Extending Hardware Transactional Memory Capacity

via

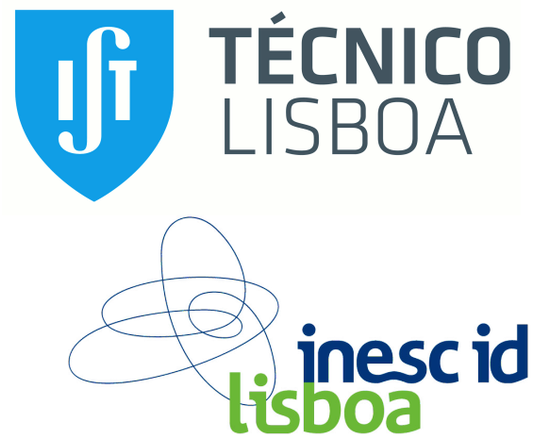
Rollback-Only Transactions and Suspend/Resume

Shady Issa

Pascal Felber

Alexander Matveev

Paolo Romano



Extending Hardware Transactional Memory Capacity

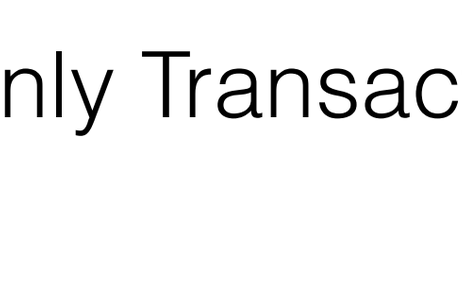
via

Rollback-Only Transactions and Suspend/Resume

POWER8-TM

Shady Issa

Pascal Felber



Alexander Matveev

Paolo Romano

Transactional Memory

- alternative paradigm for parallel programming
- easy to use
- potential of fine-grained locking performance

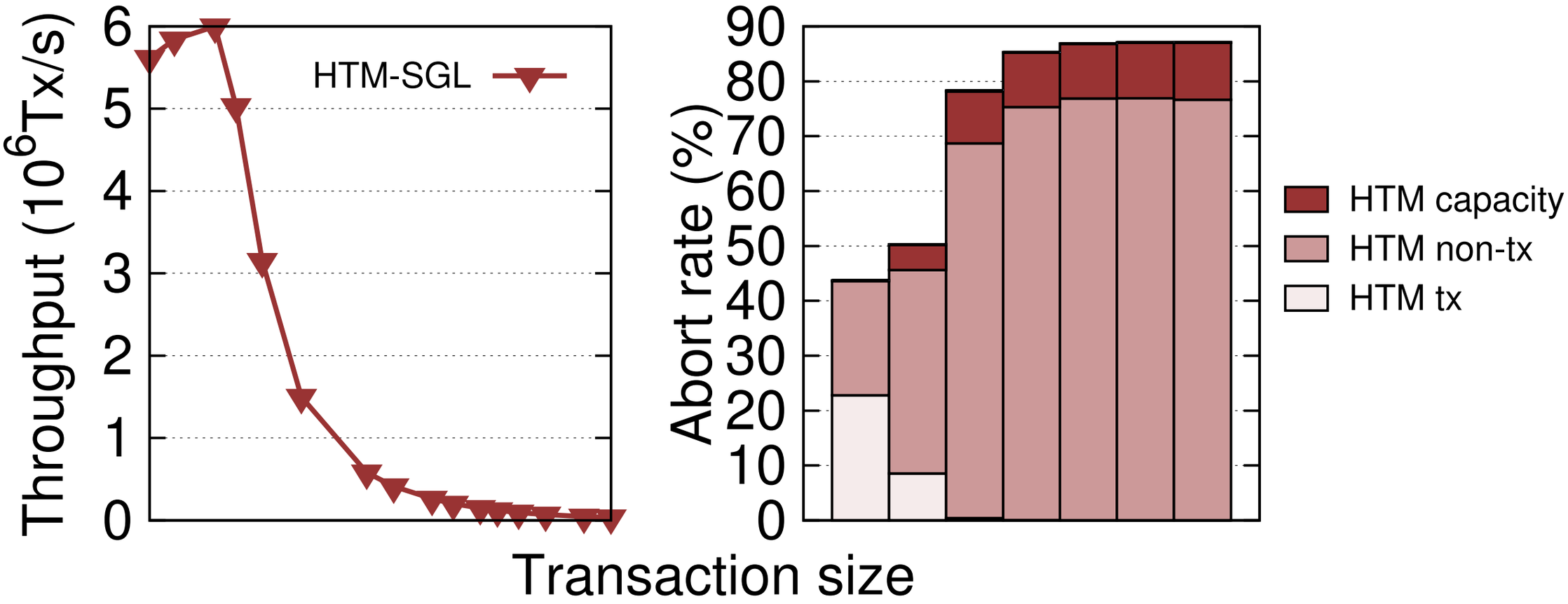
```
withdraw(account, value){
__transaction{
    if account.balance > value:
      account.balance -= value;
      return account.balance;
    else
      return -1;
  }
}
```

Transactional memory implementation

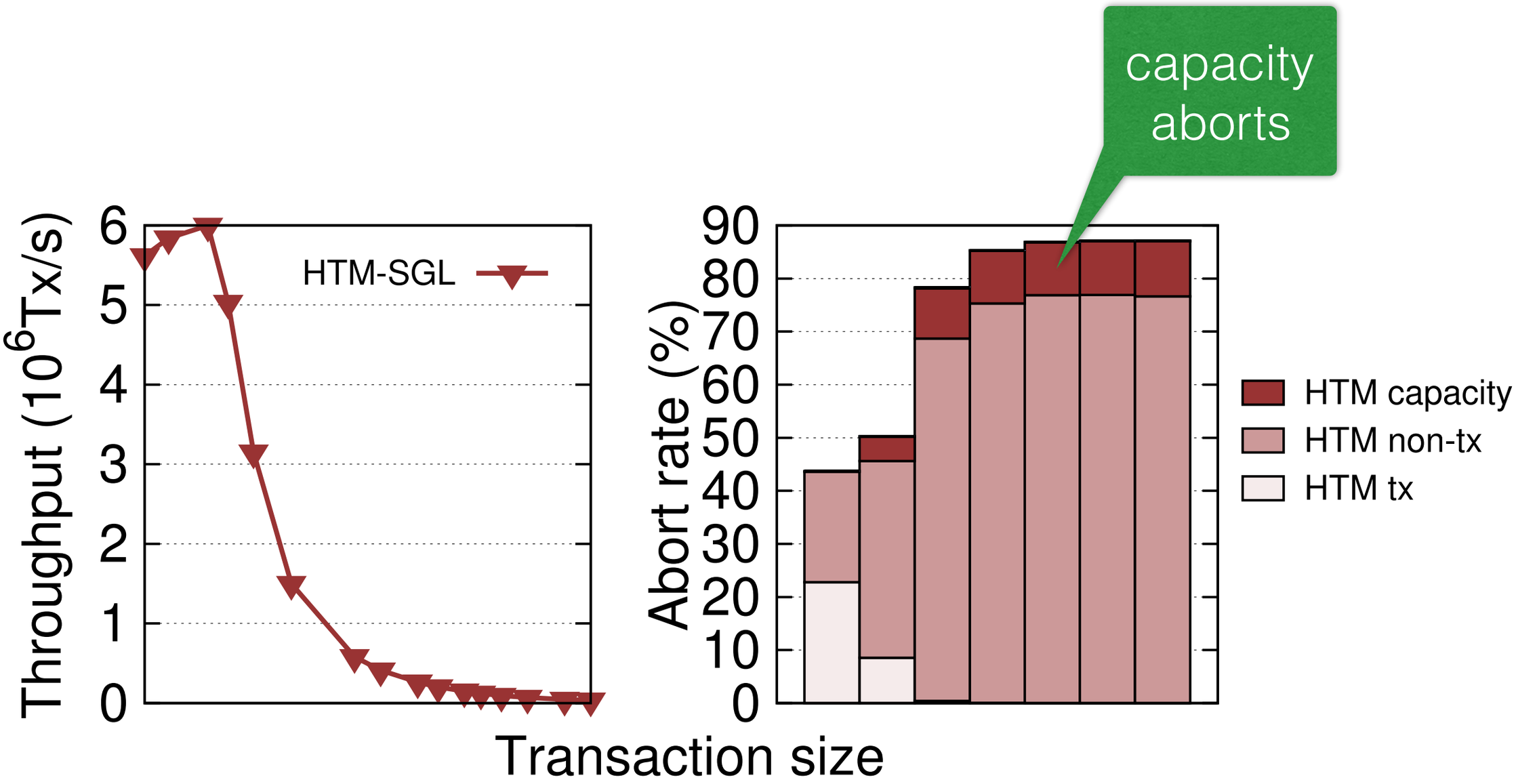
Hardware Transactional Memory

- Intel and IBM processors
- implemented in the cache coherence protocol
- cache line granularity
- best effort
  - S/W fallback is needed

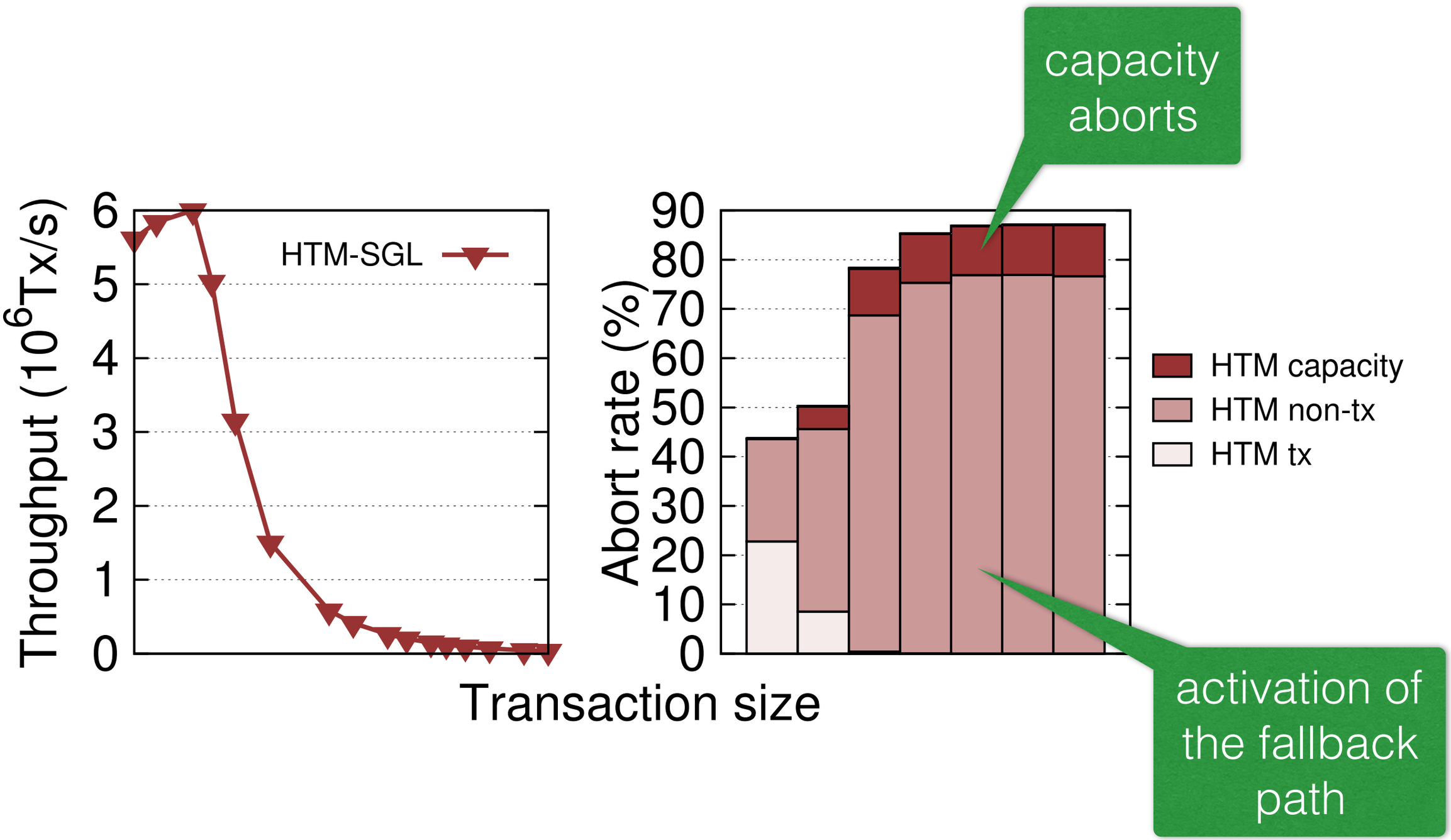
Capacity Limitations



Capacity Limitations



Capacity Limitations



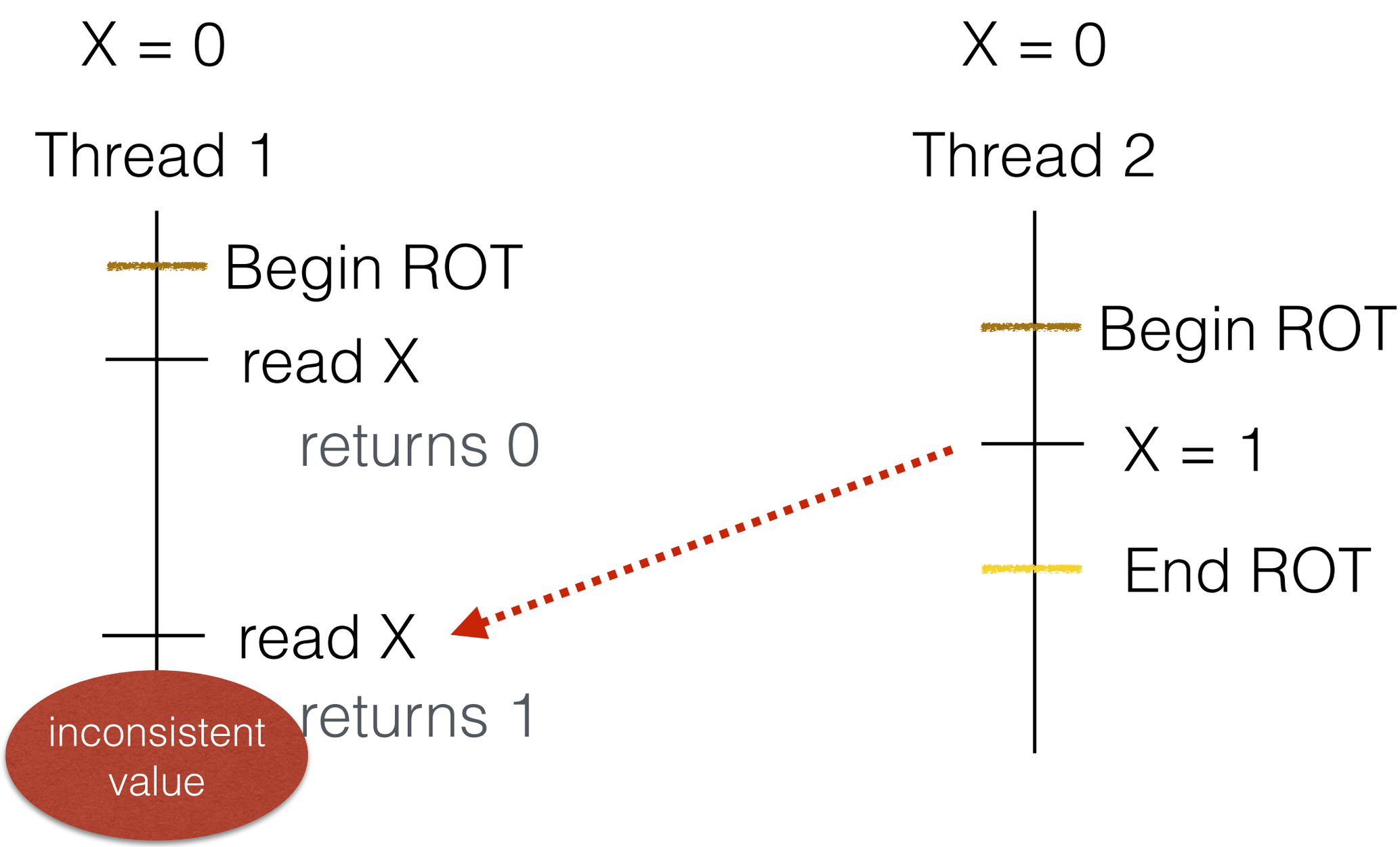
POWER8-TM

- hardware/software co-design
- utilises specific features available in POWER8:
  - suspend/resume
  - ROTs
- to support execution of larger transactions

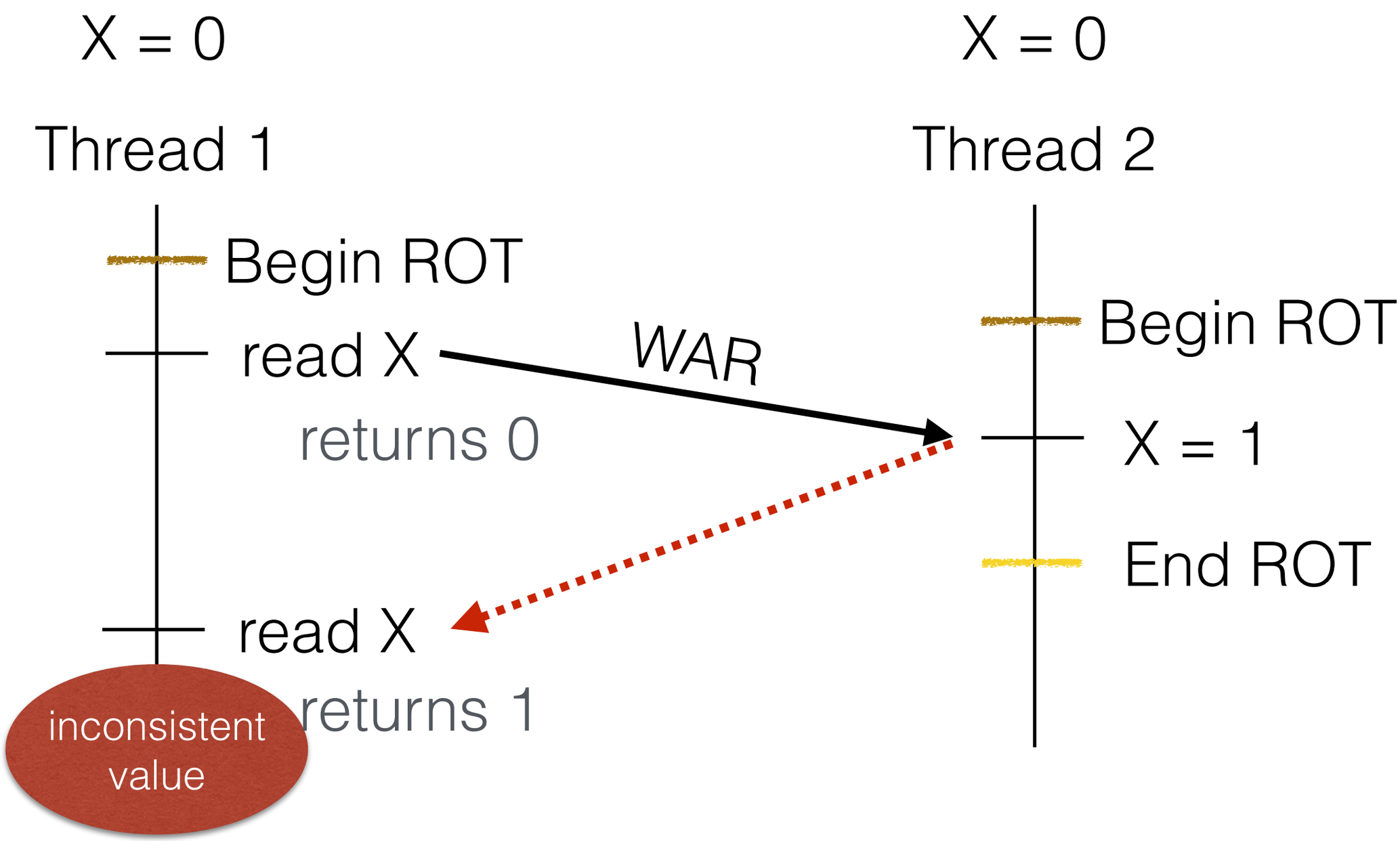
Rollback-only Transaction

- lightweight transaction type
- updates are applied atomically
- does not track the reads
  - theoretically infinite read-set
- not **serialisable**

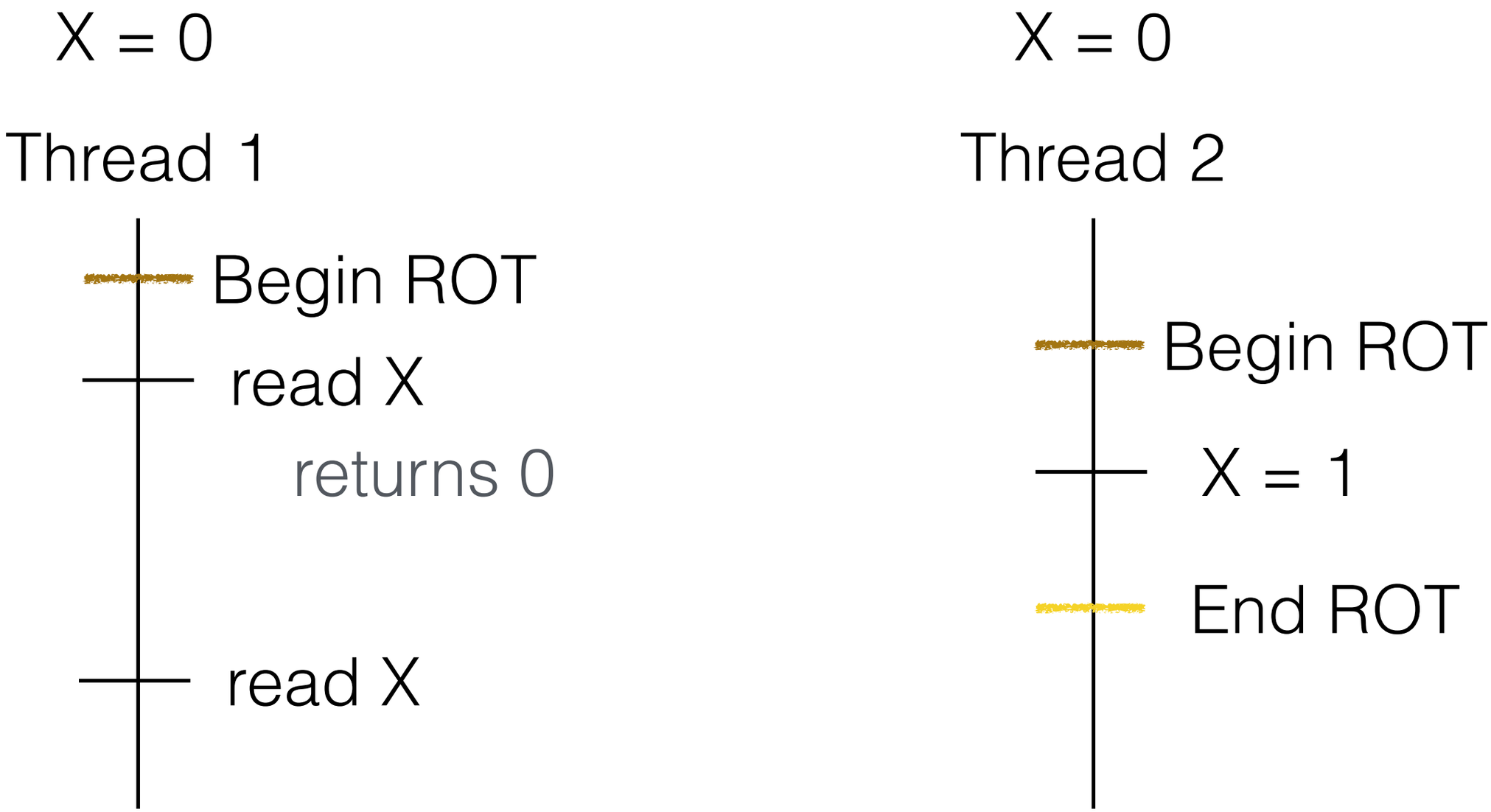
ROTs


X = 0
X = 0
Thread 1
Thread 2
Begin ROT
read X
returns 0
Begin ROT
X = 1
End ROT
read X
returns 1
inconsistent value

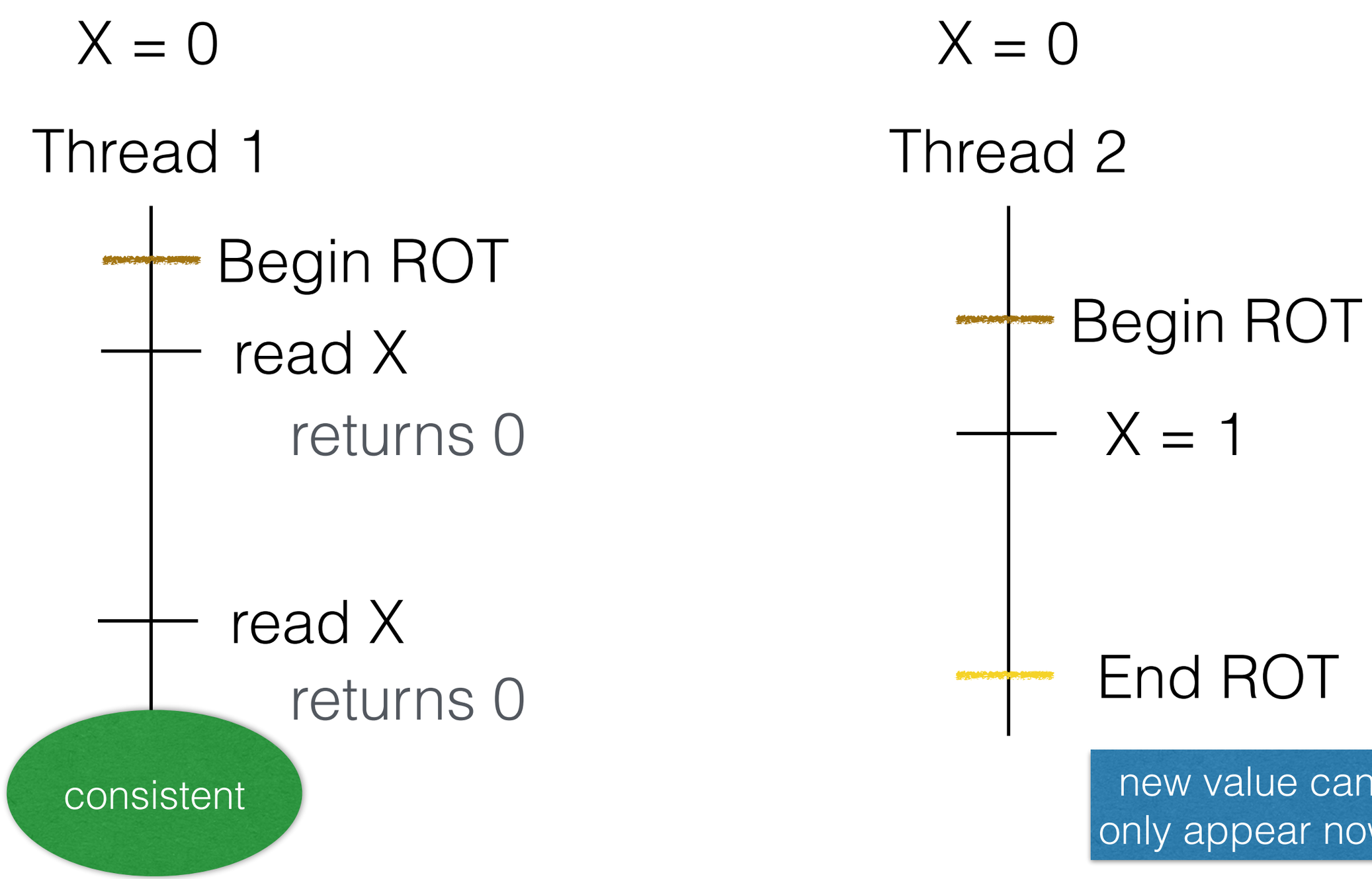
ROTs


X = 0
X = 0
Thread 1
Thread 2
Begin ROT
Begin ROT
read X
WAR
returns 0
X = 1
End ROT
read X
returns 1
inconsistent value

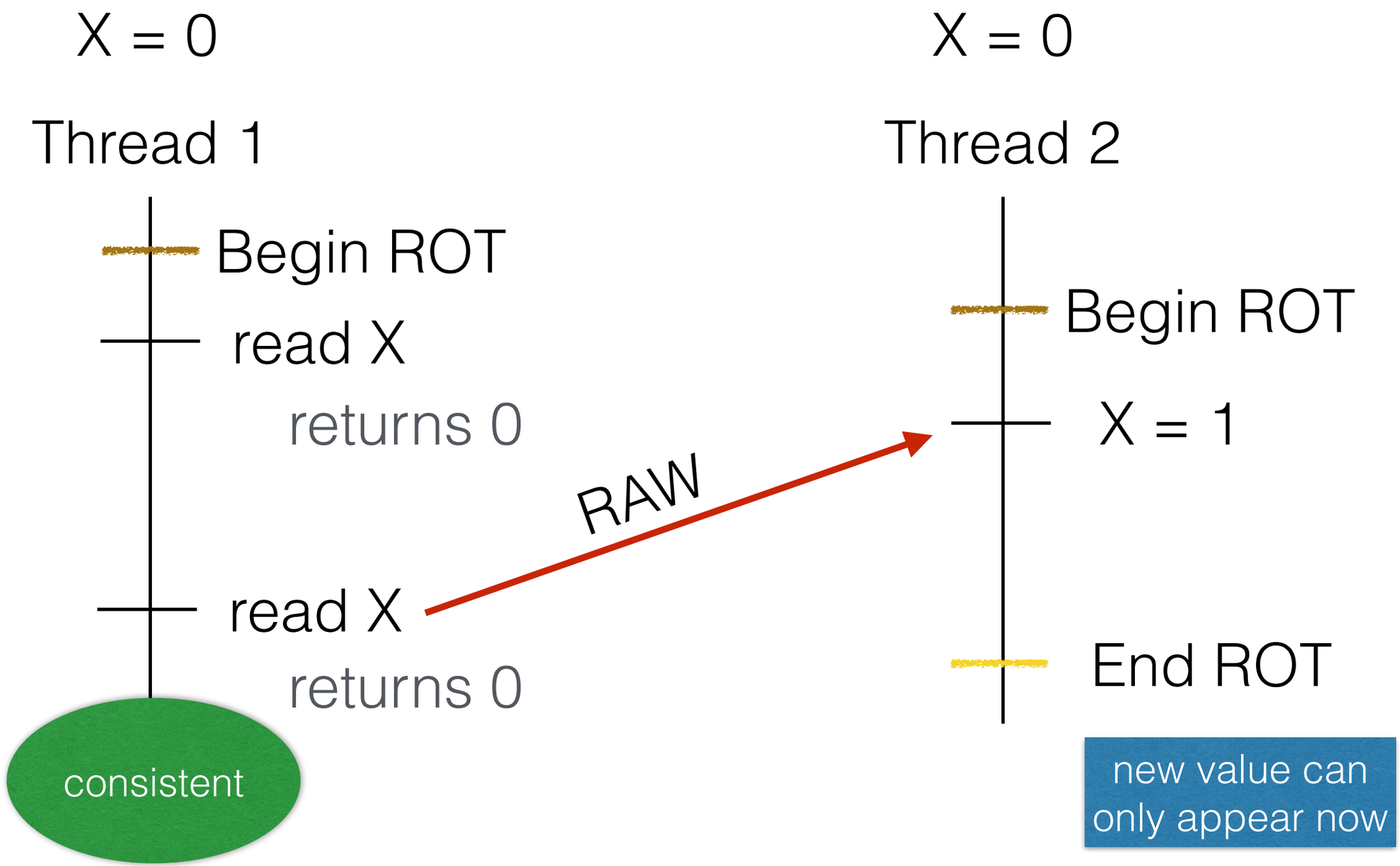
ROTs


X = 0
Thread 1
Begin ROT
read X
returns 0
read X
X = 0
Thread 2
Begin ROT
X = 1
End ROT

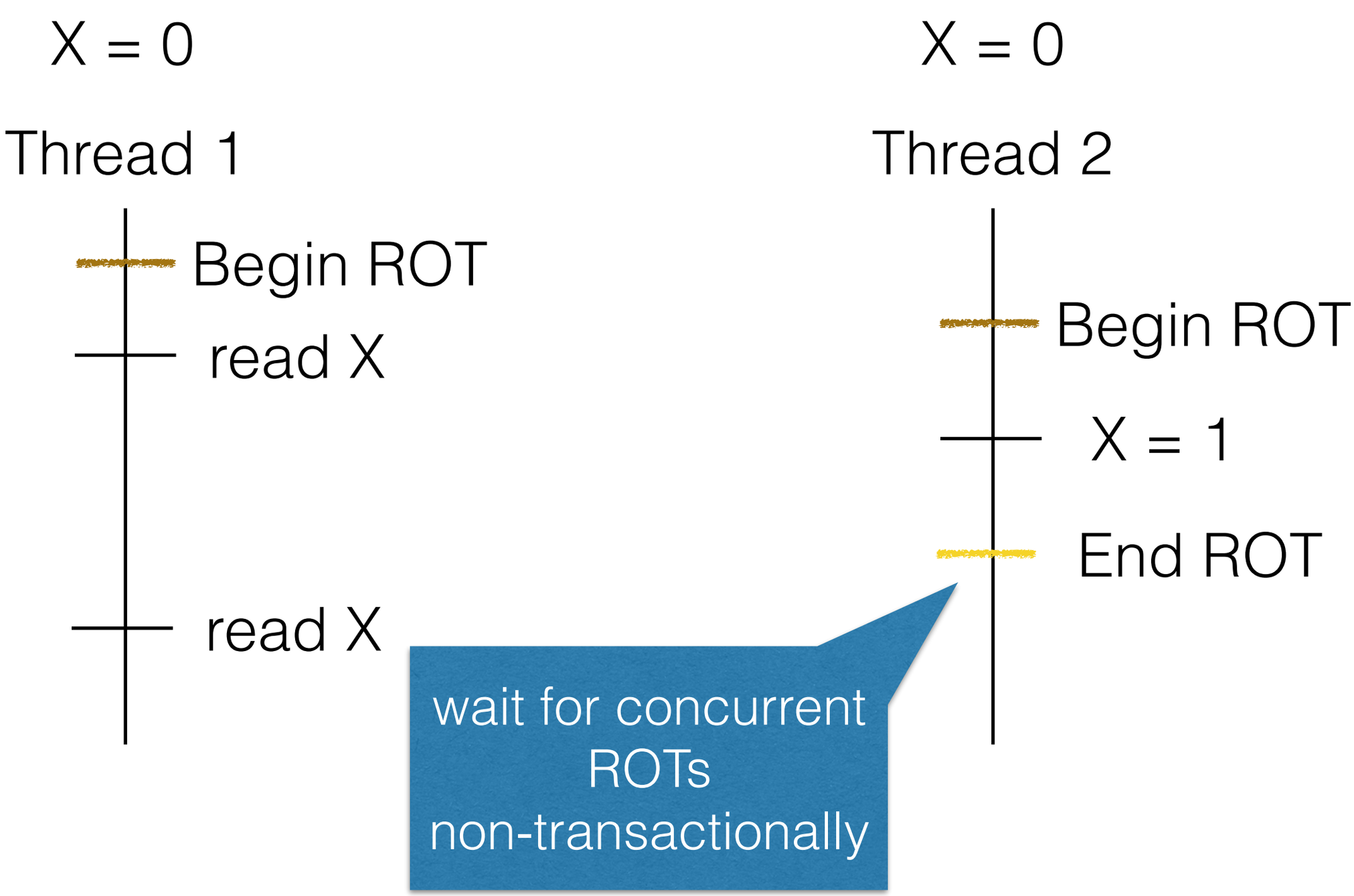
ROTs


X = 0
Thread 1
Begin ROT
read X
returns 0
read X
returns 0
consistent
X = 0
Thread 2
Begin ROT
X = 1
End ROT
new value can only appear now

ROTs


X = 0
X = 0
Thread 1
Thread 2
Begin ROT
read X
returns 0
Begin ROT
X = 1
RAW
read X
returns 0
End ROT
consistent
new value can only appear now

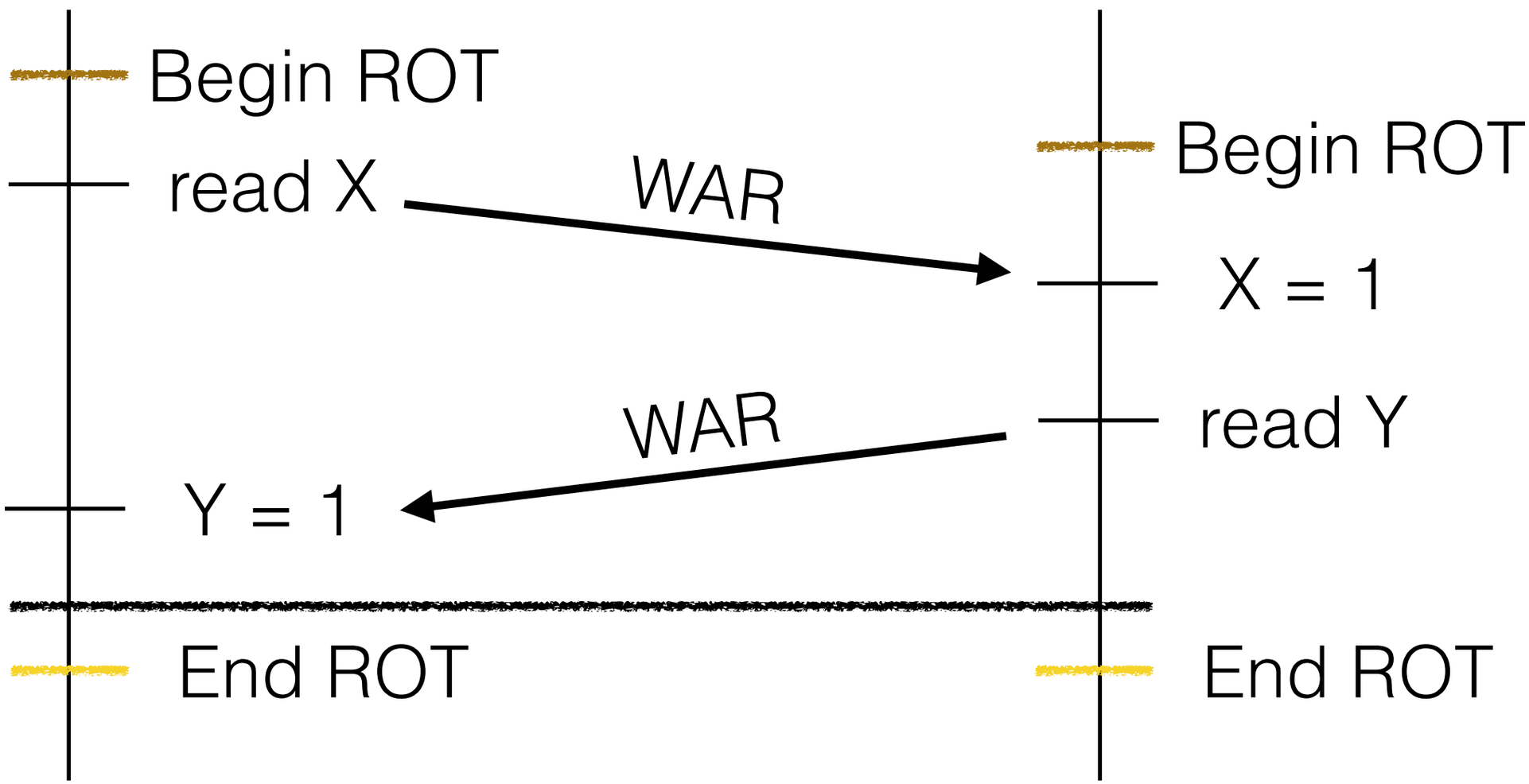
ROTs


X = 0
X = 0
Thread 1
Thread 2
Begin ROT
read X
Begin ROT
X = 1
End ROT
read X
wait for concurrent ROTs non-transactionally

ROTs

X = 0
Y = 0

X = 0
Y = 0



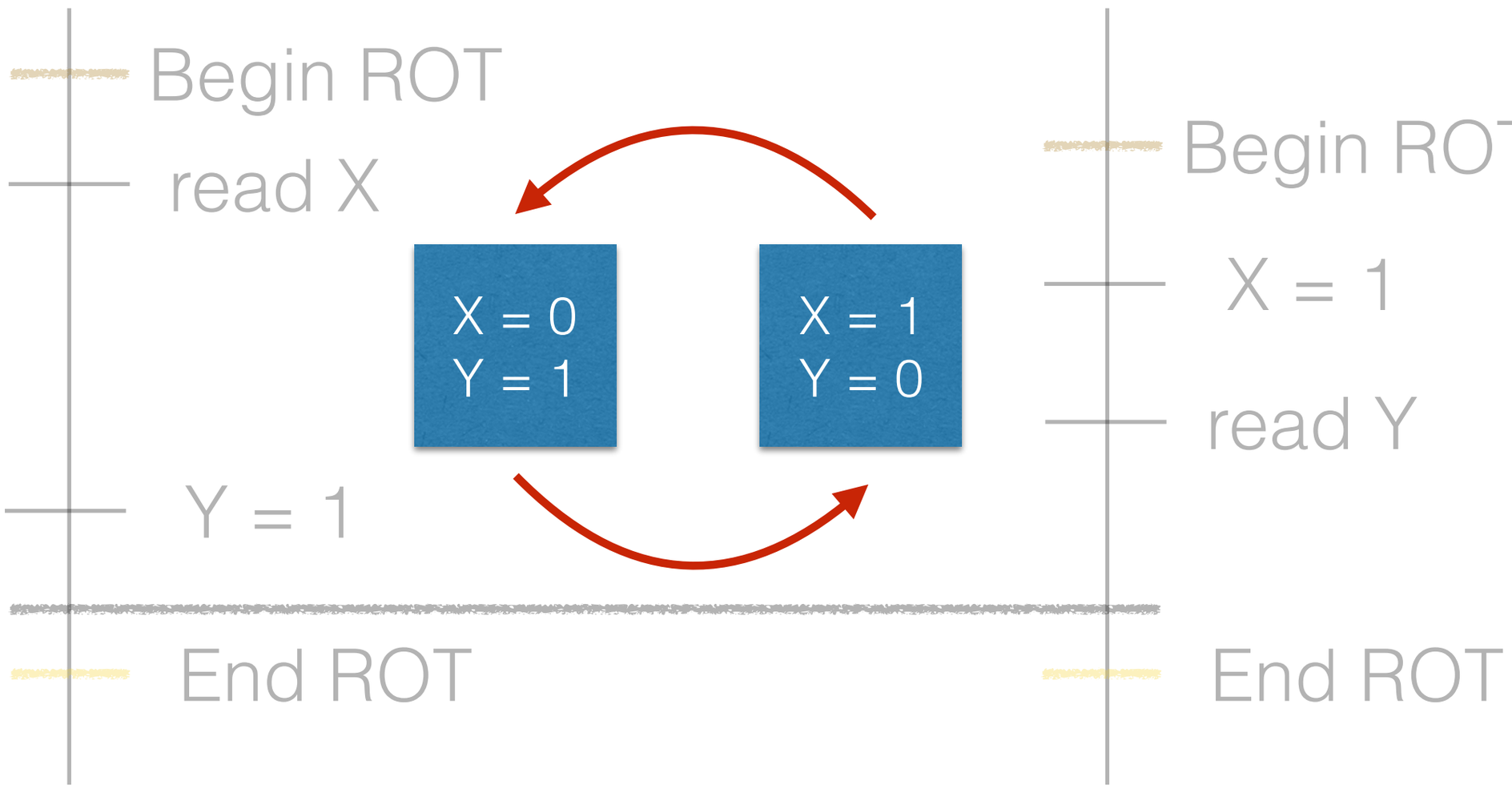
ROTs

X = 0
Y = 0

Thread 1

X = 0
Y = 0

Thread 2



Touch-to-Validate

- core algorithm of P8TM
- to make concurrent execution of ROTs safe and serialisable
- basic intuition: convert WAR to RAW

T2V

X = 0
Y = 0

Thread 1

Begin ROT

read X

write Y

End ROT

X = 0
Y = 0

Thread 2

Begin ROT

write X

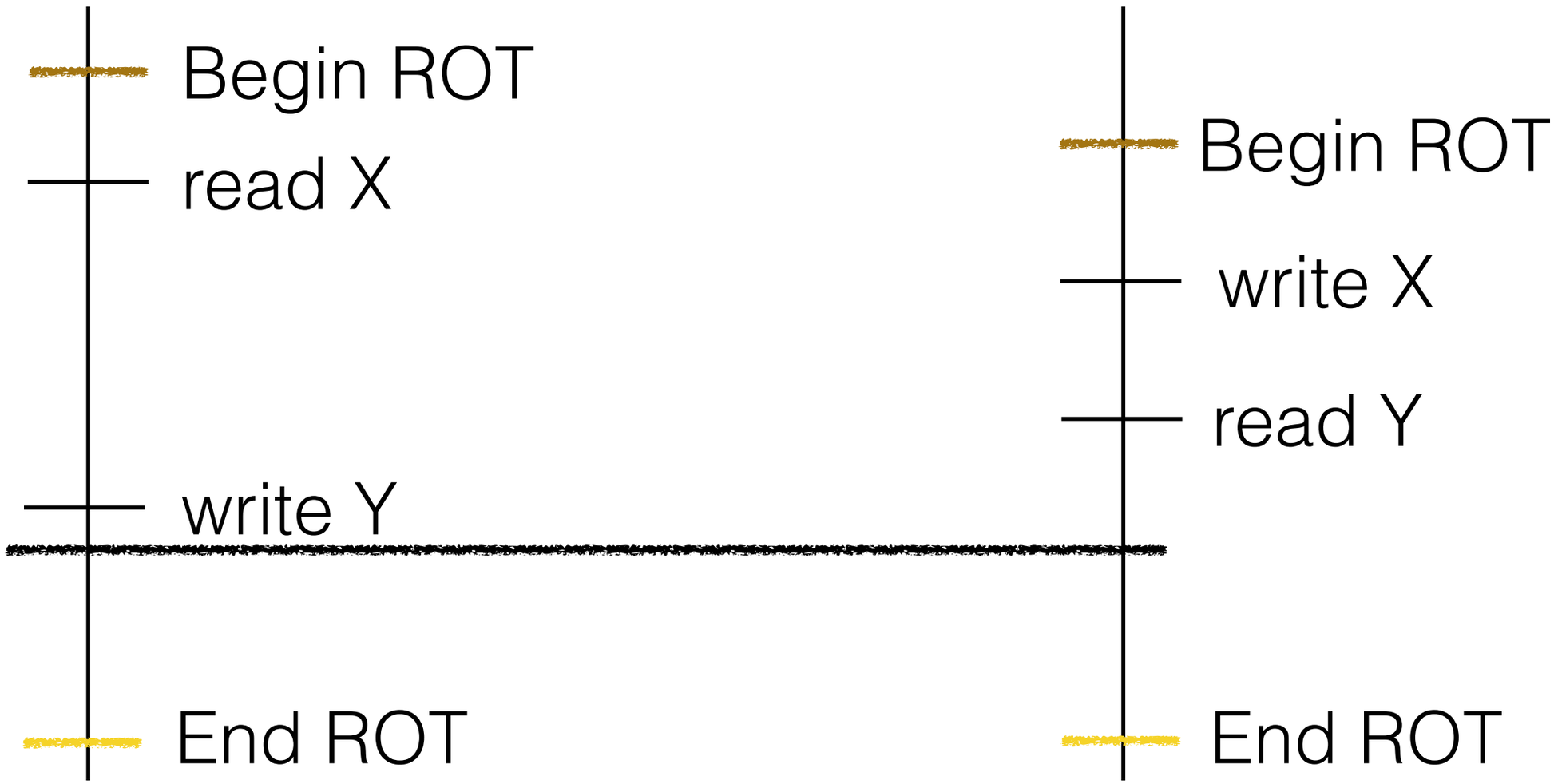
read Y

End ROT

T2V

X = 0
Y = 0

X = 0
Y = 0



T2V

X = 0
Y = 0

Thread 1

Begin ROT

read X

write Y

re-read X

End ROT

X = 0
Y = 0

Thread 2

Begin ROT

write X

read Y

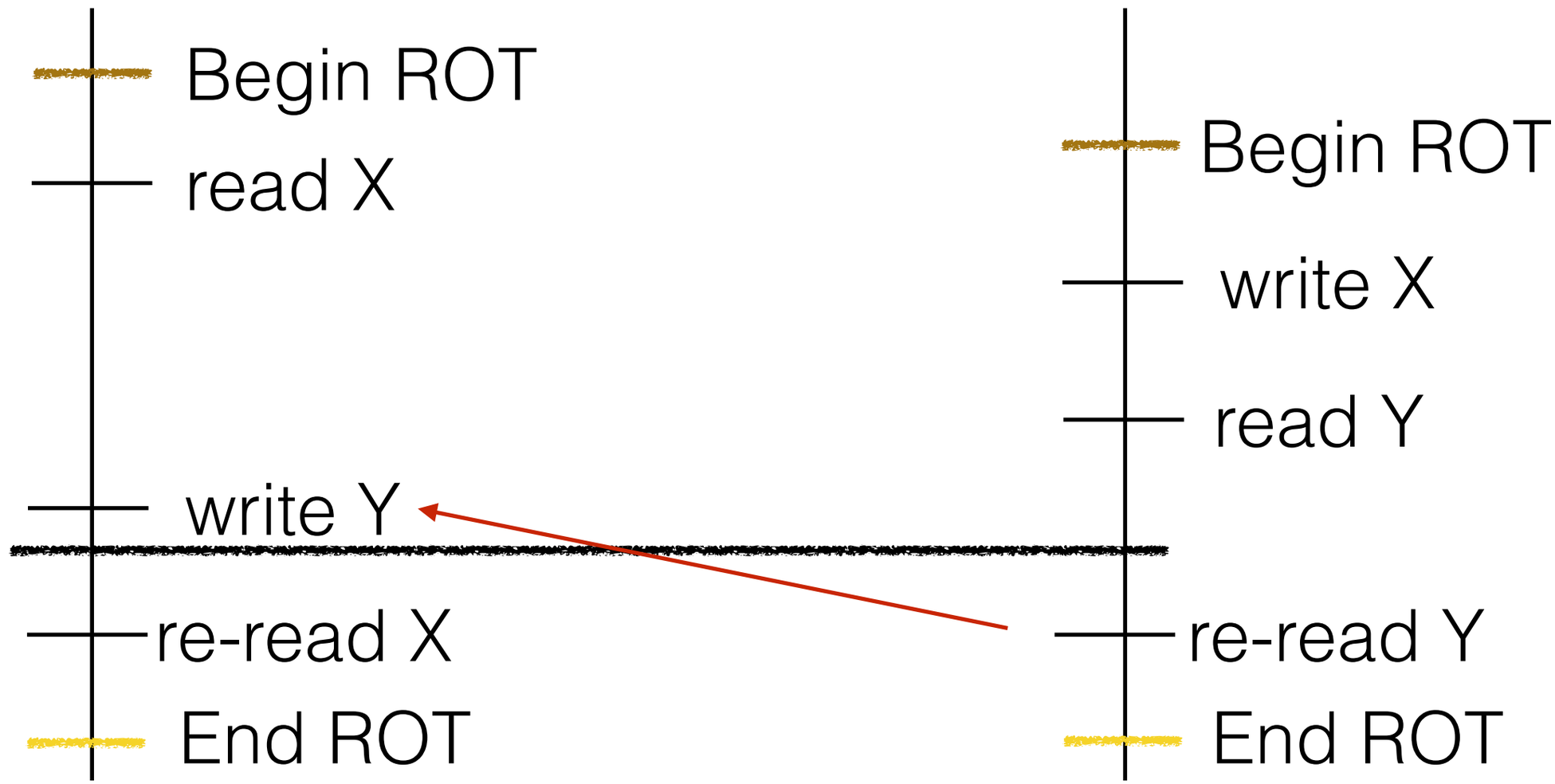
re-read Y

End ROT

T2V

X = 0
Y = 0

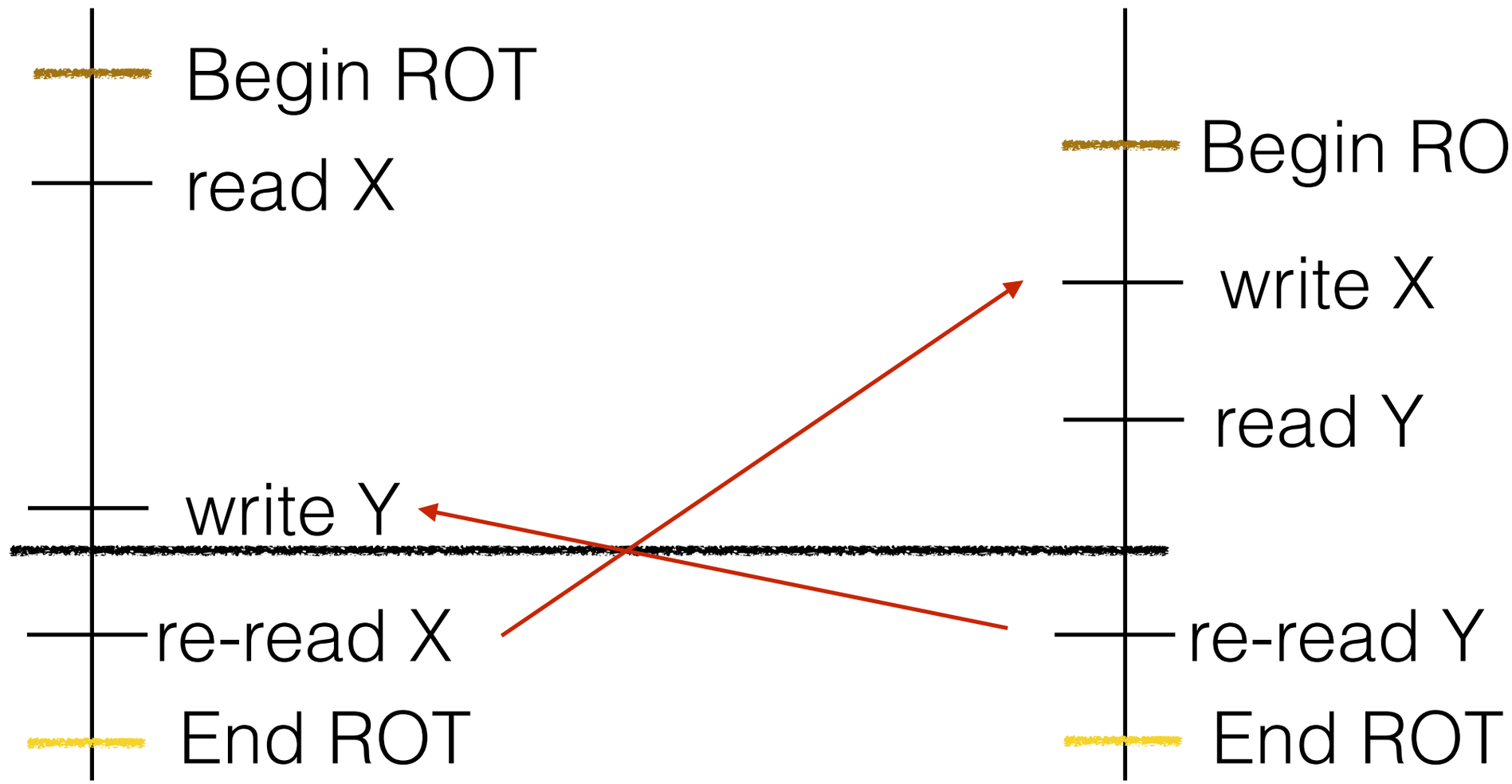
X = 0
Y = 0



T2V

X = 0
Y = 0

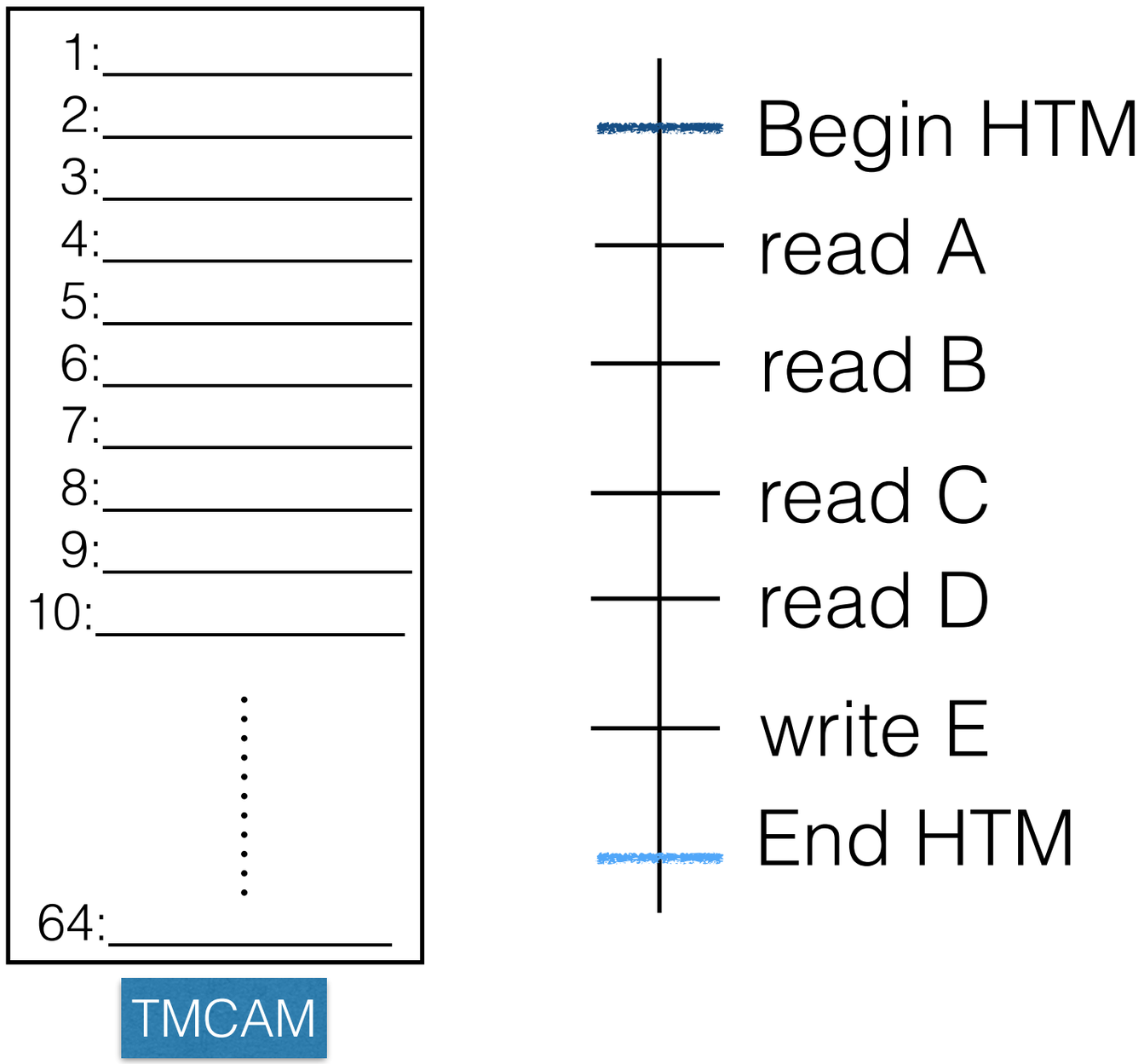
X = 0
Y = 0



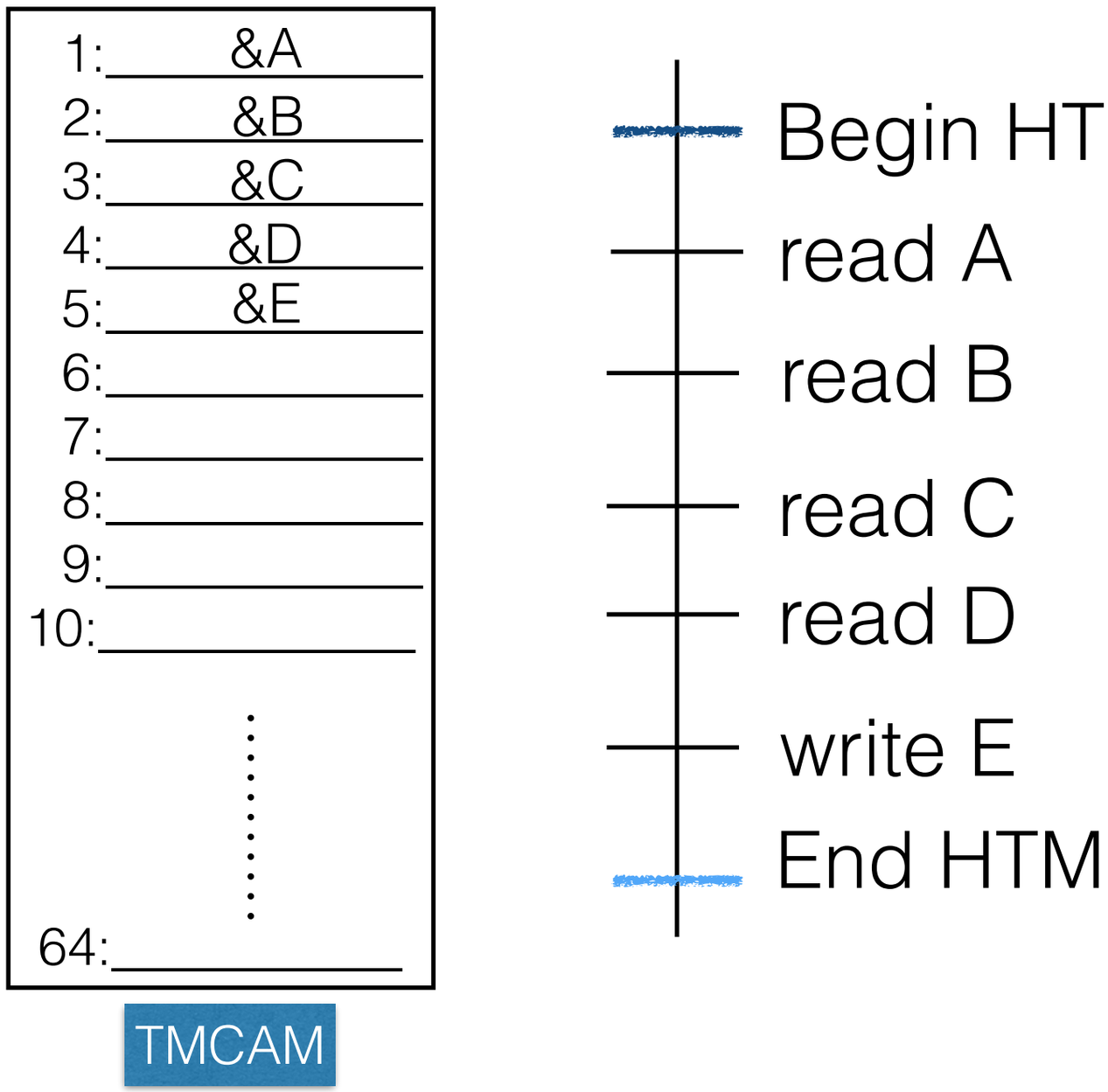
T2V

- needs to track only the addresses
- this must be done in software
- how can software outperform hardware?

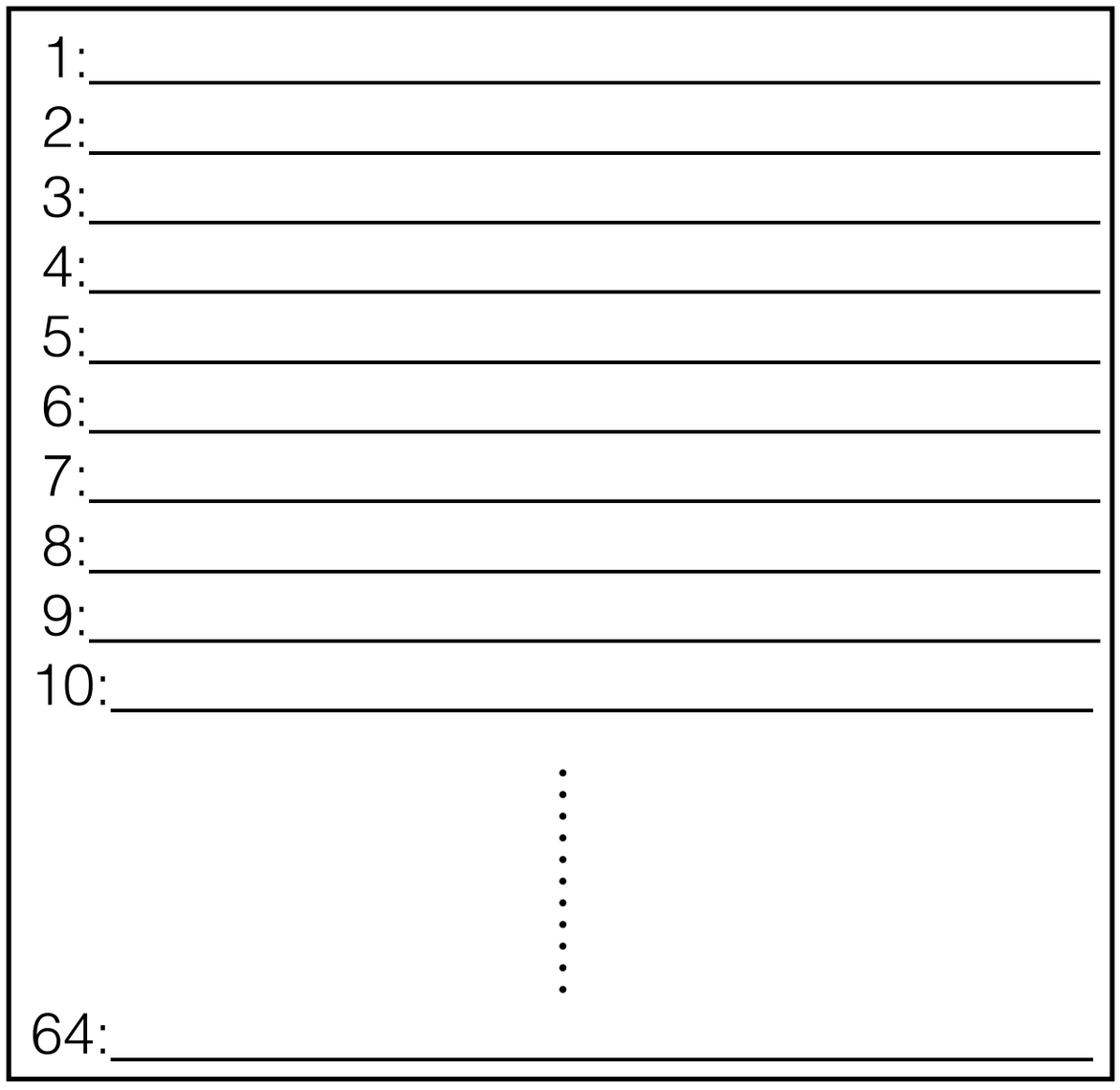
TMCAM

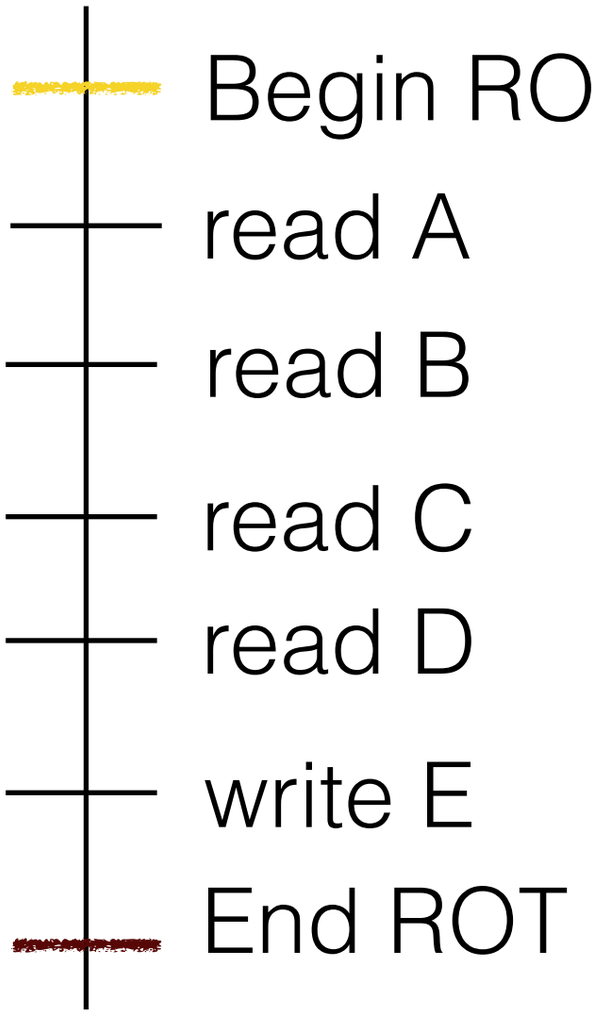

1:
2:
3:
4:
5:
6:
7:
8:
9:
10:
64:
TMCAM
Begin HTM
read A
read B
read C
read D
write E
End HTM

TMCAM

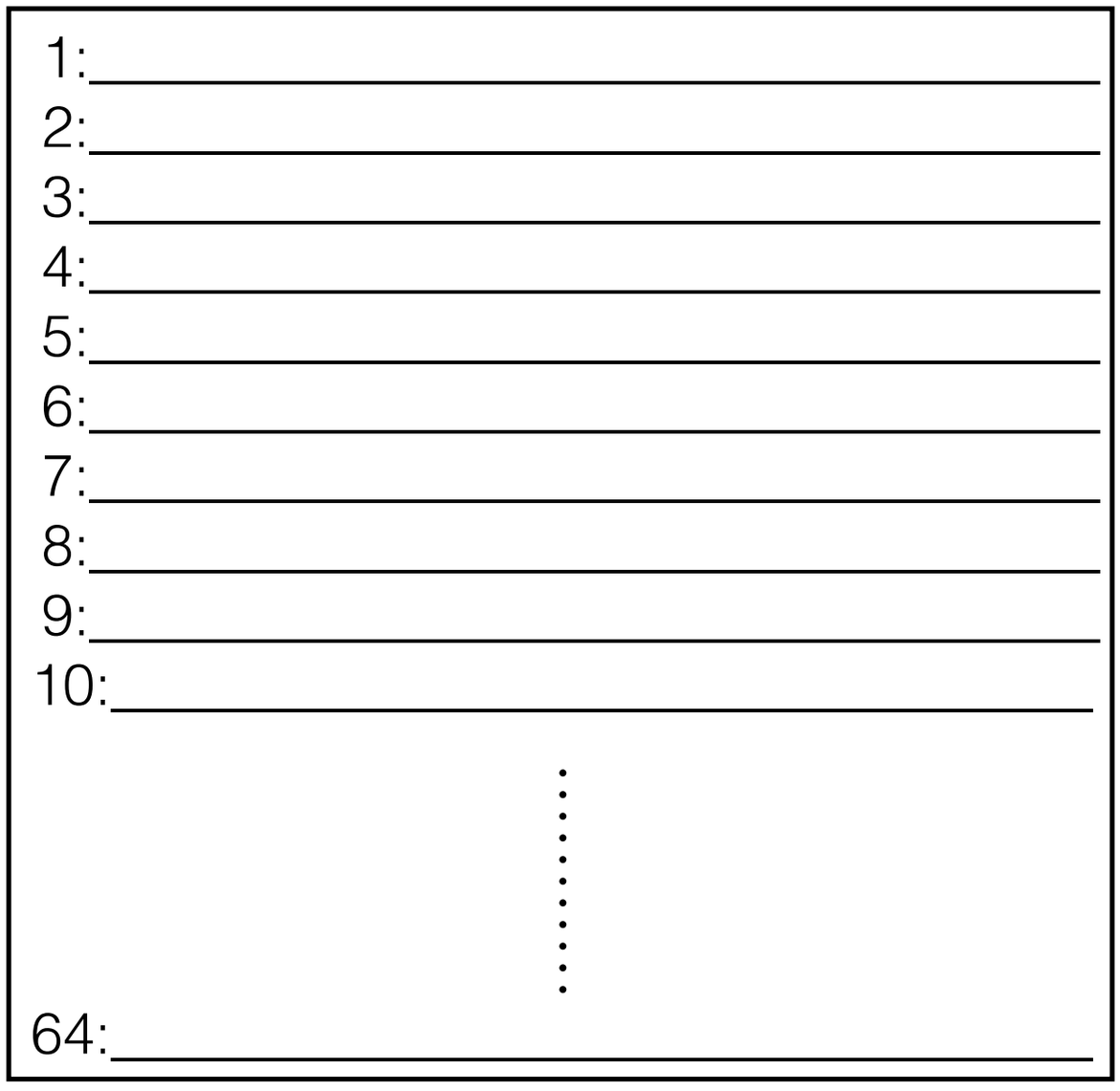

1: &A
2: &B
3: &C
4: &D
5: &E
6:
7:
8:
9:
10:
64:
TMCAM
Begin HTM
read A
read B
read C
read D
write E
End HTM

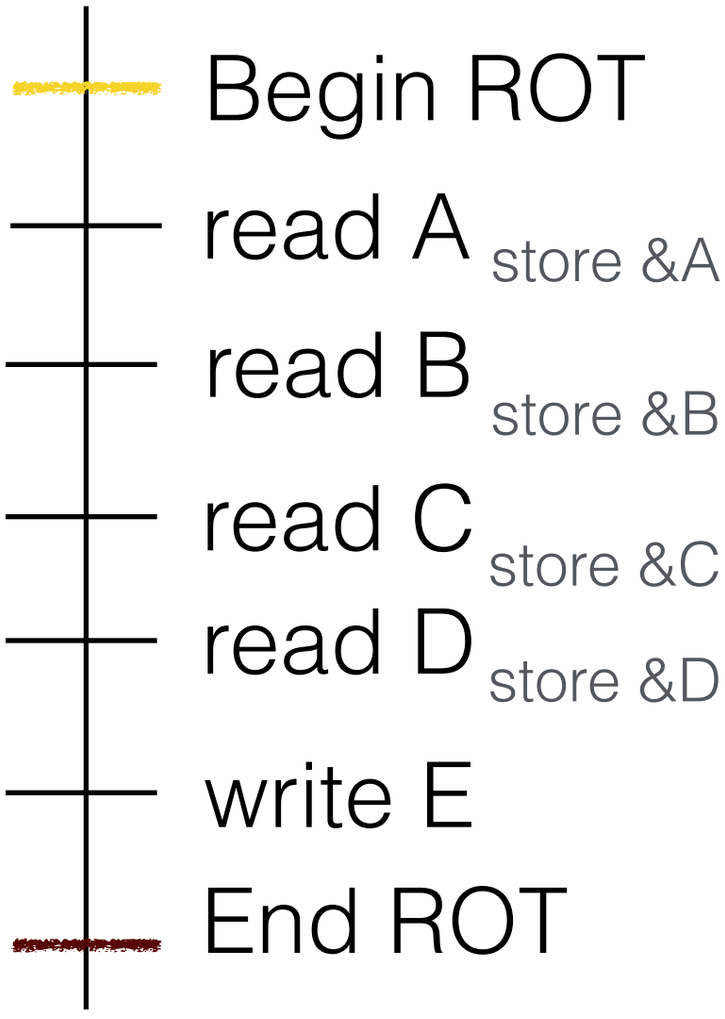
Read-set Tracking



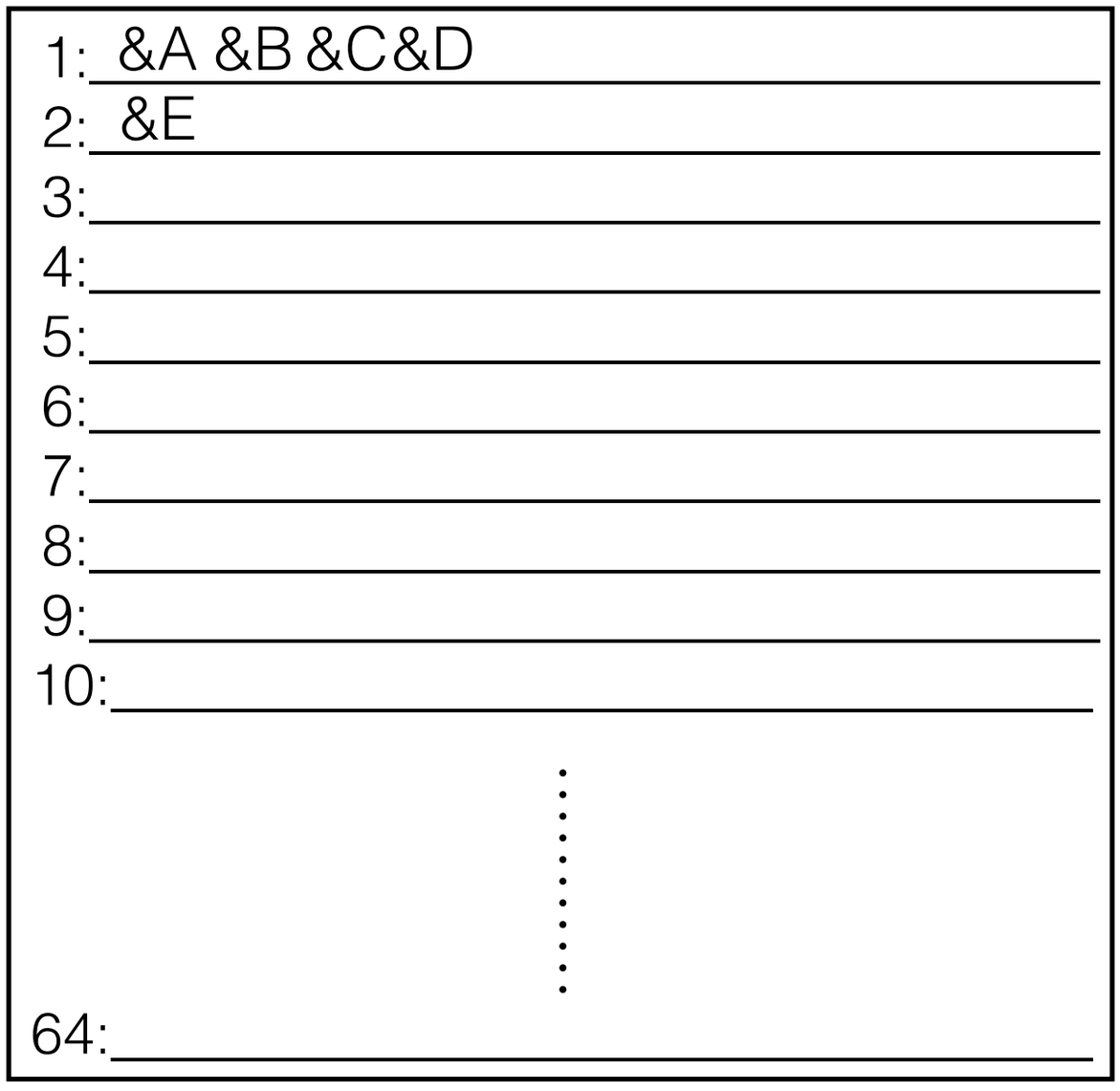


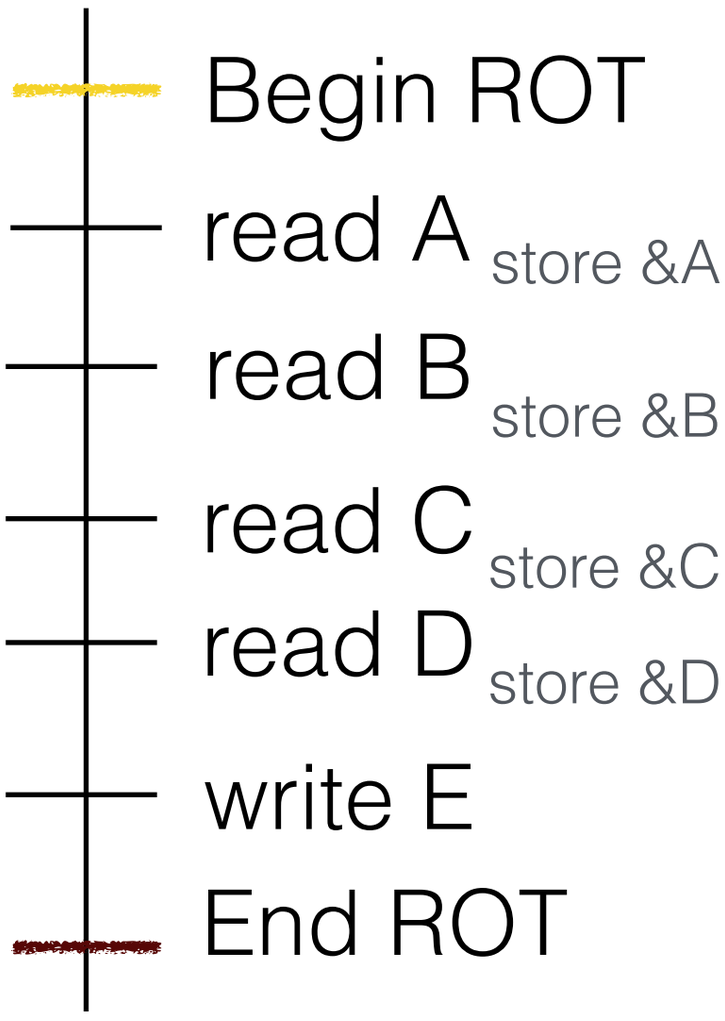
Read-set Tracking



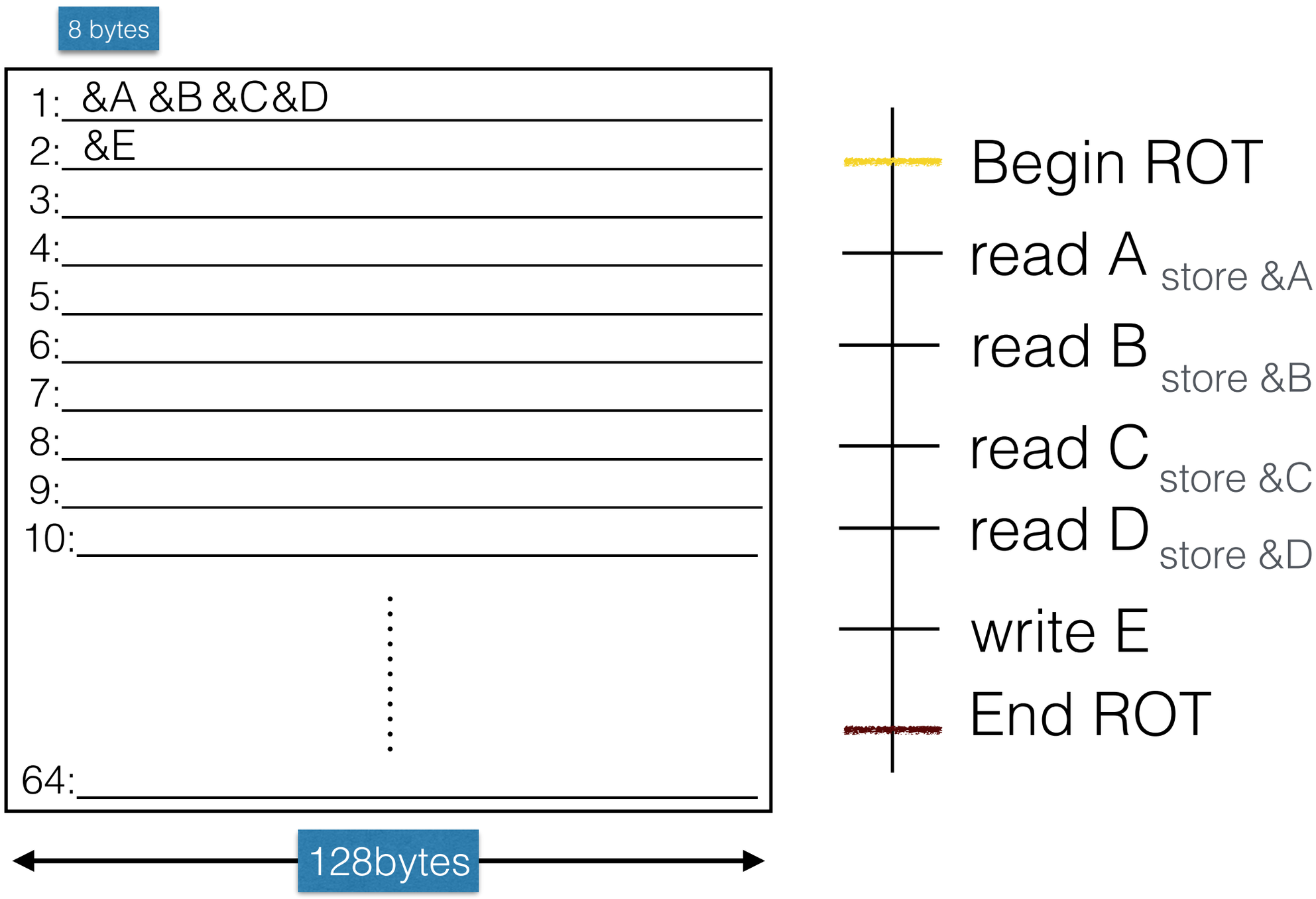


Read-set Tracking

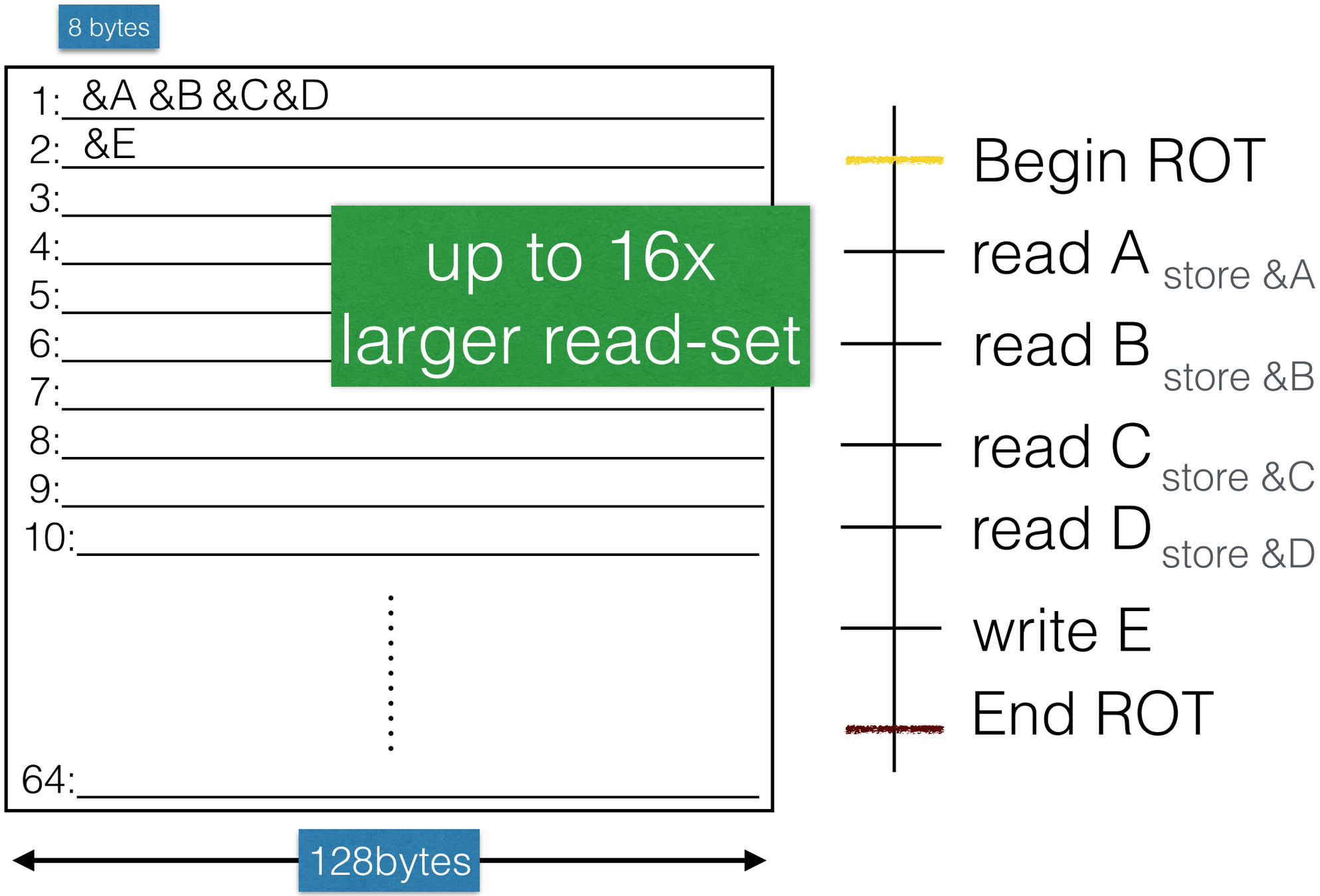




Read-set Tracking



Read-set Tracking



HTM

- transactions may fit in HTM
- we need to avoid extra overheads of using ROTs
- try first in HTM, if it overflows, fallback to ROT
- how can HTMs and ROTs run concurrently?

HTM + ROT

X = 0
Y = 0

Thread 1

Begin HTM

read X

Y = 1

End HTM

X = 0
Y = 0

Thread 2

Begin ROT

X = 1

End ROT

HTM + ROT

X = 0
Y = 0

X = 0
Y = 0

Thread 1

Thread 2

Begin HTM

read X

Y = 1

End HTM

Begin ROT

X = 1

End ROT

HTM is protected by H/W

HTM + ROT

X = 0
Y = 0

Thread 1

Begin HTM

read X

Y = 1

End HTM

HTM is protected by H/W

X = 0
Y = 0

Thread 2

Begin ROT

End ROT

HTM + ROT

X = 0
Y = 0

Thread 1

Begin HTM

read X

Y = 1

End HTM

HTM is protected by H/W

X = 0
Y = 0

Thread 2

Begin ROT

read Y

read Y

End ROT

HTM + ROT

X = 0
Y = 0

Thread 1

Begin HTM

read X

Y = 1

End HTM

HTM is protected by H/W

X = 0
Y = 0

Thread 2

Begin ROT

read Y
returns 0

read Y
returns 1

inconsistent value

End ROT

HTM + ROT

X = 0
Y = 0

X = 0
Y = 0

Thread 1

Thread 2

Begin HTM

read X

Y = 1

End HTM

Begin ROT

read Y
returns 0

read Y
returns 1

T2V

End ROT

inconsistent value

HTM is protected by H/W

HTM + ROT

X = 0
Y = 0

X = 0
Y = 0

Thread 1

Thread 2

Begin HTM

read X

Y = 1

Begin ROT

read Y

returns 0

using S/R

consistent value

read Y

returns 0

End HTM

T2V

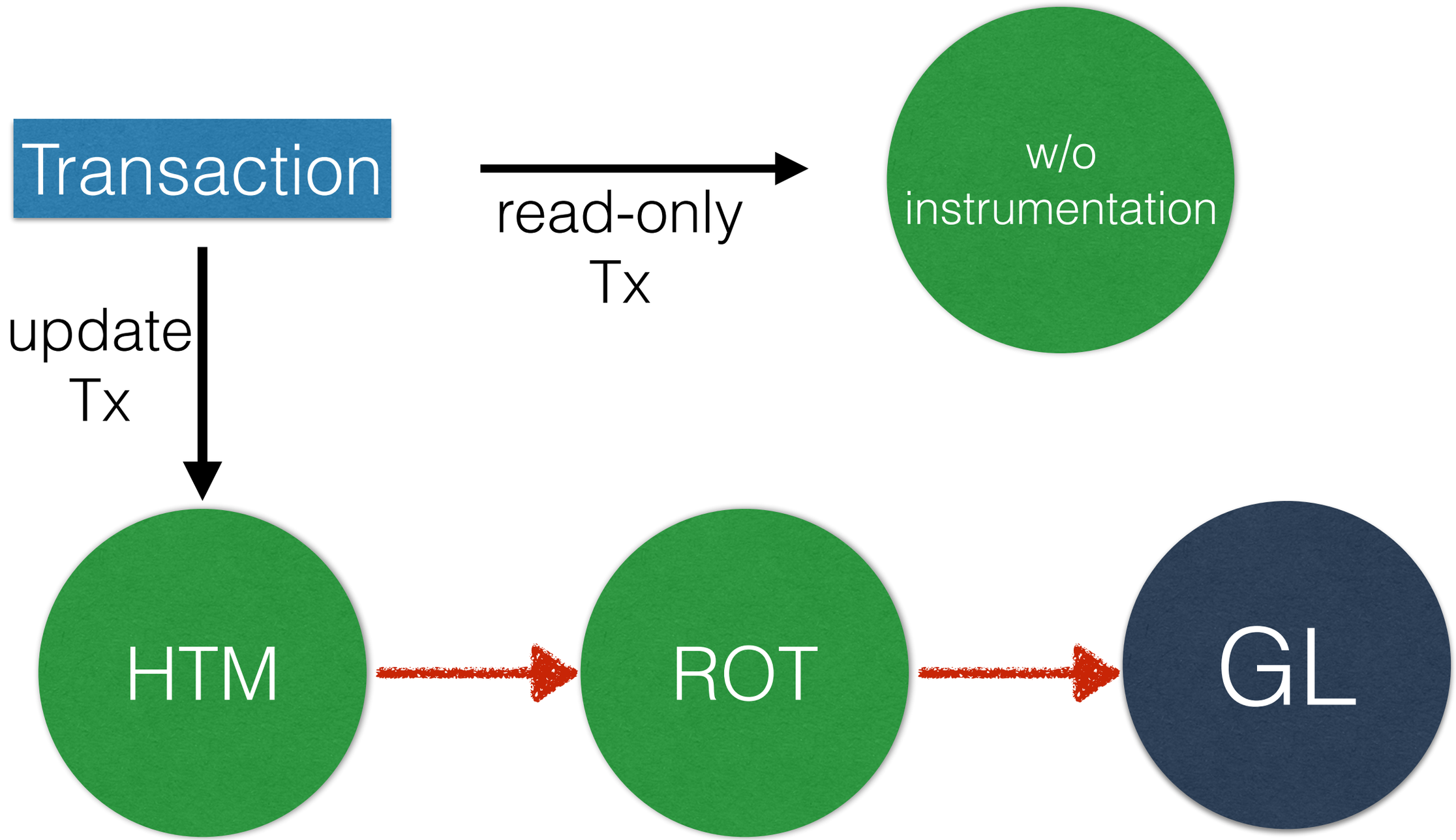
End ROT

HTM is protected by H/W

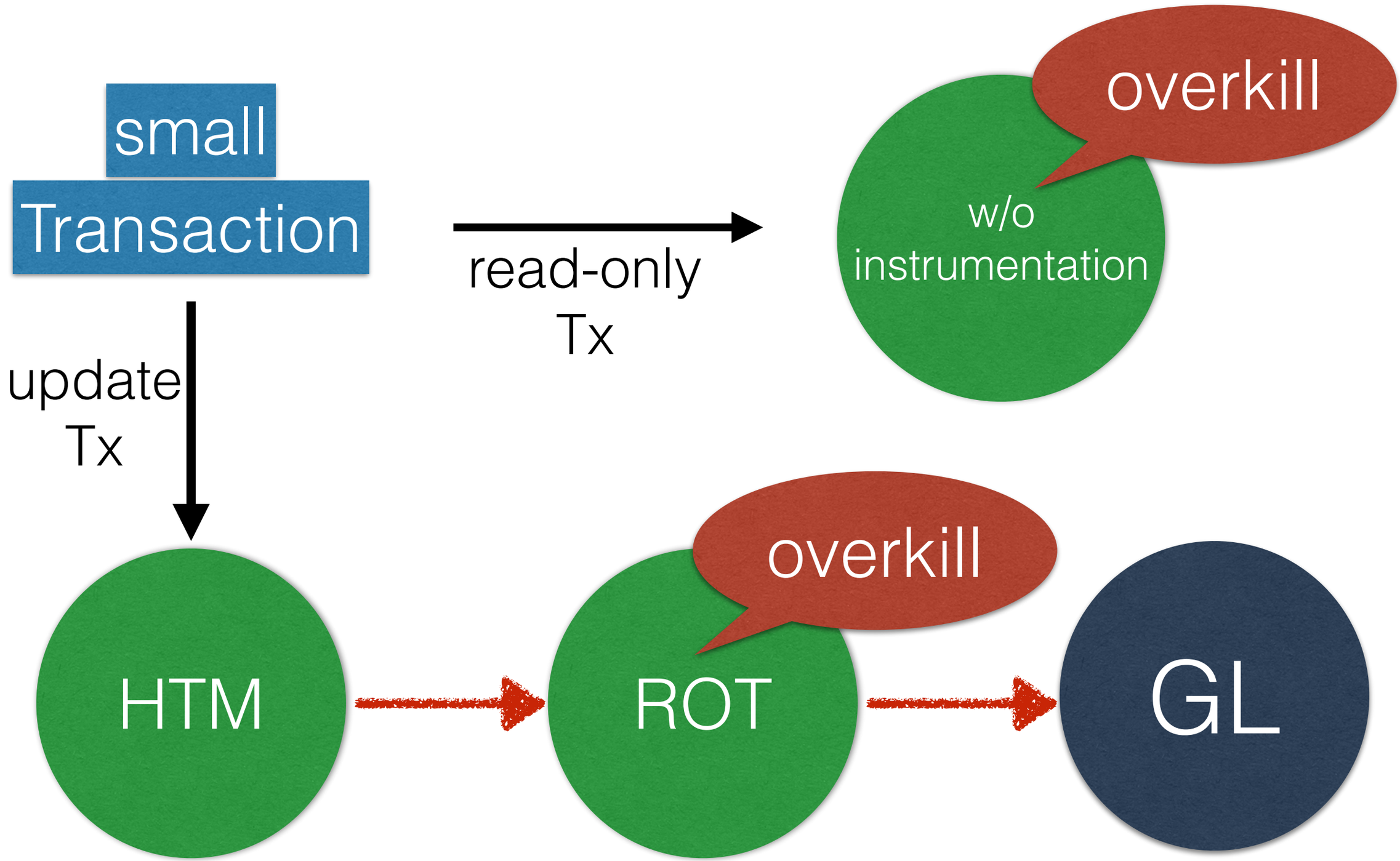
Uninstrumented Read-only

- read only transactions without any instrumentation
- outside the context of HTM or ROT
- no bounds on Tx size
- HTMs and ROTs must wait for UROs

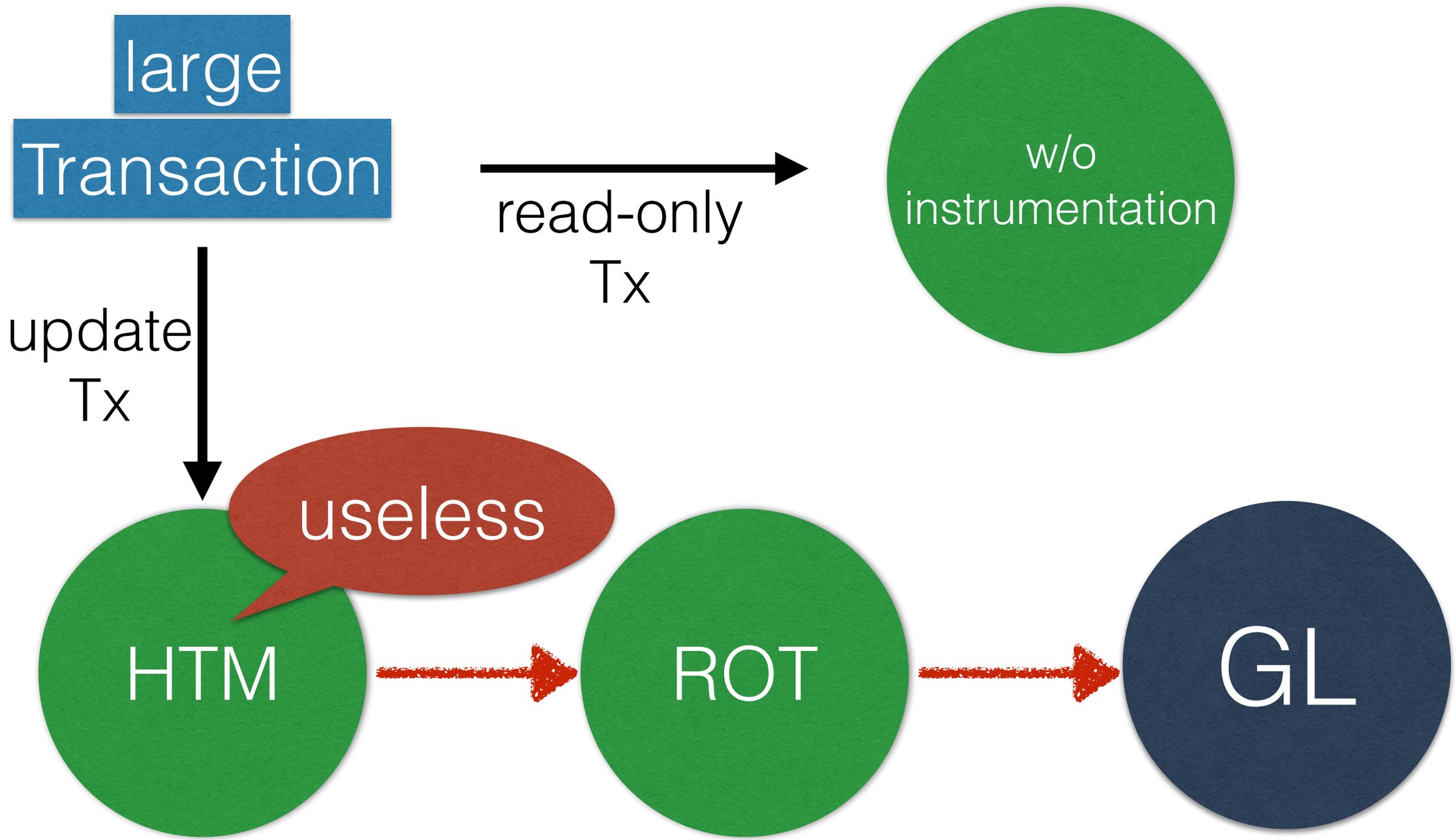
POWER8-TM


Transaction
read-only Tx
w/o instrumentation
update Tx
HTM
ROT
GL

POWER8-TM


small
Transaction
update
Tx
read-only
Tx
overkill
w/o
instrumentation
HTM
ROT
overkill
GL

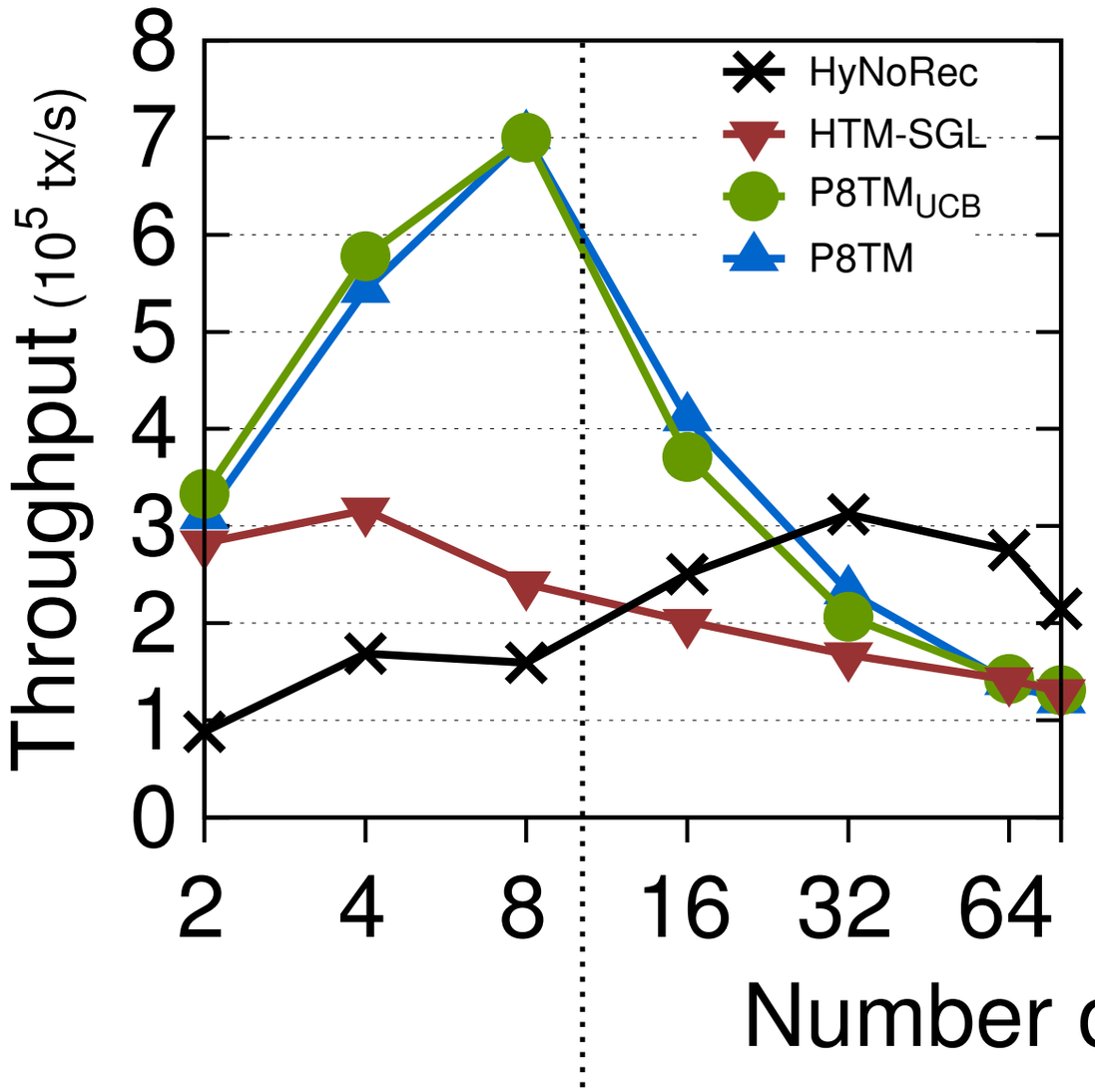
POWER8-TM

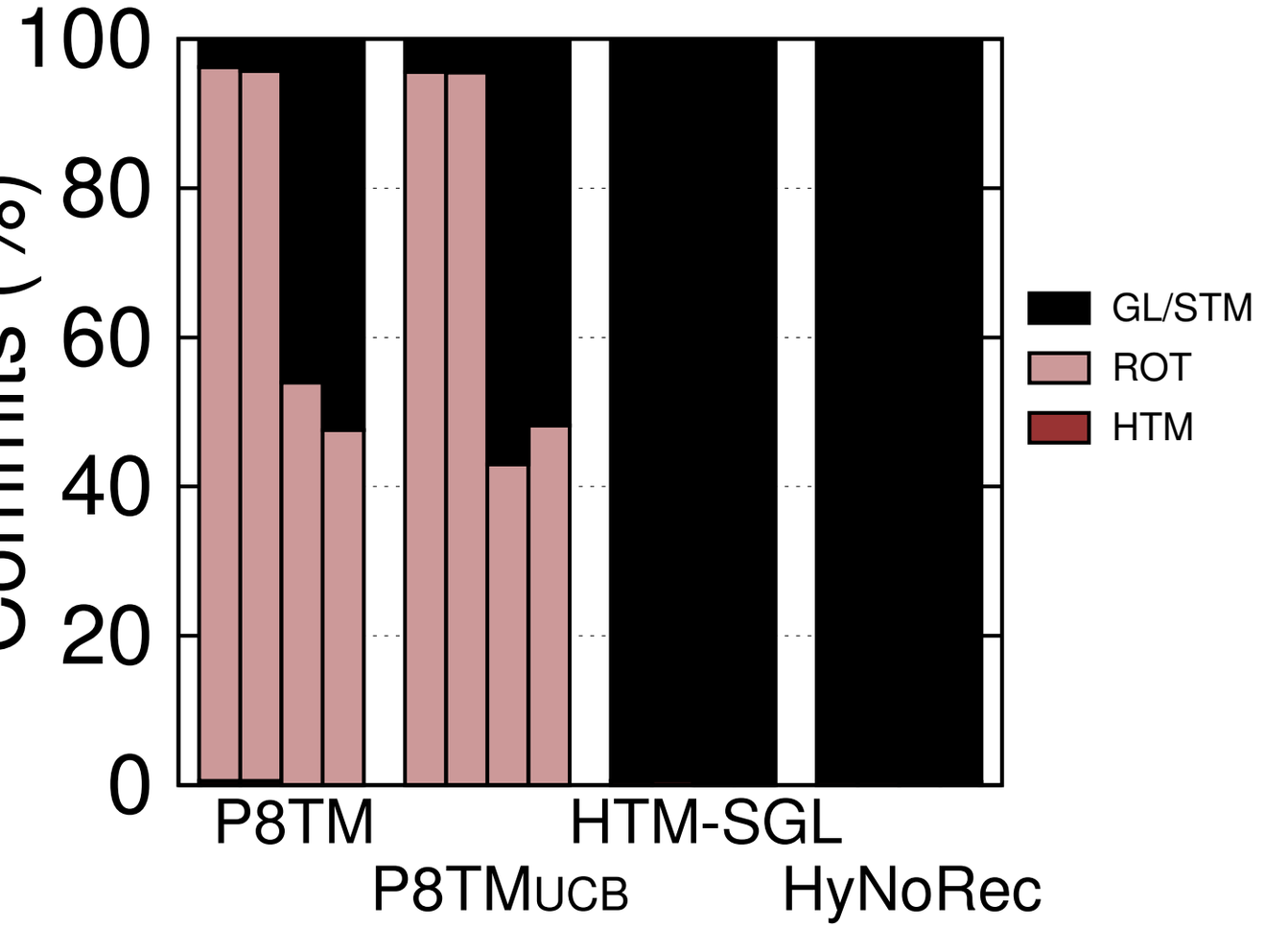

large
Transaction
read-only
Tx
w/o
instrumentation
update
Tx
useless
HTM
ROT
GL

Self-tuning

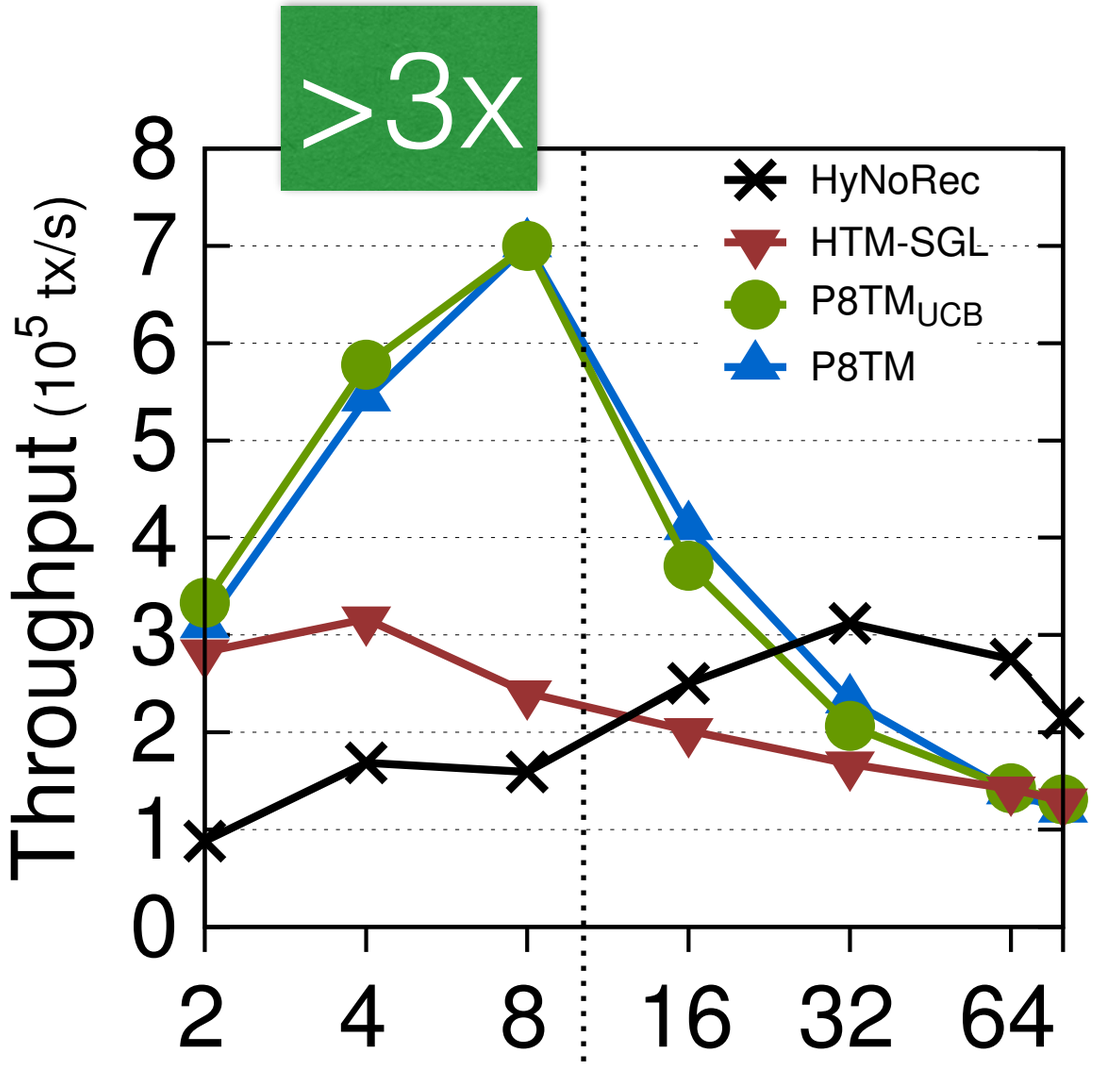
- lightweight, online reinforcement learning
- determine execution path:
  - HTM —> GL : small Txs
  - ROT —> GL : large Txs
  - HTM —> ROT —> GL : mixed workload

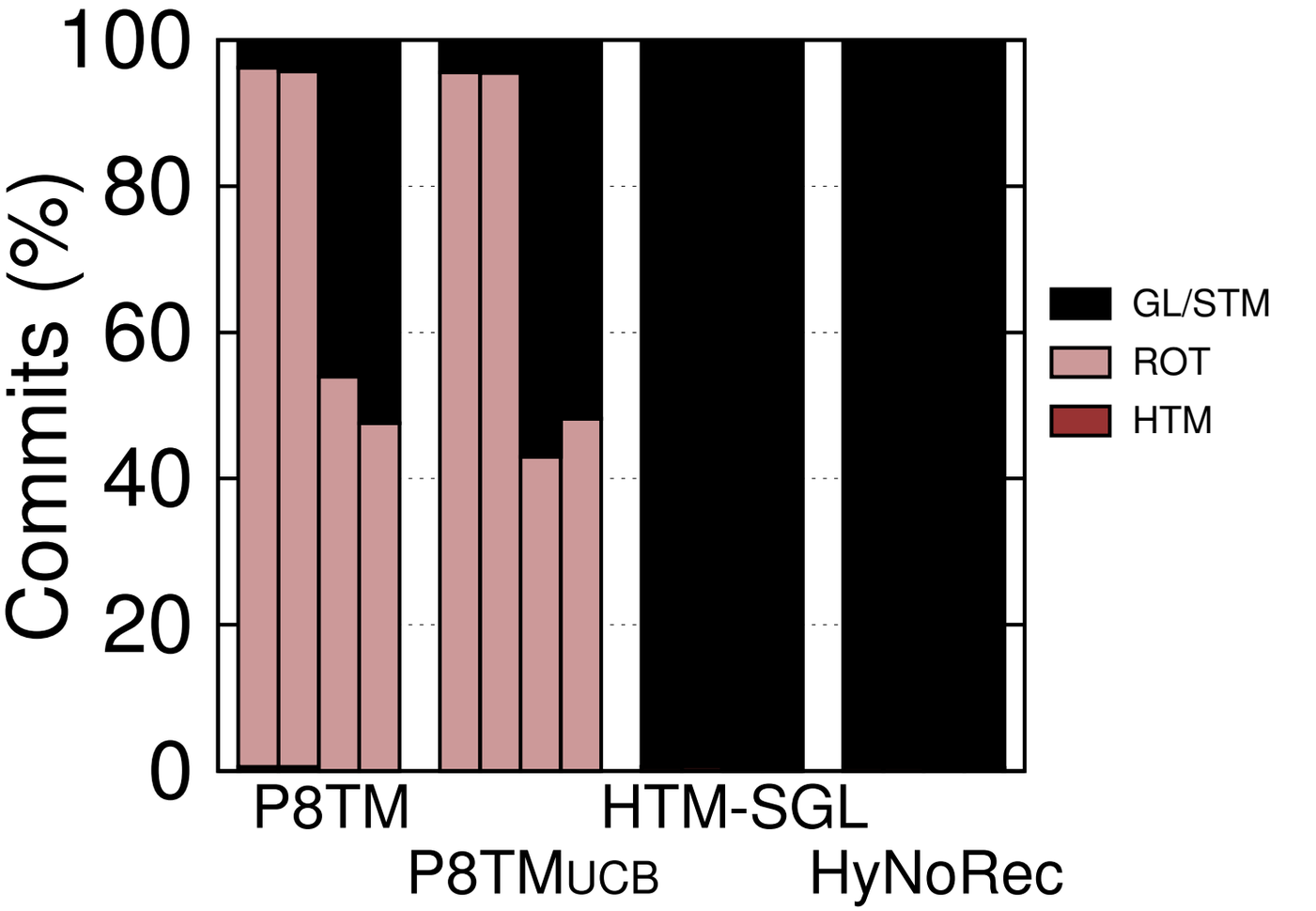
Evaluation: Vacation



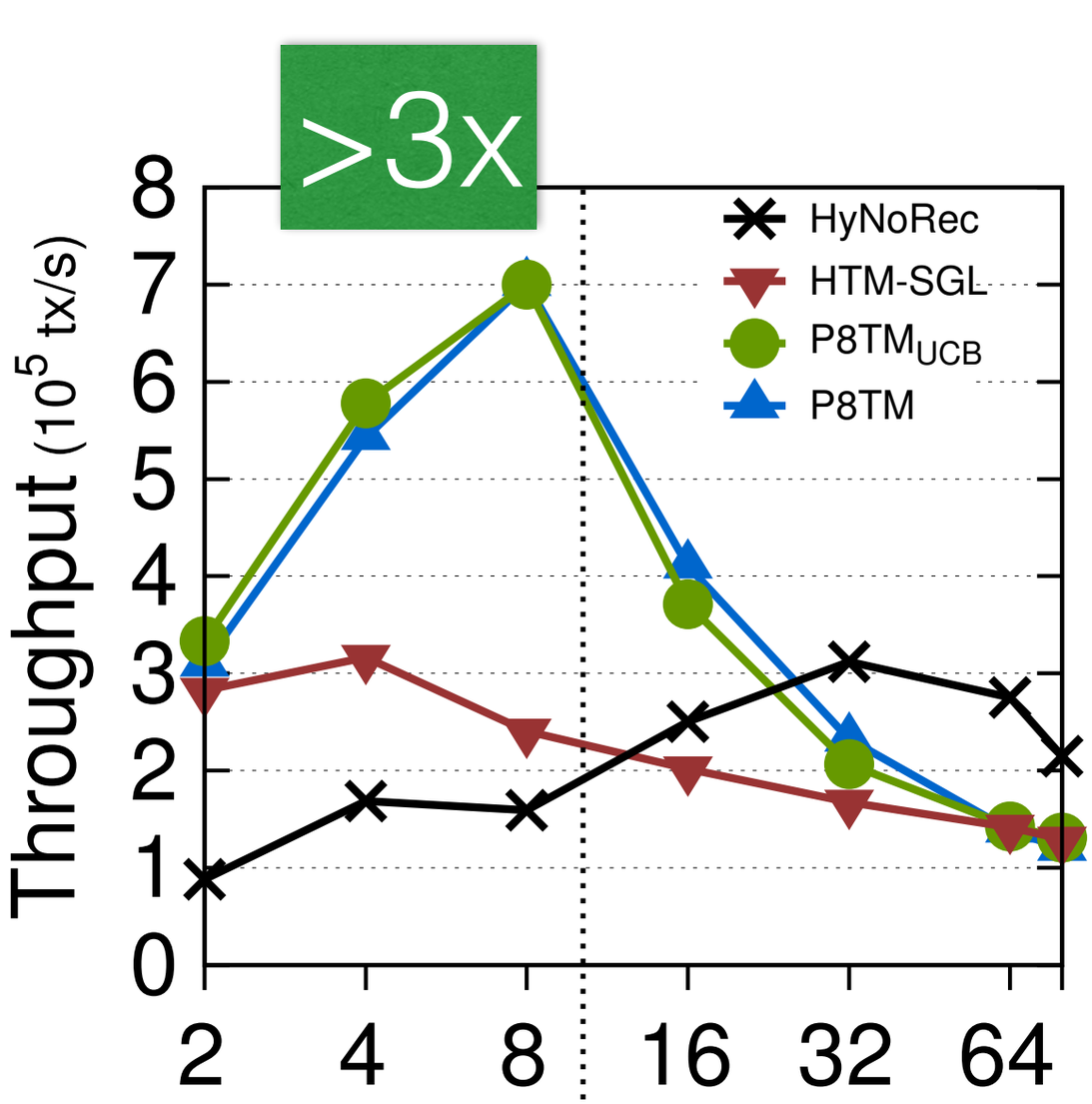


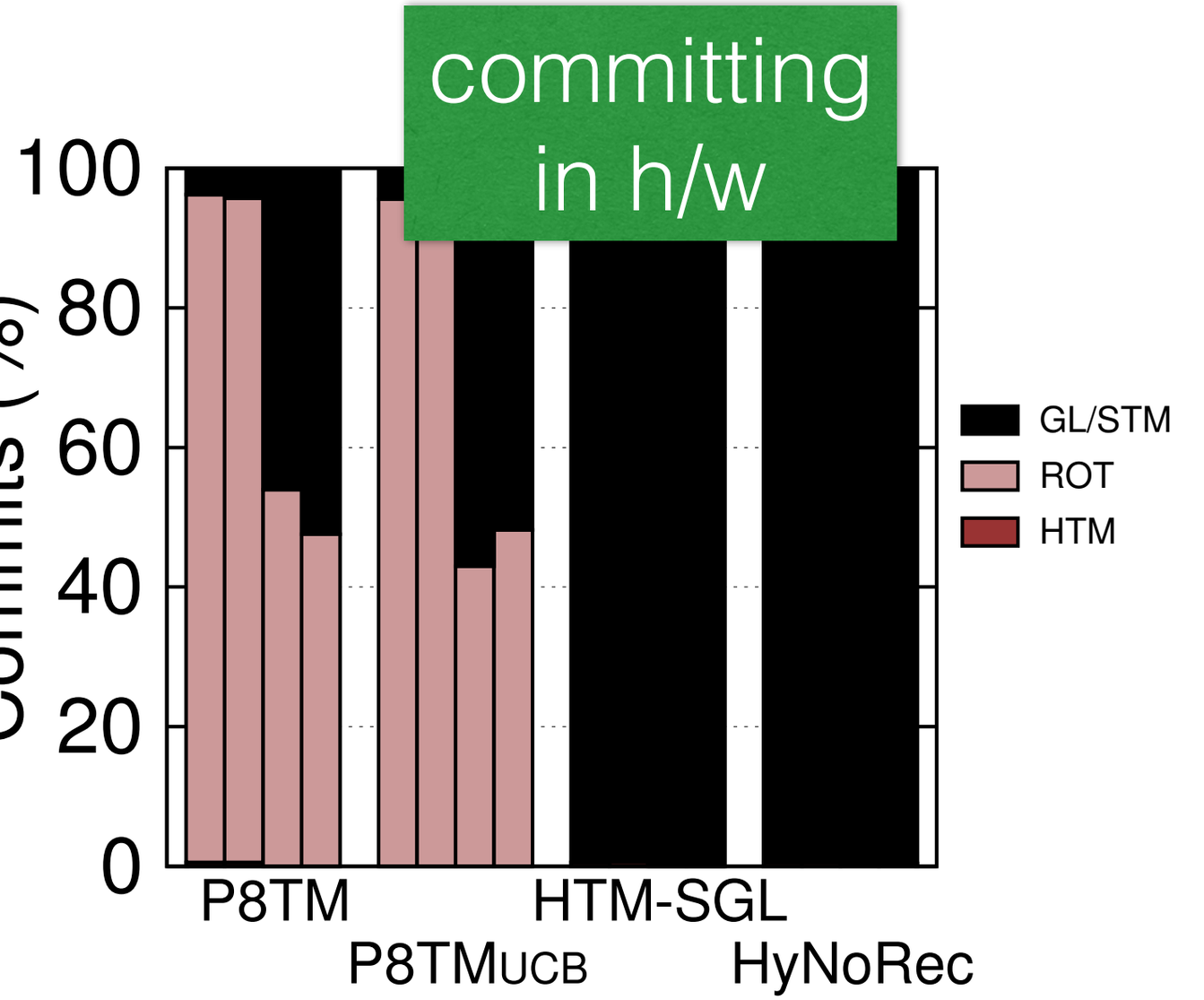
Evaluation: Vacation



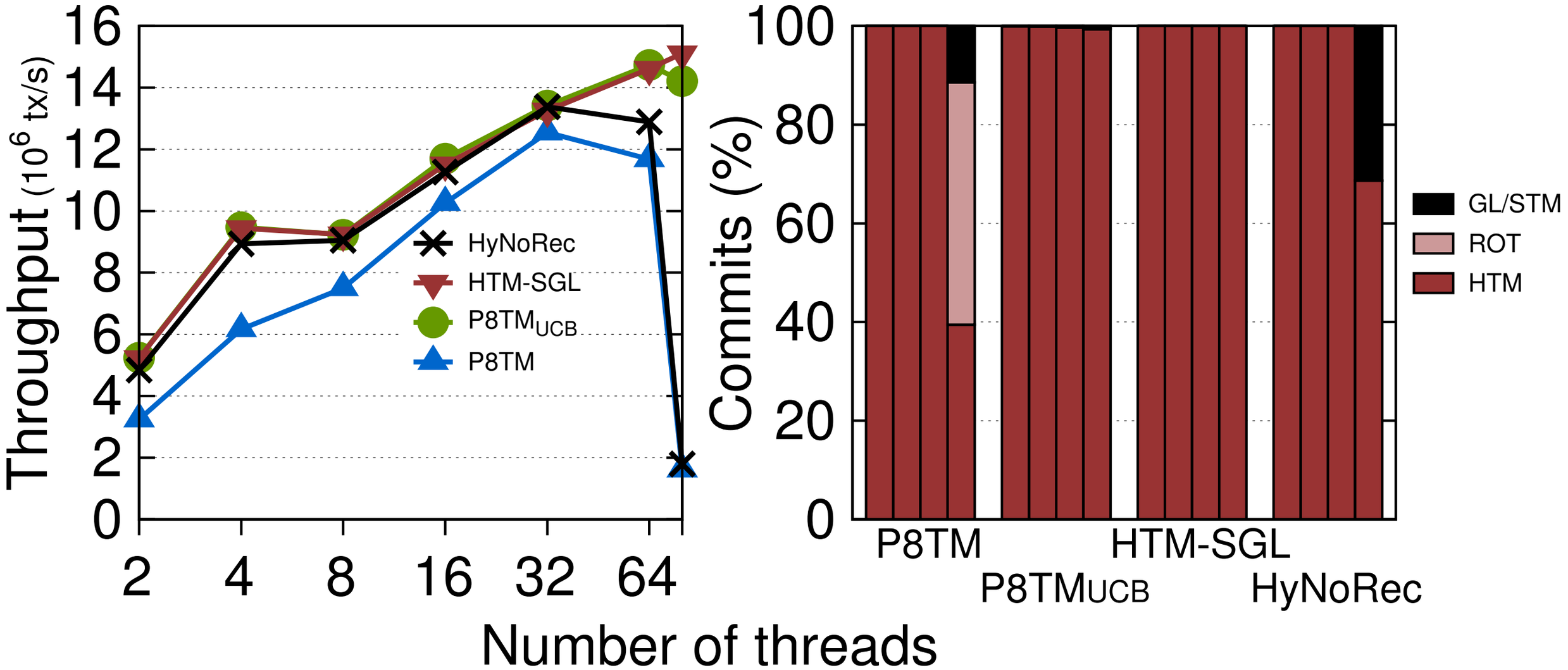


Evaluation: Vacation

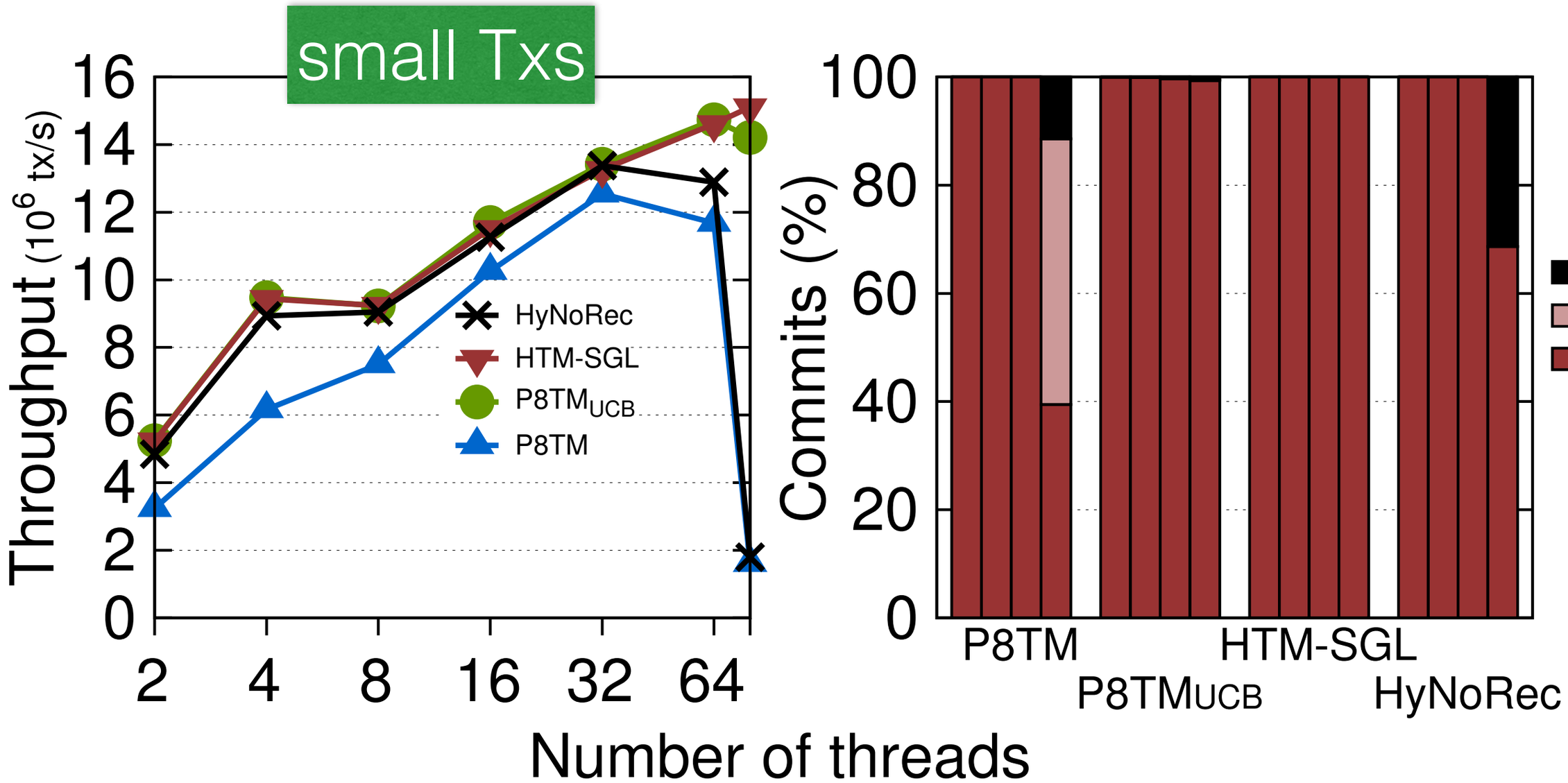




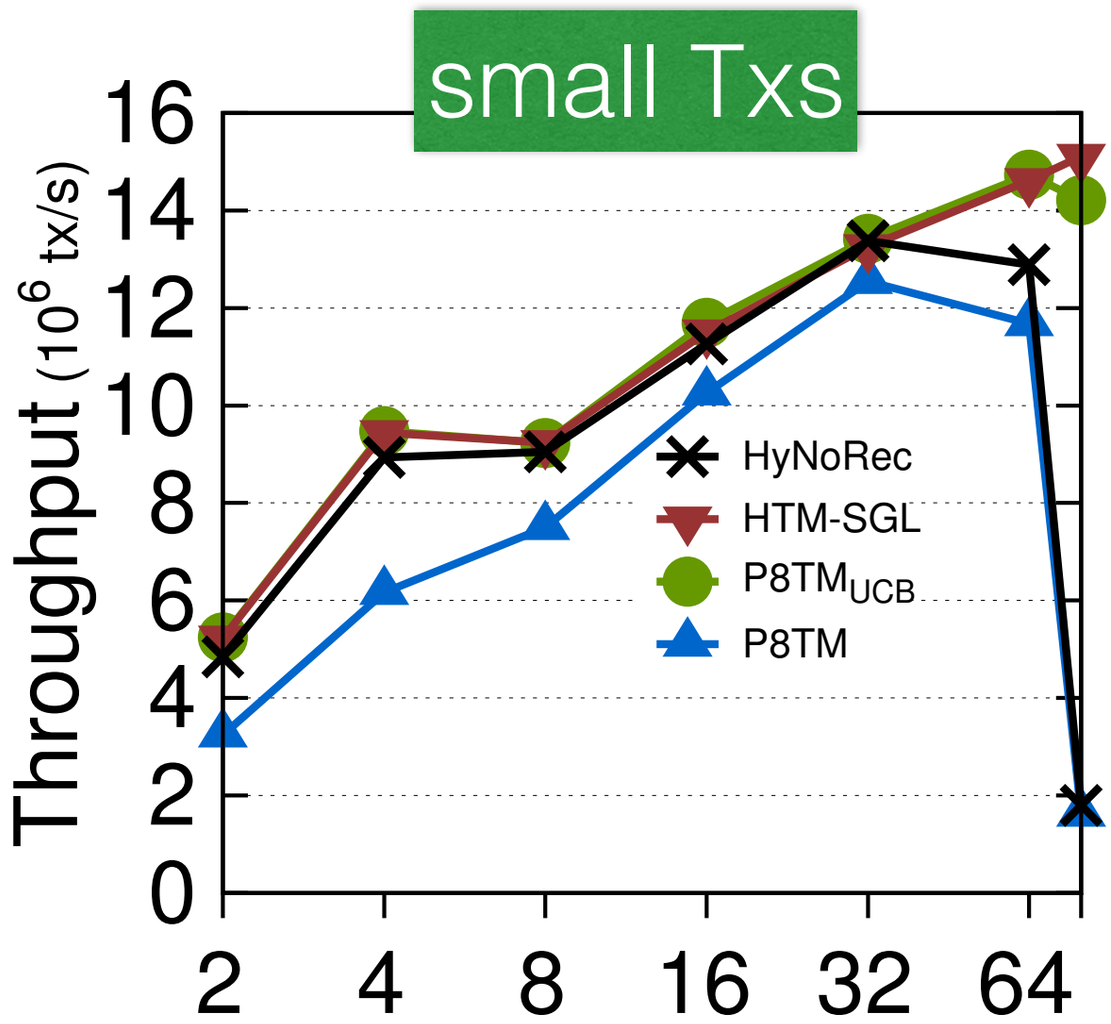
Evaluation: SSCA2

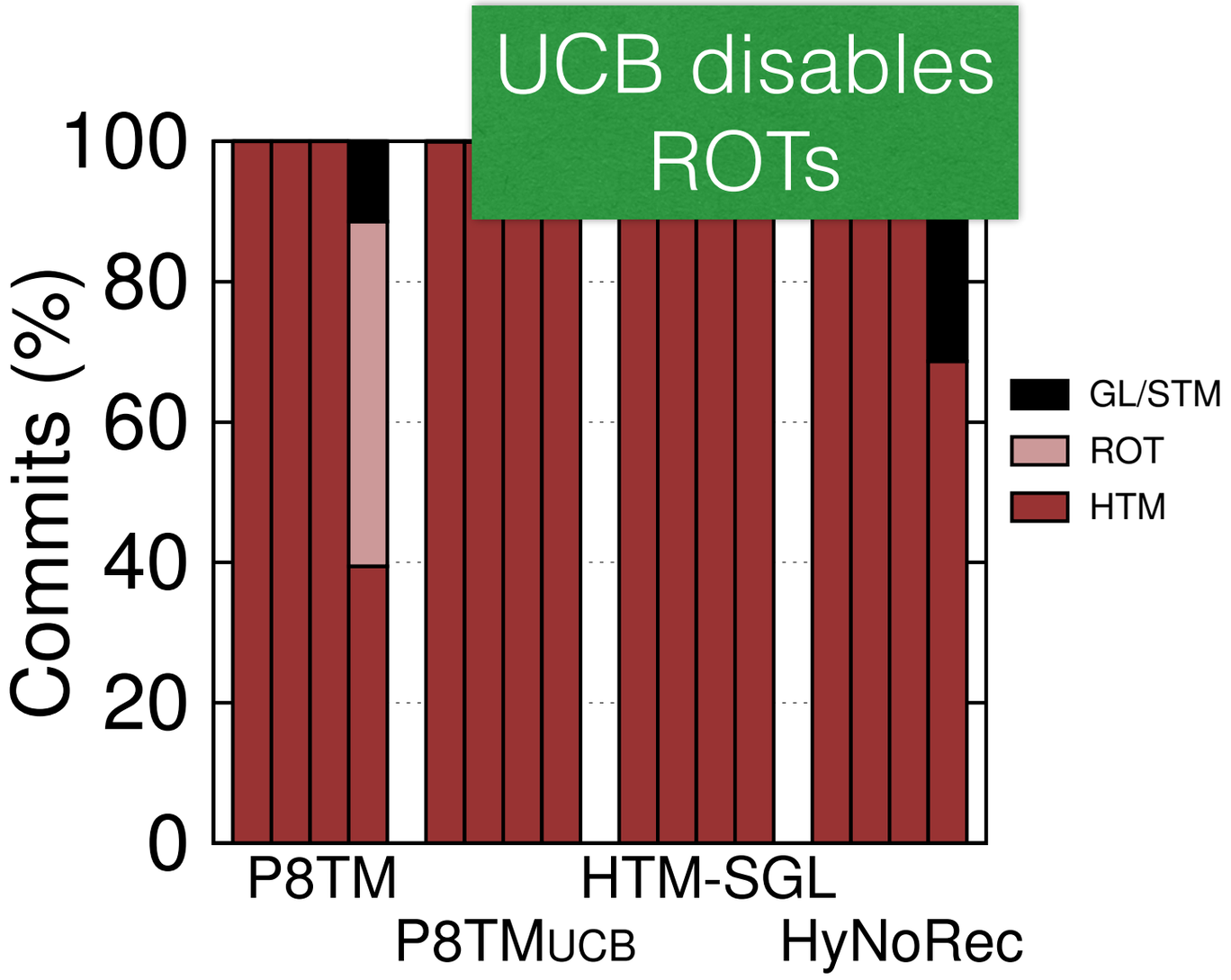


Evaluation: SSCA2



Evaluation: SSCA2



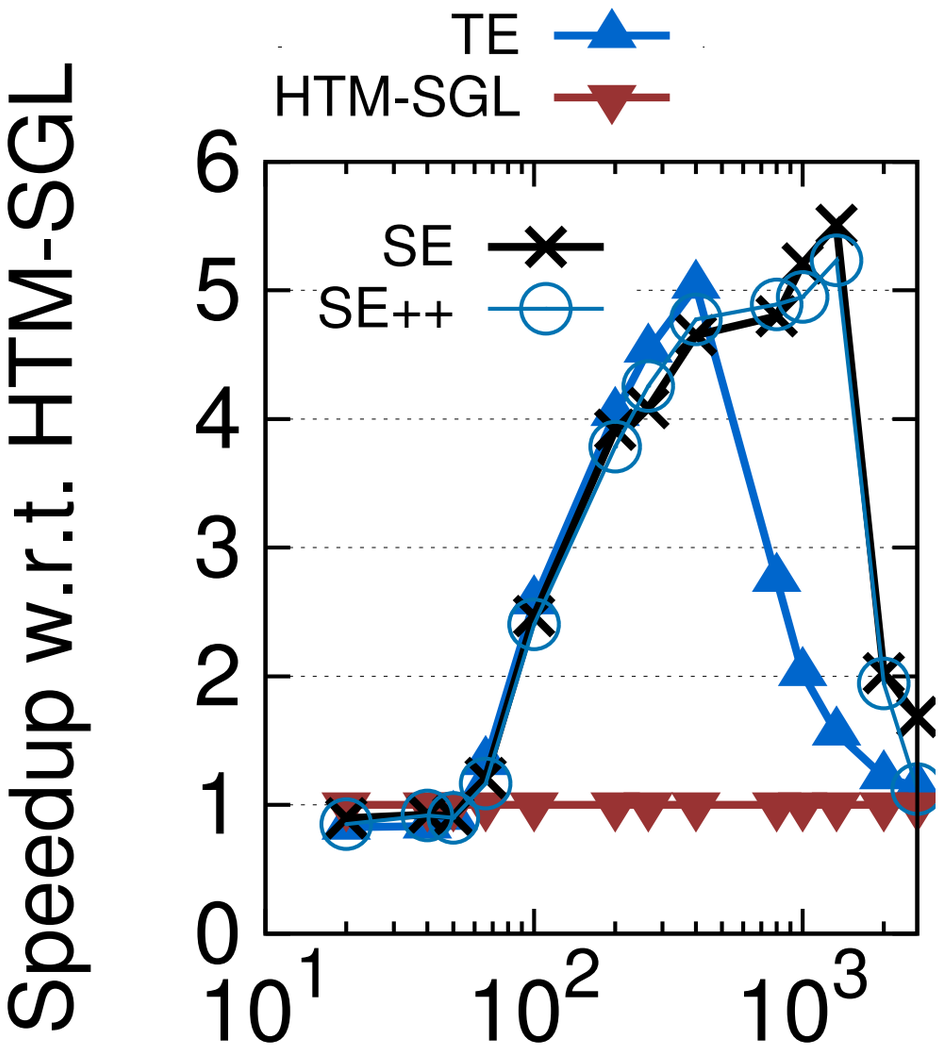


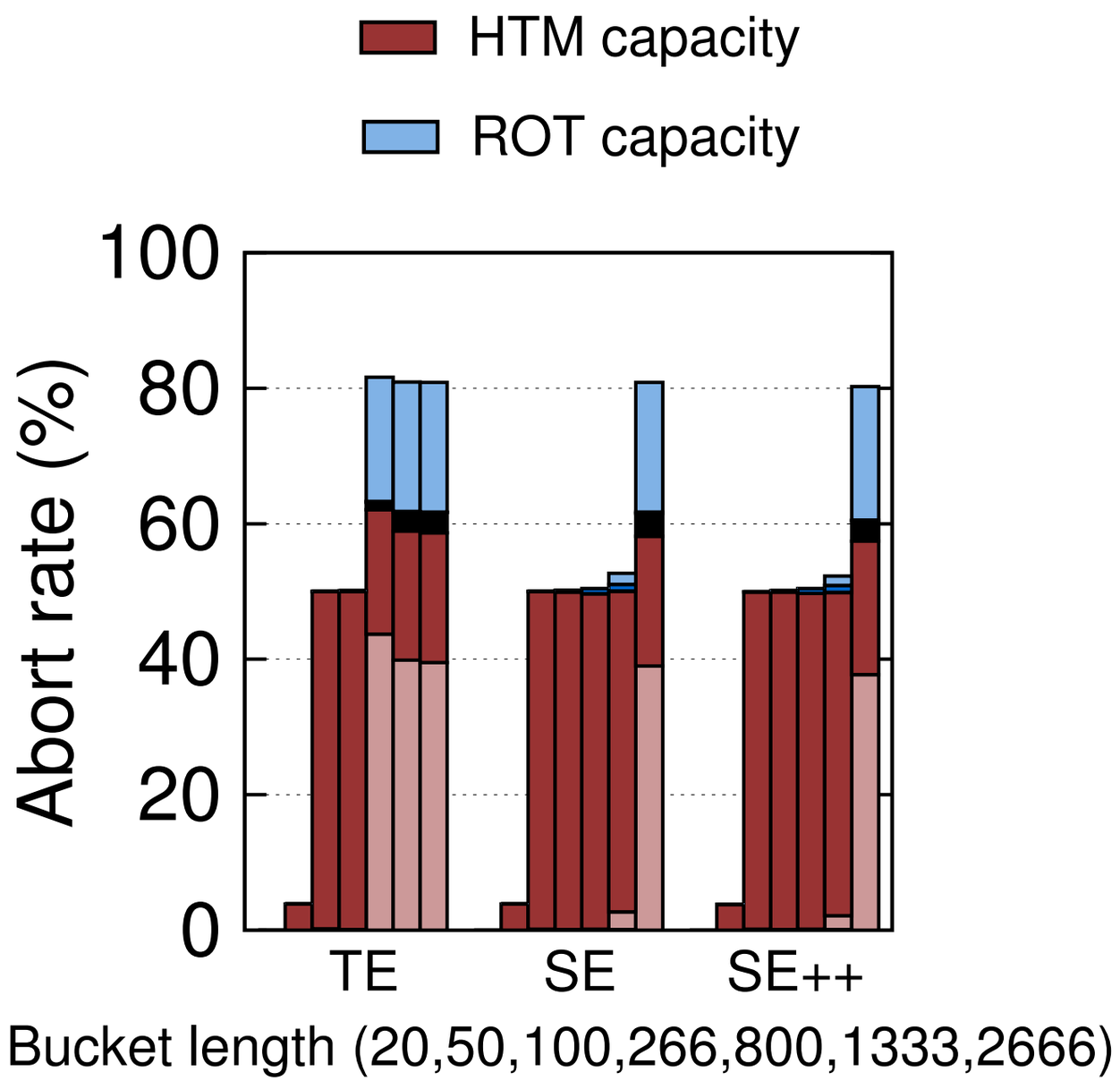
Number of threads

Conclusion

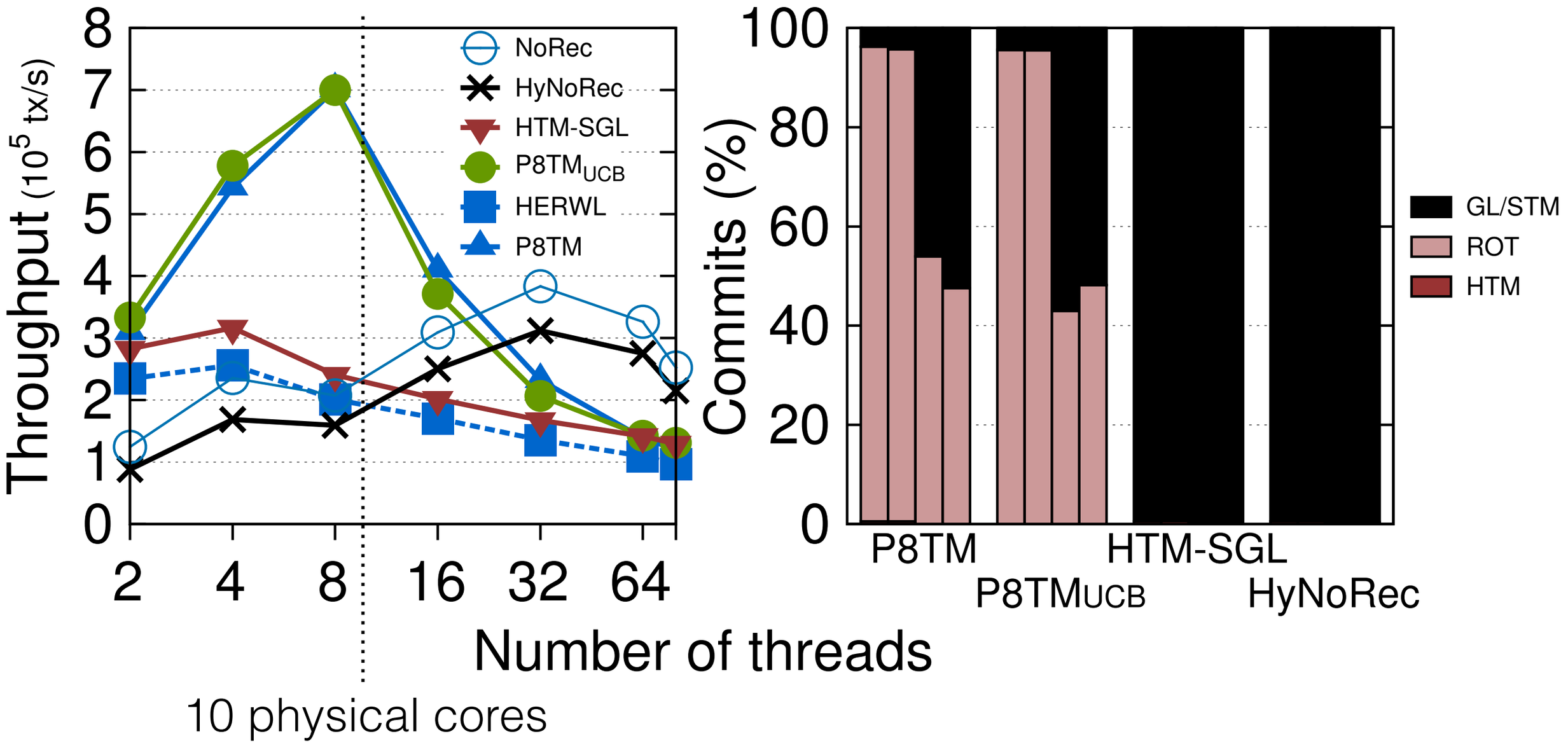
- POWER8-TM was able to exploit ROTs and suspend/resume to expand the capacity limitations
- TMCAM aware read-set tracking was necessary
- Self-tuning was effective in adapting to different workloads
- POWER8-TM promotes the importance of such features that can be used in innovative techniques to mitigate hardware limitations

Results: read-set tracking

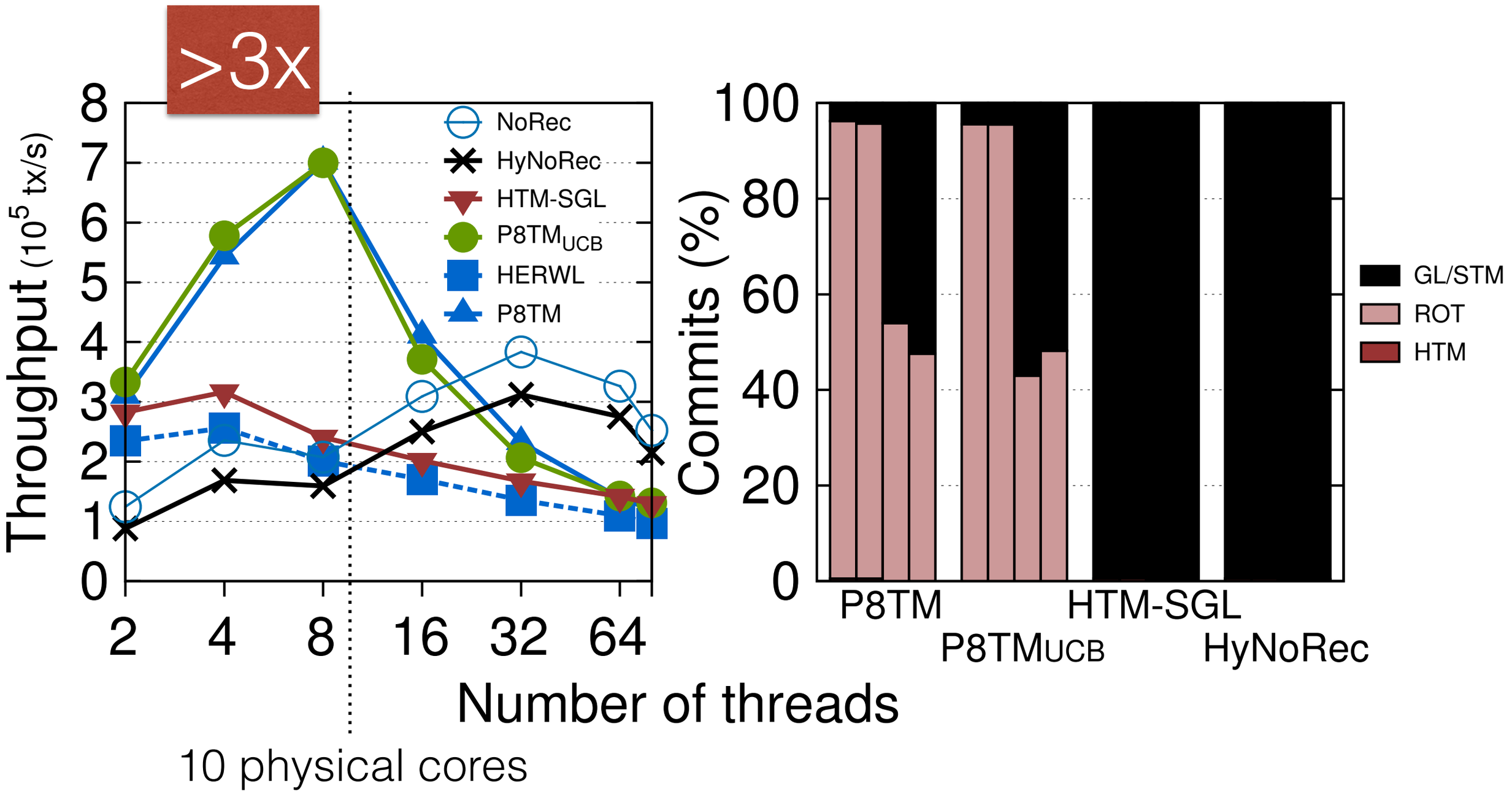




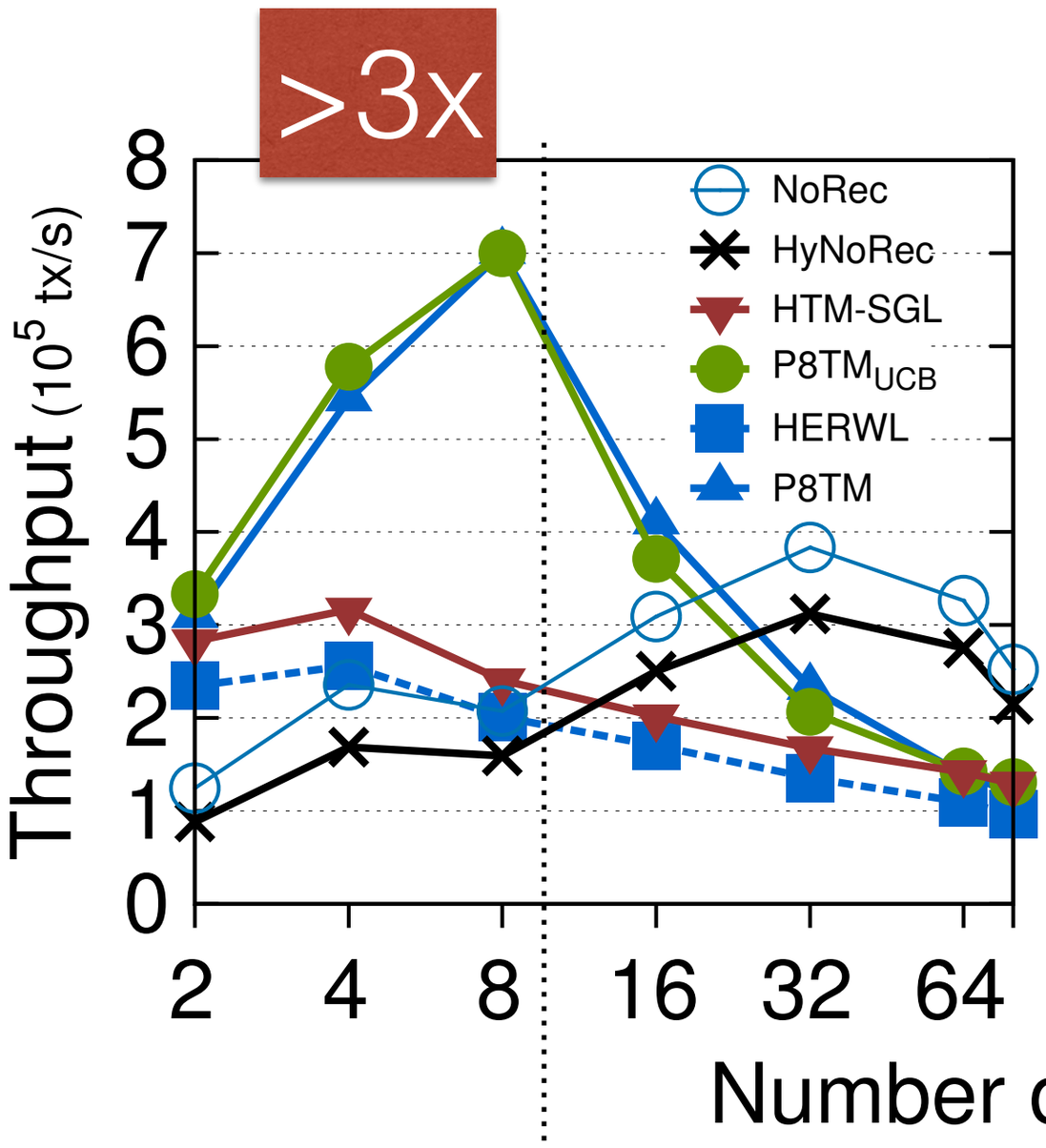
Evaluation: Vacation

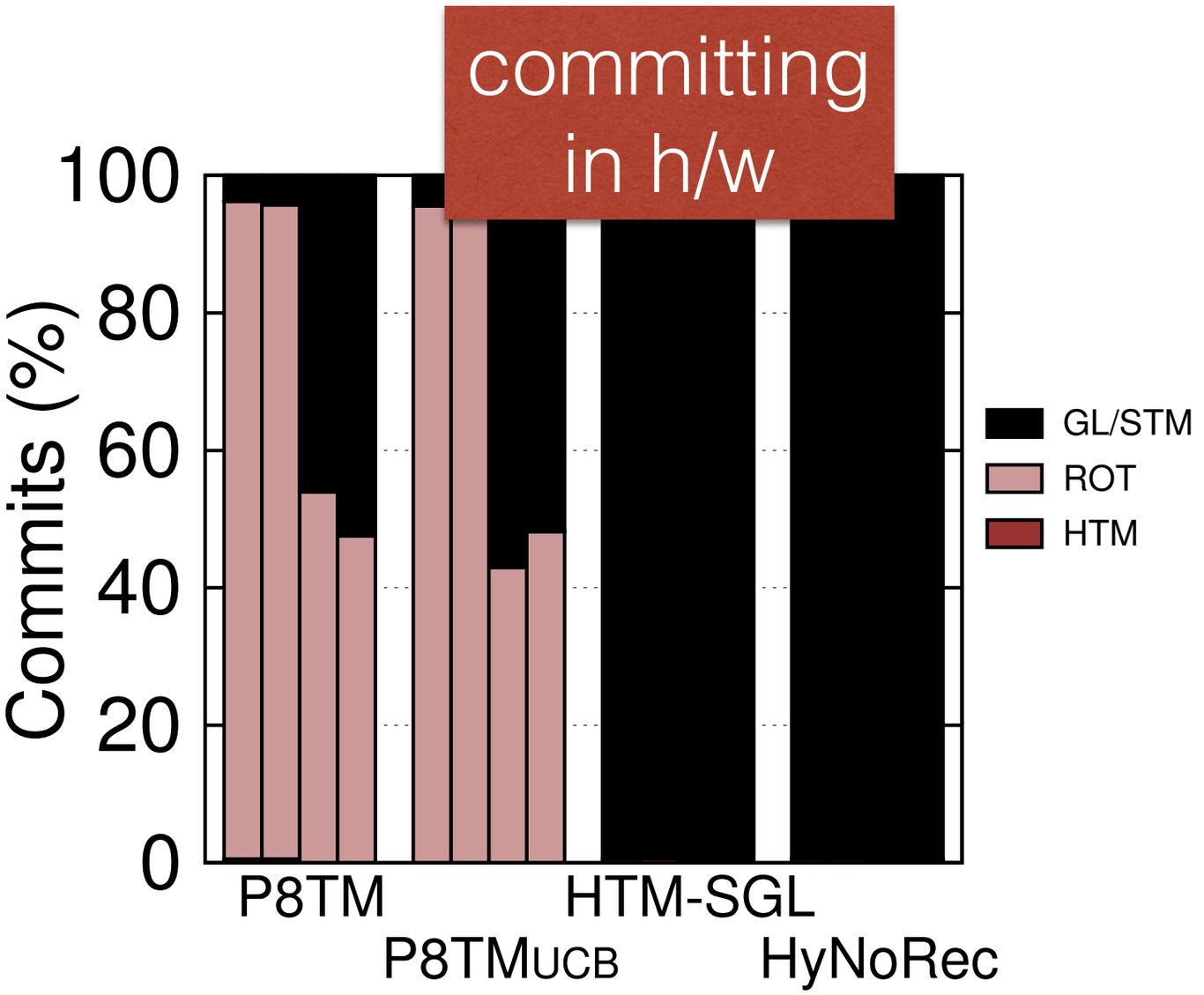


Evaluation: Vacation

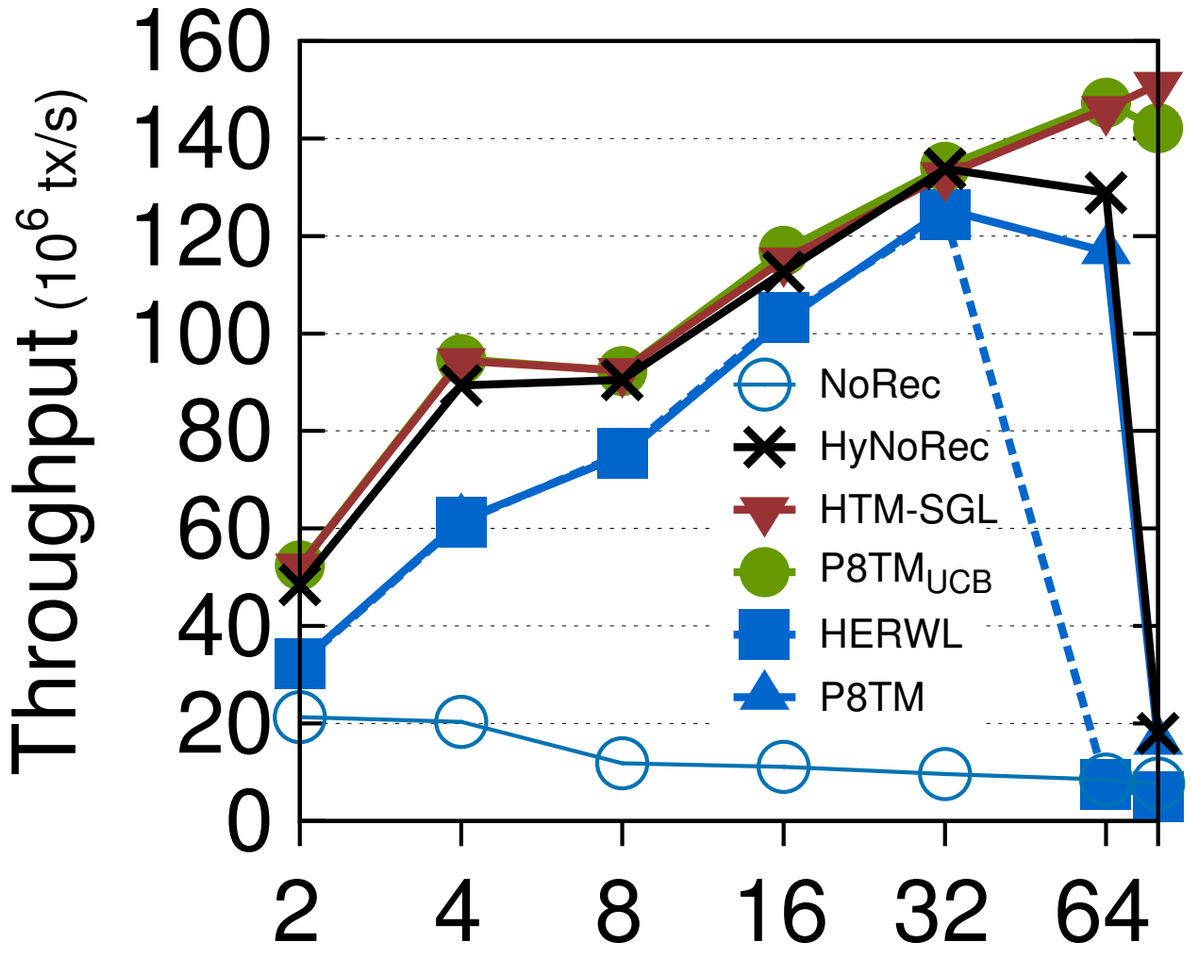


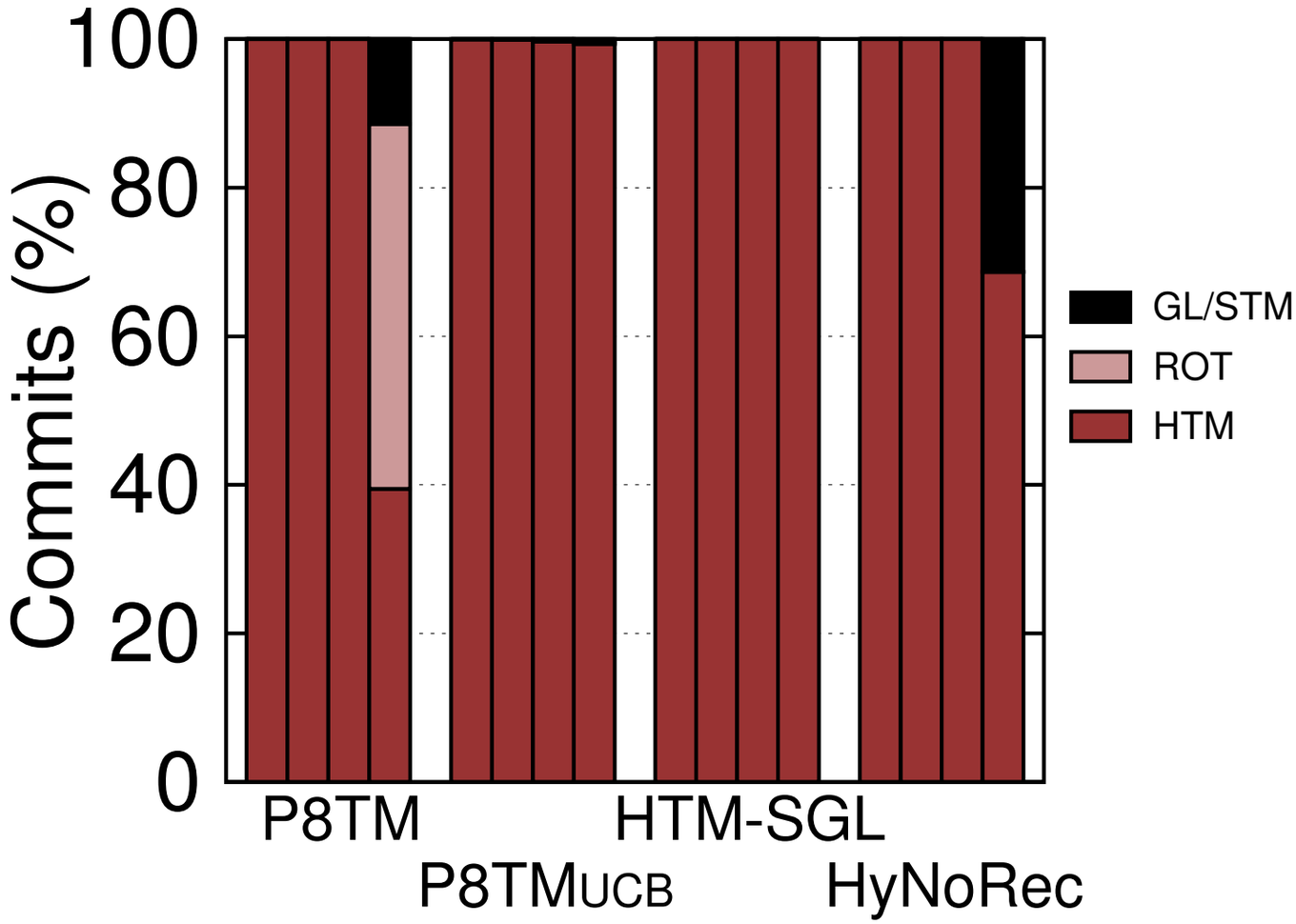
Evaluation: Vacation



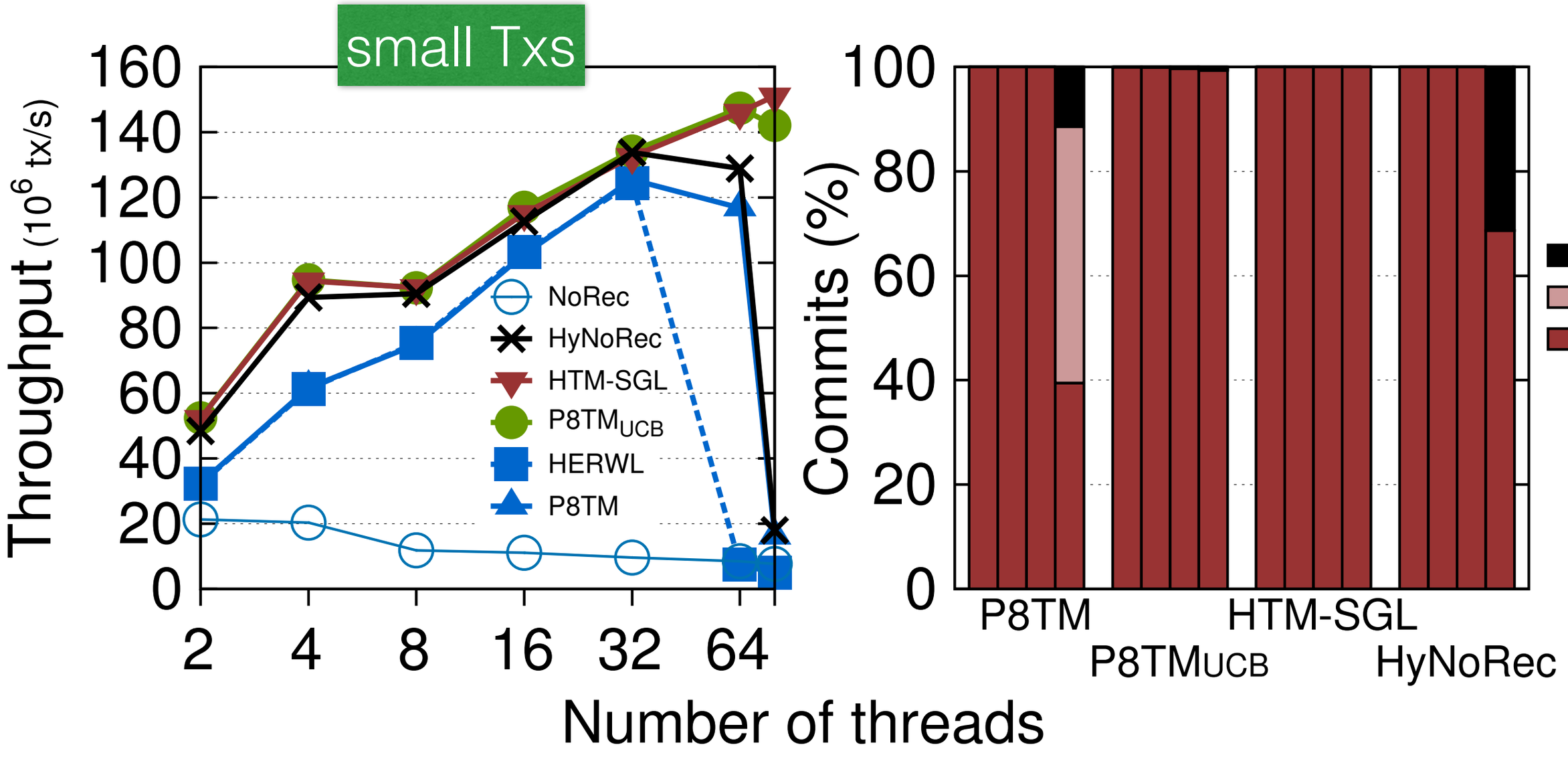


Evaluation: SSCA2





Evaluation: SSCA2



Evaluation: SSCA2