Deconstructing RDMA-enabled Distributed Transactions: Hybrid is Better!

Xingda Wei, Zhiyuan Dong, Rong Chen, and Haibo Chen

Shanghai Jiao Tong University

OSDI'18

Background

- There are two RDMA styles for transactions: One-sided and Two-sided
- It’s not obvious which one is “better”
  - Address known vs. Lookup
  - Message size can affect performance
  - RNIC/CPU coupling

One-sided vs. two-sided primitives (Verbs)

- One sided: READ, WRITE, ATOMIC
- Two sided: SEND, RECV
  - Server polls requests from a receiver queue, calls local RPC routine, and posts results back to the sender queue

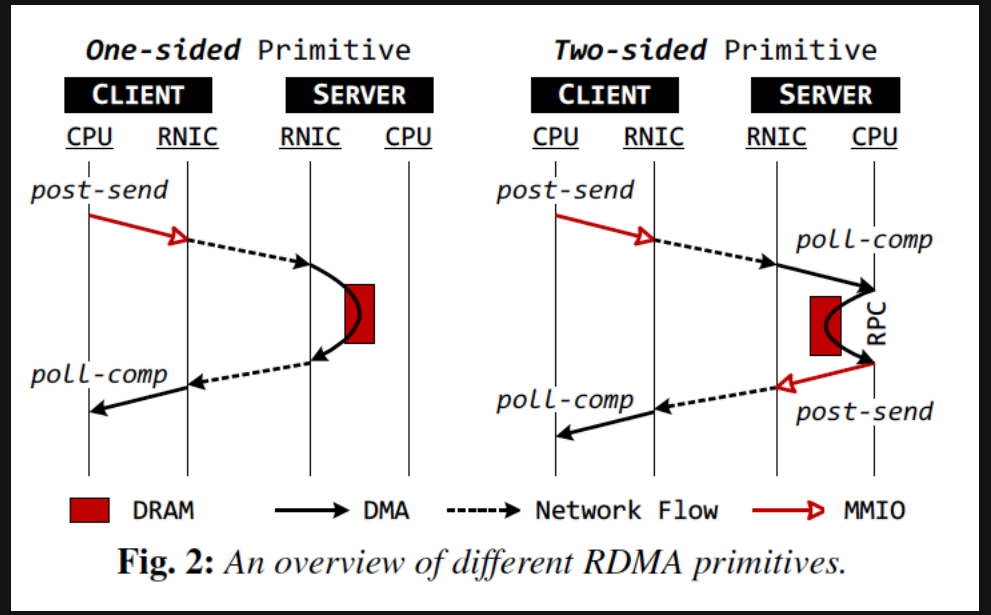


Fig. 2: *An overview of different RDMA primitives.*

OCC (Optimistic Concurrency Control)

- “Optimistic” - Assumes other requests won’t clash
- Runs transactions without locks, only checks at the end
  - Why? Locks induce latency
  - Most transactions don’t conflict
- Good for OLTP (online transaction processing)

OCC: 4 Phases

- Execution - Records a read-set (keys + versions) and a write-set (keys + new values)
- Validation - Executes a commit protocol, which locks the records in the write set and validates the records in the read set is unchanged
- Logging - If there is no conflicting transaction, the coordinator sends transaction's updates to each backup and waits for the accomplishment
- Commit - Upon successful, the transaction will be committed by writing and unlocking the records at the primary node

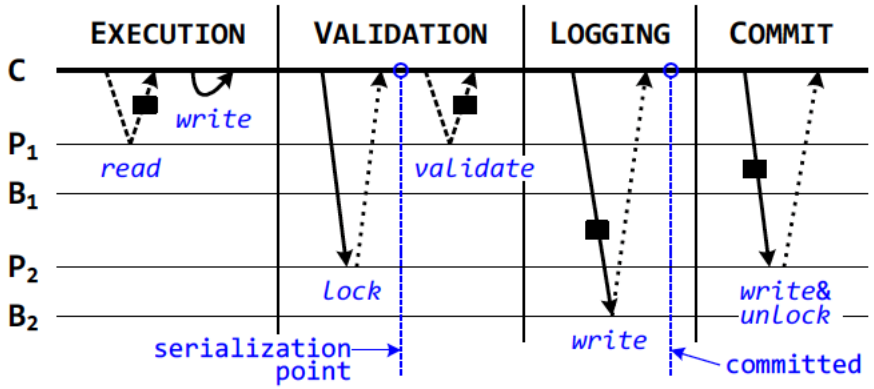


Fig. 1: *A phase-by-phase overview of transaction processing with OCC. **C**, **P**, and **B** stand for the coordinator, the primary and the backup of replicas, respectively. P_1 is read and P_2 is written. The dashed, solid, and dotted lines stand for read, write, and hardware ack operations, and rectangles stand for record data.*

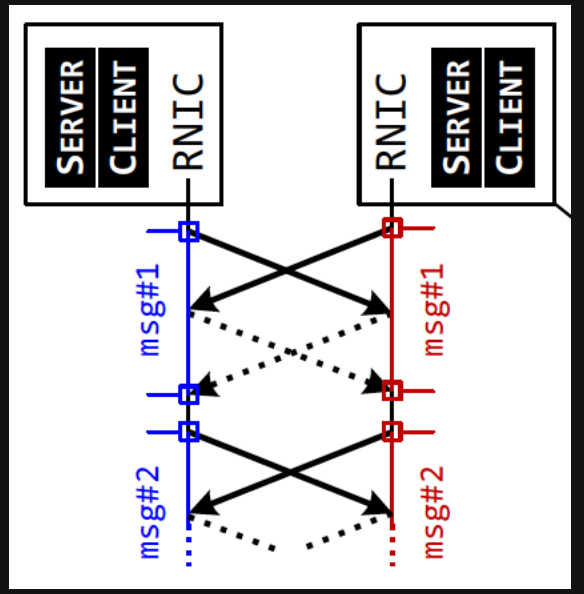
OCC: 2-Phase Read-only Transaction

- Read - Reads all records
- Validation – Validates all of them have not been changed

Execution Model

Controls - Cluster and model

- Symmetric Model: Each machine is both client and server
  - This removes asymmetry/skew as a confounder
- Register memory with huge pages (2 MB) for RNIC
  - This reduces page-translation cache misses
  - As a result, fewer PCIe reads on TLB misses



Controls - QP creation

- A per-thread device context for QP creation rather than a shared context
- Why?
  - Mellanox's driver allocates a pre-mapped MMIO buffer per context
  - A shared context forces QPs to share that buffer, causing synchronization on the MMIO buffer and up to 63% throughput drop as threads scale
    - (mlx4 has 7 dedicated buffers + 1 shared buffer)

Baselines

- Use standard Verbs API
- One-sided:
  - Each thread manages *n* Reliable Connected (RC) QPs to *n* peers
  - Inline <= 64B; overlap outstanding ops
- Two-sided:
  - SEND/RECV verbs over Unreliable Datagram (UD) QPs
    - Has better performance in symmetric settings
    - One-sided based RPC unlikely to outperform UD based RPC especially for small messages
    - For RPC communications, two WRITES are required (one for send and one for reply)
- Justify comparing RC to UD because RDMA network assumes a lossless link layer

Optimizations

Coroutine (CO)

- Run a small set of coroutines per thread; each coroutine issues RDMA ops, then yields while waiting, letting others run
- Hides µs-scale RNIC/network latency by pipelining requests across transactions; boosts per-core throughput
- Small number is sufficient (e.g. 8)
- Execution & Validation – multiple remote ops

Outstanding Requests (OR)

- Issue multiple requests from the same transaction in parallel instead of waiting for each to finish (per-transaction pipelining)
- Better RNIC utilization and lower end-to-end transaction latency; eliminates per-op serialization
- Read/write set of many OLTP transactions can be known in advance. So it’s possible to issue these reads and writes in parallel
- Execution (multi-GET/multi-read)
- Commit/Logging – writing multiple records

Doorbell Batching (DB)

- Use one doorbell MMIO to let the NIC DMA-fetch a batch of WQEs, instead of many MMIOs
- MMIOs are expensive (hundreds of cycles); batching cuts CPU MMIOs and PCIe transactions
- Better usage of PCIe bandwidth
- Easier for two-sided UD
- Use w/ high request rates to same peer

Passive ACK (PA)

- Take ACKs off the critical path
  - One-sided: mark requests unsignaled; confirm their completion later when polling the next signaled op
  - Two-sided: piggyback the reply to message #1 on the next request message (#2) in symmetric flows (“reply-on-request”)
- Cuts RNIC bandwidth/completions; in symmetric two-sided RPC, can halve reply messages
- Commit/Logging (ACK isn’t immediately required)

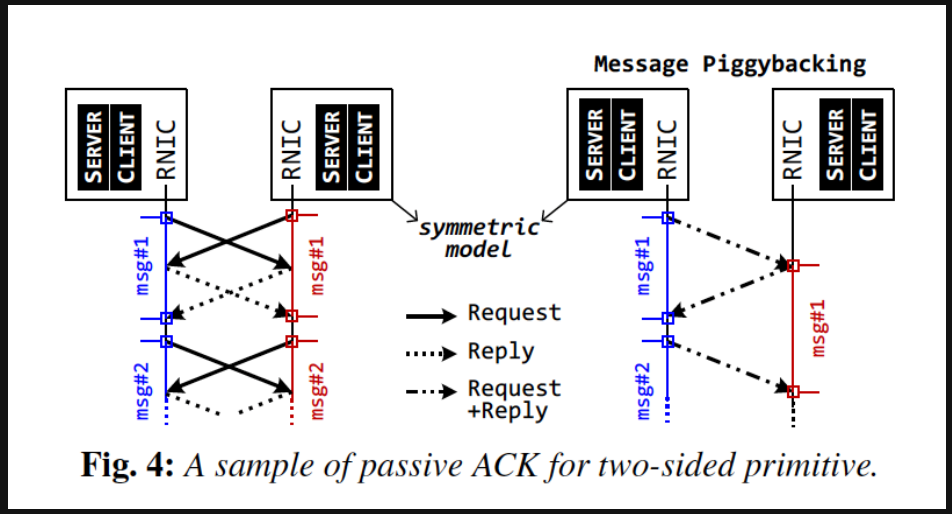


Fig. 4: *A sample of passive ACK for two-sided primitive.*

Table 2: *A summary of optimizations on RDMA primitives at different phases (§3.2).* ***OR***, ***DB***, ***CO*** *and* ***PA*** *stand for outstanding request, doorbell batching, coroutine, and passive ACK.* ***RW*** *and* ***RO*** *stand for read-write and read-only transactions.* `I` *and* `II` *stand for one-sided and two-sided primitives.*

		OR		DB		CO		PA	
		I	II	I	II	I	II	I	II
RW	E	✗	✗	✗	✓	✓	✓	✗	✗
	V	✓	✓	✓	✓	✓	✓	✗	✗
	L	✓	✓	✗	✓	✓	✓	✗	✗
	C	✓	✓	✓	✓	✓	✓	✓	✓
RO	R	✗	✗	✗	✓	✓	✓	✗	✗
	V	✓	✓	✗	✓	✓	✓	✗	✗

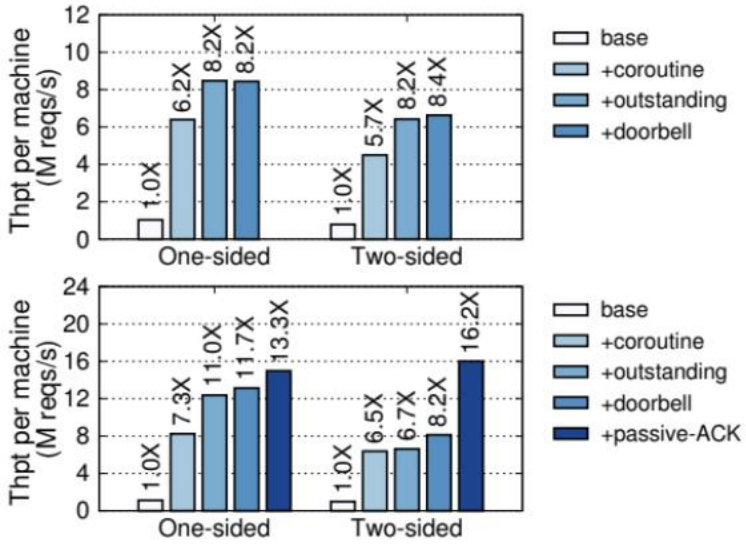


Fig. 6: *A comparison of one-sided and two-sided primitives for multiple-object (a) reads and (b) writes with 64-byte payloads.*

Benchmarks and Setup

- Workloads:
  - TPC-C (CPU intensive)
  - SmallBank (network-intenstive)
- Partitioned datastore: rows sharded across all machines
- High availability: 3-way logging/replication - each primary has two backups

Benchmark	Focus	Locality	Scale details
TPC-C/**no**	CPU-intensive	forced **distributed** (New-Order only)	384 warehouses / 16 machines
SmallBank	Network-intensive	**all networked**	100k accts/thread; **4%** of records accessed by **90%** txns

Phase by Phase Analysis – Execution & Validation

- Execution: One-sided/Cache
- Validation: One-sided

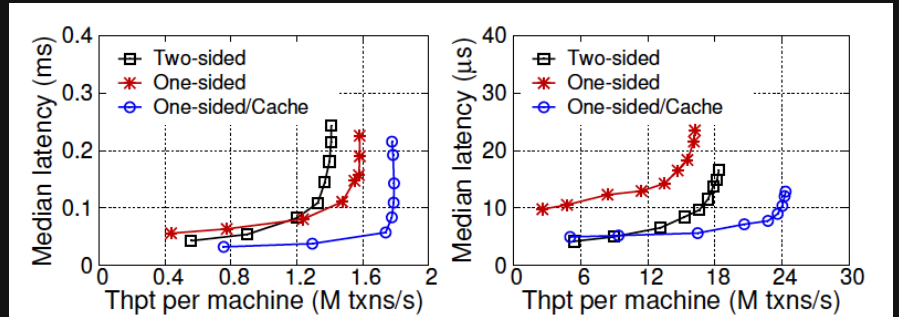


Fig. 7: *The performance of (a) TPC-C/no and (b) SmallBank with different implementations of Execution phase.*

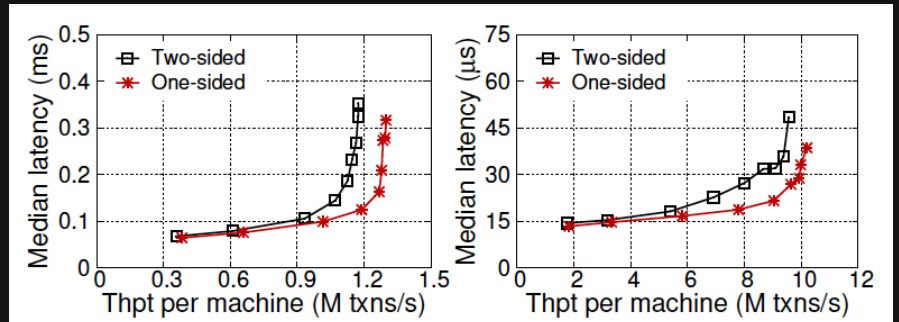


Fig. 8: *The performance of (a) TPC-C/no and (b) SmallBank with different implementations of locking in Validation phase.*

Phase by Phase Analysis – Commit & Logging

- Commit: Two-sided + PA
- Logging: One-sided

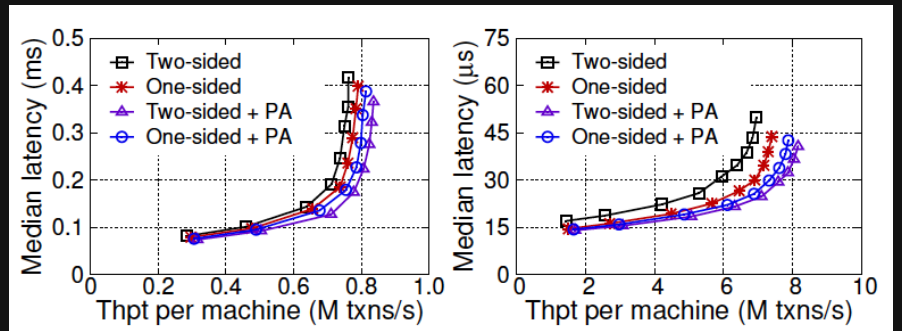


Fig. 9: *The performance of (a) TPC-C/no and (b) SmallBank with different implementations of Commit phase.*

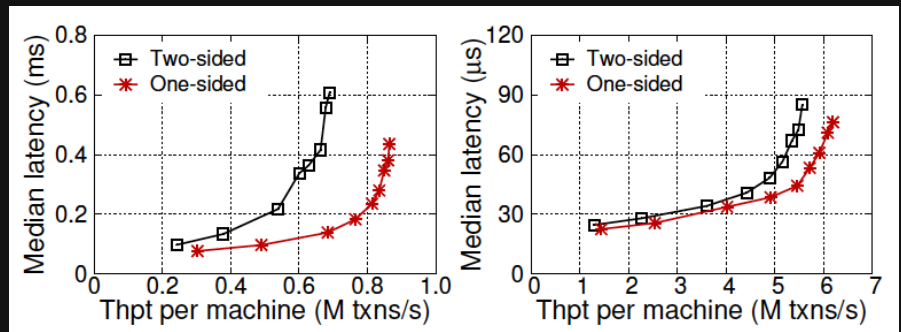


Fig. 10: *The performance of (a) TPC-C/no and (b) SmallBank with different implementations of Logging phase.*

Phase by Phase Analysis – 2 phase Read & Validate

- One-sided/Cache for Read
- One-sided for Validation

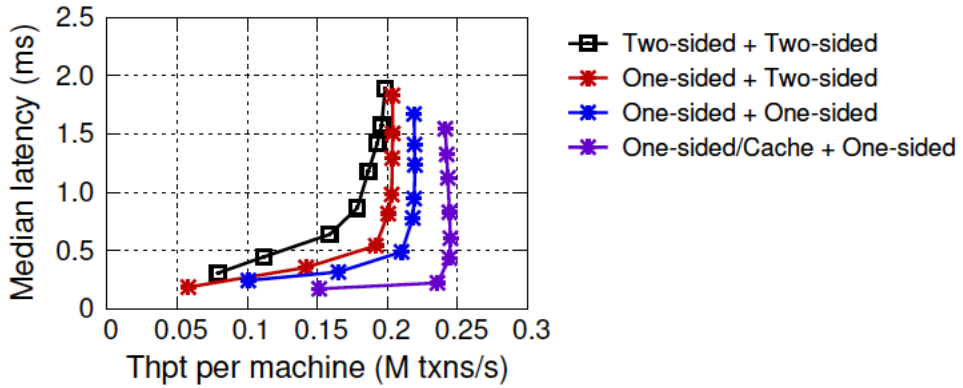


Fig. 11: *The performance of* `customer-position` *in TPC-E with different implementations of the read-only transaction (Read and Validation phases).*

Hybrid Design – DrTM+H

- Execution: Hybrid reads — if record address is cached → one-sided READ; on miss → two-sided RPC (fetch record + address).
- Validation: use one-sided ATOMIC to lock/check if RNIC atomics don't conflict with local CPU accesses; otherwise two-sided (RNIC atomics can slow local ops).
- Logging/Replication: one-sided WRITEs to push logs to all backups; two-sided later to reclaim log space.
- Commit/Install: if validation used one-sided atomics → one-sided WRITEs; else two-sided RPC + Passive-ACK.
- Speculative exec + OR: even without knowing the full read/write set, speculatively continue and fetch independent records in parallel to shrink per-txn lifespan.

DrTM+H Performance

- Throughput: ~7.3M tps (TPC-C/no) and ~90.4M tps (SmallBank) on 16 nodes
- Scales well: With an emulated 80-node QP setting, throughput drops only 5% (TPC-C/no) and 9% (SmallBank); per-thread uses 80 QPs in round-robin
- Beats pure two-sided: On SmallBank, DrTM+H delivers 1.3× higher throughput than a pure two-sided design and cuts tail latency: P50 −22%, P90 −39%, P99 −49%

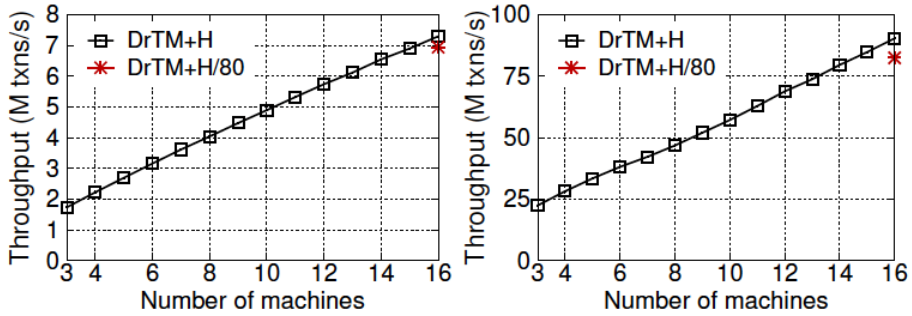


Fig. 12: *The performance of DrTM+H with the increase of machines for (a) TPC-C/no and (b) SmallBank.*

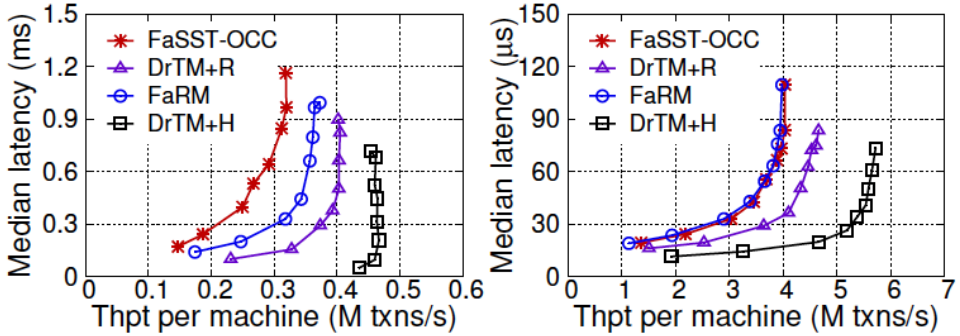


Fig. 14: *An end-to-end comparison of different designs for (a) TPC-C/no and (b) SmallBank.*