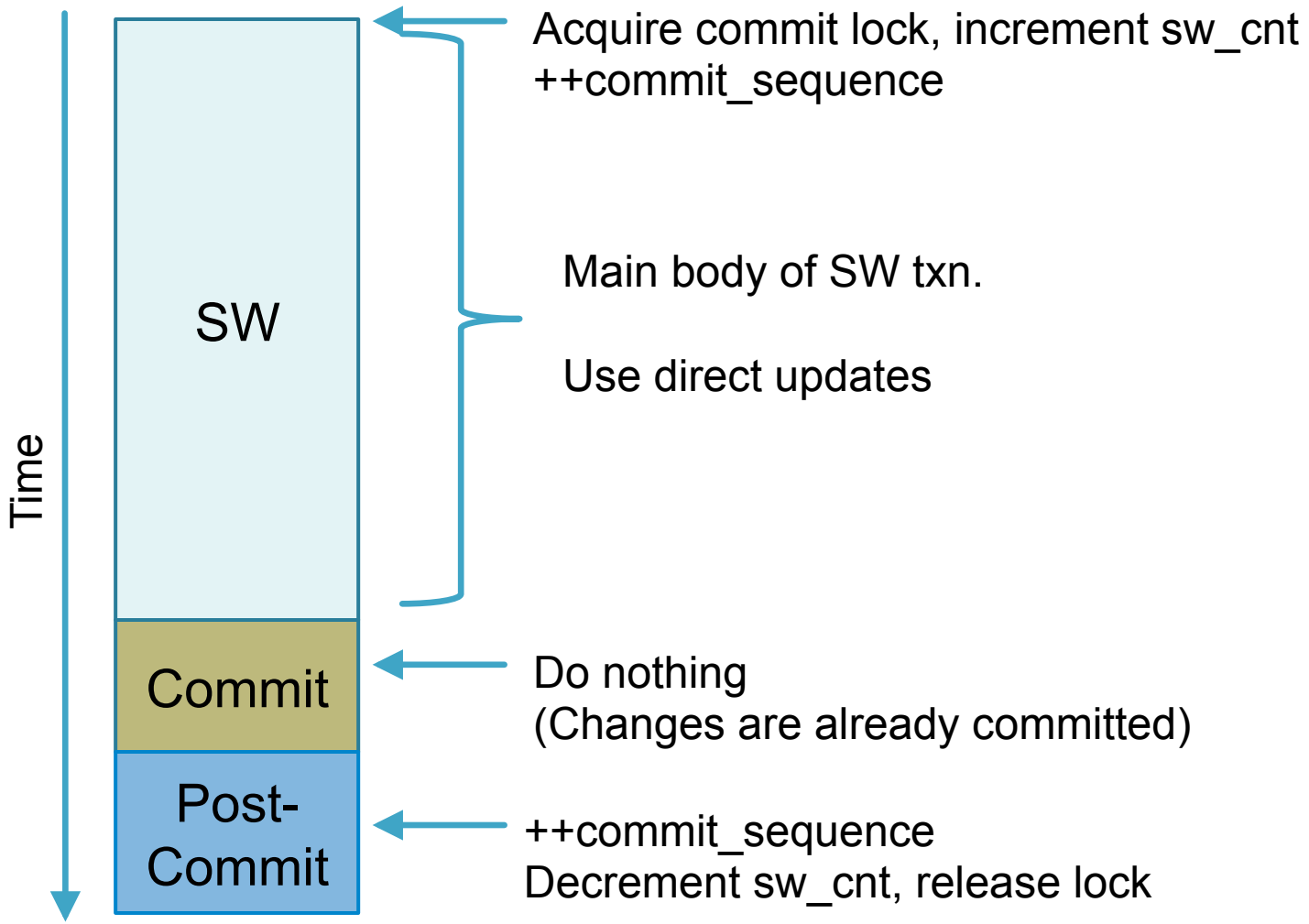


Does not abort – Guarantees Progress

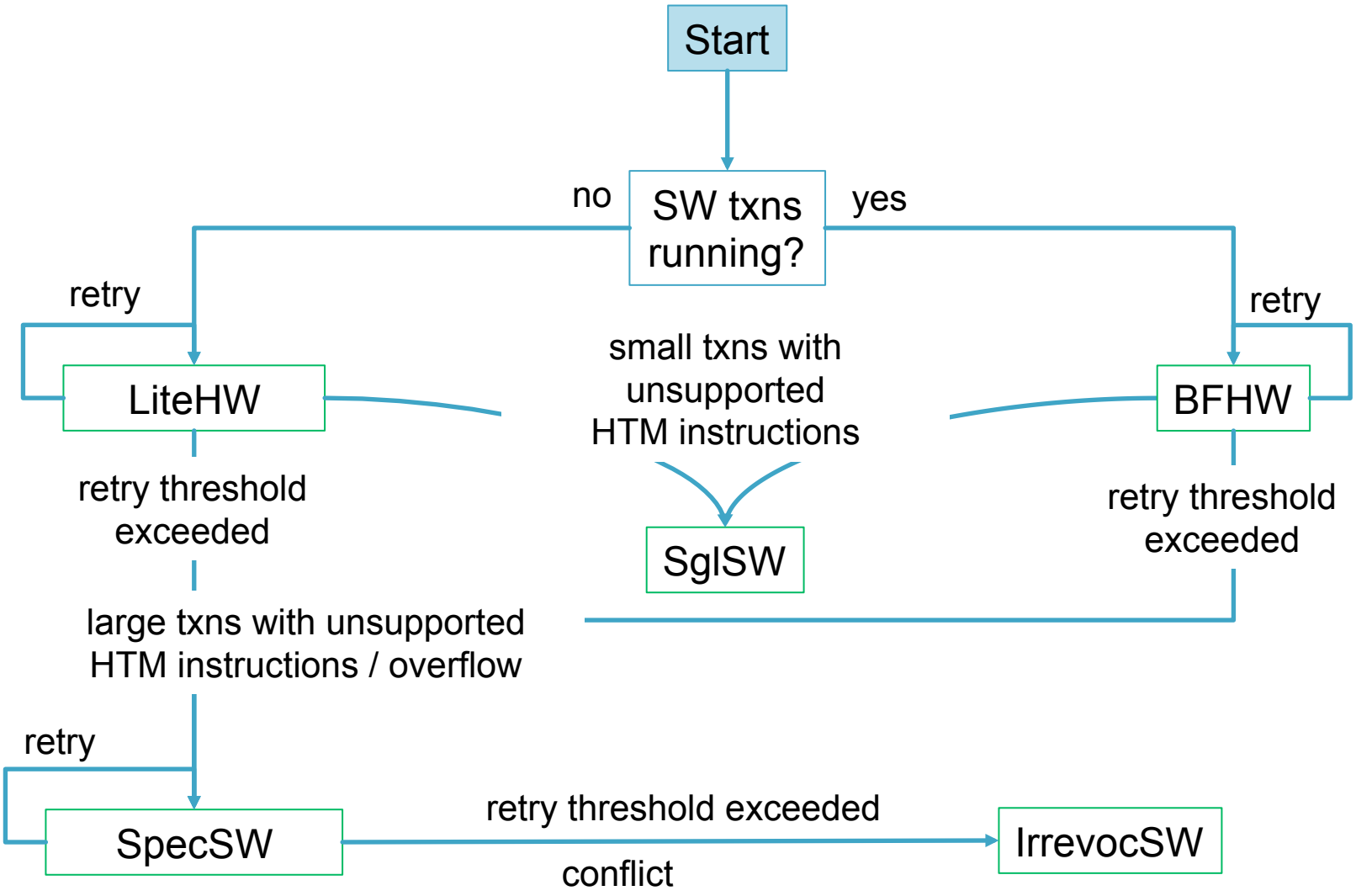
IRREVOCSW (Irrevocable Software)



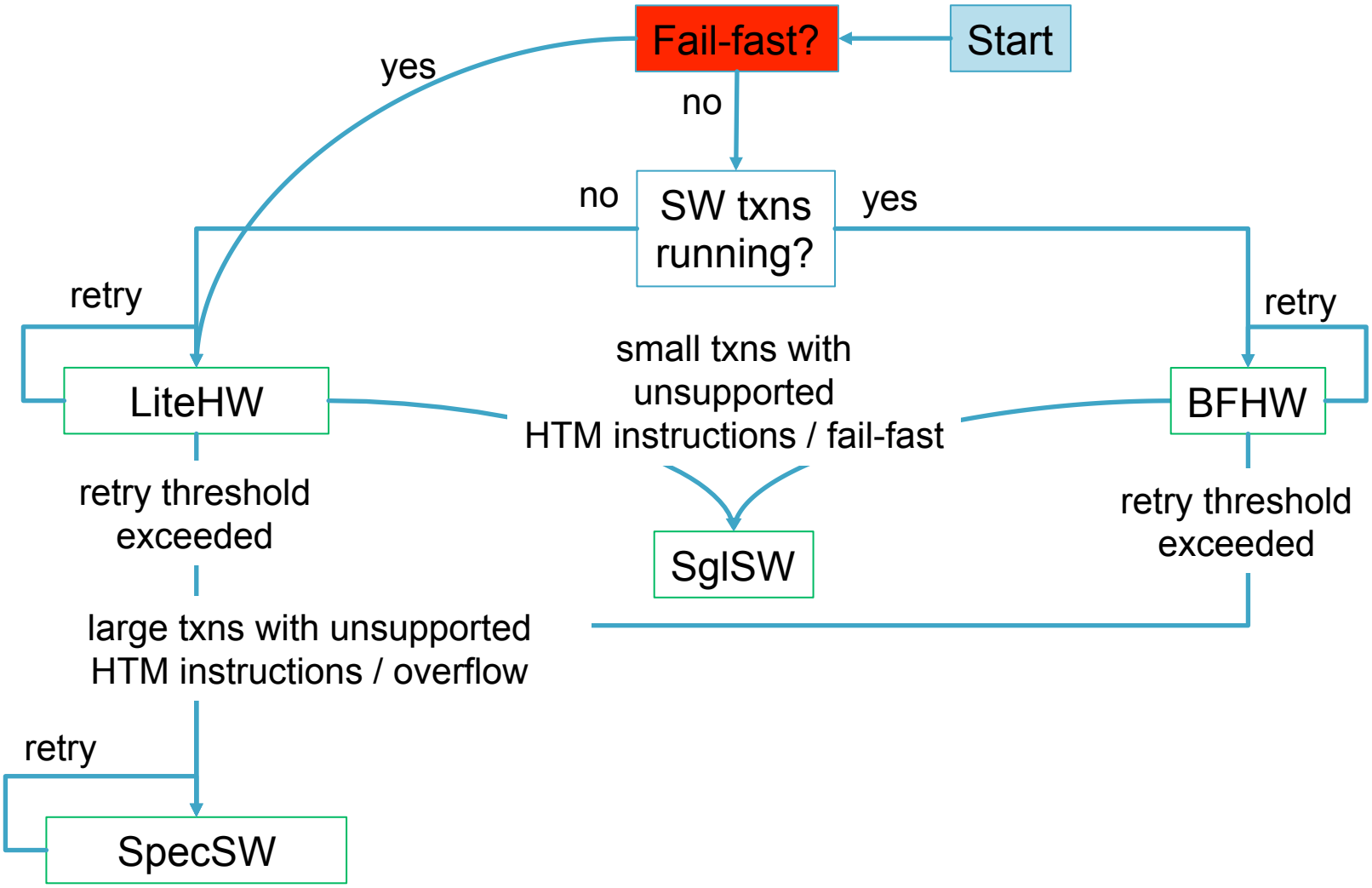
SGLSW (Single-Global-Lock Software)



Invyswell State Diagram



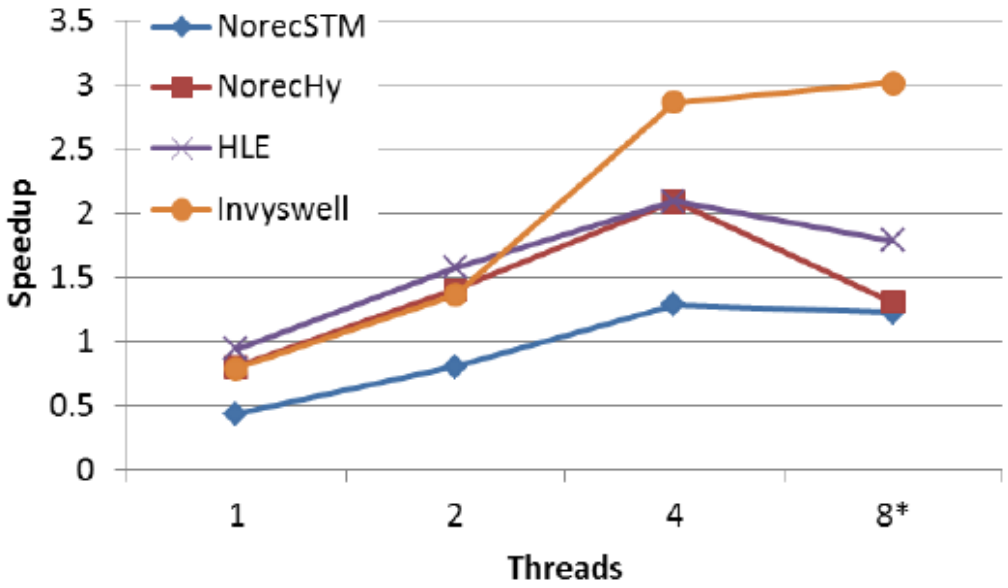
Invyswell State Diagram



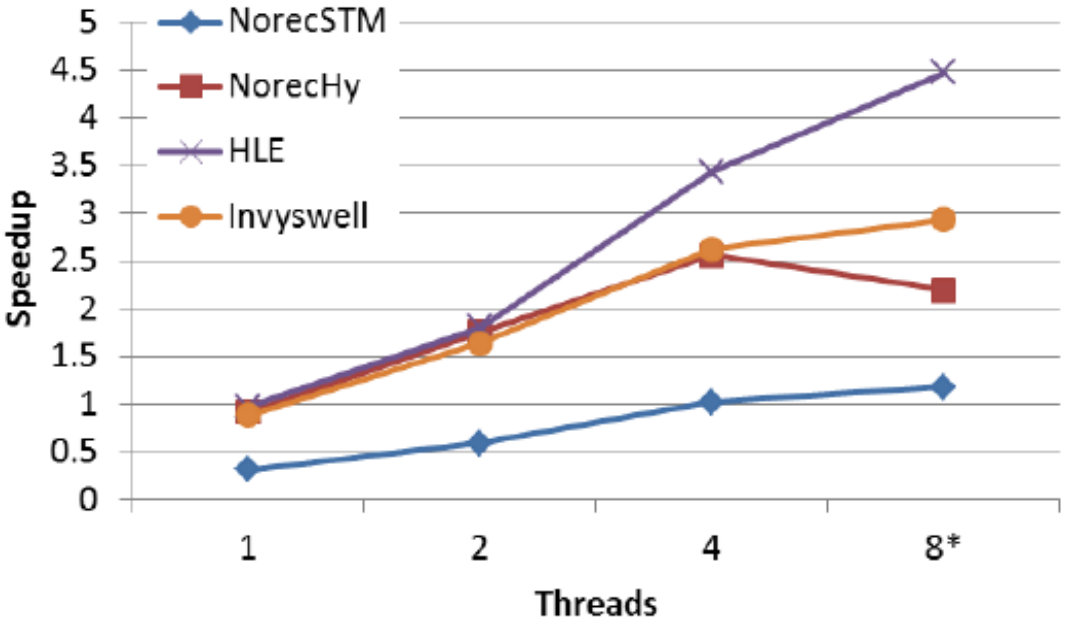
Concurrent Execution Matrix

Types	BFHW	LiteHW	SpecSW	IrrevocSW	SglSW
BFHW	yes	yes	yes	yes	yes
LiteHW	yes	yes	yes	yes	yes
SpecSW	yes	yes	yes	yes	no
IrrevocSW	yes	yes	yes	no	no
SglSW	yes	yes	no	no	no

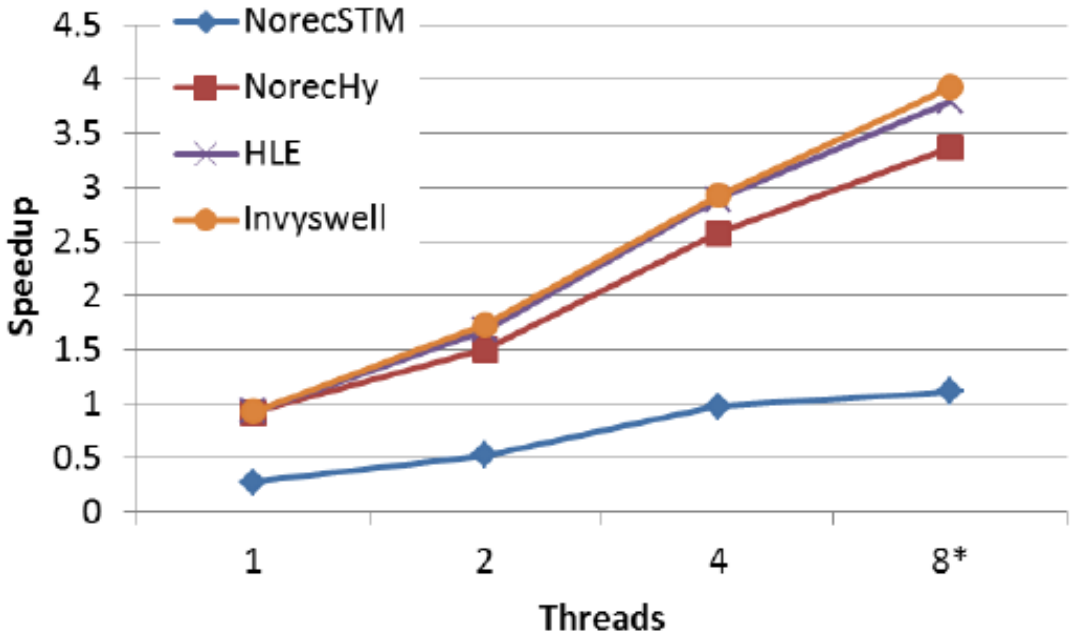
Speedup



Intruder.

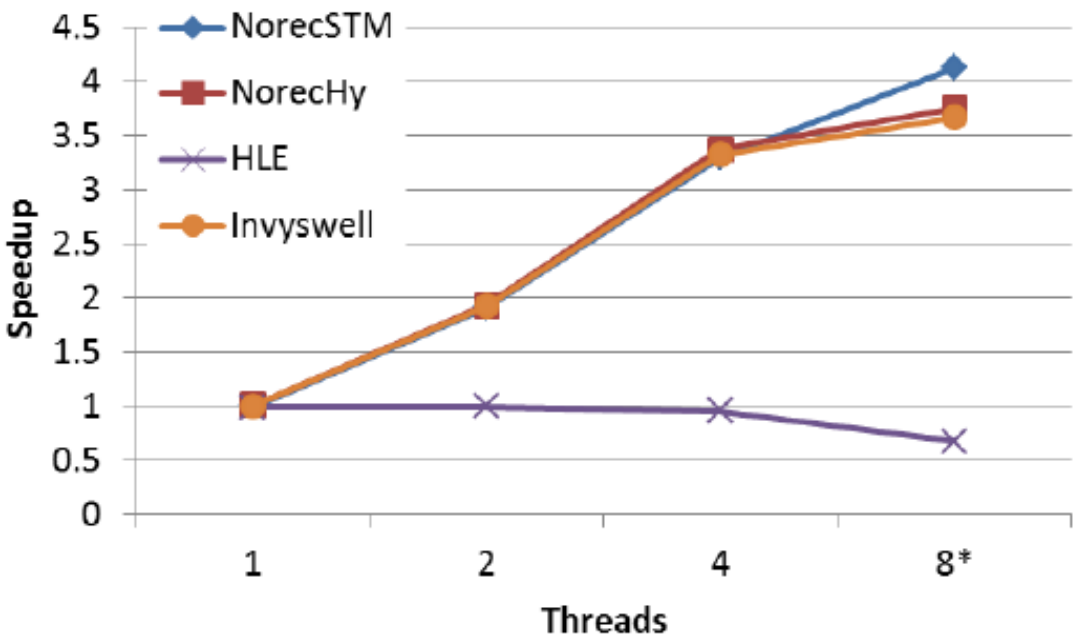


kmeans low.



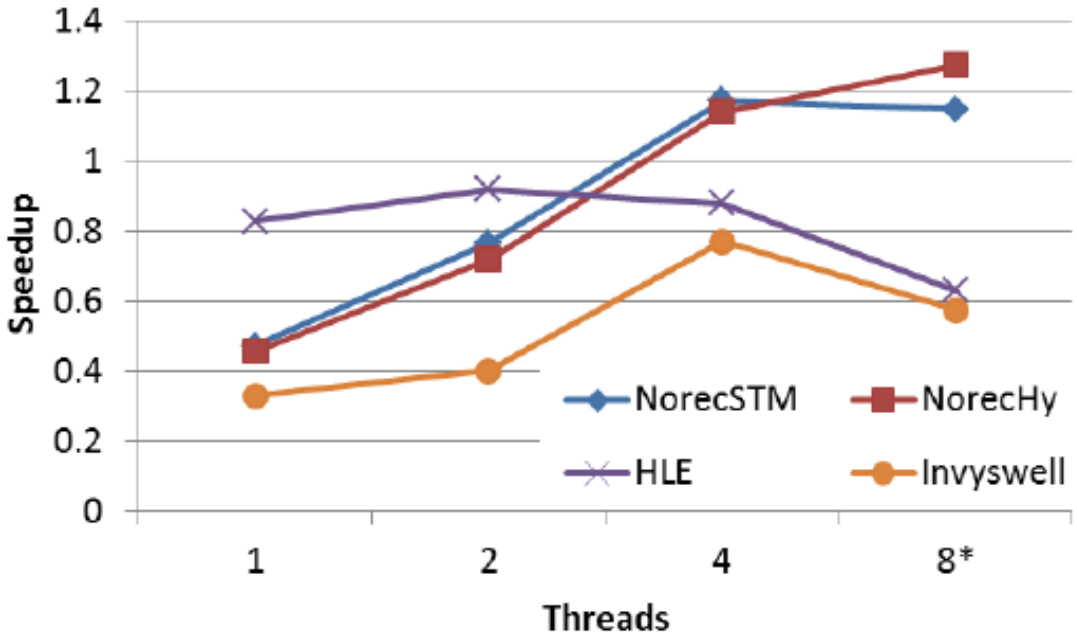
ssca2.

Speedup



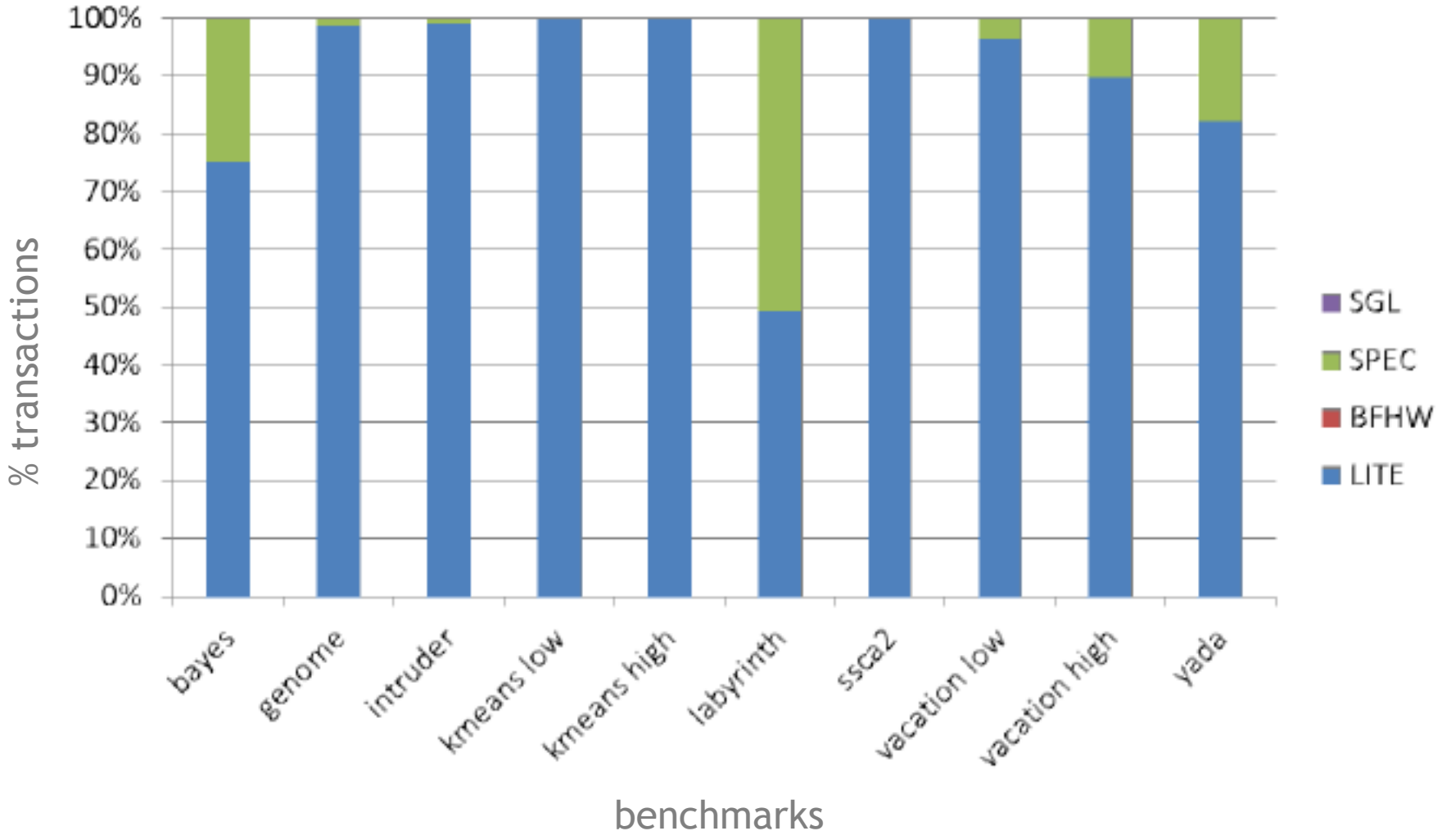
Labyrinth.

Speedup

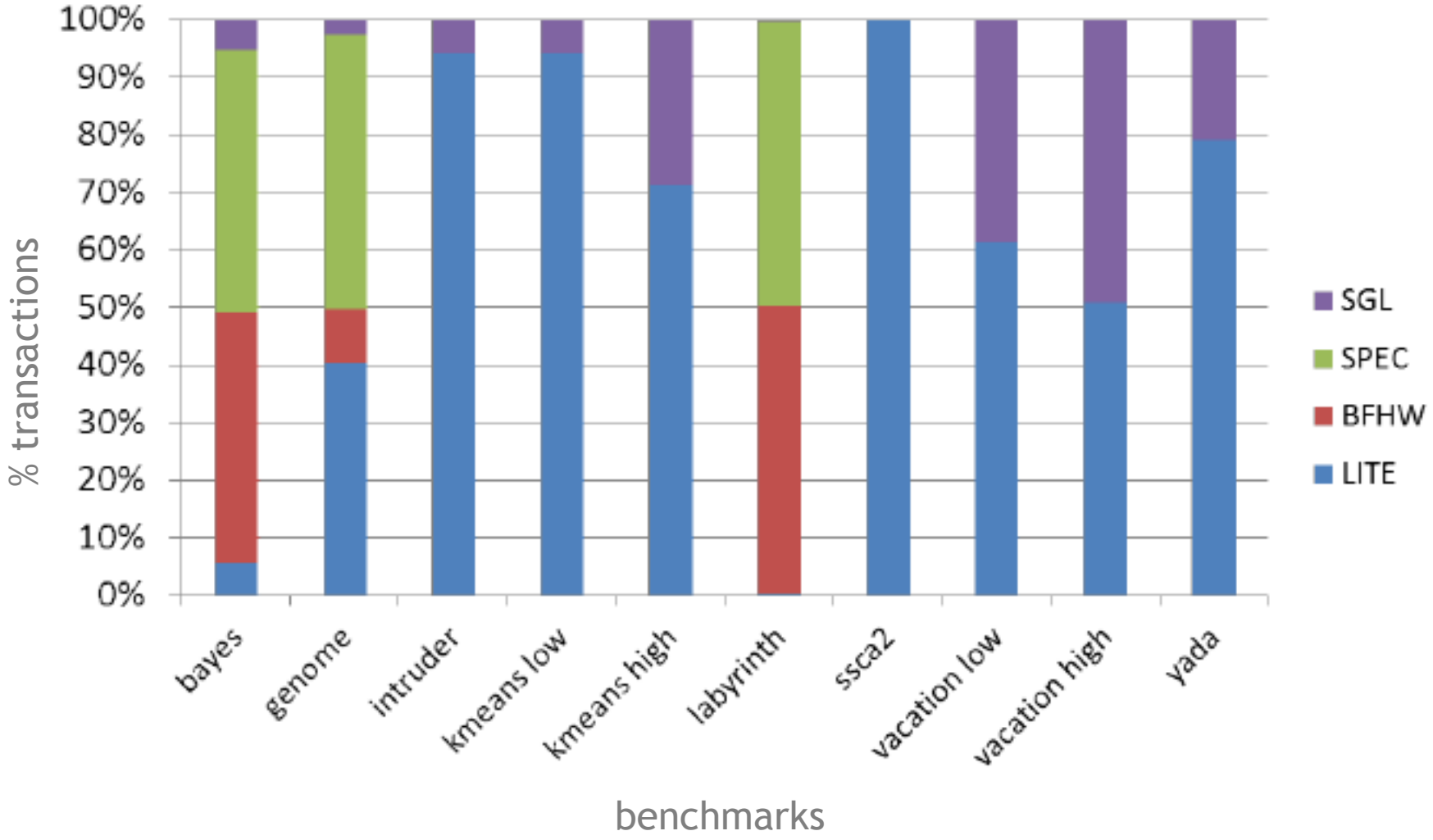


Yada.

Transaction Types - 1 Thread



Transaction Types - 8 Threads



Conclusions

- HLE and RTM w/ SGL fallback are not enough
- Invyswell is 35% faster than NOrec, 18% faster than Hybrid NOrec and 25% faster than HLE across all STAMP benchmarks

Thank you!

- http://cs.brown.edu/~irina
- irina@cs.brown.edu