Remote Invalidation: Optimizing the Critical Path of Memory Transactions

Ahmed Hassan, Roberto Palmieri, Binoy Ravindran

Virginia Tech

{hassan84, robertop, binoy}@vt.edu

IPDPS 2014

28th IEEE International Parallel & Distributed Processing Symposium

Virginia Tech

Synchronization using locks

- Coarse-grained Locking:
  - Easy to implement, good for low number of small threads.
  - **But** minimizes concurrency.
- Fine-grained Locking:
  - Allows more concurrency.
  - **But** error prone.
- How to solve this trade-off?? Transactional Memory.

Transactional Memory

- Use an underlying TM framework to guarantee consistency, atomicity, isolation, deadlock freedom, …

```
Thread 1

@Atomic
foo1()
{
   if(B.balance > 500)
        A.balance+=500;
        B.balance-=500;
}
```

- Programmable (like coarse-grained locking).
- Allows concurrency (like fine-grained locking).

Transactional Memory

- Software Transactional Memory (STM):
  - Everything is controlled by SW.
  - Portable to any HW.
- Hardware Transactional Memory (HTM):
  - Rely on a specific HW features (e.g. a modified cache coherent protocol).
- Hybrid Transactional Memory (Hybrid TM):
  - HTM transactions fall-back to STM

Transactional Memory Gains Traction!!

- Intel Haswell Processor: TSX Extensions.
- IBM and AMD.
- STM support in GCC (4.7).

Motivation: Issues in STM

- Progress guarantees, support for nesting, interaction with non-transaction code, irrevocable transactions, …
- However, performance and scalability remain as the most important issues.
- Main STM overhead: meta-data handling.
  - Handling meta-data adds an overhead for any STM algorithm (with respect to serial execution), even for single-thread execution.

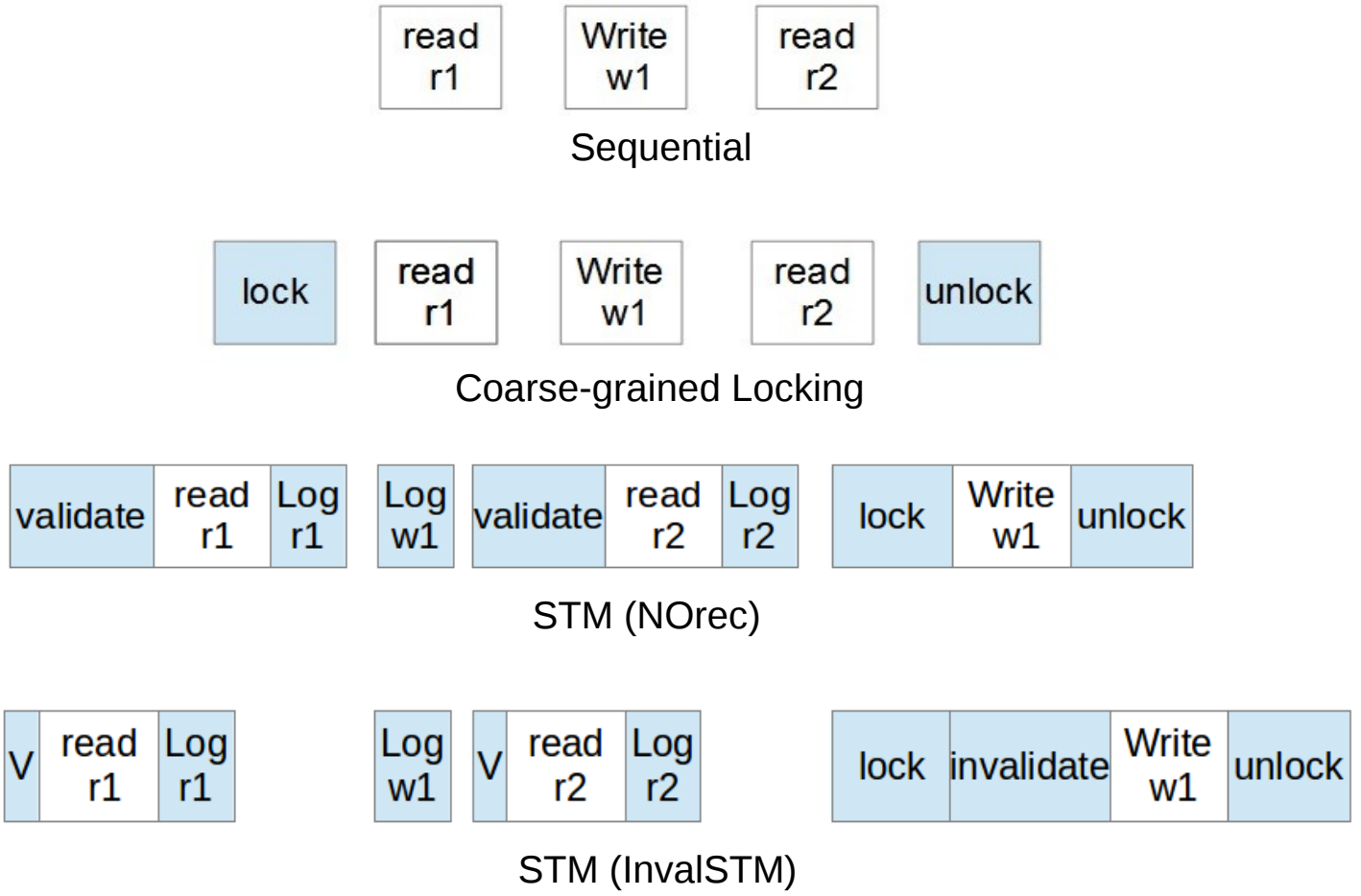
Transaction's critical path

- Operations in the path of STM transaction's execution:
  - Logging reads and writes.
  - Validation.
  - Locking.
  - Commit
  - Abort (contention Management).

- These parameters interfere with each other.

- Reducing the negative effect of one parameter (e.g., validation) may increase the negative effect of another (i.e., commit), resulting in an overall degradation in performance for some workloads.

Transaction's critical path

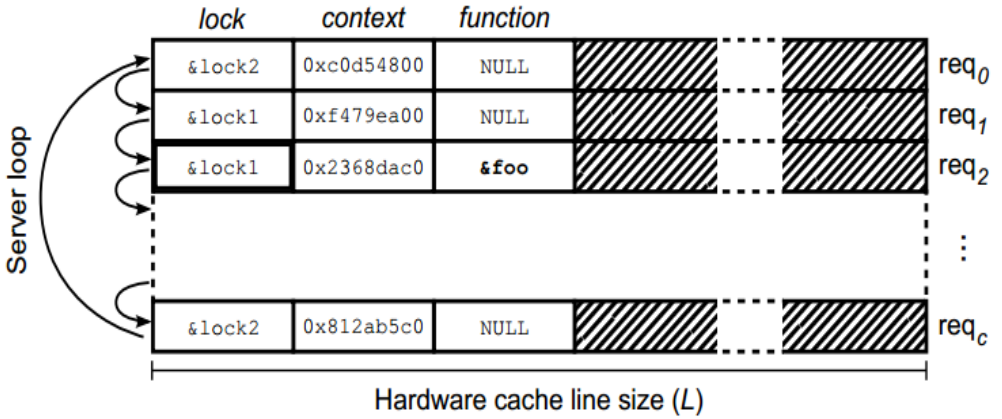


Critical path overheads: locking

- Granularity: Various algorithms with different design decisions.
  - Single lock, bloom filters, ownership records, ...
- Mechanism: Most STM frameworks use spin locking.
- In Spin locking, all threads spin on the same shared locks:
  - Cache misses.
  - Harmful CAS operations.

Remote Core Locking *(Lozi et al. – ATC'12)*

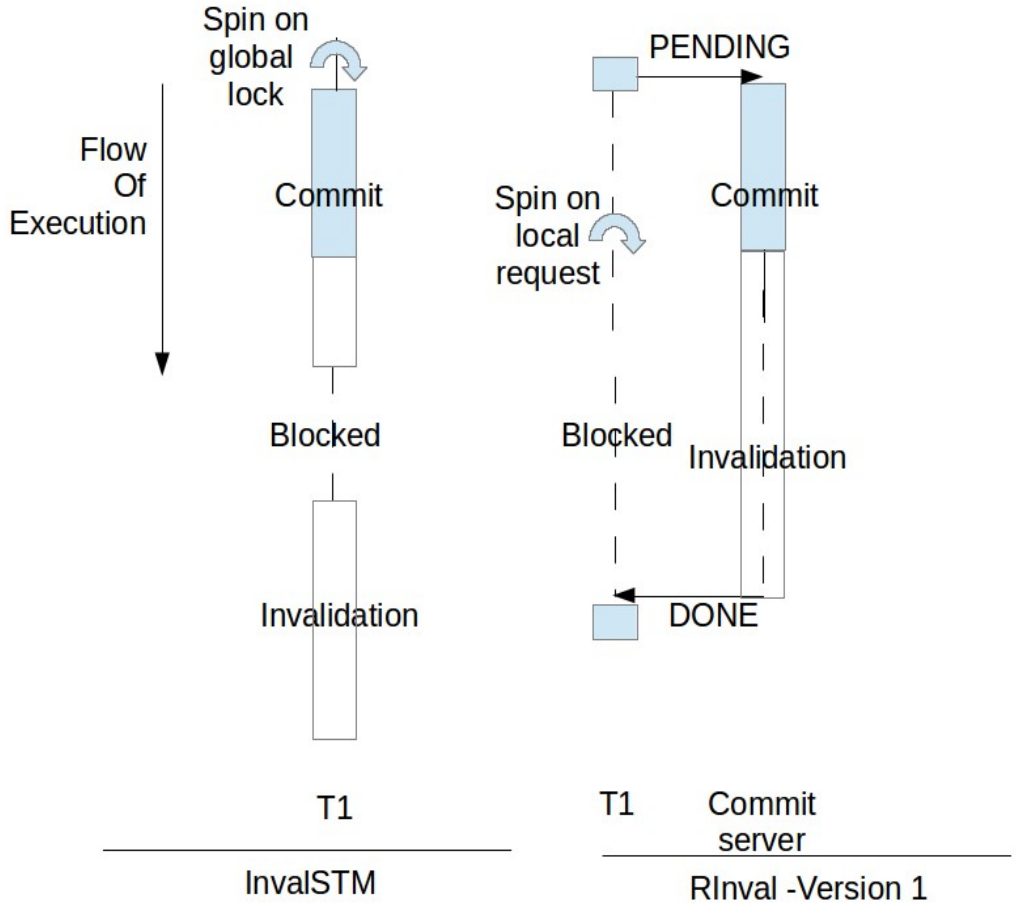
- Execute critical sections in dedicated server cores.
- Spin on local rather than shared variables.
- Reduce the overhead of:
  - Cache misses
  - CAS operations
- Issues:
  - Lock-based applications.
  - Complicated Servers.



From RCL [ATC'12]

Remote Invalidation: Version 1

- Replaces spin locking with RCL



Remote Invalidation: Version 1

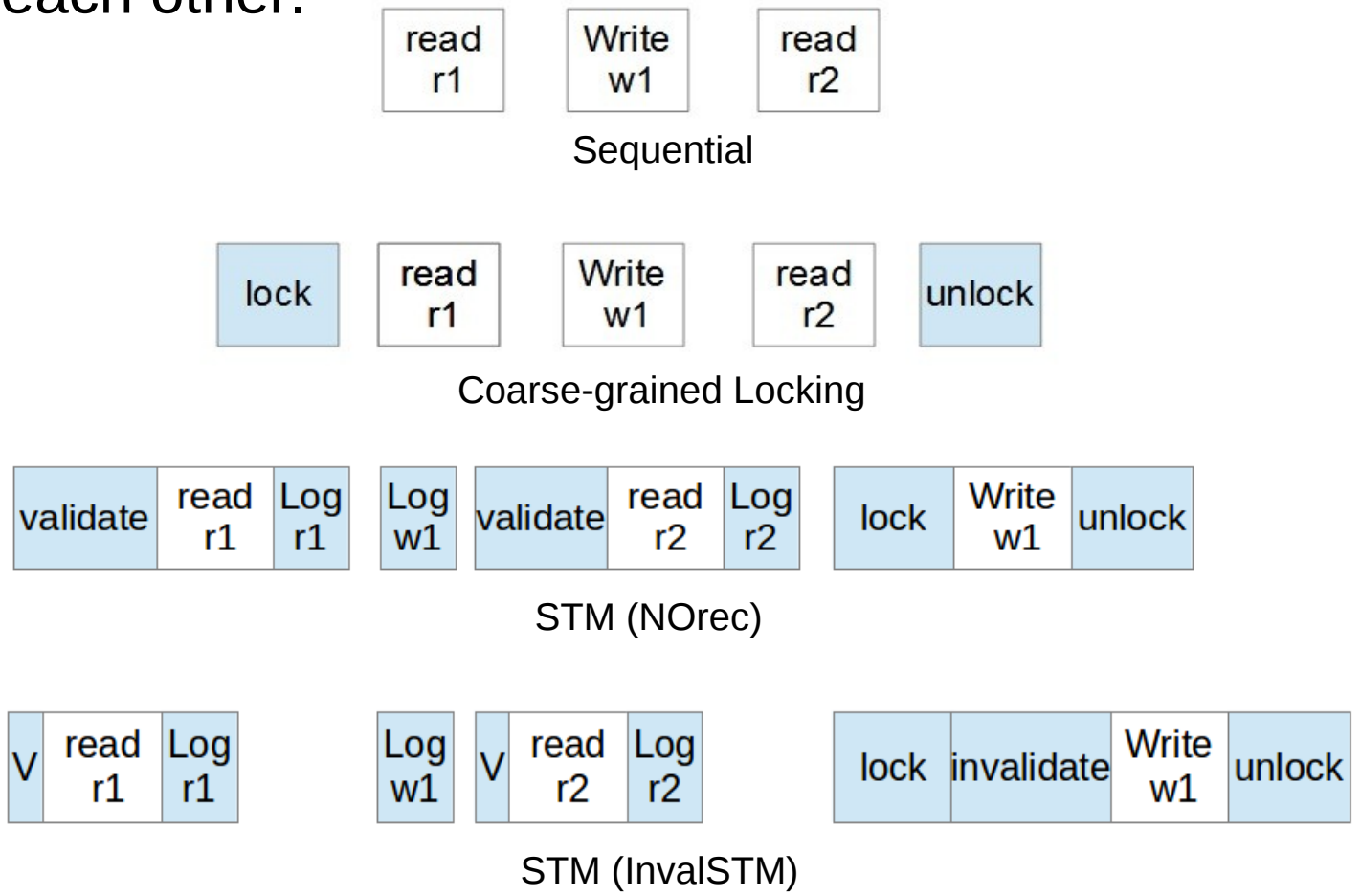
- InvalSTM + RCL = RCL benefits + …
  - Simpler server routines than RCL.
    Critical section is well defined (InvalSTM's commit).
  - Only one server.
    InvalSTM uses a single global lock at commit.
  - No CAS operations at all!!
    Both InvalSTM and RCL uses CAS operations.
  - Allows invalidation routines to run in parallel.
    By adding invalidation servers (without adding any CAS operations)

Critical path overheads: Validation vs Commit

- Validation overhead and commit overhead interleave with each other.

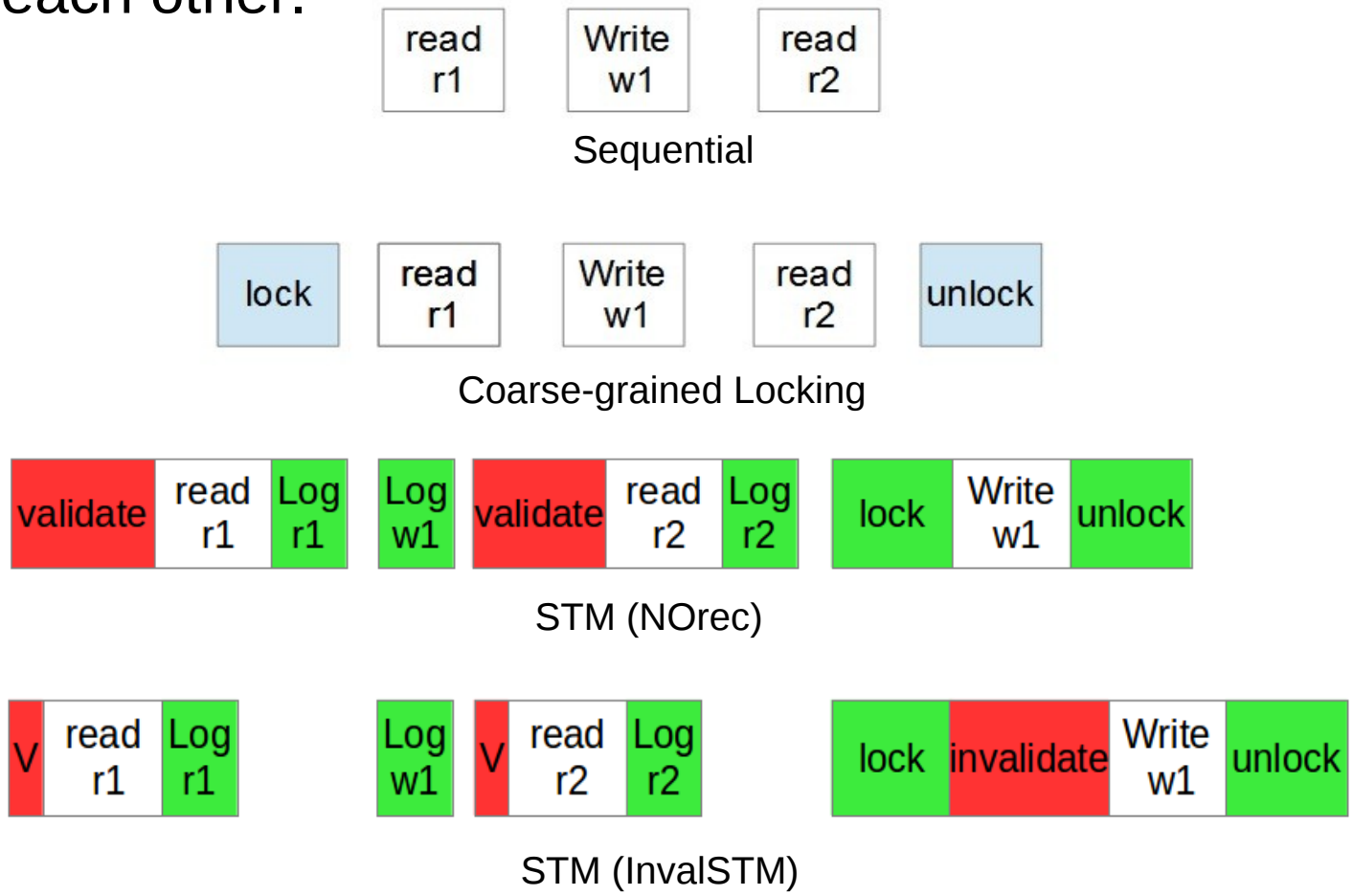
Critical path overheads: Validation vs Commit

- Validation overhead and commit overhead interleave with each other.

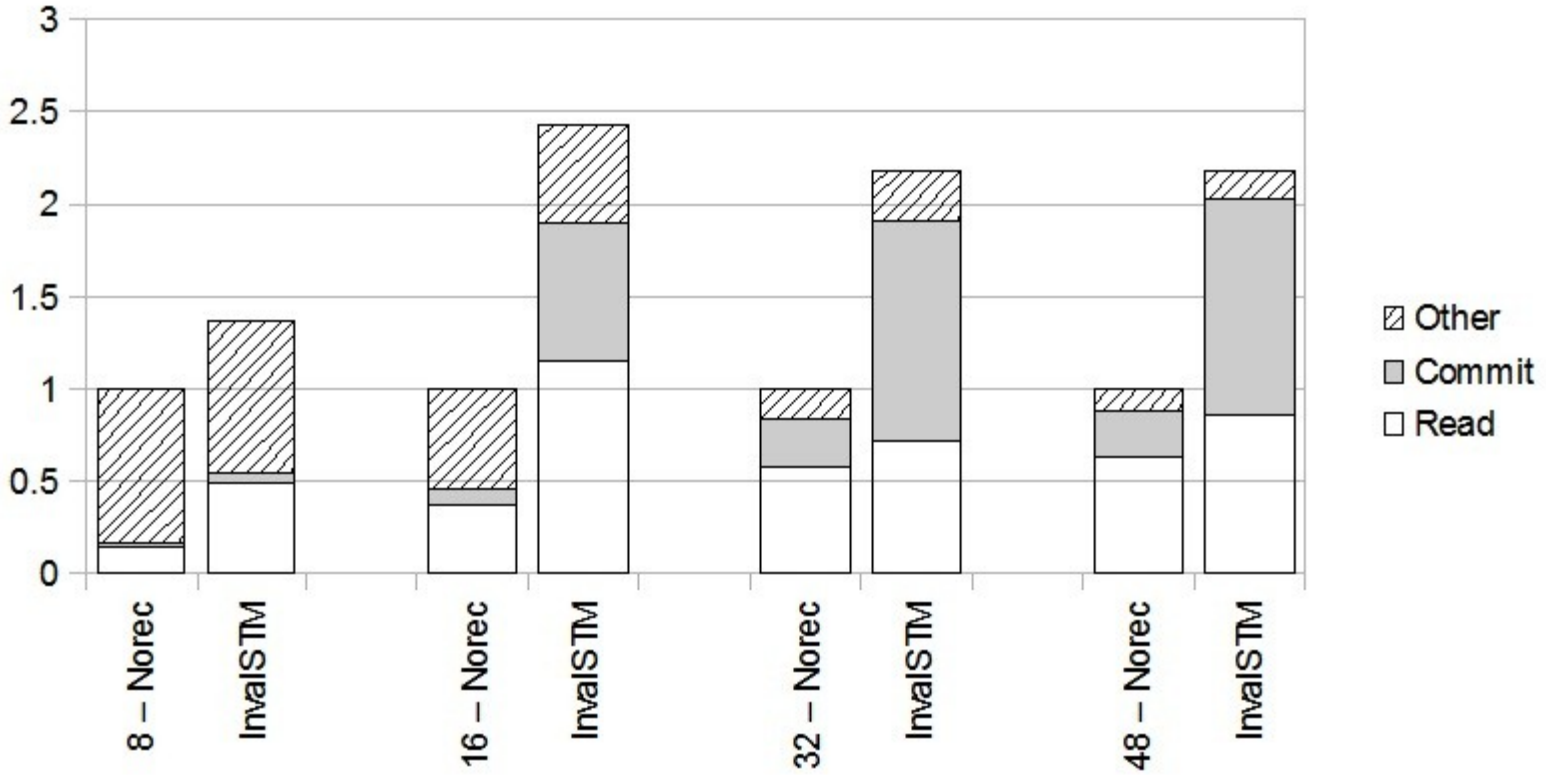


Critical path overheads: Validation vs Commit

- Validation overhead and commit overhead interleave with each other.

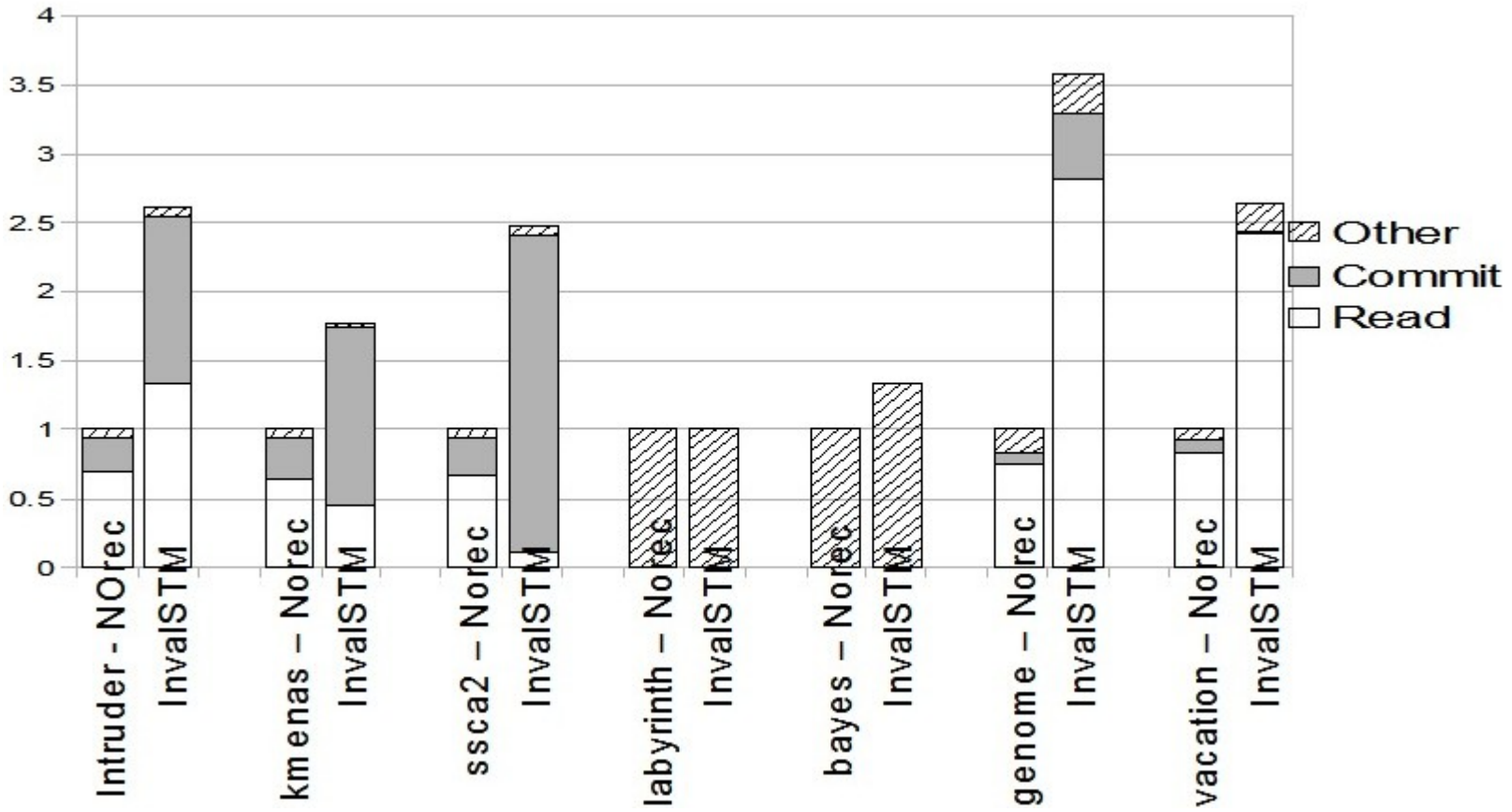


Critical path overheads: Validation vs Commit



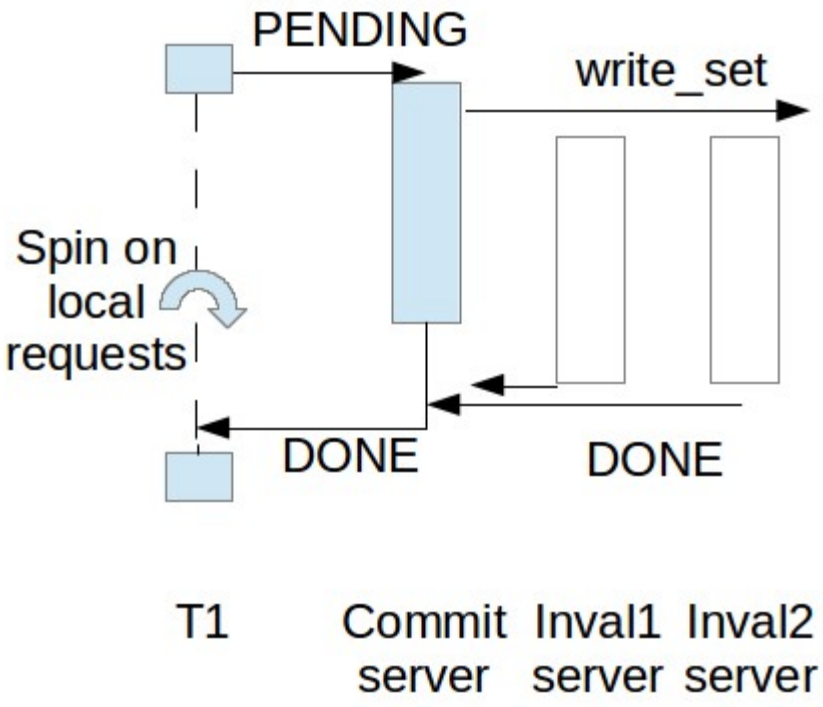
Percentage of overheads on RB-Tree (normalized to NOrec)

Critical path overheads: Validation vs Commit



Percentage of overheads on STAMP (normalized to NOrec)

Remote Invalidation: Version 2

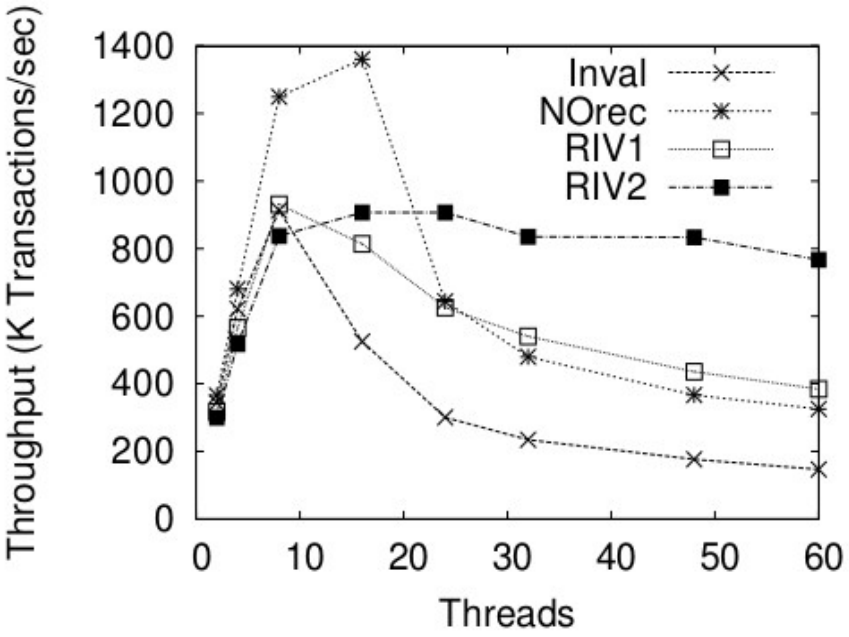


Remote Invalidation: Version 3

- In Version 2, commit server waits for all invalidation servers before handling new requests.

- In Version 3:
  - Requests whose invalidation servers finish their execution can be handled immediately.
  - Allows commit server to be n steps ahead of the invalidation servers.
  - More robust is special cases (like OS descheduling of invalidation servers).

Performance Evaluation

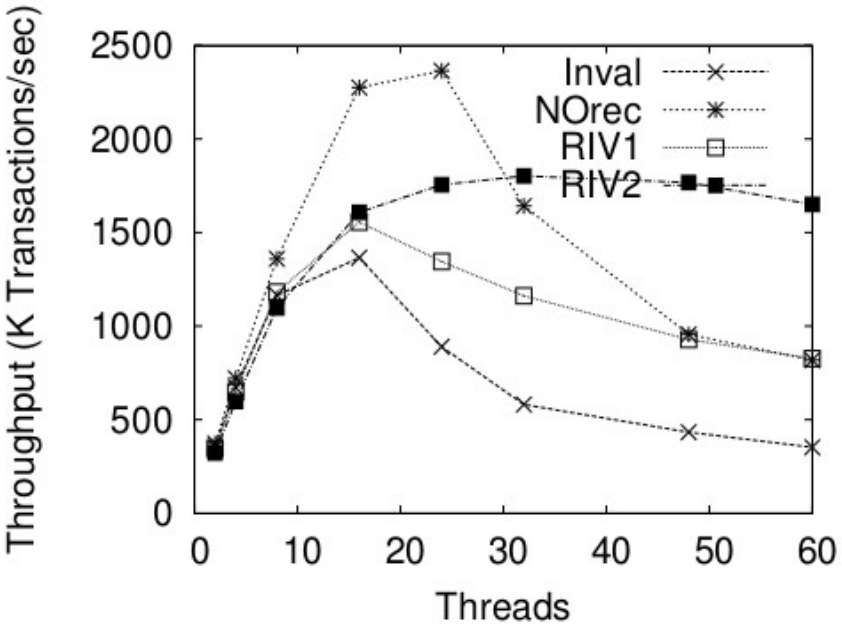


RB-Tree – 64K nodes

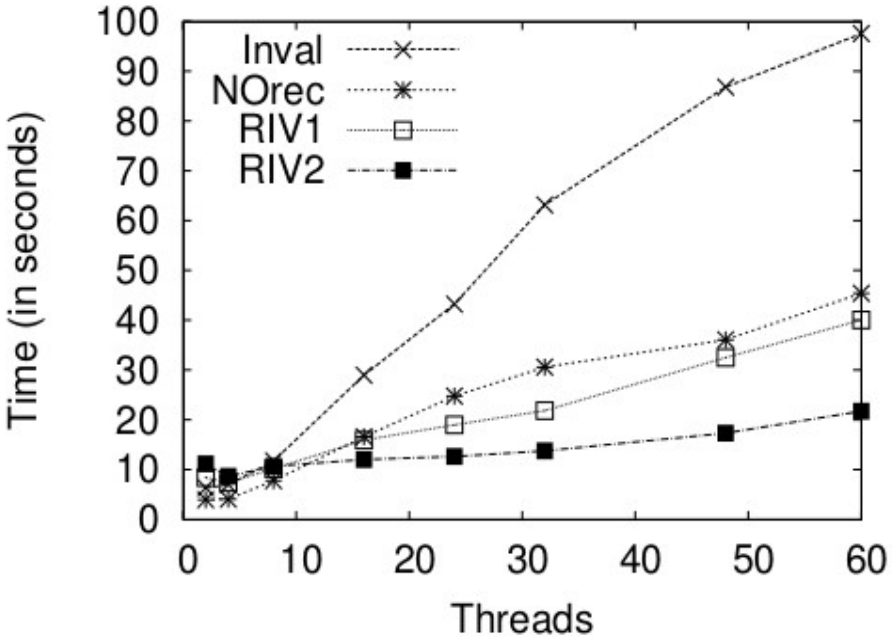
50% reads

RB-Tree – 64K nodes

80% reads

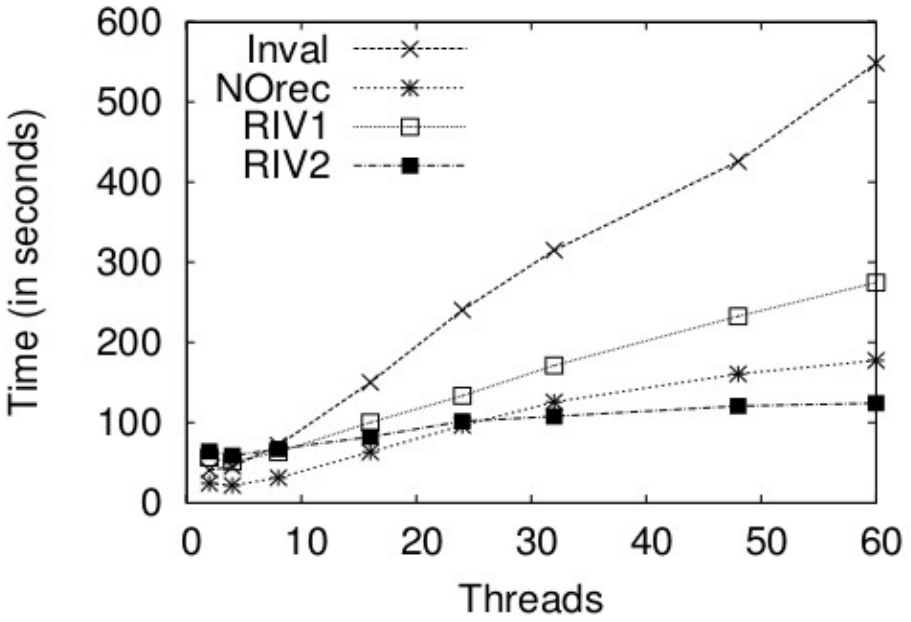


Performance Evaluation



Kmeans

Intruder



Conclusions

- STM is a promising alternative to lock-based applications.
- Overheads in the critical path of STM transactions: logging, locking, validation, commit, abort.
- Remote Invalidation:
  - Replaces spin locking with efficient remote core locking.
  - Optimize validation/commit trade-off by running commit and invalidation routines in parallel on different servers.

Thanks!

{hassan84, robertop, binoy}@vt.edu

http://www.hyflow.org

Questions