PennState

Brief Announcement: Recyclable Optimistic-Traversal Data Structures

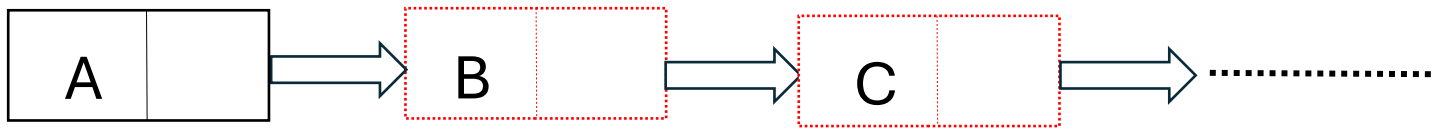
Md Amit Hasan Arovi, arovi@psu.edu, The Pennsylvania State University, USA

Ruslan Nikolaev, rnikola@psu.edu, The Pennsylvania State University, USA

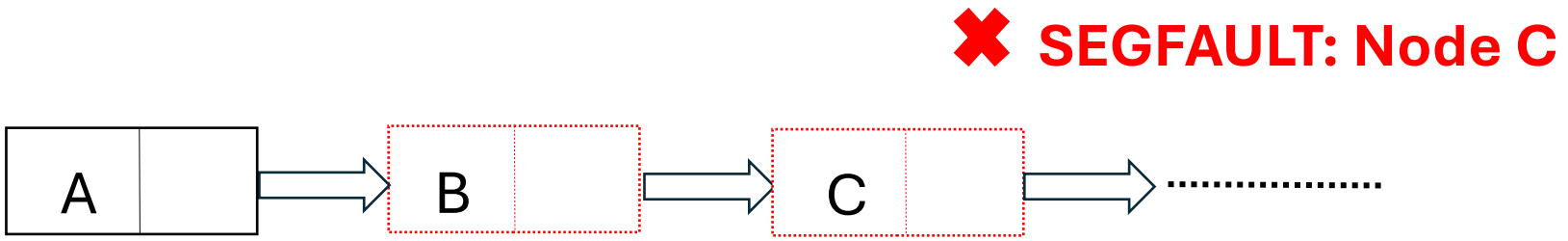
Non-Blocking Data Structures

- Allow multiple threads to operate without mutual exclusion.
- Ensure system-wide progress: at least one thread always makes progress in *lock-free* data structures.
- But they require memory reclamation techniques: Epoch-Based Reclamation (EBR), Hazard Pointers (HP), etc.
  - EBR is fast and easy-to-use but has unbounded memory usage
  - HP is more difficult to use

Optimistic Traversal: Harris' linked list



Optimistic Traversal: Harris' linked list



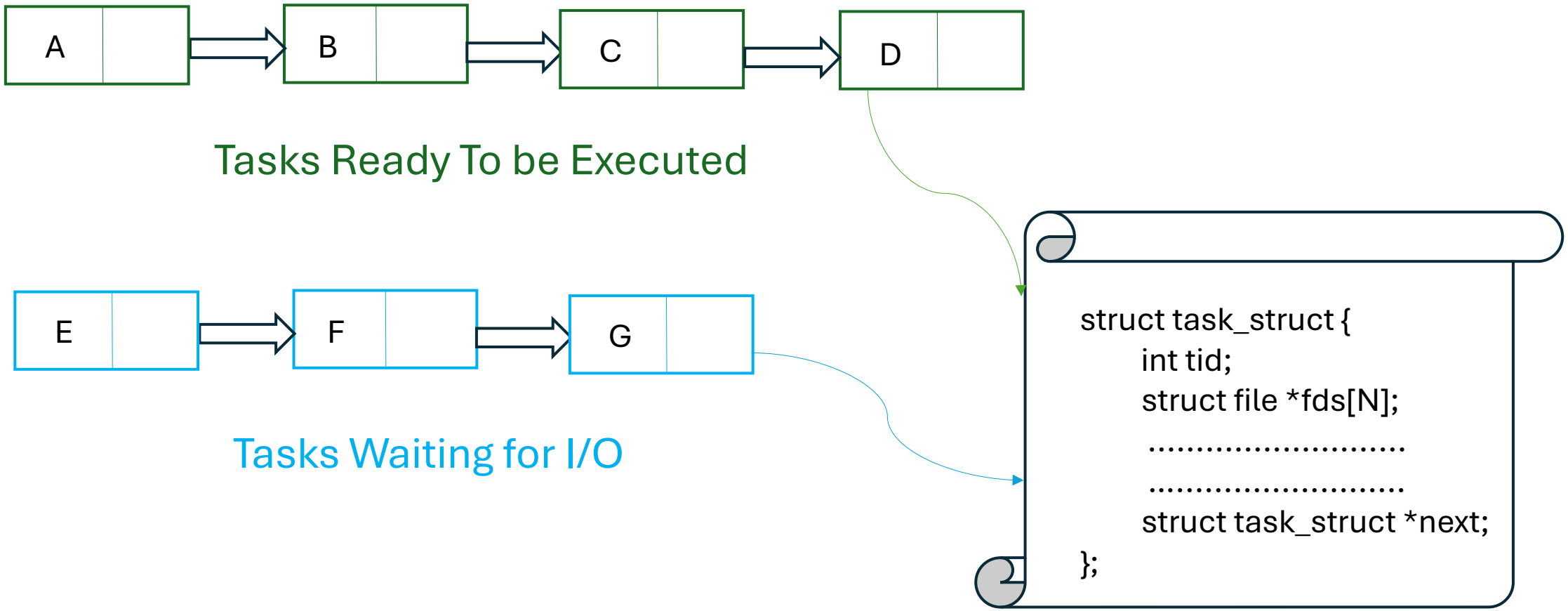
Optimistic Traversals

- Many HP-like **robust** memory reclamation schemes historically failed to support **optimistic traversals** used in many lock-free algorithms:
  - Harris' linked list (had to use Michael's modification)
  - Natarajan-Mittal tree
  - Many others (skip lists, hash tables, etc.)
- **Safe Concurrent Optimistic Traversals (SCOT)** [SPAA '25 BA, PPoPP '26] enables optimistic traversals with many robust schemes:
  - Hazard Pointers [TPDS '04]
  - Hazard Eras [SPAA '17]
  - IBR [PPoPP '18]
  - Hyaline-1S [PLDI '21]

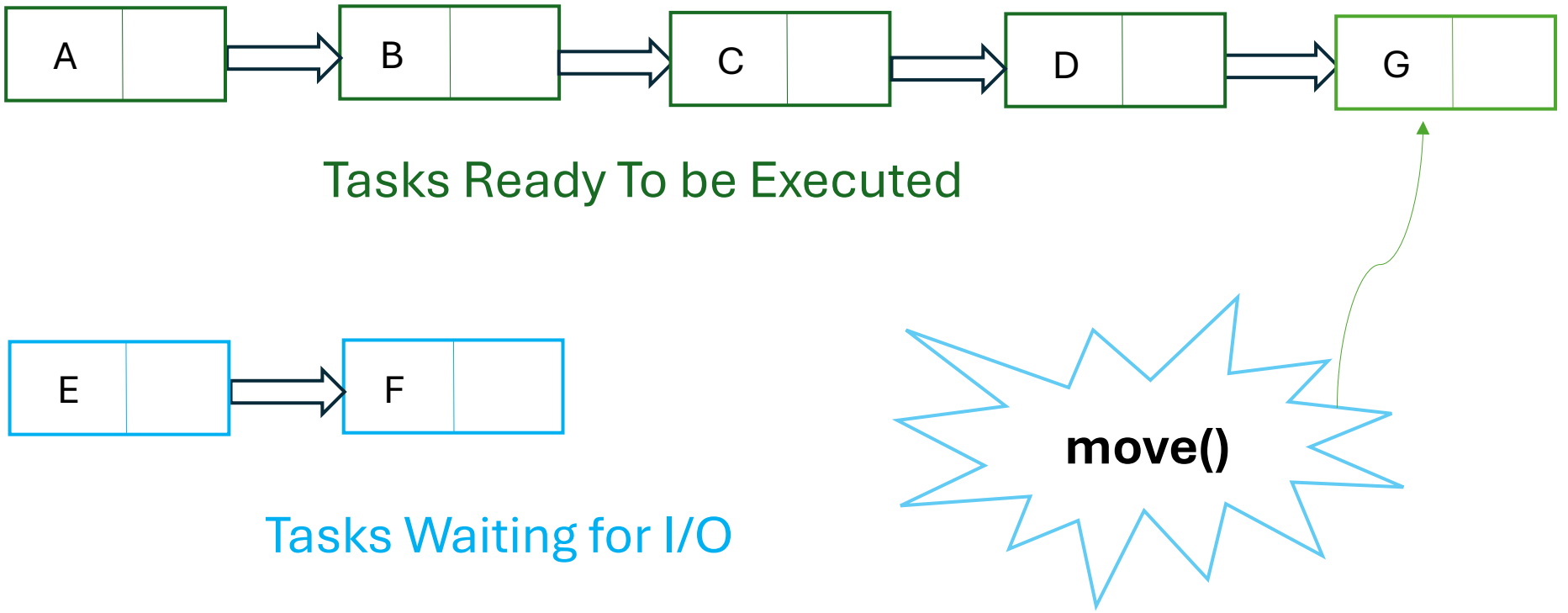
Contribution: Recyclable data structures with optimistic traversals (R-SCOT)

- **RRR-SMR** [PLDI ’25] enables safe node reuse, e.g., an object can be moved from one data structure to the other one.
  - But it implemented all linked lists *without optimistic traversals* (using Michael’s approach) and did not consider the Natarajan-Mittal tree for robust schemes.
- **R-SCOT** combines both SCOT and RRR-SMR and solves additional challenges for optimistic traversals.

Recycling Example



Recycling Example



Among other things, we must keep adjacent ABA tags to every pointer.

R-SCOT: Key Challenge: Tag changes are ambiguous

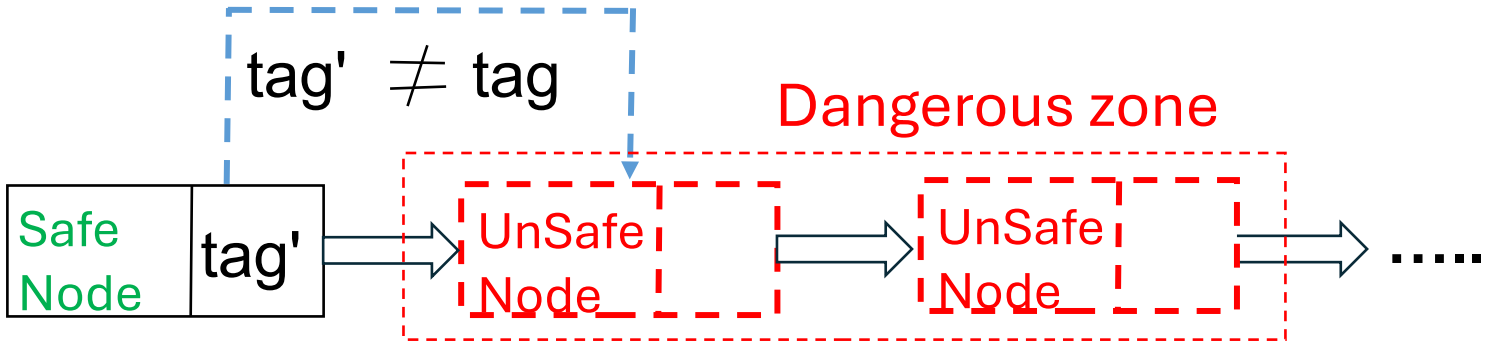
In recyclable Harris' linked list, a tag change may indicate either:

- Logical deletion, or
- Benign concurrent update.

In SCOT, we must validate whether the last *safe* node still points to the first *unsafe* node:

- Treating **every** tag change as deletion causes unnecessary restarts.

R-SCOT: Tag mismatch does not always mean Deletion



Previous tag does not match: Was the Safe Node moved to a different list?

Naive rule:

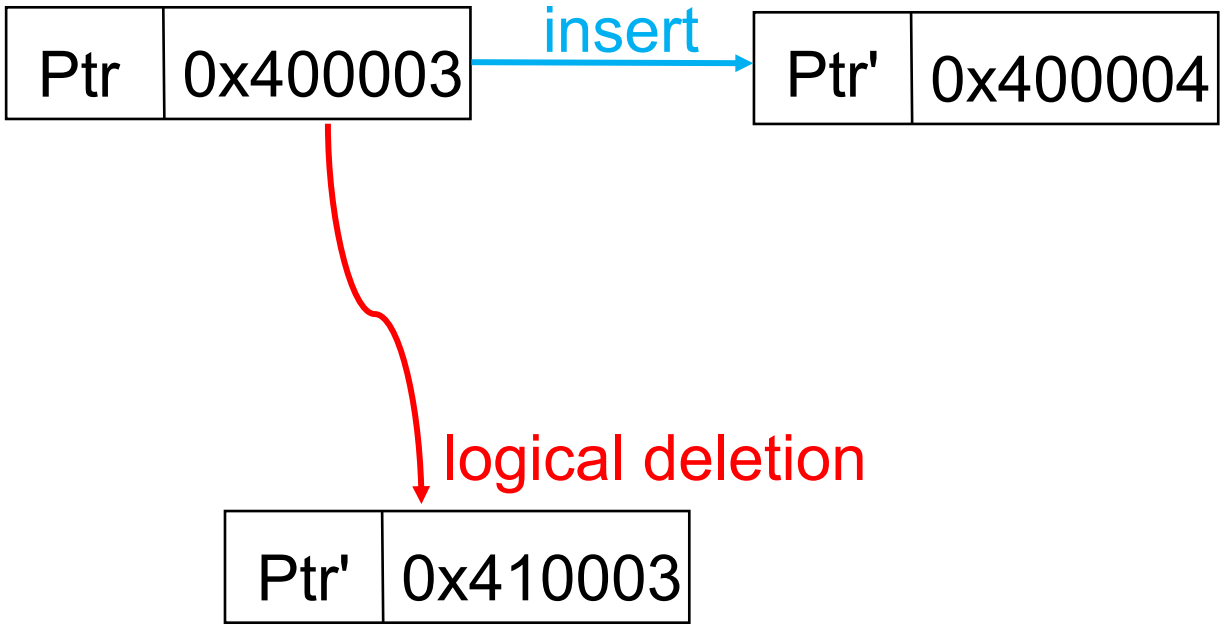
if tag changed:
restart

But this is too conservative:

- Insertion after the safe node can change the tag
- These are benign updates, not necessarily deletion

Treating every mismatch as deletion causes unnecessary restarts.

R-SCOT: Key Idea: Separate Deletion from benign Updates



tag difference < LDEL:

benign update

recover locally from the safe node

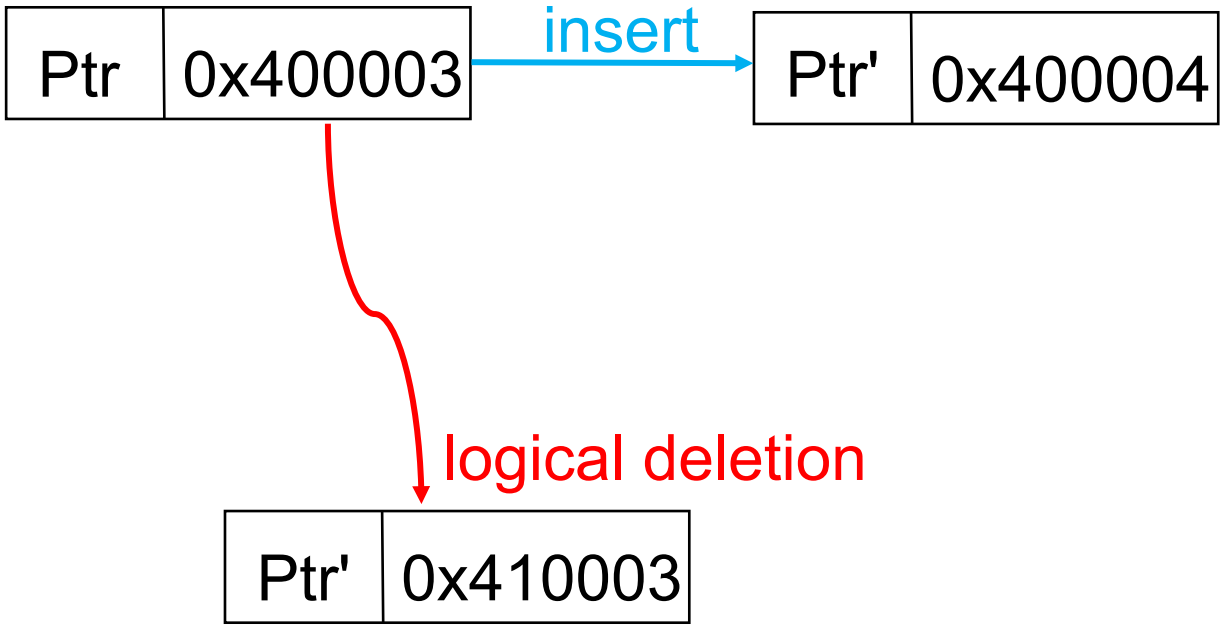
tag difference ≥ LDEL:

likely logical deletion

restart from the beginning

This re-enables the performance benefit of optimistic traversal in the recyclable setting.

R-SCOT: Key Idea: Separate Deletion from benign Updates



tag difference < LDEL:

benign update

recover locally from the safe node

tag difference ≥ LDEL:

likely logical deletion

restart from the beginning

LDEL must be large enough to avoid false restarts, but small enough to preserve ABA safety. We use LDEL = 2^16 as a reasonable compromise.

Recyclable Natarajan-Mittal Tree: with Robust SMR

- RRR-SMR previously demonstrated recyclable Natarajan-Mittal tree with EBR.

- Unlike EBR, robust SMR schemes require that every reclaimable object is explicitly protected.

Recyclable NMTree uses multiple object types:

- regular tree nodes
- immutable blocks

Traversal must carefully reserve and update protection for both nodes and blocks. The main challenge is implementation complexity, not changing the high-level idea.

Evaluation Setup

- Platform: AMD EPYC 9754, 128-core, 256 threads, 384 GiB of RAM.

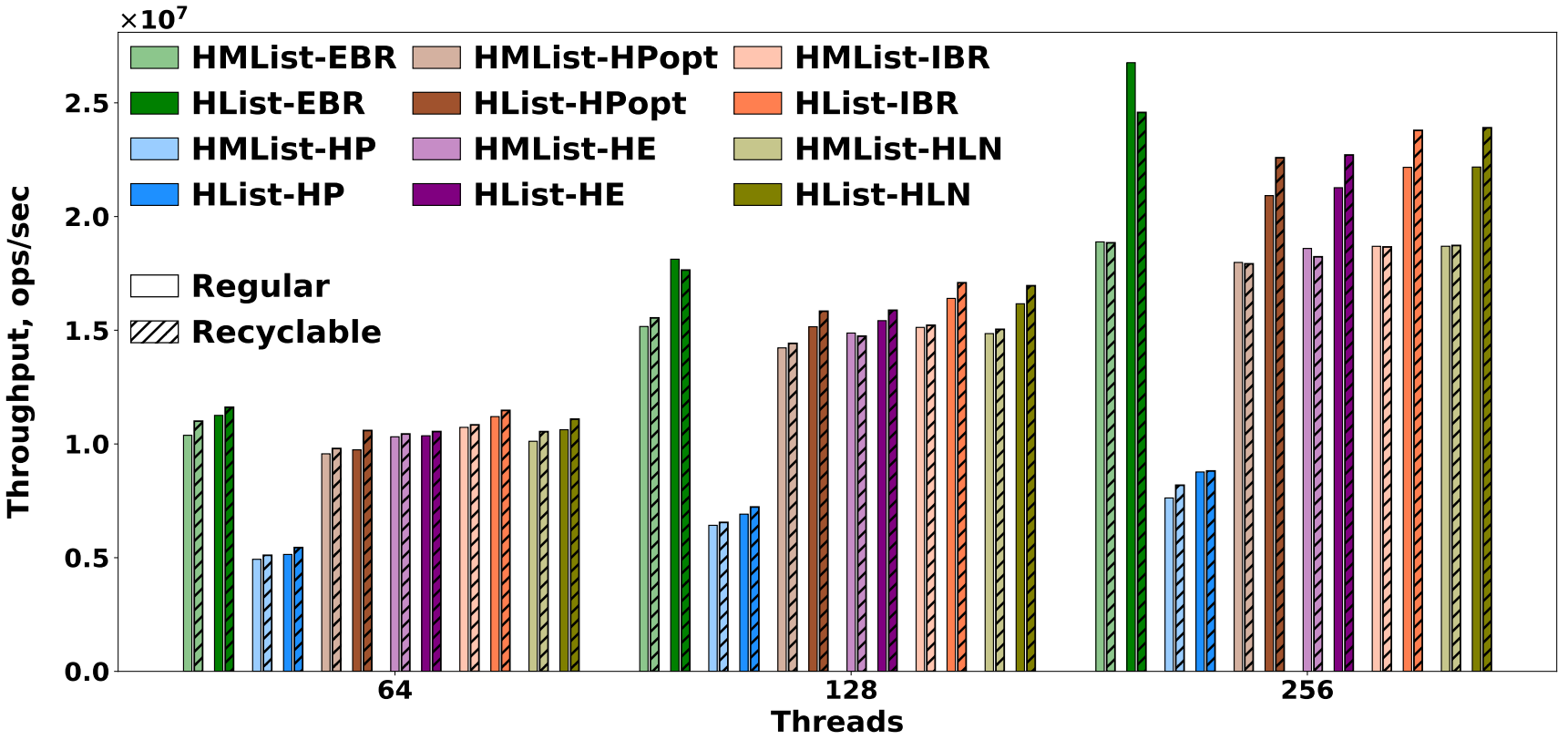
- Data Structures: Harris' Linked List (HList), Harris-Michael Linked List (HMList), and the Natarajan-Mittal Tree (NMTree).

- SMR Schemes: EBR, HP, HPopt, HE, IBR, and Hyaline-1S.

- Payload: 0B, to isolate the overhead of recyclability support.

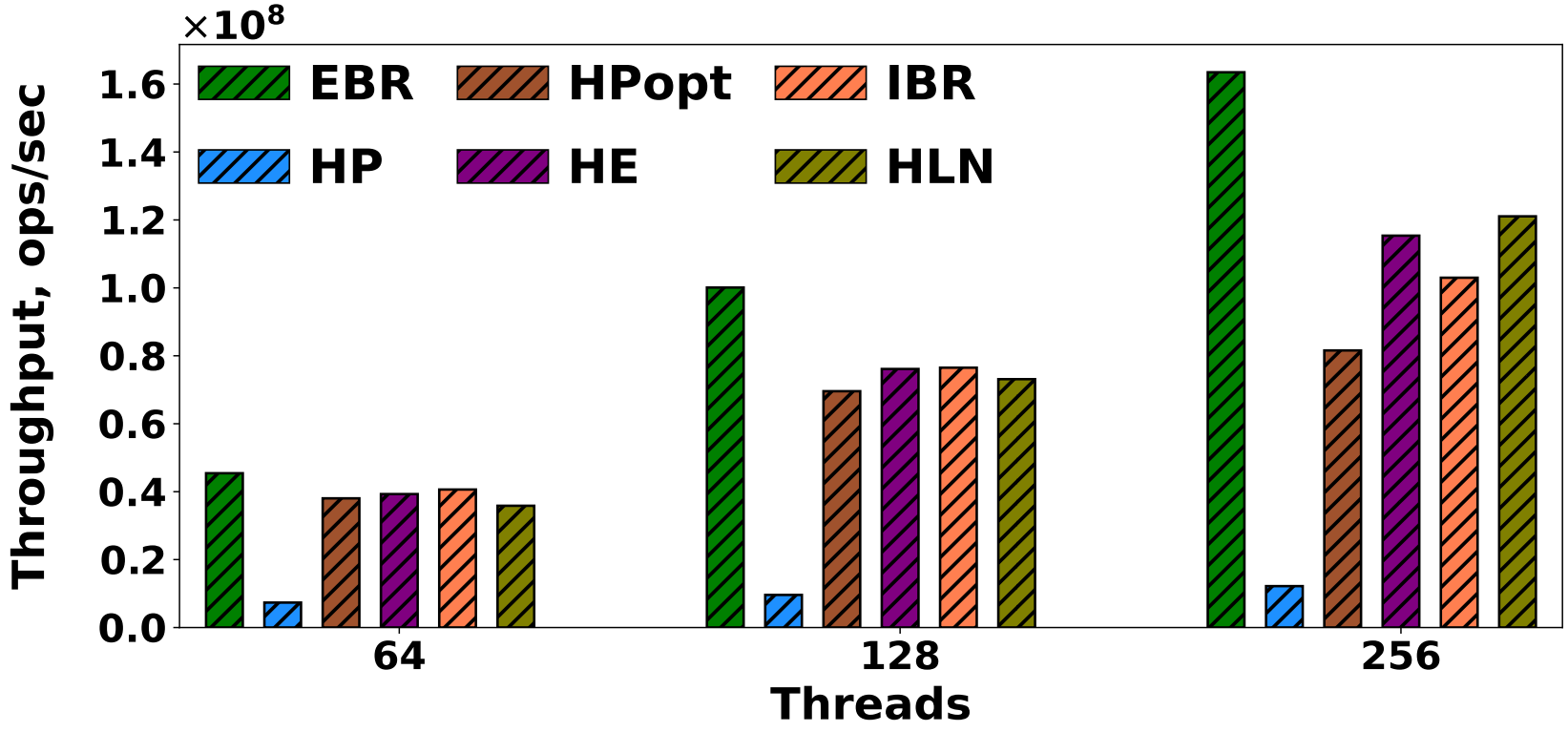
- Measured throughput across 64, 128, and 256 threads.

Evaluation: Harris vs. Harris-Michael list (Throughput)



Key Range = 2048, 10% recycling

Evaluation: The Natarajan-Mittal tree (Throughput)



Key Range = 65,536

Takeaways

- For **recyclable** data structures, the traditional distinction between "**easy**" EBR and "**hard**" HP no longer exists:
  - Both require SCOT-like techniques for optimistic traversals.
  - Both need LDEL optimization for better performance.

Code Availability

- The code repository:

https://github.com/rusnikola/r-scot

Code Availability

- The code repository:

https://github.com/rusnikola/r-scot

THANK YOU! QUESTIONS?