Interval-Based Memory Reclamation

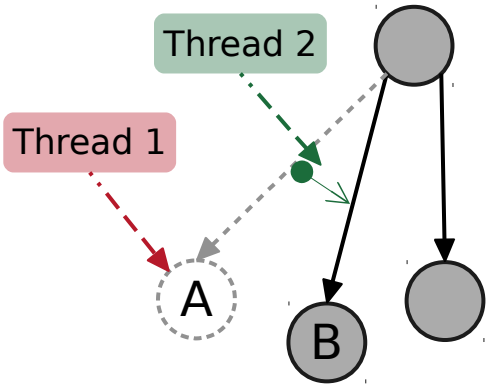
Haosen Wen, Joseph Izraelevitz, Wentao Cai,
H. Alan Beadle and Michael L. Scott

University of Rochester

PPoPP'18

Background

- Unlike lock-based concurrent data structures, non-blocking ones allow updates to happen concurrently with other accesses.
  - Specifically, a thread might try to reclaim a block while others still have access to it.

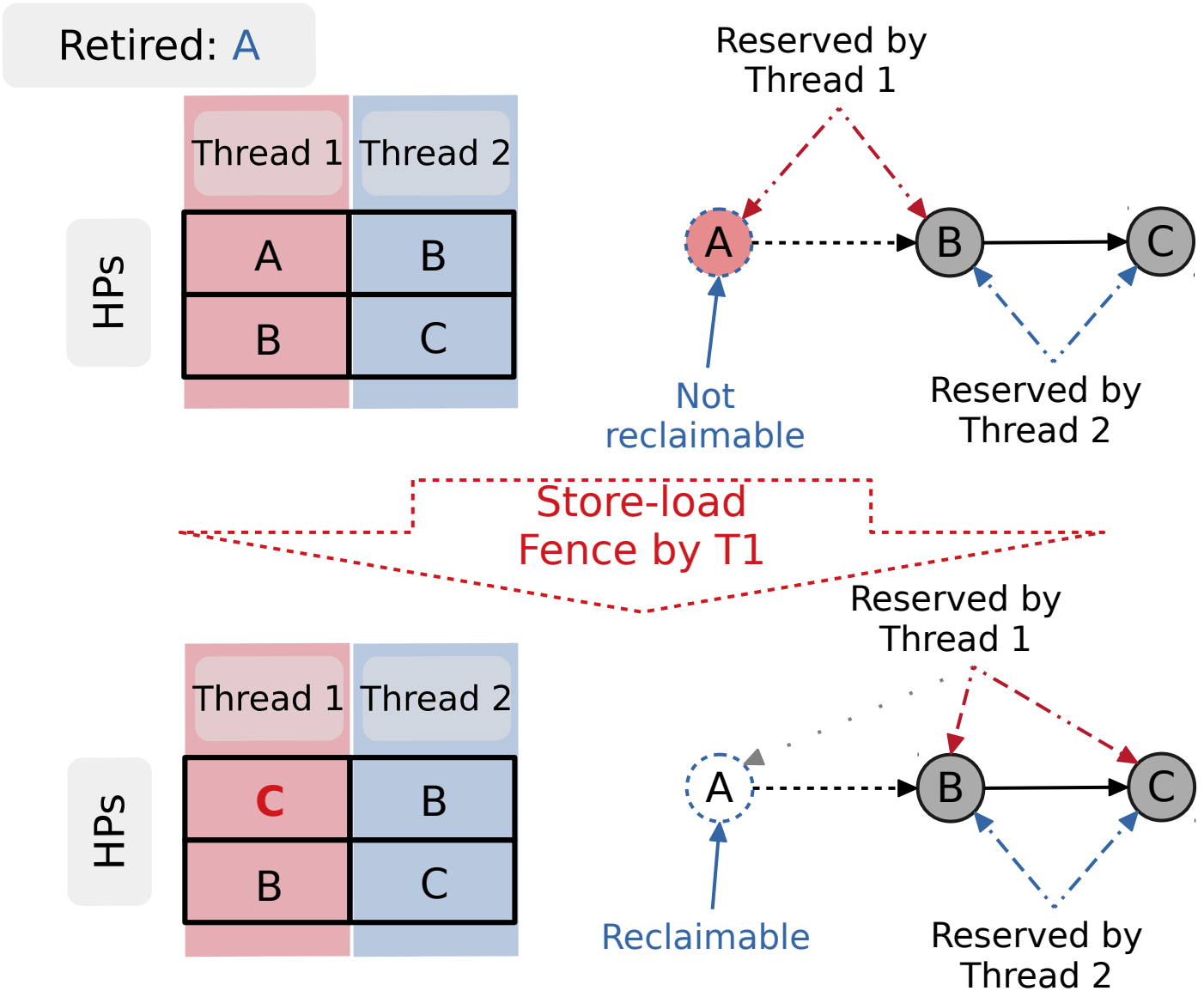


- (Thread-safe) garbage collecting languages tend to bring high overhead.

The Problem

- Manual approaches are majorly based on "reservations," a global metadata, which require expensive store-load fences to update:
  - Hazard Pointers (HP) [Michael, PODC'02] reserves a minimum number of blocks per thread, but updates reservation every time a thread follows a shared pointer.
  - Epoch Based Reclamation (EBR) [Fraser, thesis'04]; [Hart et al., 2007] only issues memory fences at beginnings and ends of operations, but a stalling thread may cause an unbounded amount of blocks to be unreclaimable.
- **Our approach improves EBR by making it robust to thread stalling.**

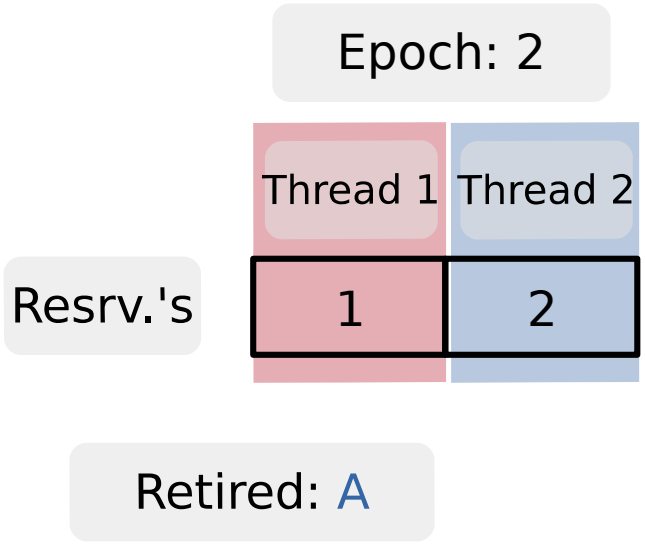
Hazard Pointers (HP)

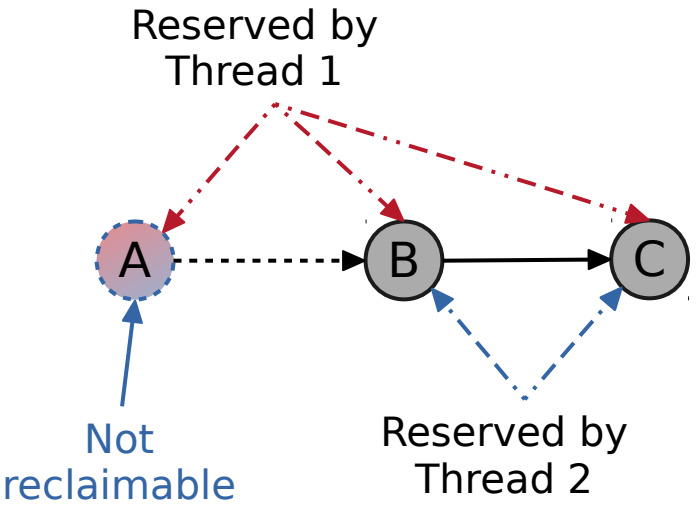


- Thread 1 is traversing a linked list and Thread 2 is retiring block A.
- Blocks in global array of HPs are reserved from reclamations.

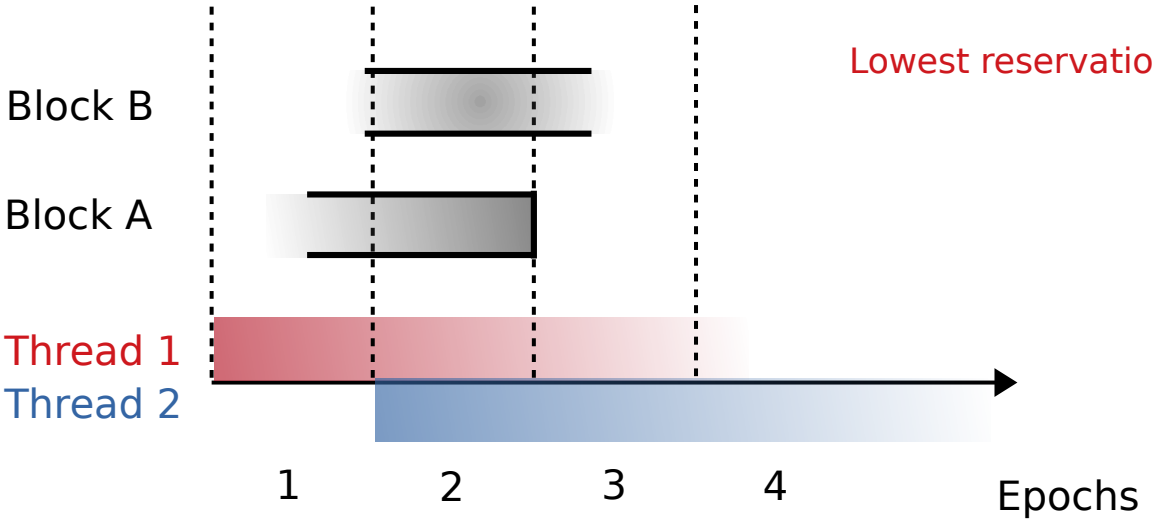
- Store-load fences are issued on every HP update.
- Number of HPs per thread is usually small, but can be unbounded in some cases.

Epoch-Based Reclamation (EBR)

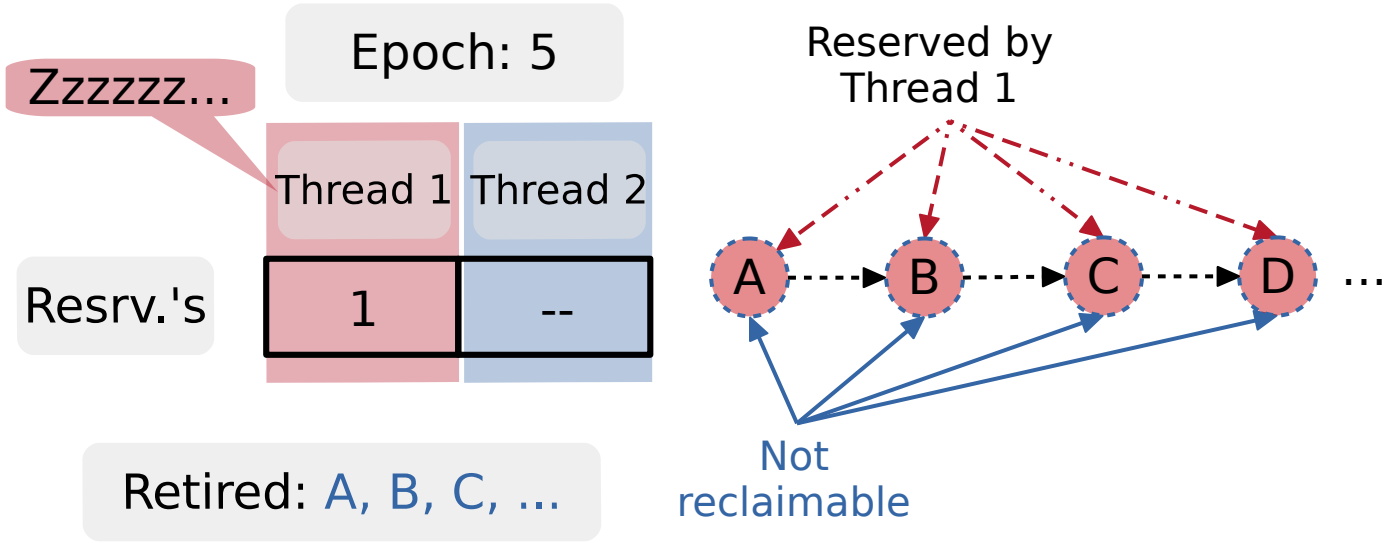


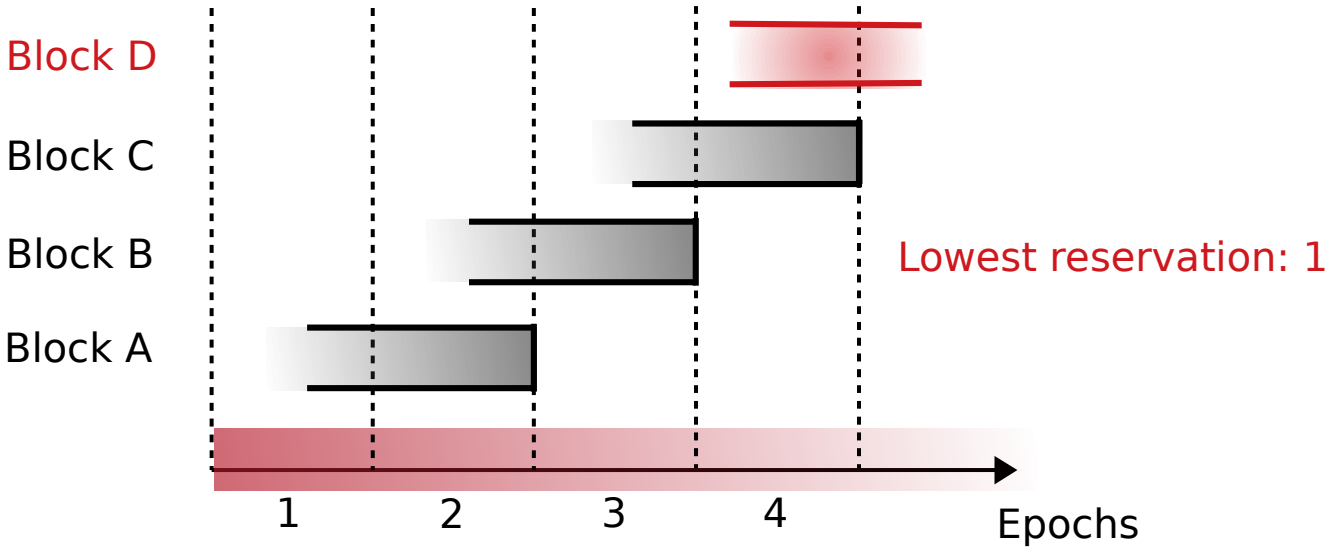


- The Epoch counter is a slow-ticking "clock"
- Each thread puts the current epoch *E* in reservation at the beginning of operations, reserving all objects retired **on and after epoch *E***.
- As a result, only blocks retired before the lowest reservation can be reclaimed.



Epoch-Based Reclamation (EBR)





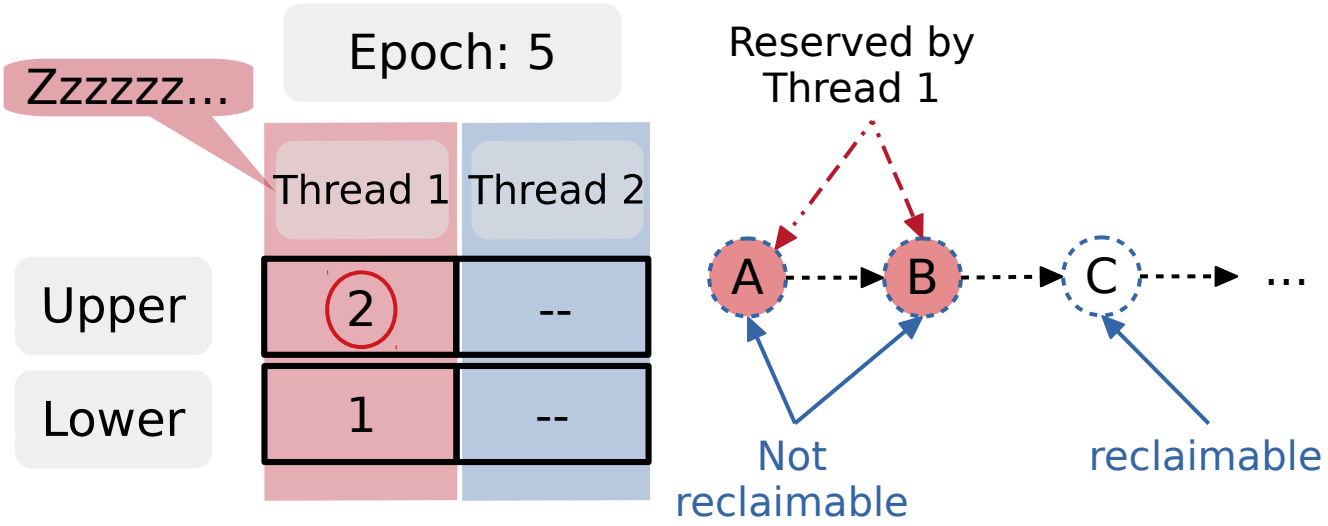
- The Epoch counter is a slow-ticking "clock"
- Each thread puts the current epoch *E* in reservation at the beginning of operations, reserving all objects retired **on and after epoch *E***.
- As a result, only blocks retired before the lowest reservation can be reclaimed.
- Unbounded numbers of blocks may be tied up if some thread is stalled: EBR is **not robust** to thread stalling.

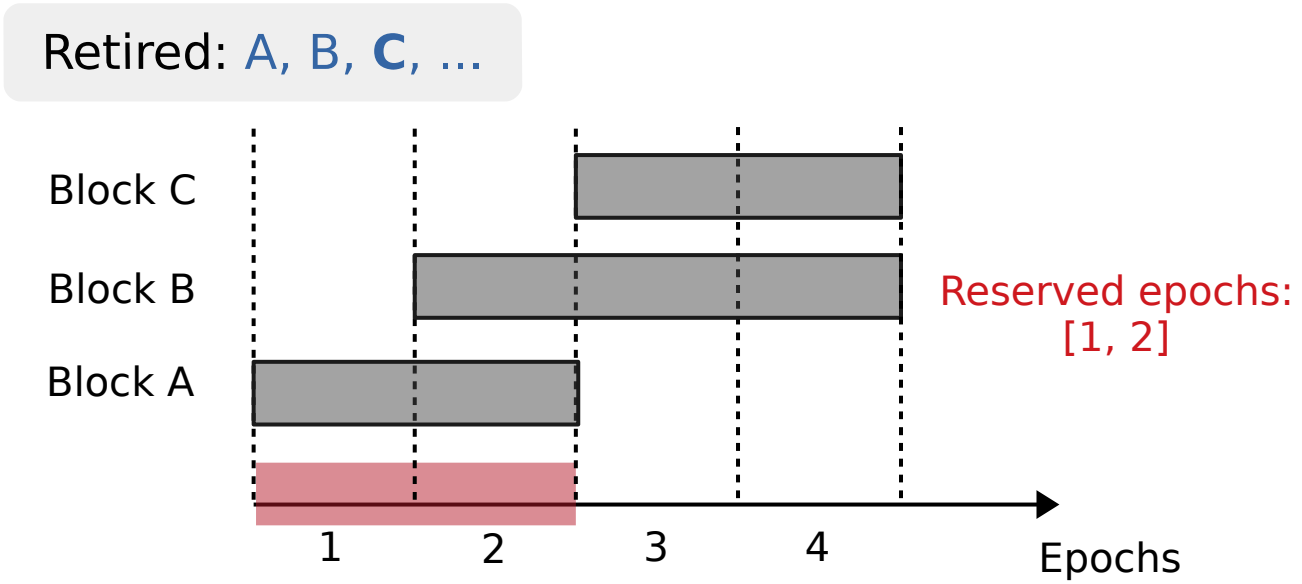
Thoughts about EBR

- EBR is **not robust** [Dice et al., 2016]: a stalled thread can end up reserving an unbounded number of blocks, including blocks created after it stalled.
- If reservation of one thread can only hold a bounded **range** of epochs, then a stalled thread can only reserve a finite number of blocks.
- To ensure correctness, a block should be reserved if its "life **interval**" ("lifetime" between its **birth epoch** and **retire epoch**) intersects with any reservation(s).

Introducing Interval-Based Reclamation (IBR)

Interval-Based Reclamation (IBR)

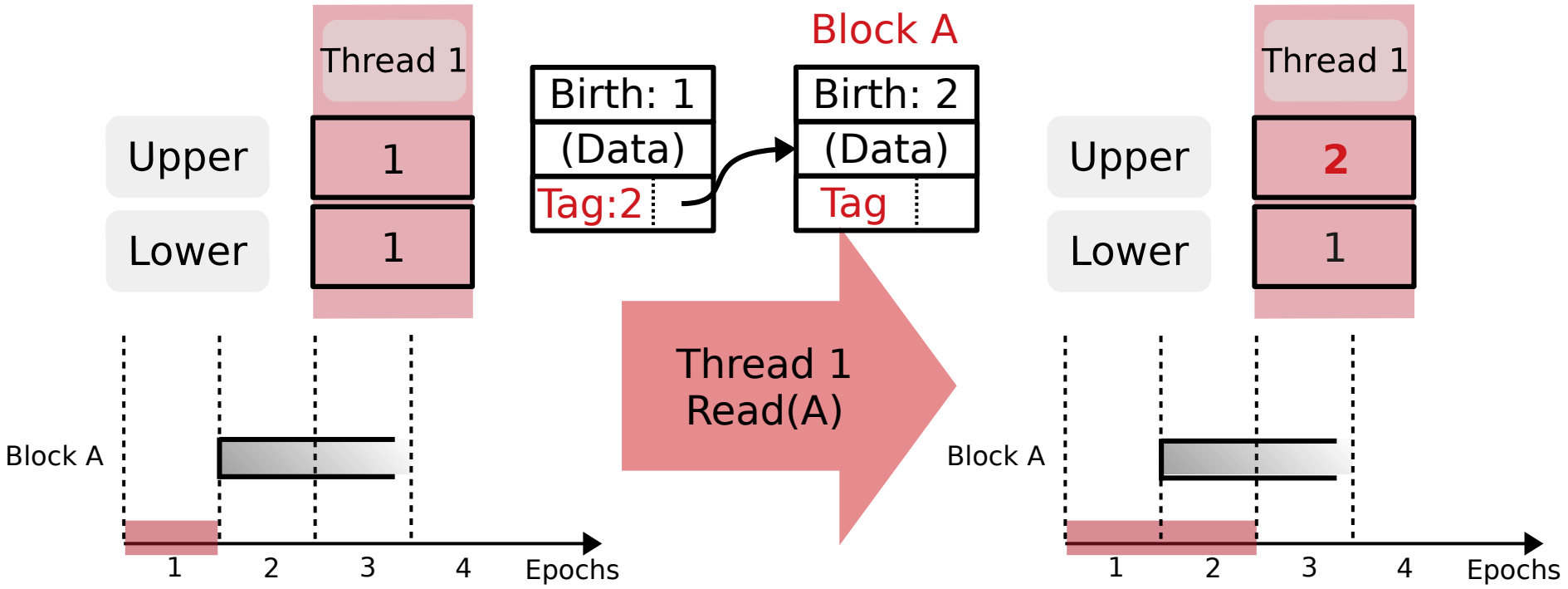




- IBR tracks the life interval (hence the name) of all blocks.
- A block is reclaimable if its life interval does not intersect with reservations of any thread.
- The reservation of each thread contains a finite range of epochs; a stalled thread won't reserve any block born after the upper bound of its reservation.
- A thread updates its upper reservation as it progresses.

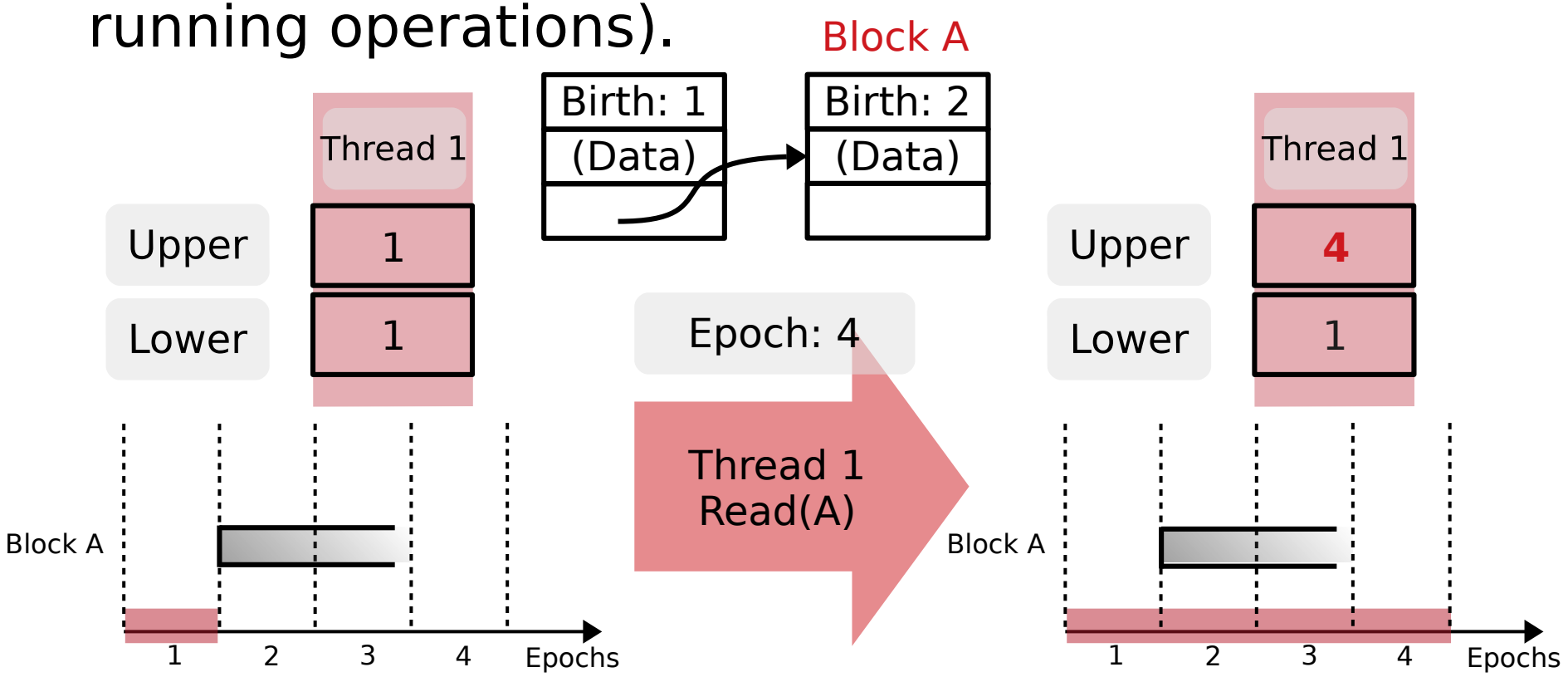
Tagged Pointer IBR (TagIBR)

- Update reservations when following shared pointers. **Goal: reserve the target block before pointer dereference**.
- A **tag** in the pointer is guaranteed to be greater than or equal to the birth epoch of its target.



2 Global Epoch IBR (2GEIBR)

- Always update upper reservations to the current global epoch – faster (or simpler*).
- There is a potential trade-off between space bound and throughput (or simplicity*) (in long-running operations).



*with different TagIBR variants.

Persistent Object IBR (POIBR)

- The most straightfarward implementation of IBR: every thread can only reserve one epoch.
- Suitable only for data structures who persists histories. For example, one whose internal pointers are immutable.

Performance Results

Experimental Setup

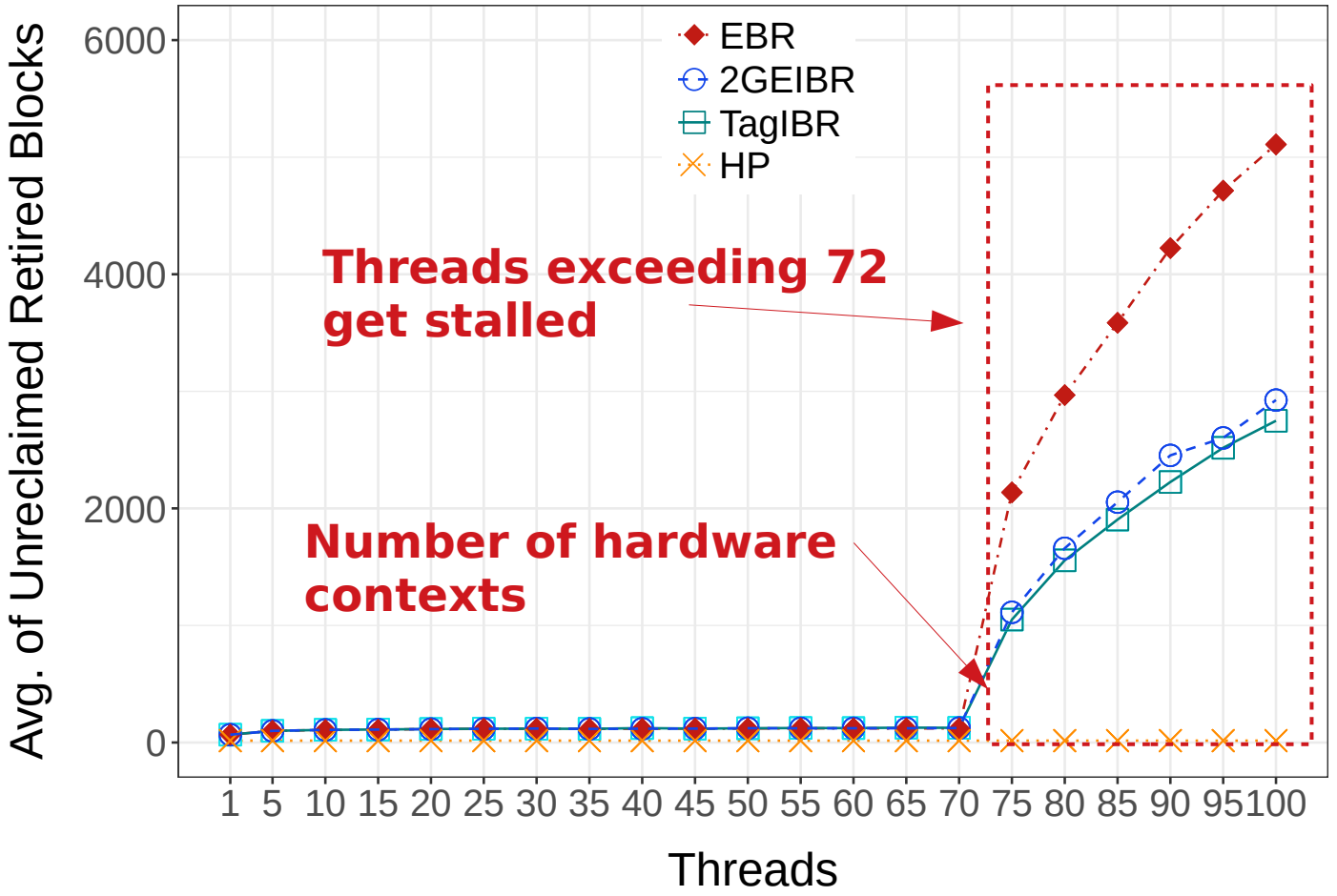
- Platform: Intel(R) Xeon(R) CPU E5-2699 v3.
- Processor: 2 sockets, 18 cores,
  2 hyperthreads on each core:
  72 hyperthreads in total.
  (Threads >72, some get stalled)
- Thread pinning strategy:
  1 thread per core on one socket ->
  hyperthreads on the same socket ->
  next socket.

Schemes in the test

- HP: Hazard Pointers
- EBR: Epoch-based reclamation
- TagIBR
  - (sub-variants: TagIBR-FAA, TagIBR-WCAS in paper.)
- 2GEIBR: 2 Global Epoch IBR
- No MM
- (POIBR in paper)

Average retired-but-not-reclaimed objects per operation

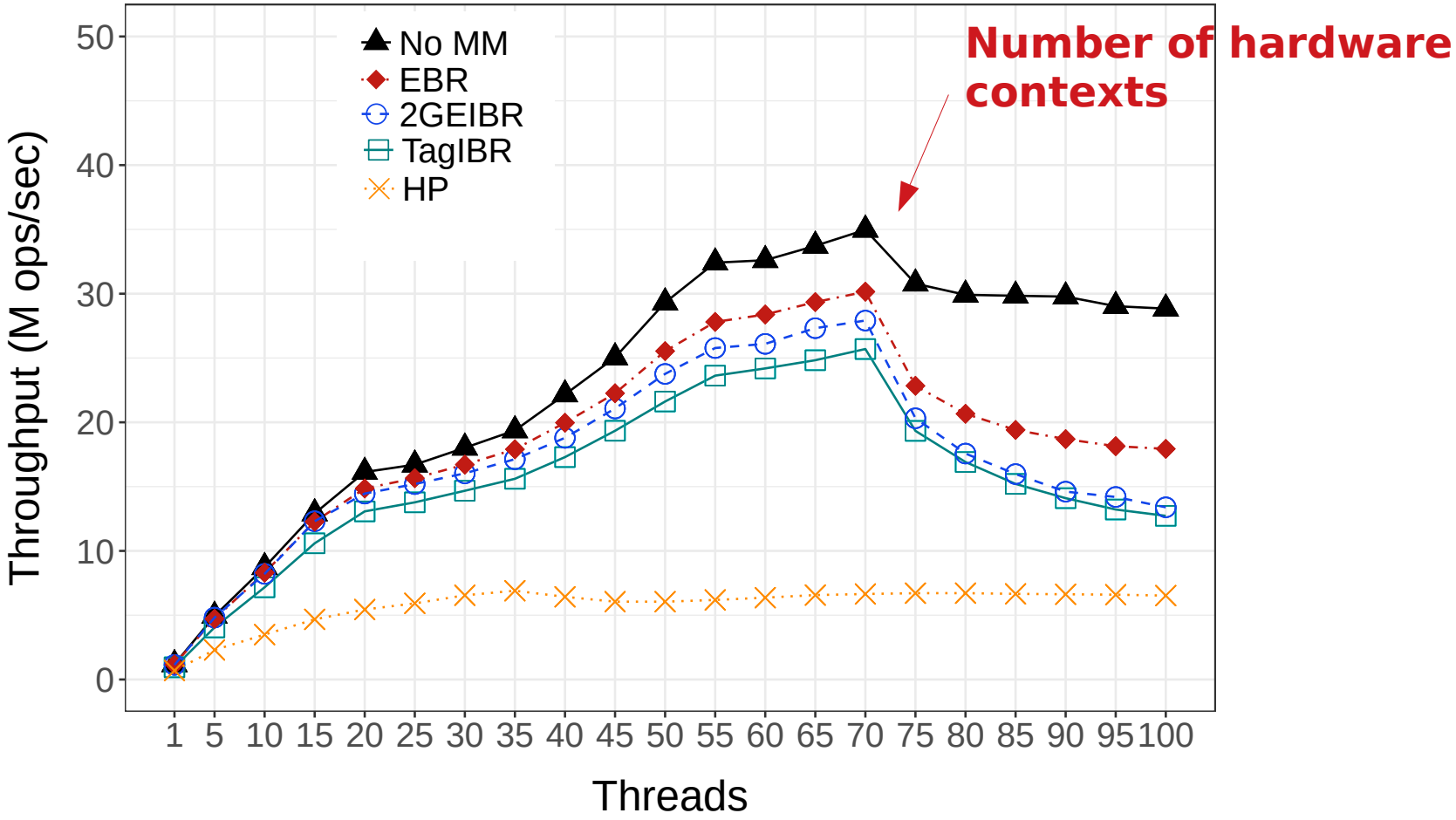
Natarajan & Mittal's Tree



- Michael's Hash Map has similar performance

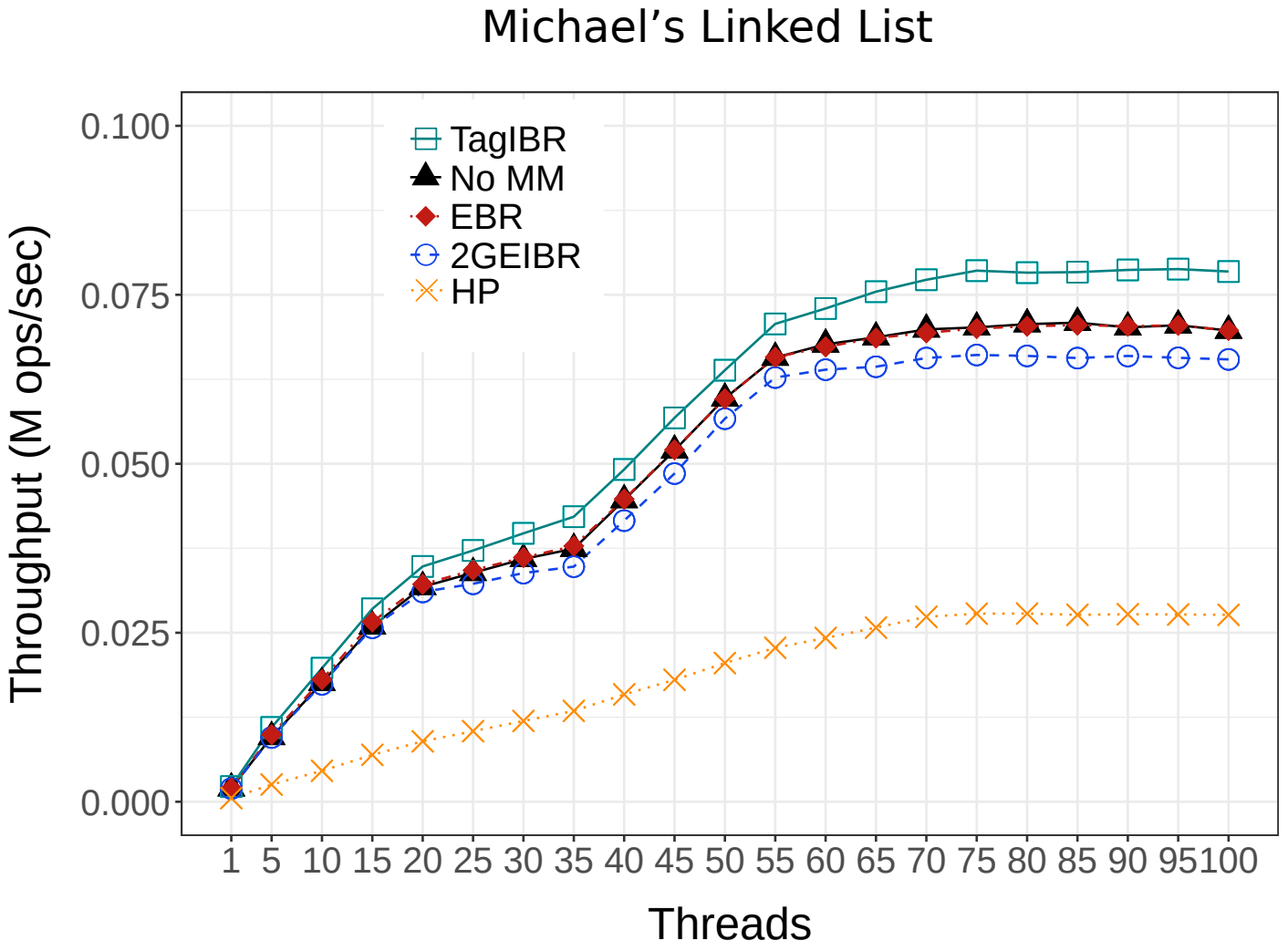
Throughput (M ops/s)

Natarajan & Mittal's Tree



- Michael's Hash Map has similar performance

Throughput (M ops/s)



UNIVERSITY of ROCHESTER

Summary

- We presented Interval-Based Memory Reclamation, a family of memory management schemes for non-blocking concurrent data structures.
- These showed throughput comparable to the fastest existing approach(es), and are robust to thread stalling.
- In theory, TagIBR is more suitable for data structures with long operations working on old data; 2GEIBR for (almost) the rest.
- The artifact is available at: https://zenodo.org/record/1168572