Neutralization Based Reclamation (NBR)

Ajay Singh, Trevor Brown and Ali Mashtizadeh

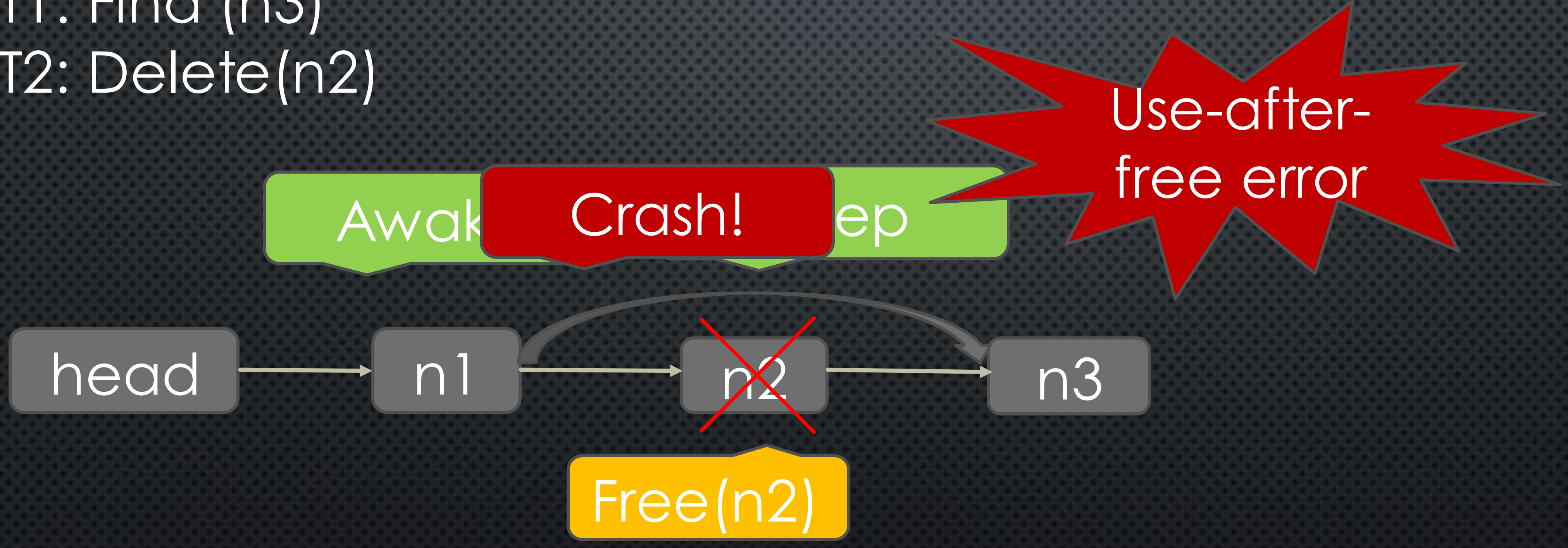
(Multicore Lab, University of Waterloo)

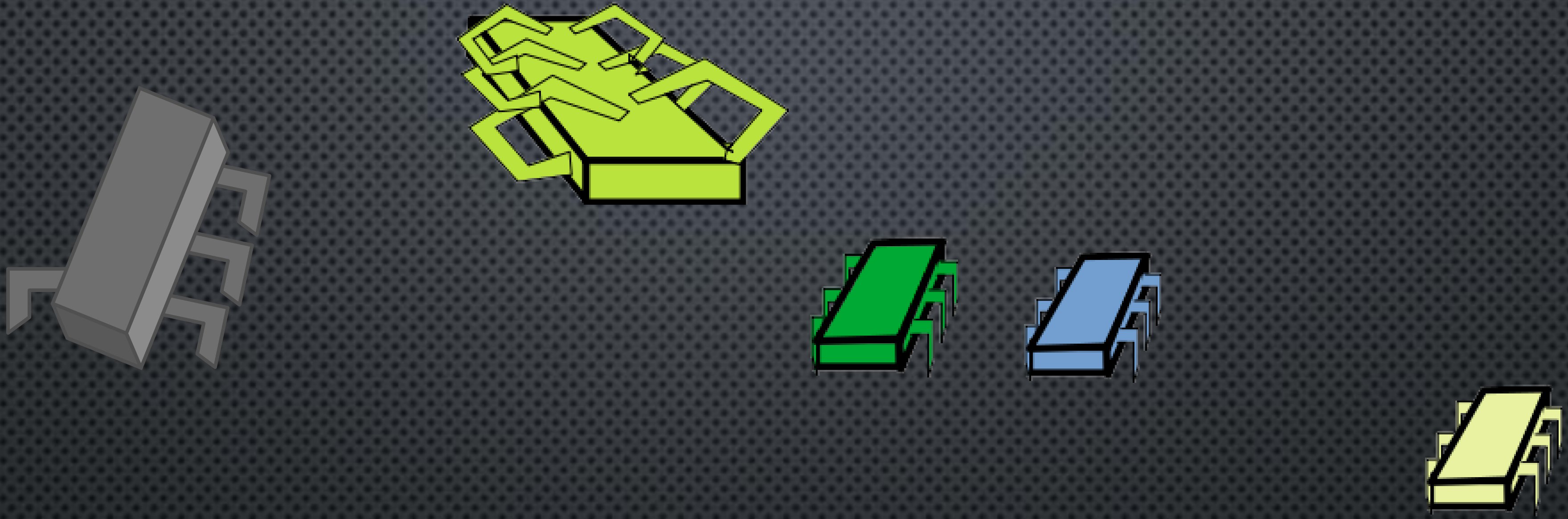
OUTLINE

- What is a safe memory reclamation (SMR) problem?
- Desirable properties in SMR algorithms
- Issues in Existing SMR algorithms
- Our contribution motivated by problems in existing algorithms
  - NBR
  - NBR+ (faster NBR)
- Results
- Conclusion

WHEN IS IT SAFE TO FREE A NODE?

T1: Find (n3)
T2: Delete(n2)

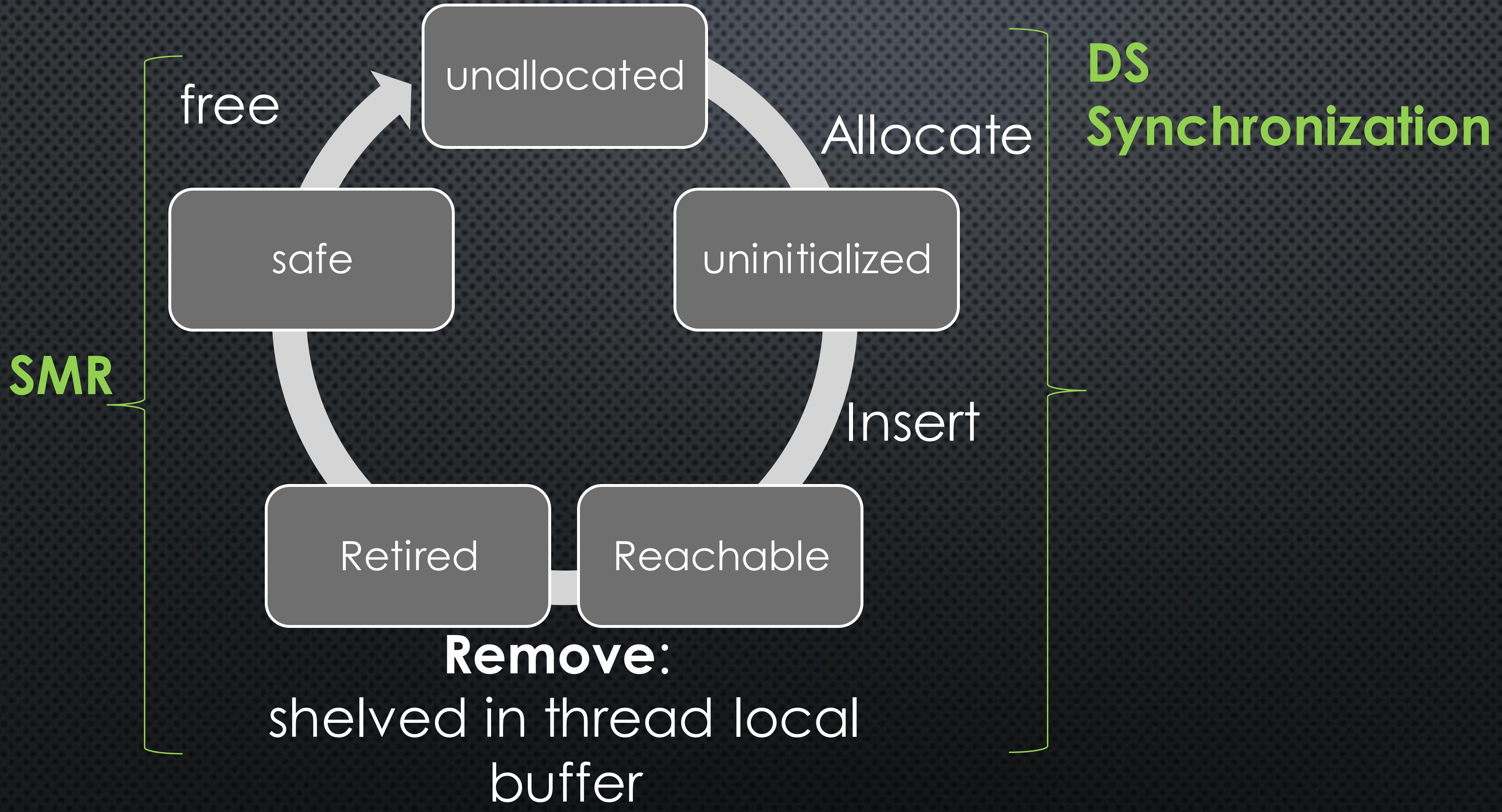




SAFE MEMORY RECLAMATION (SMR)

Safe memory reclamation:
problem of deciding when it is safe to free a record in concurrent data structure using dynamic memory so that use-after-free error do not occur.

LIFECYCLE OF A RECORD IN SMR?



DESIRABLE PROPERTIES IN AN SMR ALGORITHM

- Performance
- Bounded Garbage
- Usable
- Applicable

WHERE EXISTING SMR ALGORITHM STAND IN REGARDS WITH THE DESIRABLE PROPERTIES?

	Reference counting based	**Epoch based**	**Hazard pointer based**
Performance	low	high	**medium**
Bound on garbage	conditional	unbounded	**bounded**
Usability	medium	high	low
Applicability	*	high	**low**

HP doesn't apply to 16/18 data structures surveyed

SELECTED EXAMPLES

HHLMSS05	Lazy linked list
EFRB10	External binary search tree
HJ12	
S13	
NM14	
DVY14	
EFRB14	
BER14	
RM15	
BPA20	External interpolation search tree

MOTIVATION FOR NBR : INTERESTING DESIGN PATTERN OF DATA STRUCTURES

```
void operation()
{
RETRY:
    pred = head;
    curr = pred.next;
    while(key ≤ curr.key) {
        pred=curr;
        curr=cur.next;
    }
    . . .
    lock(pred); lock(curr);
    if (!validate()) {
        unlock(pred); unlock(curr);
        goto RETRY;
    }

    do update
    unlock(pred); unlock(curr);
}
```

restart

can I restart?

Read-phase:

reservation-phase:

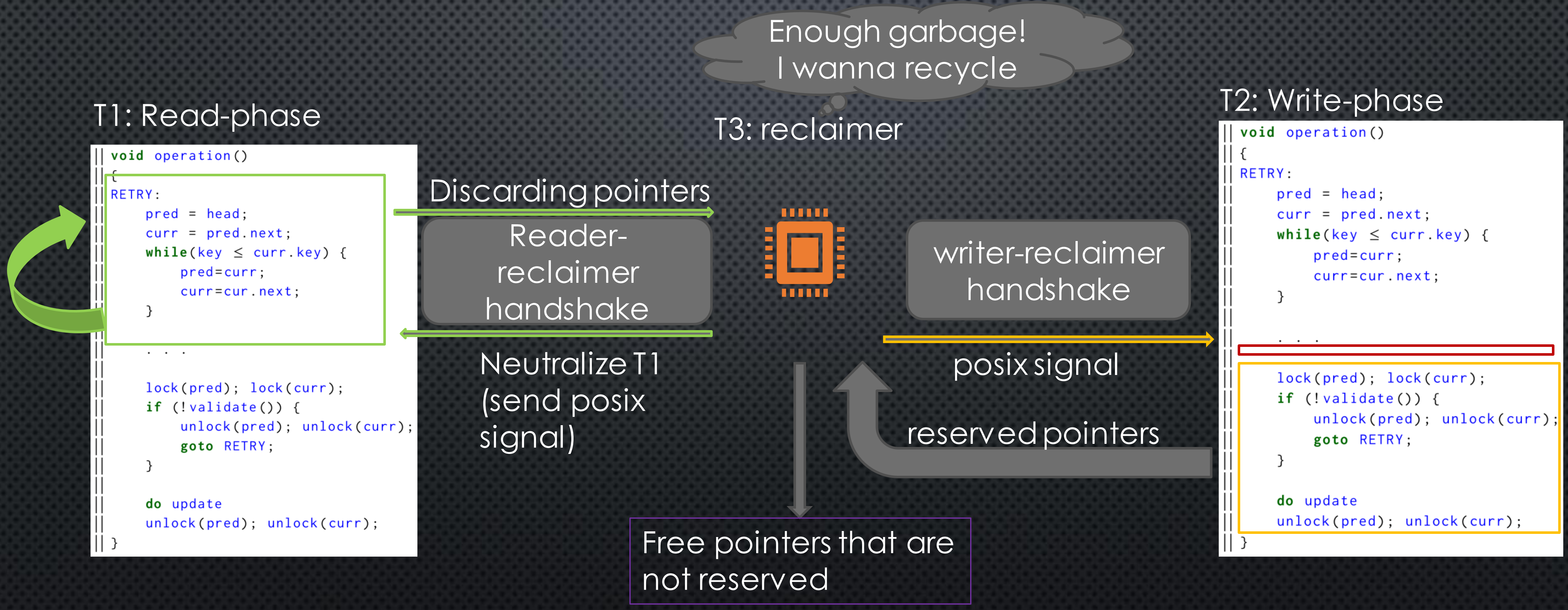
write-phase

Many concurrent data structures have a pattern where long searches are followed by short (optional) updates.

Operations consist of (or can be presented in) two phases:

1. **Read-phase:** threads only read the underlying data structure.
2. **Write-phase:** threads modify the underlying data structure.

NBR: HIGH LEVEL OVERVIEW



PERFORMANCE BOTTLENECK IN NBR

Cost of signals: Every time a thread reclaims it sends POSIX signals to neutralize all other threads.

More wasted work for threads in read phase due to restarts.

Can we do better?

OBSERVATION: IN NBR THREADS CAN PIGGYBACK ON A THREAD ALREADY BROADCASTING SIGNALS

~~Enforced quiescence~~

Records retired before t1 are safe to free

threads either discard pointers and restart **or** do not restart but have reserved pointers.

t1: start signaling

t2: end signaling

ADVANTAGE:

- Less number of signals
- Lower amount of wasted work for readers.
- FASTER NBR

Signal broadcast by a thread is enough for all threads to reclaim their buffers.

NBR+: HIGH LEVEL OVERVIEW

All threads maintain two thresholds *C1* & *C2* in its buffer. (*C1*< *C2*)

At C2, a thread T_J enforces quiescence and reclaims its buffer as in NBR. Additionally, maintains a SWMR timestamp which it increments once at t1 and at t2.

After reaching C1, a thread T_I passively monitors for some T_J that could be sending signals so that it could piggyback on signals sent by T_J to reclaim its own buffer.

EXPERIMENTS

BINARY SEARCH TREE [DGT15]

(A,B) TREE [BROWN17]

LAZYLIST (IN PAPER*)

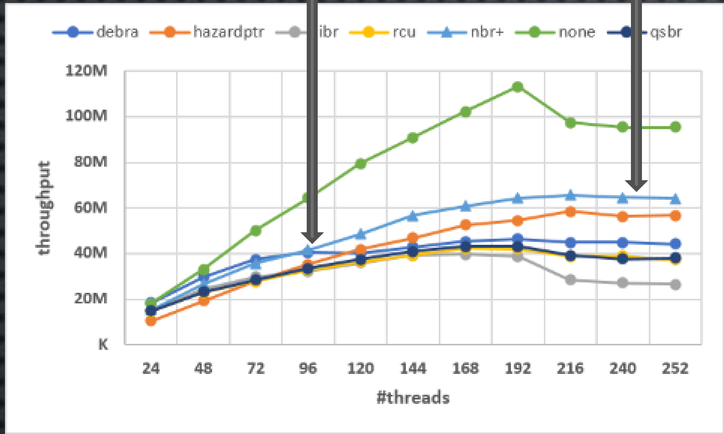
HARRIS MICHAEL LIST (IN PAPER*)

- 4x Intel Xeon Platinum 8160
- 192 hardware threads
- Ubuntu 18.04, g++ 7.4, -O3
- 5 second timed trials

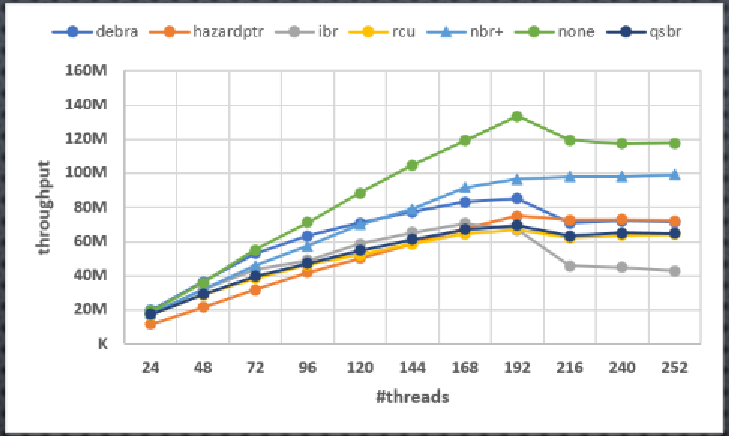
- DEBRA
- HP
- IBR
- QSBR
- RCU

Crosses Debra

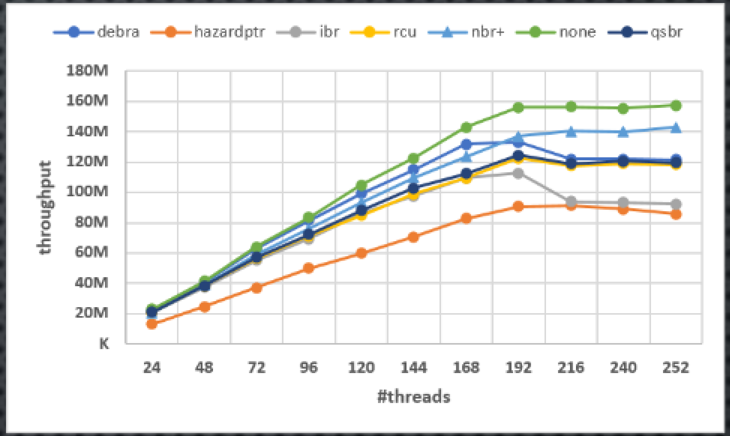
oversubscription



50% inserts – 50% deletes



25% inserts – 25% deletes

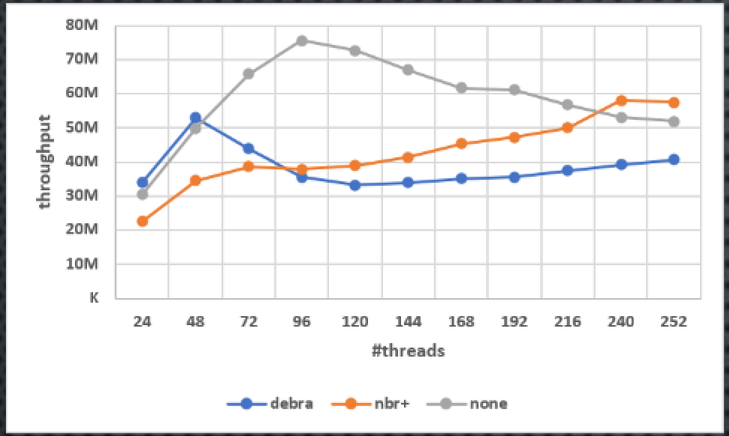


5% inserts – 5% deletes

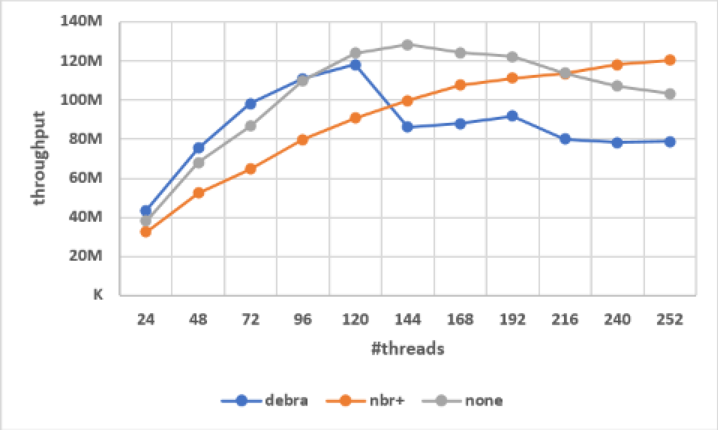
Workload types

Data Structure:

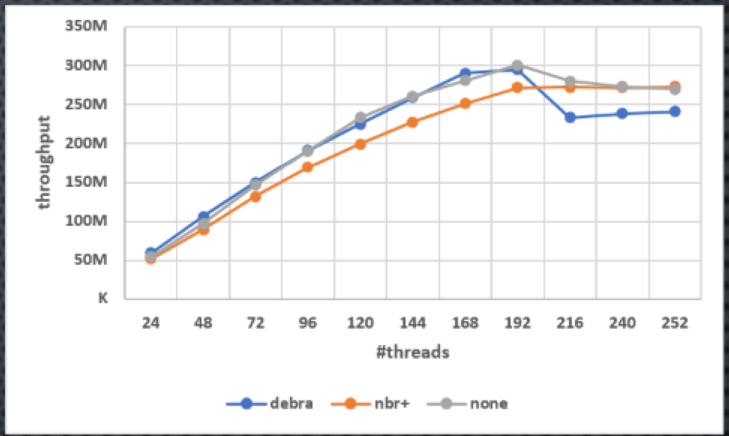
- External Binary search Tree (DGT15)
- Size: 2M keys



50% inserts – 50% deletes



25% inserts – 25% deletes



5% inserts – 5% deletes

Workload types

Data Structure:

- Brown's (a,b)-Tree
- Size: 2M keys

CONCLUSION

- Fast
- Bounded Garbage
- Usable
- Applicable

Data structure	HP	EBR	NBR
Lazy list [Heller et al 2005]	✗	✓	✓
Harris list [Harris 2001]	✓	✓	✓
EFRB BST [Ellen et al 2010]	✗	✓	✓
External (a,b) tree [Brown et al, 2017]	✗	✓	✓
Howey-Joney internal BST [2017]	✗	✓	✓
External chromatic tree [Brown et al, 2014]	✗	✓	✓
External AVL tree [Brown et al, 2017]	✗	✓	✓

TAKE HOME
COOL THINGS I DINT TALK..
THINGS THAT COULD MAKE WANNA PPL READ PAPER.