Persistent hardware transactions can scale

João Barreto

(and Daniel Castro, Alexandro Baldassin, Paolo Romano)

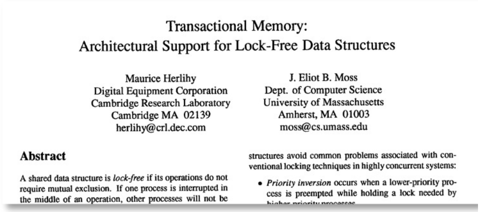
The **transactional memory** abstraction

- Great for concurrency control
- One of the most popular abstraction for “failure-atomic critical sections” in PM literature

```
TX_BEGIN(pool) {
    TX_ADD_DIRECT(&D_RW(FOO));
    TX_ADD_DIRECT(&D_RW(BAR));
    D_RW(FOO) = D_RO(FOO)-100;
    D_RW(BAR) = D_RO(BAR)+100;
} TX_END
```

A persistent memory transaction with PMDK

Hardware transactional memory (HTM)


Transactional Memory:
Architectural Support for Lock-Free Data Structures

Maurice Herlihy
Digital Equipment Corporation
Cambridge Research Laboratory
Cambridge MA 02139
herlihy@crl.dec.com

J. Eliot B. Moss
Dept. of Computer Science
University of Massachusetts
Amherst, MA 01003
moss@cs.umass.edu

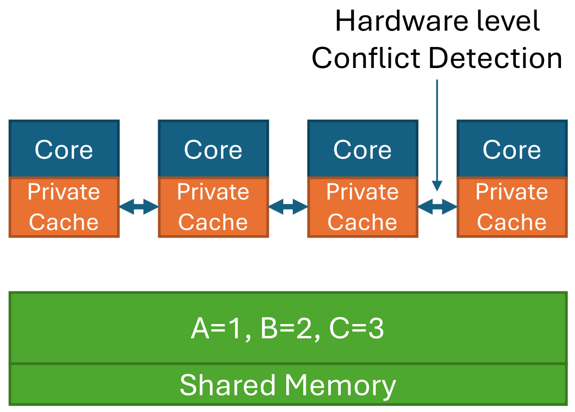
Abstract

A shared data structure is *lock-free* if its operations do not require mutual exclusion. If one process is interrupted in the middle of an operation, other processes will not be

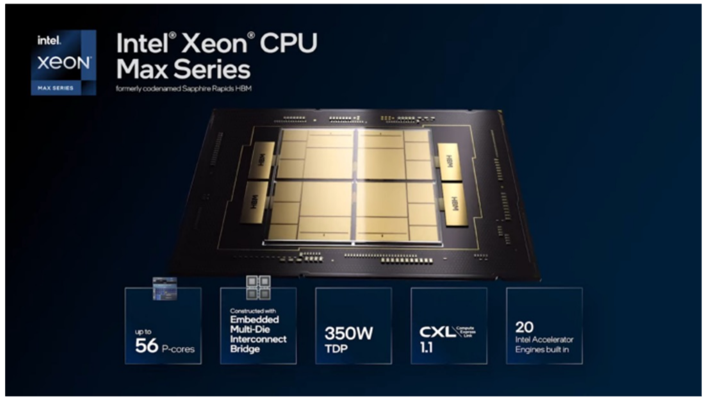
structures avoid common problems associated with conventional locking techniques in highly concurrent systems:

- *Priority inversion* occurs when a lower-priority process is preempted while holding a lock needed by

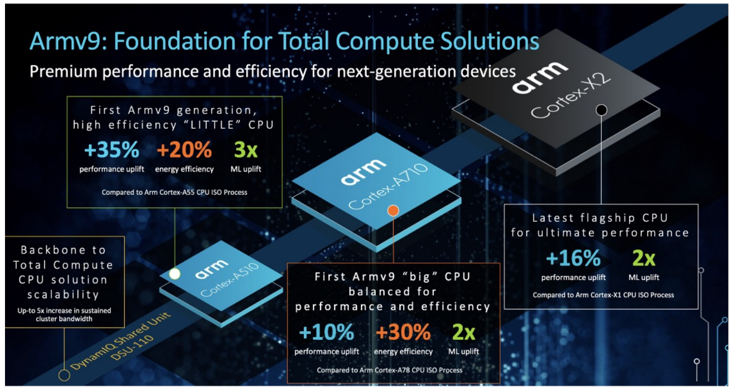
Herlihy and Moss, 1993



```
_xbegin;
foo = foo - 100;
bar = bar + 100;
_xend;
```

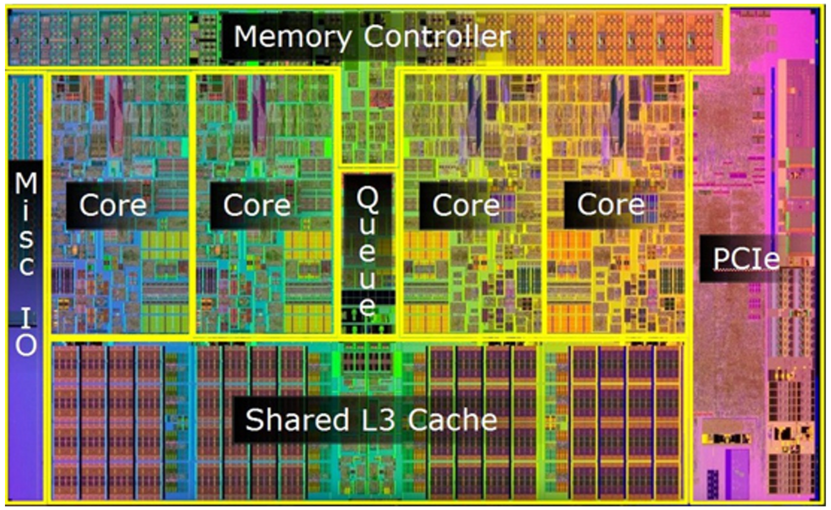


intel. xeon
Intel® Xeon® CPU Max Series
formerly codenamed Sapphire Rapids HBM
up to 56 P-cores
Constructed with Embedded Multi-Die Interconnect Bridge
350W TDP
CXL 1.1
20 Intel Accelerator Engines built in



Armv9: Foundation for Total Compute Solutions
Premium performance and efficiency for next-generation devices
First Armv9 generation, high efficiency "LITTLE" CPU
+35% performance uplift
+20% energy efficiency
3x ML uplift
Compared to Arm Cortex-A55 CPU ISO Process
arm Cortex-A710
arm Cortex-X2
arm Cortex-A510
Backbone to Total Compute CPU solution scalability
Up-to 5x increase in sustained cluster bandwidth
DynamIQ Shared Unit DSU-110
First Armv9 "big" CPU balanced for performance and efficiency
+10% performance uplift
+30% energy efficiency
2x ML uplift
Compared to Arm Cortex-A78 CPU ISO Process
Latest flagship CPU for ultimate performance
+16% performance uplift
2x ML uplift
Compared to Arm Cortex-X1 CPU ISO Process

After the multi-core revolution…
… the persistent memory (PM) revolution



HTM provides **opacity**

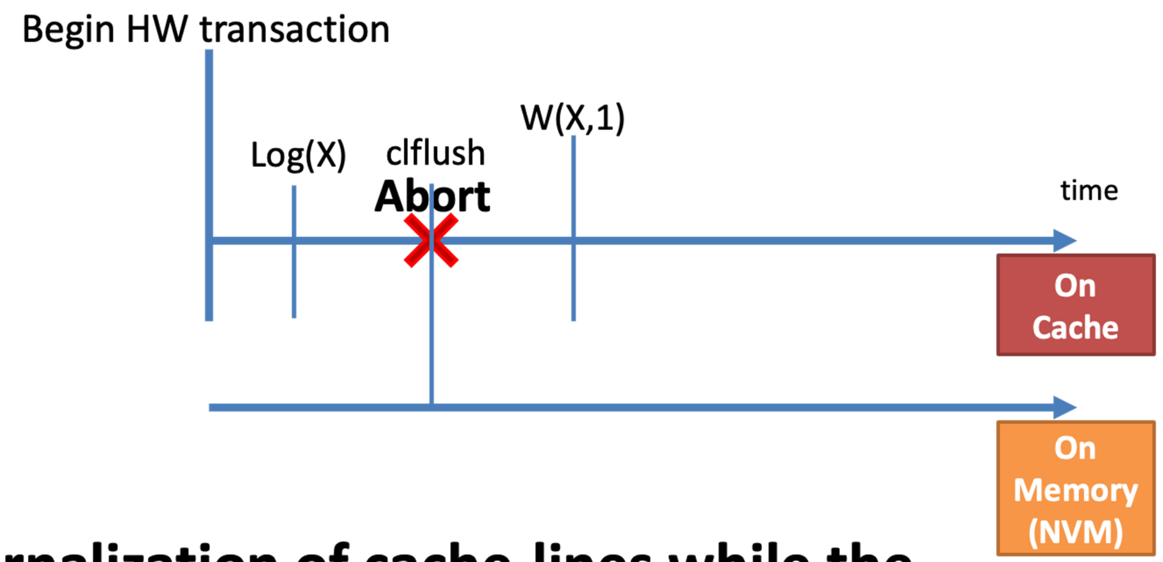
hardware transactions can access persistent objects

However, HTM doesn't guarantee **durable opacity**

How to help HTM support **persistent hardware transactions?**

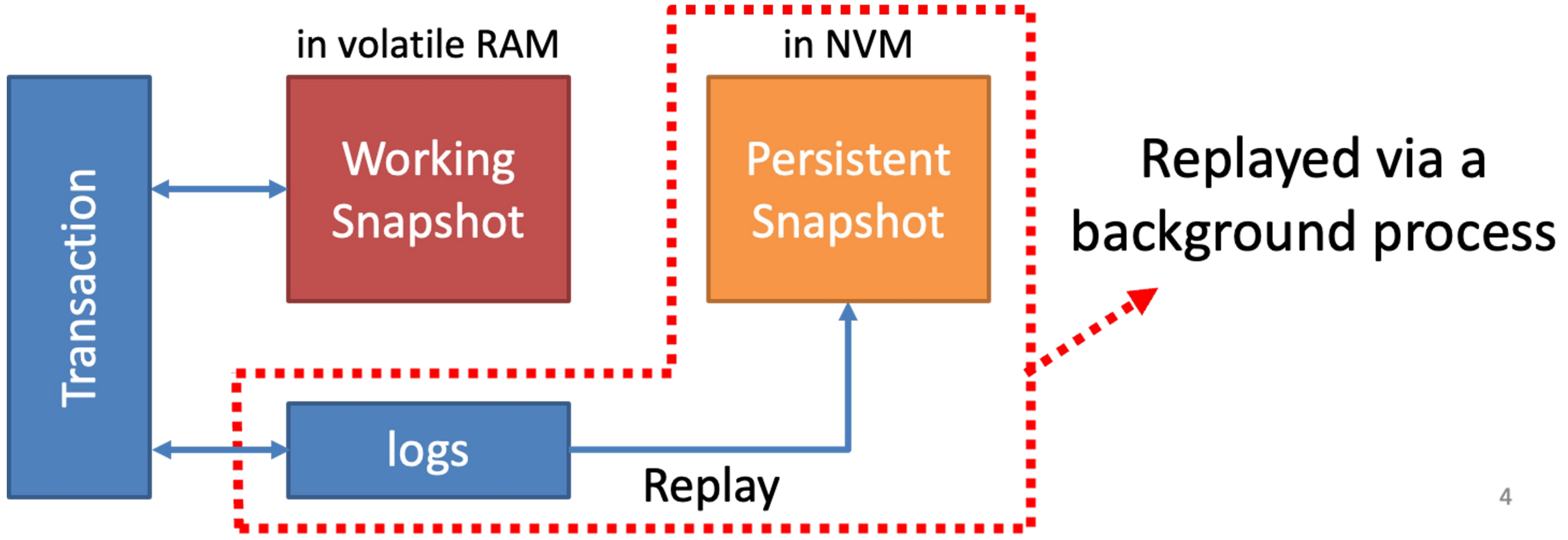
Why not just using write-ahead logging?

- Writes to PM also added to a durable redo log (in PM)
- However, we cannot flush redo log entries to PM before the HTM commits the transaction



Externalization of cache-lines while the transactions is running causes it to abort!

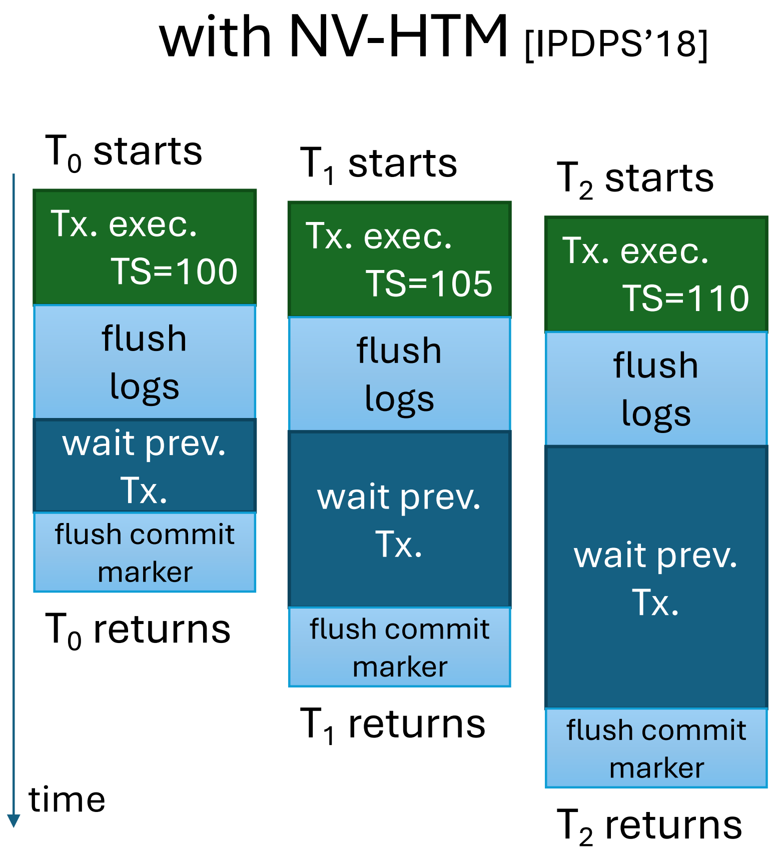
Durable hardware transactions based on a shadow copy

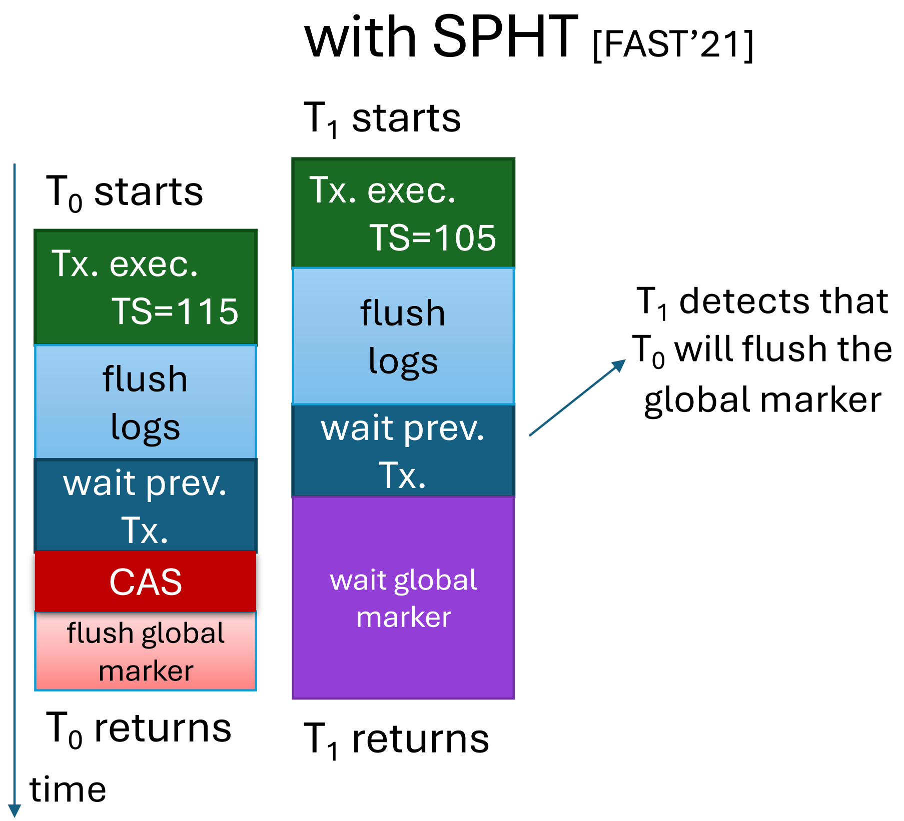


We decouple transaction isolation (via HTM) from durability (via WAL)

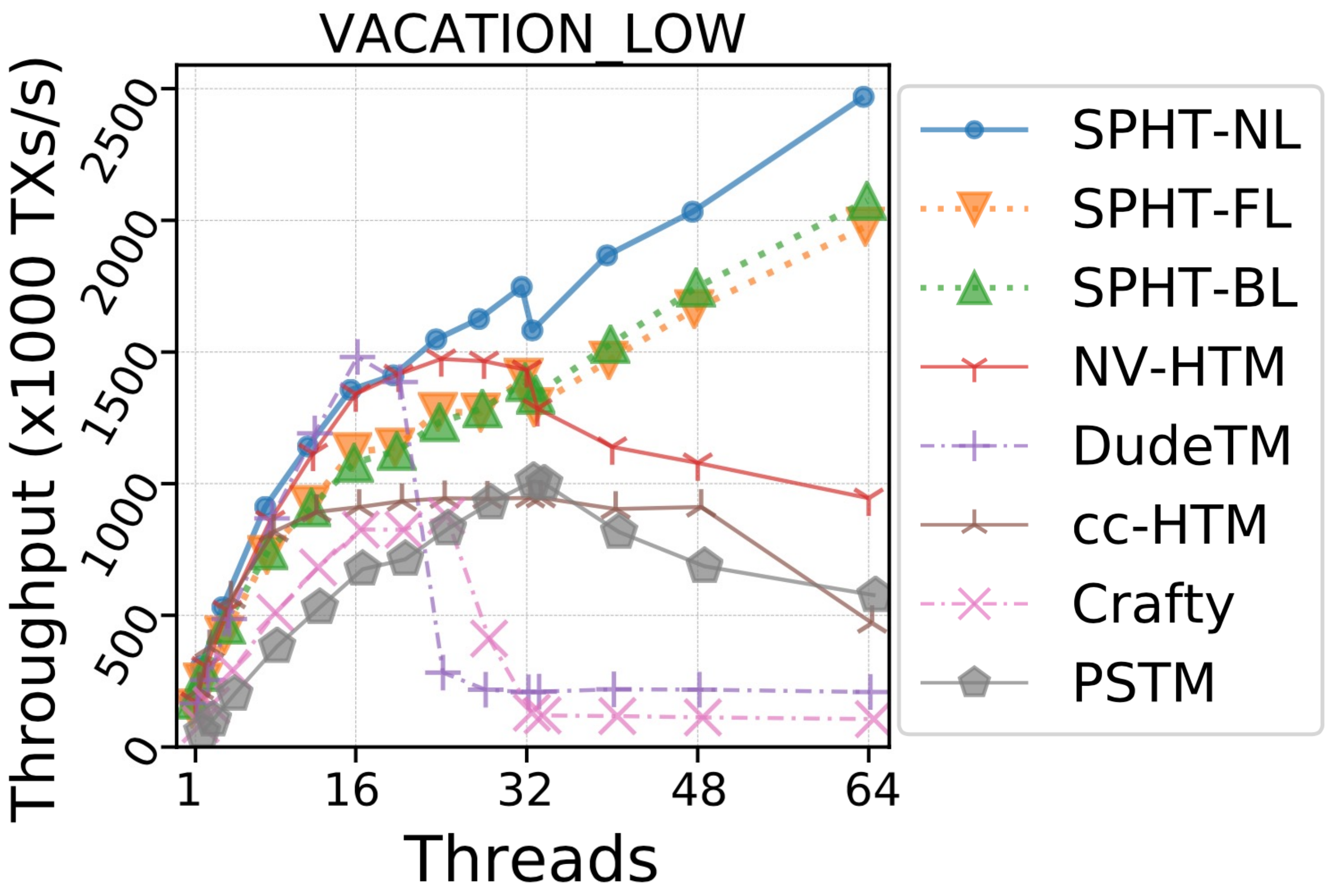
Challenge: Persistence order must be consistent with **happens-before order**

But does it scale?...





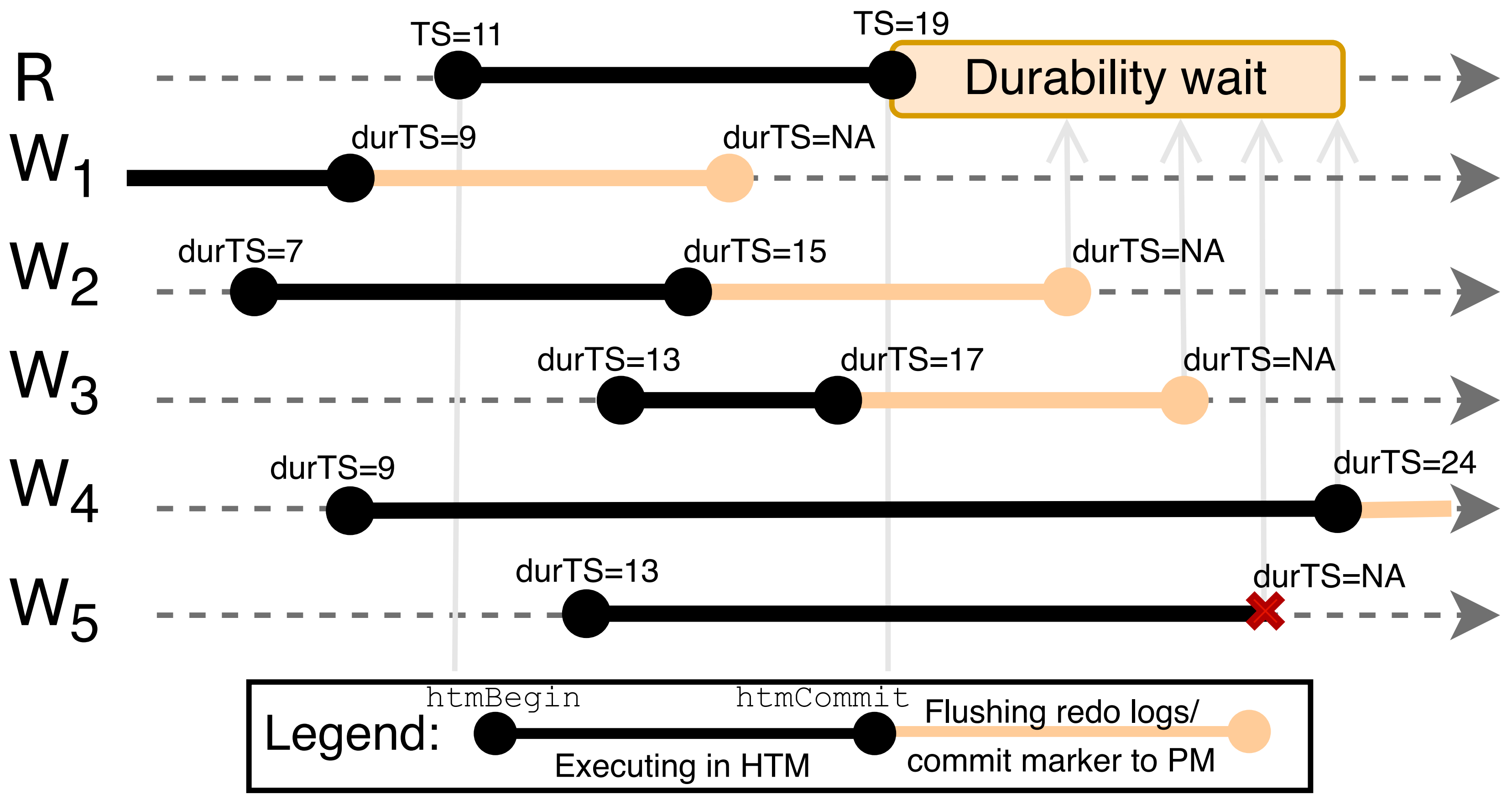
STAMP



VACATION_LOW
Throughput (x1000 TXs/s)
0
500
1000
1500
2000
2500
1
16
32
48
64
Threads
SPHT-NL
SPHT-FL
SPHT-BL
NV-HTM
DudeTM
cc-HTM
Crafty
PSTM

What about read-only transactions?

Scalability issue #1

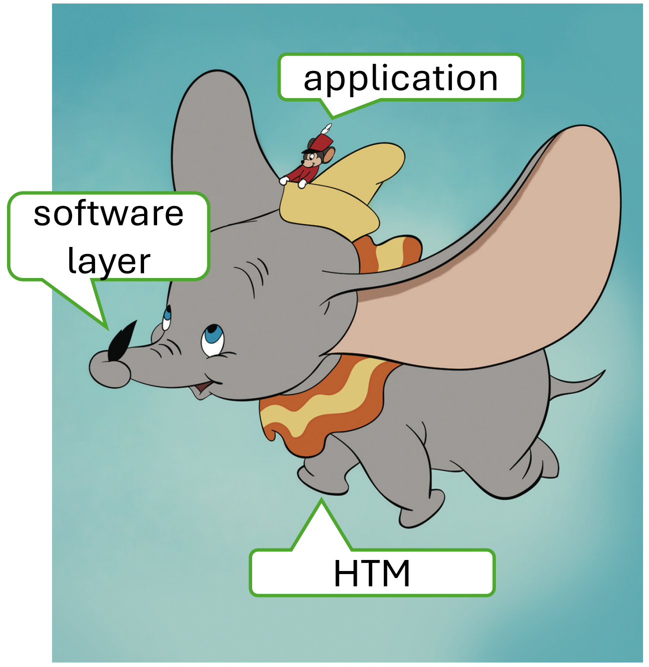


Scalability issue #2

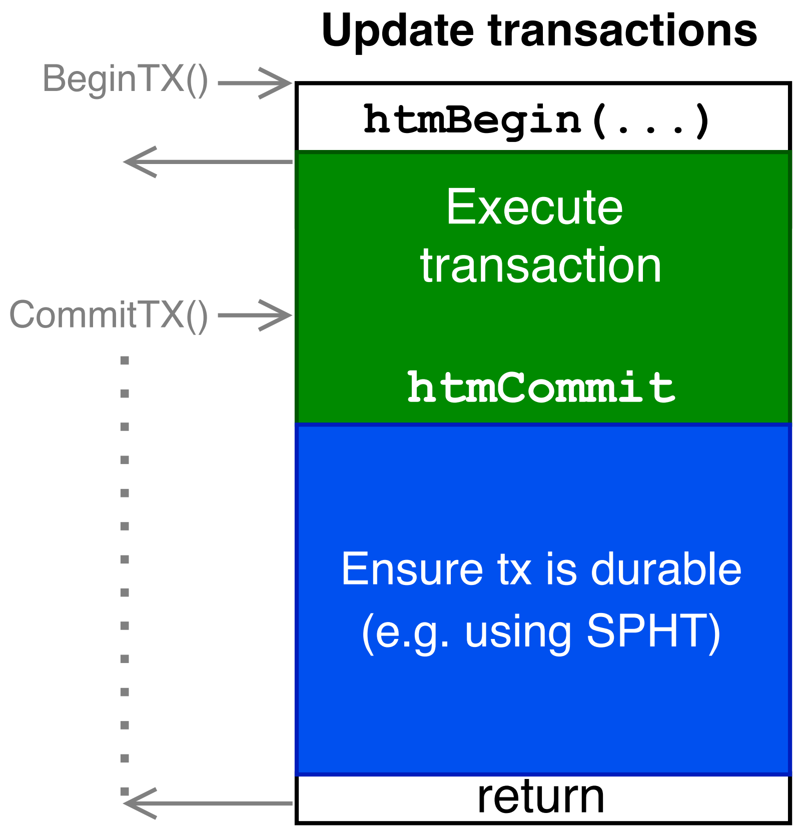
- Hardware transactions have a (very) limited read set capacity
- When a transaction exceeds its capacity, the HTM abort it
- Typically, the transaction must acquire a single global lock to execute

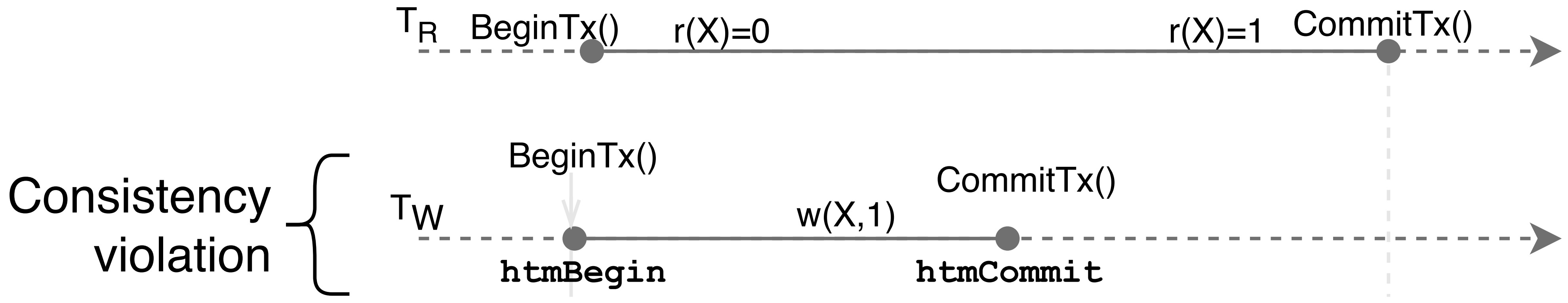
Making read-only transactions scale with DUMBO

- Scalability issue #1:
  Read-only transactions ***practically* never need to wait** for writer transactions to persist
- Scalability issue #2:
  Read-only transactions **run HTM-free** (without read instrumentation), so have **unlimited reads and never abort**

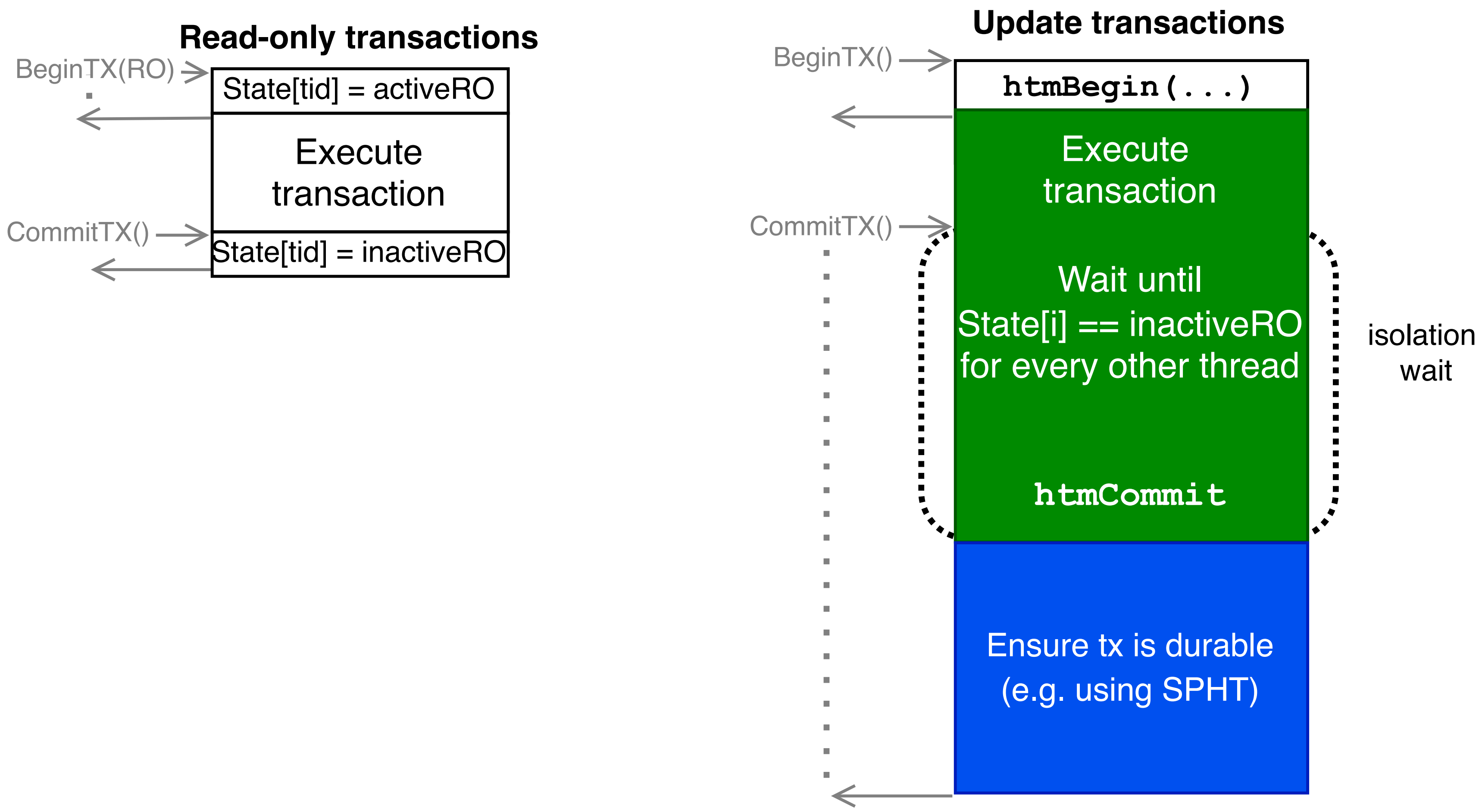


Durable Unlimited-reads Memory transactions on Best-effOrt HTM (DUMBO)


Update transactions
BeginTX()
htmBegin(...)
Execute transaction
CommitTX()
htmCommit
Ensure tx is durable (e.g. using SPHT)
return

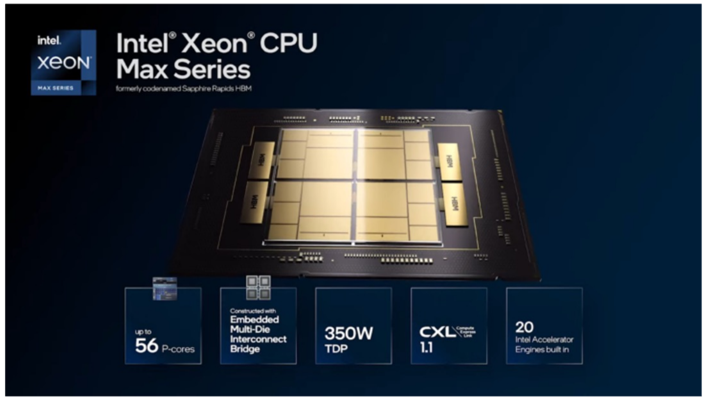


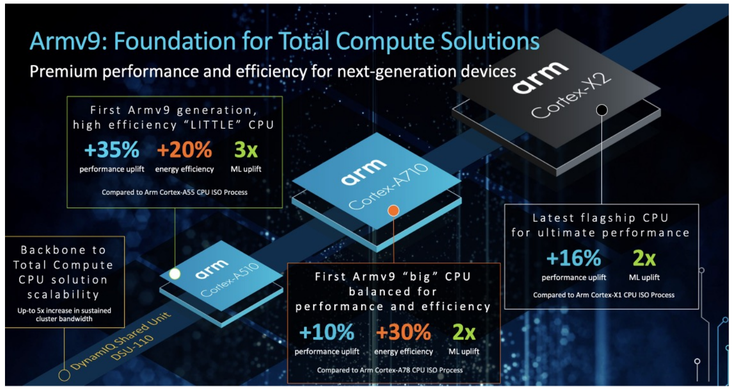
T_R
BeginTx()
r(X)=0
r(X)=1
CommitTx()
Consistency violation
BeginTx()
CommitTx()
T_W
w(X,1)
htmBegin
htmCommit



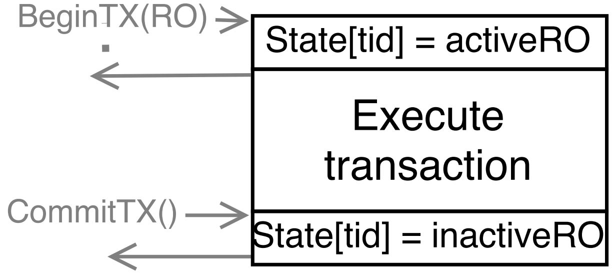
Read-only transactions
BeginTX(RO)
State[tid] = activeRO
Execute transaction
CommitTX()
State[tid] = inactiveRO
Update transactions
BeginTX()
htmBegin(...)
Execute transaction
CommitTX()
Wait until State[i] == inactiveRO for every other thread
htmCommit
isolation wait
Ensure tx is durable (e.g. using SPHT)

Existing HTMs	HTM can suspend access tracking?
ARM TME	no
Intel TSX	loads
PowerHTM	loads or full

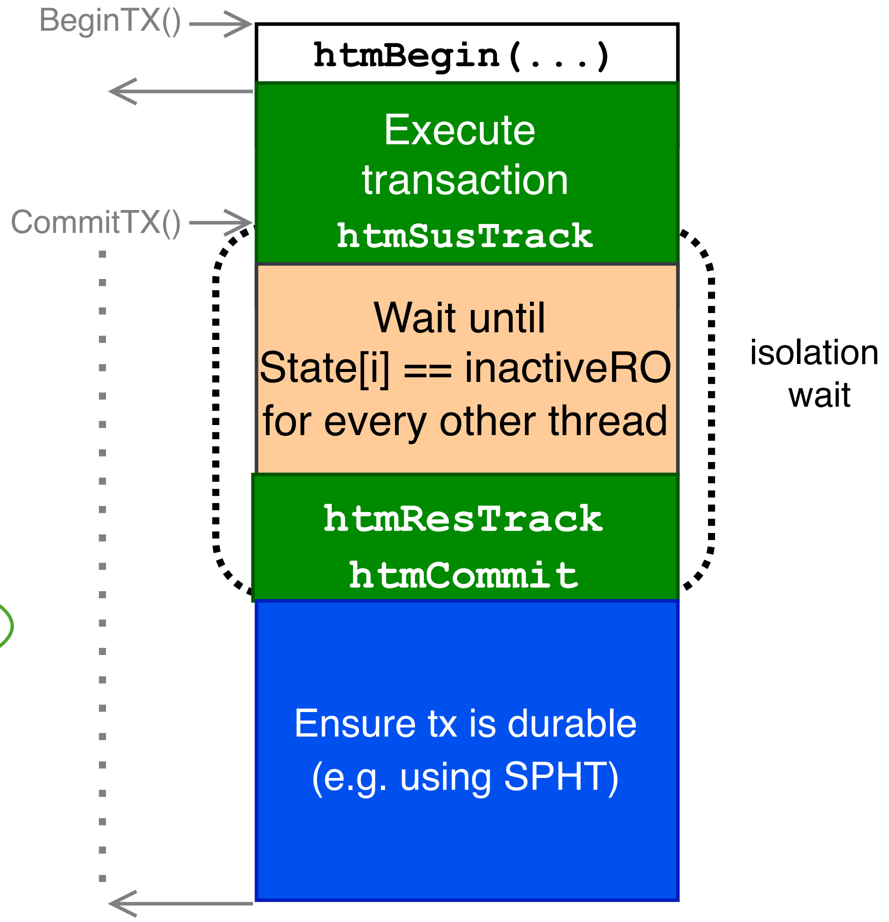

intel. xeon MAX SERIES
Intel® Xeon® CPU Max Series
formerly codenamed Sapphire Rapids HBM
HBM
up to 56 P-cores
Constructed with Embedded Multi-Die Interconnect Bridge
350W TDP
CXL Compute Express Link 1.1
20 Intel Accelerator Engines built in


Armv9: Foundation for Total Compute Solutions
Premium performance and efficiency for next-generation devices
First Armv9 generation, high efficiency "LITTLE" CPU
+35% performance uplift
+20% energy efficiency
3x ML uplift
Compared to Arm Cortex-A55 CPU ISO Process
arm Cortex-X2
arm Cortex-A710
arm Cortex-A510
Latest flagship CPU for ultimate performance
+16% performance uplift
2x ML uplift
Compared to Arm Cortex-X1 CPU ISO Process
Backbone to Total Compute CPU solution scalability
Up-to 5x increase in sustained cluster bandwidth
DynamIQ Shared Unit DSU-110
First Armv9 "big" CPU balanced for performance and efficiency
+10% performance uplift
+30% energy efficiency
2x ML uplift
Compared to Arm Cortex-A78 CPU ISO Process

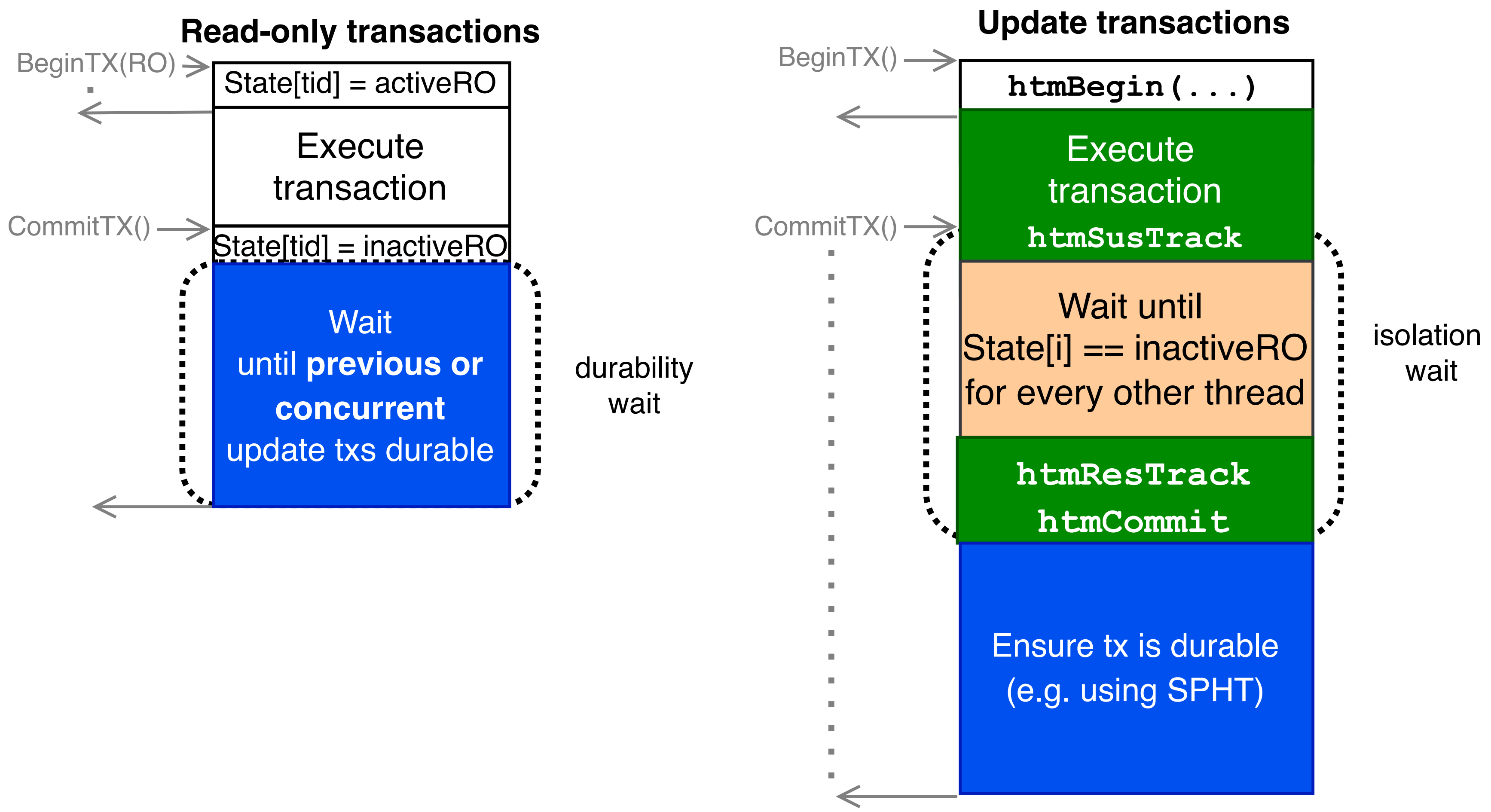
Read-only transactions



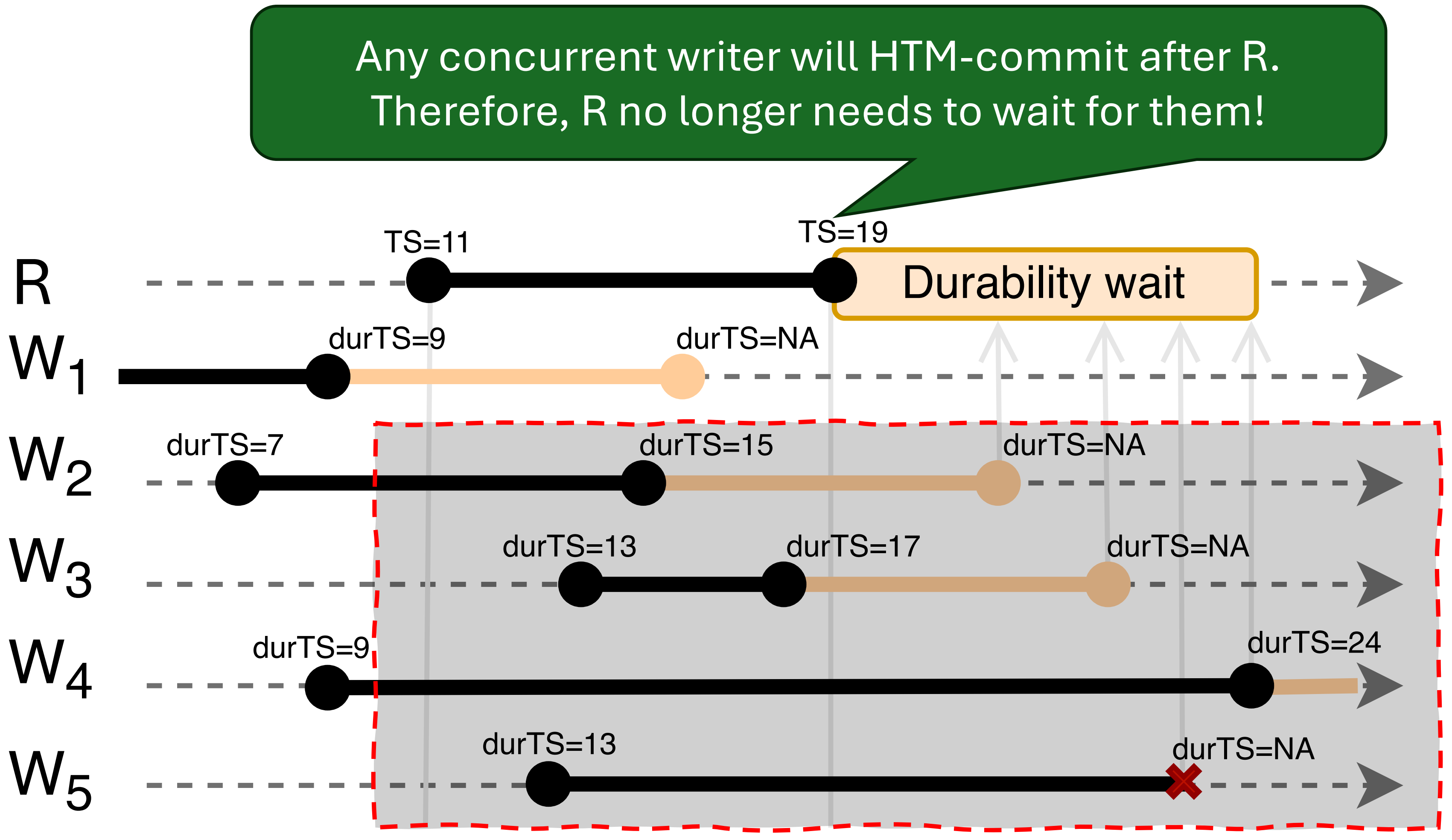
Update transactions

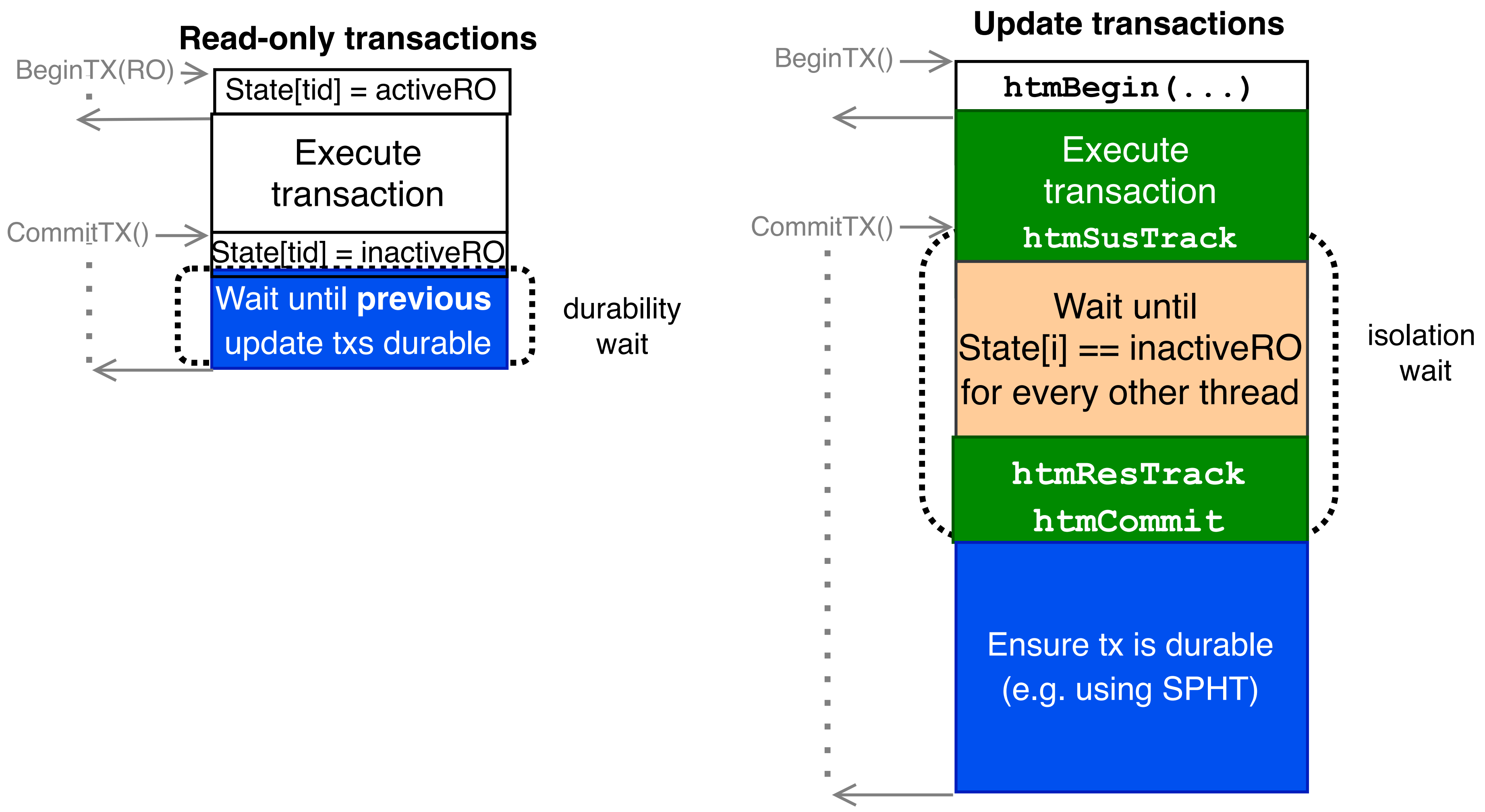


Not ready yet.
We need to fix the ”The persistence bug”
(see Erez Petrank’s previous talk)


Read-only transactions
BeginTX(RO)
State[tid] = activeRO
Execute transaction
CommitTX()
State[tid] = inactiveRO
Wait until **previous or concurrent** update txs durable
durability wait
Update transactions
BeginTX()
htmBegin(...)
Execute transaction
htmSusTrack
CommitTX()
Wait until State[i] == inactiveRO for every other thread
isolation wait
htmResTrack
htmCommit
Ensure tx is durable (e.g. using SPHT)

Revisiting the durability wait of read-only txs




Read-only transactions
BeginTX(RO)
State[tid] = activeRO
Execute transaction
CommitTX()
State[tid] = inactiveRO
Wait until **previous** update txs durable
durability wait
Update transactions
BeginTX()
htmBegin(...)
Execute transaction
htmSusTrack
CommitTX()
Wait until State[i] == inactiveRO for every other thread
isolation wait
htmResTrack
htmCommit
Ensure tx is durable (e.g. using SPHT)

Evaluation

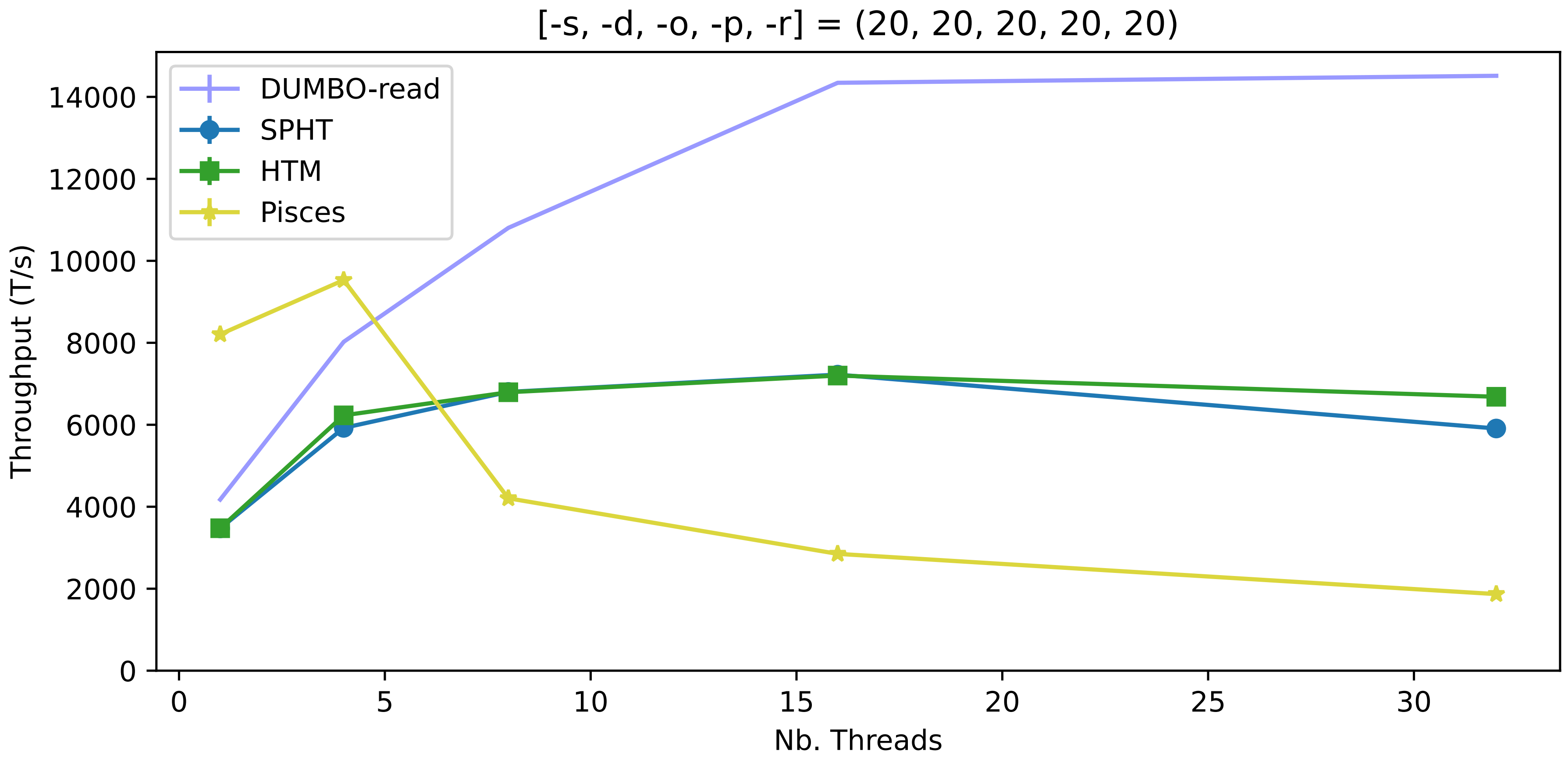
- IBM POWER9
- OS-assisted suspend/resume (HTM issues a trap) => high suspend/resume latency
- DUMBO, SPHT [FAST'21], HTM, Pisces (SI STM) [ATC'19]

- TPC-C benchmark with an even mix of operations:
  - "Stock level" and "Order status" (large read-only transactions)
  - "Payment" (small writer transactions)
  - "Delivery", "New order" (large writer transactions)

Oregon State University

Machine provided by OSU Open Source Lab

Results


[-s, -d, -o, -p, -r] = (20, 20, 20, 20, 20)
DUMBO-read
SPHT
HTM
Pisces
Throughput (T/s)
Nb. Threads
0
2000
4000
6000
8000
10000
12000
14000
0
5
10
15
20
25
30

Summary

- We can have persistent hardware transactions with HTM+"software glue", however scalability is a huge challenge
- **SPHT** [FAST'21] accelerates writer transactions with a new commit logic that mitigates scalability bottlenecks of previous alternatives
- **DUMBO** [wip] boosts read-only transactions by granting them unlimited reads and a reduced durability wait
- Suspend/resume tracking support in HTM is useful, even if through expensive OS-assisted mechanisms

João Barreto | joao.barreto@tecnico.ulisboa.pt | https://www.dpss.inesc-id.pt/~jpbarreto/

More DUMBO (not in this talk)

- On IBM POWER9, we can also suspend load+store tracking
- This also enables DUMBO to:
  - Hide redo log flush latency
  - Generlize unlimited reads to writer transactions
  - Improve durability&log replay logic