1RMA: Re-envisioning Remote Memory Access for Multi-tenant Datacenters

presented by Ray Dedhia
for 6.5810 [6.828] Fall 2022

Background: (Traditional) RDMA

RDMA (Remote Direct Memory Access)

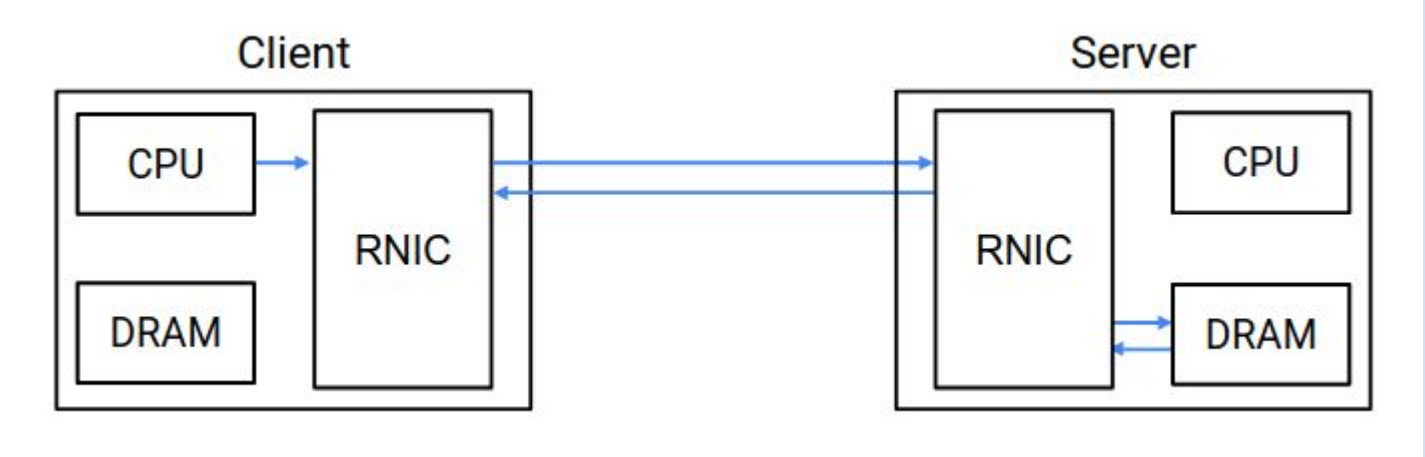
- Allows direct access to memory of remote machine

RDMA (Remote Direct Memory Access)

- Allows direct access to memory of remote machine
- High throughput and low latency
  - Less CPU cycles needed to process network packets

RDMA (Remote Direct Memory Access)

- Allows direct access to memory of remote machine
- High throughput and low latency
  - Less CPU cycles needed to process network packets

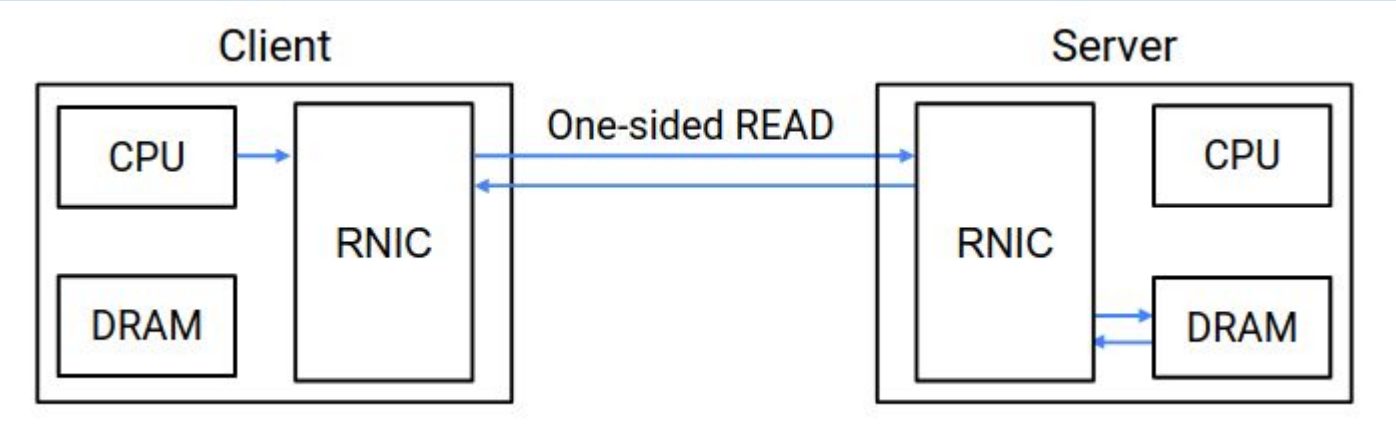


RDMA: One-Sided READ and WRITEs

- Works well with distributed systems, such as datacenters

RDMA: One-Sided READ and WRITEs

- Works well with distributed systems, such as datacenters
- One-sided reads and writes provide latency and op-rate benefits



RDMA Challenge #1: Problems from Connections

- Connection exhaustion can strand critical applications

RDMA Challenge #1: Problems from Connections

- Connection exhaustion can strand critical applications
- How?

RDMA Challenge #1: Problems from Connections

- Connection exhaustion can strand critical applications
- How?
  - RDMA-capable NIC (RNIC) caches can be overwhelmed by large workloads

RDMA Challenge #1: Problems from Connections

- Connection exhaustion can strand critical applications
- How?
  - RDMA-capable NIC (RNIC) caches can be overwhelmed by large workloads
  - Allocation policies ignore business priorities, so low priority apps can consume all available connections

RDMA Challenge #1: Problems from Connections

- Connection exhaustion can strand critical applications
- How?
  - RDMA-capable NIC (RNIC) caches can be overwhelmed by large workloads
  - Allocation policies ignore business priorities, so low priority apps can consume all available connections

RDMA Challenge #1: Problems from Connections (cont.)

- Connection sharing: independent workloads multiplexed on same connection

RDMA Challenge #1: Problems from Connections (cont.)

- Connection sharing: independent workloads multiplexed on same connection
  - Leads to **induced ordering**: required FIFO execution of same-type ops within a single connection

RDMA Challenge #1: Problems from Connections (cont.)

- Connection sharing: independent workloads multiplexed on same connection
  - Leads to **induced ordering**: required FIFO execution of same-type ops within a single connection
- Problems

RDMA Challenge #1: Problems from Connections (cont.)

- Connection sharing: independent workloads multiplexed on same connection
  - Leads to **induced ordering**: required FIFO execution of same-type ops within a single connection
- Problems
  - If one op fails, all others in same connection will fail, often unnecessarily

RDMA Challenge #1: Problems from Connections (cont.)

- Connection sharing: independent workloads multiplexed on same connection
  - Leads to **induced ordering**: required FIFO execution of same-type ops within a single connection
- Problems
  - If one op fails, all others in same connection will fail, often unnecessarily
  - RNIC in-hardware order recovery makes complexity worse

RDMA Challenge #2: Connection-centric Security

- Access control tied to connections

RDMA Challenge #2: Connection-centric Security

- Access control tied to connections
- Encryption tied to connections

RDMA Challenge #2: Connection-centric Security

- Access control tied to connections
- Encryption tied to connections
  - No ready way to manage encryption keys

RDMA Challenge #2: Connection-centric Security

- Access control tied to connections
- Encryption tied to connections
  - No ready way to manage encryption keys
  - Applications need to re-connect to rotate encryption keys

RDMA Challenge #2: Connection-centric Security

- Access control tied to connections
- Encryption tied to connections
  - No ready way to manage encryption keys
  - Applications need to re-connect to rotate encryption keys
    - Expensive

RDMA Challenge #2: Connection-centric Security

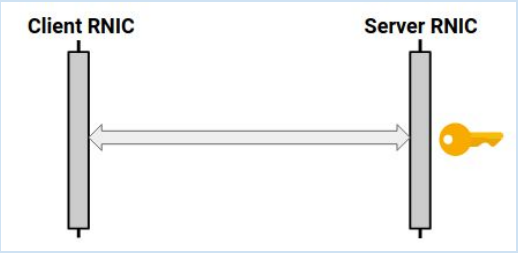
- Access control tied to connections
- Encryption tied to connections
  - No ready way to manage encryption keys
  - Applications need to re-connect to rotate encryption keys
    - Expensive
    - Creates impractical period of unavailability

RDMA Challenge #2: Connection-centric Security

- Access control tied to connections
- Encryption tied to connections
  - No ready way to manage encryption keys
  - Applications need to re-connect to rotate encryption keys
    - Expensive
    - Creates impractical period of unavailability

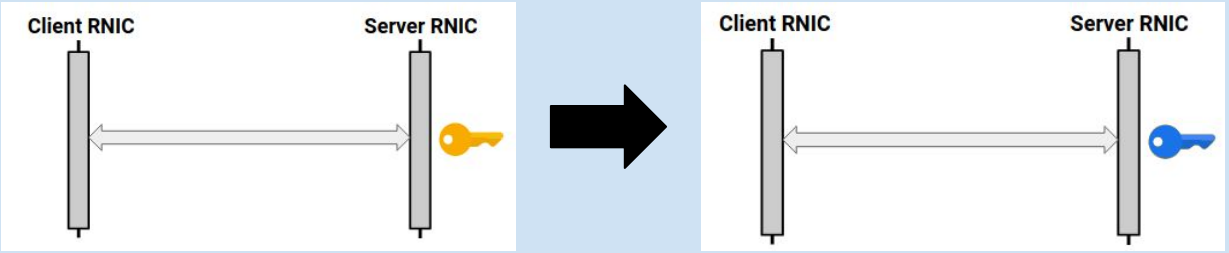
RDMA Challenge #2: Connection-centric Security (cont.)

- Encryption key rotated



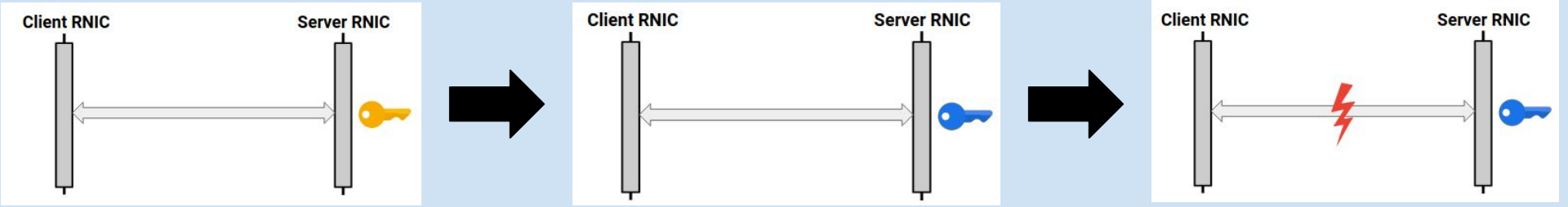
RDMA Challenge #2: Connection-centric Security (cont.)

- Encryption key rotated => new key being used



RDMA Challenge #2: Connection-centric Security (cont.)

- Encryption key rotated => new key being used => temporary connection teardown due to authentication failure while connection is set up again



RDMA Challenge #3: Rigid Congestion Control & Loss Recovery

- Priority flow control (PFC) is needed to provide a near lossless network

RDMA Challenge #3: Rigid Congestion Control & Loss Recovery

- Priority flow control (PFC) is needed to provide a near lossless network
- PFC doesn't work in commercial datacenters
    - Head-of-line blocking, poor at-scale failure isolation, and risk of deadlock

RDMA Challenge #3: Rigid Congestion Control & Loss Recovery

- Priority flow control (PFC) is needed to provide a near lossless network
- PFC doesn't work in commercial datacenters
  - Head-of-line blocking, poor at-scale failure isolation, and risk of deadlock
- Solutions that reduce reliance on PFC bake congestion response and loss recovery into hardware
  - Problem: Can't update congestion control algorithms post-deployment

RDMA Challenges: Root Causes

- RDMA ill suited to multi-tenancy because of its two basic design attributes
  - (1) Connection orientedness
  - (2) Complex policies baked into hardware (inflexible)

Solution: 1RMA

RDMA vs. 1RMA

Responsibility	RDMA	1RMA
Inter-op ordering	NIC	Software
Failure recovery	NIC	Software
Flow and congestion control	NIC and Fabric	NIC and Software
Security ops (*e.g.*, Rekey)	None	NIC

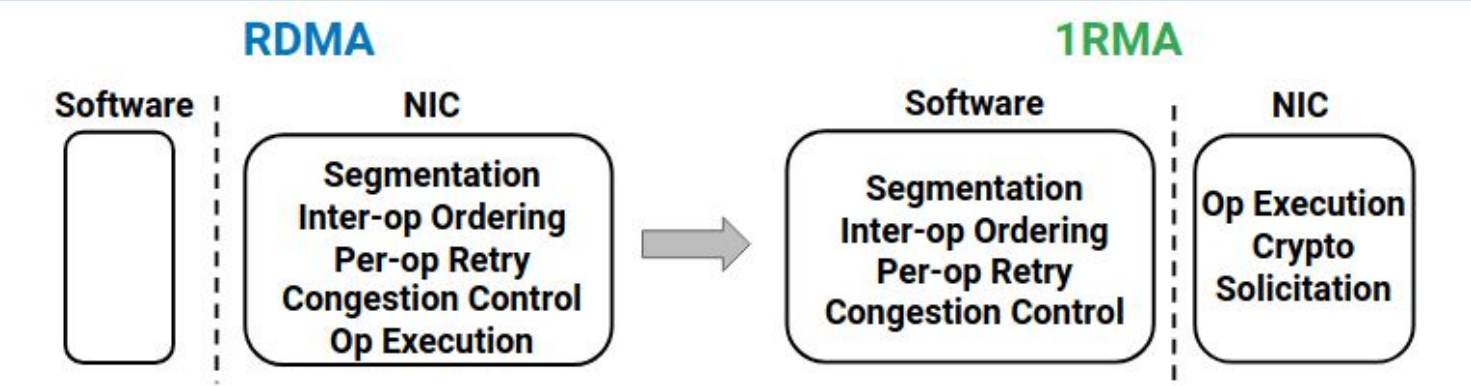
Table 1: Division of responsibilities in standard RDMA and 1RMA. Moving a subset of functionality to software simplifies hardware and enables more flexibility/rapid iteration.

1RMA's Key Ideas

- (1) No connections
- (2) Small fixed-size ops, with solicited transfers
- (3) Software-driven congestion control
- (4) Finite resource allocation
- (5) Security-focused

1RMA Key Idea #1: No Connections

- 1RMA NIC treats ops as independent of each other
  - Software handles iter-op ordering => less complexity
  - Failures less costly



1RMA Key Idea #1: No Connections (cont.)

- Hardware state doesn't grow with number of endpoint pairs
- Provides **fail-fast behavior**: if op doesn't complete within fixed time, NIC will cancel it and deliver fast and precise failure notifications to the software

Discussion Questions

- 1RMA times out delayed ops by converting slow ops to failures. Is there a situation in which a op with a long runtime will never be able to run because it is consistently converted to a failure?

Discussion Questions

- 1RMA times out delayed ops by converting slow ops to failures. Is there a situation in which a op with a long runtime will never be able to run because it is consistently converted to a failure?
- Would this be the desired behavior in this situation?

1RMA Key Idea #2: Small Ops with Solicited Transfers

- 1RMA NIC acts on **small**, **fixed-size ops** (max 4KB)
  - Large ops chunked into smaller ops
  - Enables isolation and prioritization

1RMA Key Idea #2: Small Ops with Solicited Transfers

- 1RMA NIC acts on **small**, **fixed-size ops** (max 4KB)
  - Large ops chunked into smaller ops
  - Enables isolation and prioritization
- Solicited transfers: 1RMA only gets data it requests

1RMA Key Idea #2: Small Ops with Solicited Transfers

- 1RMA NIC acts on **small**, **fixed-size ops** (max 4KB)
  - Large ops chunked into smaller ops
  - Enables isolation and prioritization
- Solicited transfers: 1RMA only gets data it requests
  - Only requests data transfers when there's space in on-NIC SRAM
    - On-NIC solicitation window keeps track of space

1RMA Key Idea #2: Small Ops with Solicited Transfers

- 1RMA NIC acts on **small**, **fixed-size ops** (max 4KB)
  - Large ops chunked into smaller ops
  - Enables isolation and prioritization
- Solicited transfers: 1RMA only gets data it requests
  - Only requests data transfers when there's space in on-NIC SRAM
    - On-NIC solicitation window keeps track of space
  - Enables responsive congestion control

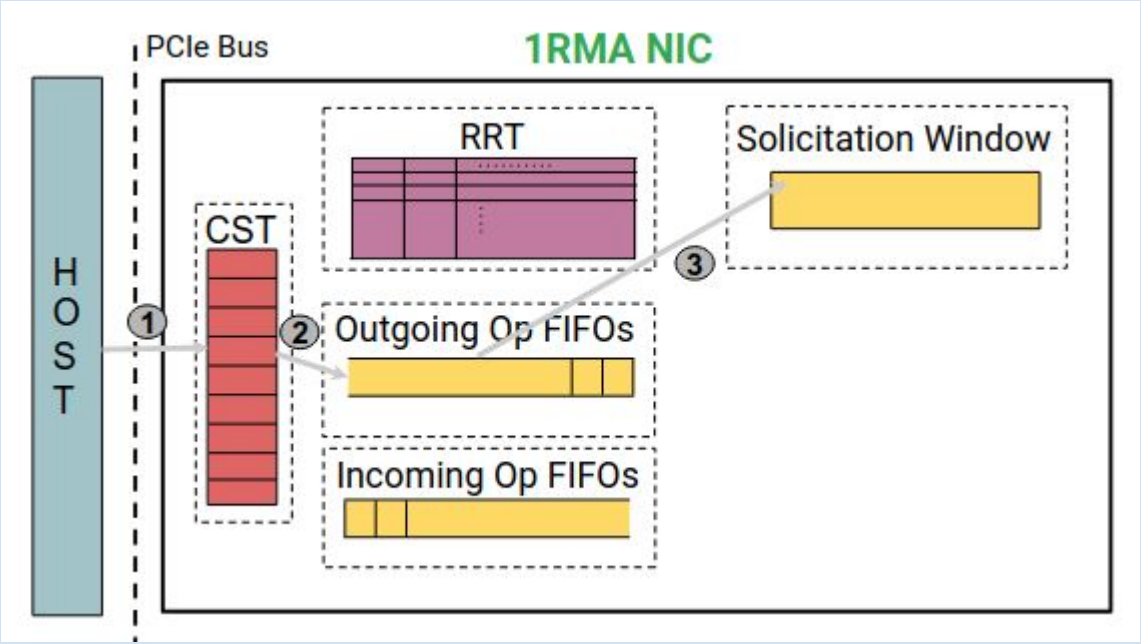
1RMA Key Idea #2: Small Ops with Solicited Transfers

- 1RMA NIC acts on **small**, **fixed-size ops** (max 4KB)
  - Large ops chunked into smaller ops
  - Enables isolation and prioritization
- Solicited transfers: 1RMA only gets data it requests
  - Only requests data transfers when there's space in on-NIC SRAM
    - On-NIC solicitation window keeps track of space
  - Enables responsive congestion control
  - Prevents large incasts (many to one communication patterns) by bounding number of incoming bytes, preventing TCP failure

1RMA Key Idea #2: Small Ops with Solicited Transfers

- 1RMA NIC acts on **small**, **fixed-size ops** (max 4KB)
  - Large ops chunked into smaller ops
  - Enables isolation and prioritization
- Solicited transfers: 1RMA only gets data it requests
  - Only requests data transfers when there's space in on-NIC SRAM
    - On-NIC solicitation window keeps track of space
  - Enables responsive congestion control
  - Prevents large incasts (many to one communication patterns) by bounding number of incoming bytes, preventing TCP failure
- Tolerates unordered responses by tracking byte arrivals

1RMA Key Idea #2: Small Ops with Solicited Transfers (cont.)



Discussion Questions

- What is the overhead cost of chunking large operations? Do you think it's worth it?

Discussion Questions

- What is the overhead cost of chunking large operations? Do you think it's worth it?
- What is the overhead cost of tracking byte arrivals to tolerate unordered responses?

1RMA Key Idea #3: Software-Driven Congestion Control

- Hardware provides:
  - Delay statistics for each op
  - Specific failure messages

1RMA Key Idea #3: Software-Driven Congestion Control

- Hardware provides:
  - Delay statistics for each op
  - Specific failure messages
- Software uses these values to:
  - Take specific actions to fix congestion issues (can make rapid policy changes)
  - Avoid wasted bandwidth and allocate bandwidth fairly

Delay Signals

- 1RMA hardware measures both local and remote delay
  - total_delay = how long it took to execute the operation
  - local_delay = issue_delay = how long it took for the request to enter service at the initiator
  - remote_delay = total_delay - issue_delay

Delay Signals

- 1RMA hardware measures both local and remote delay
  - total_delay = how long it took to execute the operation
  - local_delay = issue_delay = how long it took for the request to enter service at the initiator
  - remote_delay = total_delay - issue_delay
- Congestion window variables
  - One rCWND per (remote destination, direction) pair for computing remote delay
  - Single ICWD for computing local delay

Delay Signals (cont.)

- This allows its software to distinguish between local and remote congestion
  - Better performance (see evaluation section later)

Delay Signals (cont.)

- This allows its software to distinguish between local and remote congestion
  - Better performance (see evaluation section later)
- This is unique to 1RMA

1RMA Key Idea #4: Finite Resource Allocation

- NIC Resource Pools are explicitly finite

1RMA Key Idea #4: Finite Resource Allocation

- NIC Resource Pools are explicitly finite
- Result
  - Software assigns resources based on business priority
    - Prevents low priority applications from monopolizing network

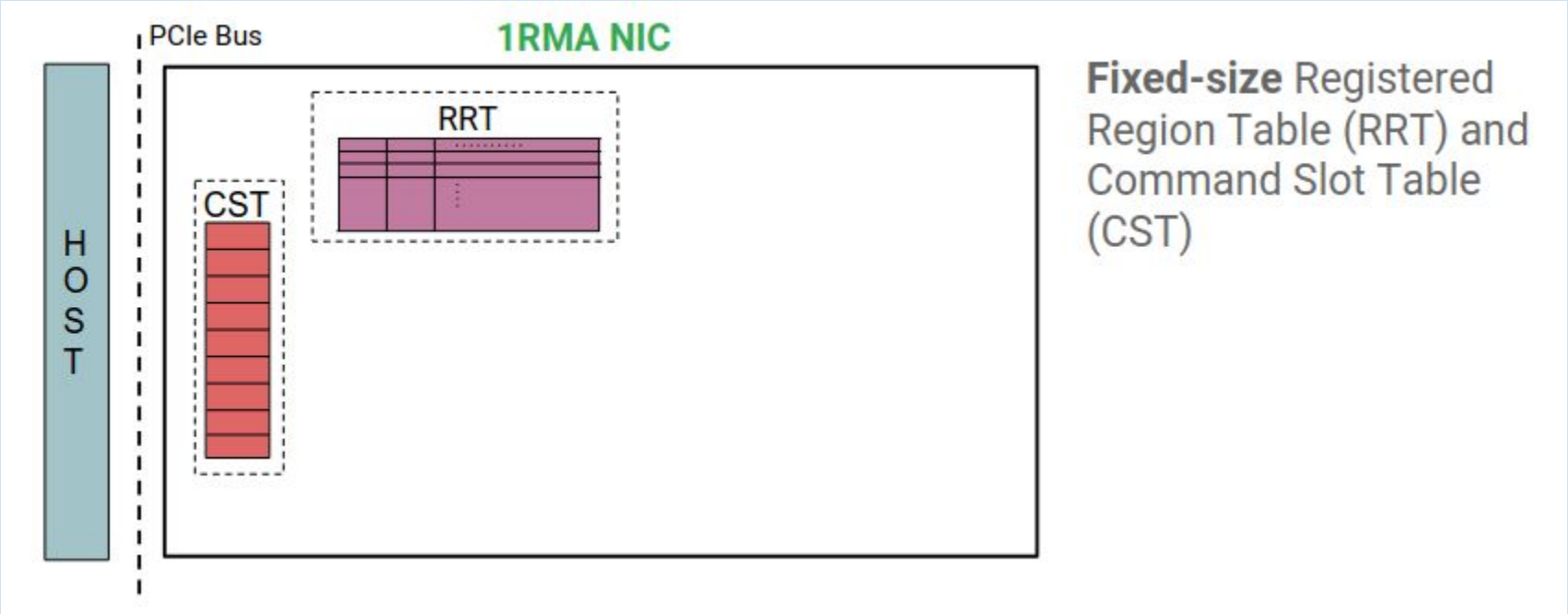
1RMA Key Idea #4: Finite Resource Allocation

- NIC Resource Pools are explicitly finite
- Result
  - Software assigns resources based on business priority
    - Prevents low priority applications from monopolizing network
  - Simplifies hardware

1RMA Key Idea #4: Finite Resource Allocation

- NIC Resource Pools are explicitly finite
- Result
  - Software assigns resources based on business priority
    - Prevents low priority applications from monopolizing network
  - Simplifies hardware
  - Provides predictable performance

1RMA Key Idea #4: Finite Resource Allocation (cont.)



1RMA Key Idea #5: Security-Focused

- All transfers encrypted and work without connections

1RMA Key Idea #5: Security-Focused

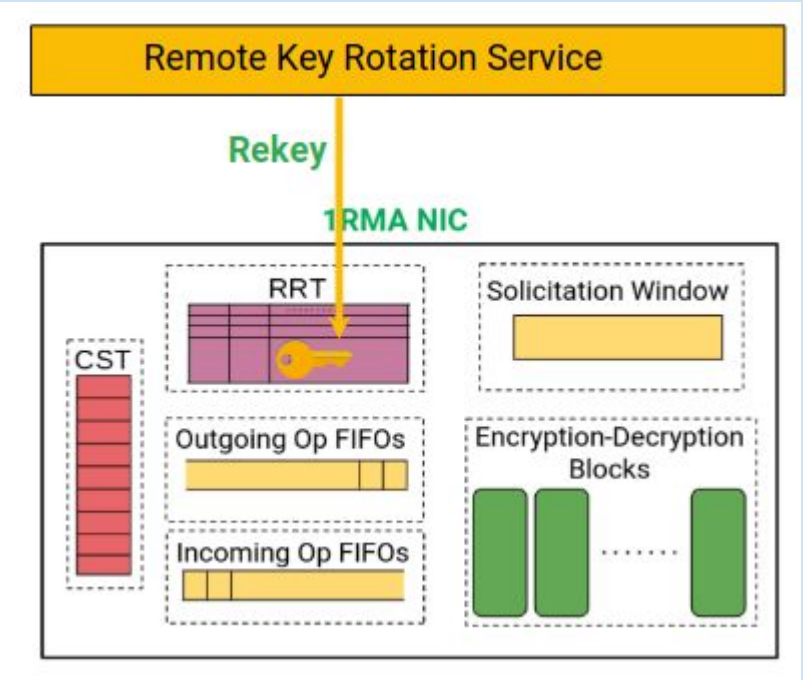
- All transfers encrypted and work without connections
- Encryption process for each NIC salted with ascending message counters
    - Protects key integrity + protects against replay attacks

1RMA Key Idea #5: Security-Focused (cont.)

- Apps directly manage encryption keys; don't need to trust infrastructure

1RMA Key Idea #5: Security-Focused (cont.)

- Apps directly manage encryption keys; don't need to trust infrastructure
- Can do key rotation:
  - Without trusting local software stack
  - Without using local CPU



1RMA: Operations

- Reads
- Writes via request-to-reads
- Rekeys

1RMA: Reads

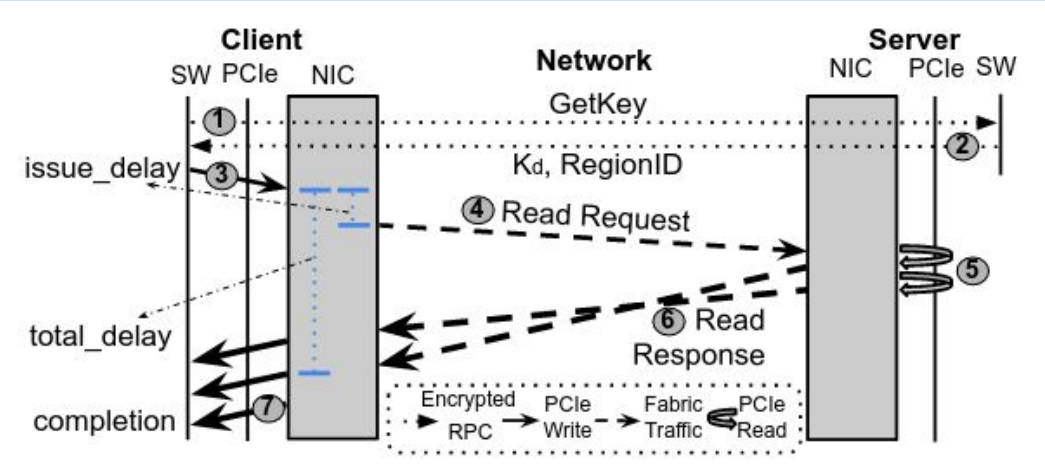


Figure 3: Execution of a 2KB Read op in 1RMA.

(1 & 2) Client performs out-of-band communication to obtain information to access remote memory region.

(3) The client initiates the 2KB read op by writing a command into a command slot on the 1RMA NIC.

(4) The op is sent over the network subject to 1RMA's solicitation rules.

(5) The request reaches the server-side 1RMA NIC which reads the requested data via PCIe, and

(6) streams read data as individual network responses.

(7) Finally, once all the data arrives, a successful op completion is written to the client software.

1RMA: Writes via Request-to-Reads

- 4-hop write transactions
  - Write operations implemented as request-to-reads, where writer asks remote NIC to receive data via read operation
  - Adds an additional round-trip time (RTT)

1RMA: Writes via Request-to-Reads

- 4-hop write transactions
  - Write operations implemented as request-to-reads, where writer asks remote NIC to receive data via read operation
  - Adds an additional round-trip time (RTT)
- Benefits
  - Writes obey solicitation (incast burst protection)
  - Secure against replay attacks
  - Consistent behavior => timeout happens after side effects in remote memory

Discussion Questions

- The paper states that having 1RMA implement writes as remote requests to read “incurs the downside of an additional RTT”. Is there a situation in which this downside would significantly harm performance, or is it always negligible?

1RMA: Rekeys

- Specialized Rekey operation lets user cheaply install new region keys, simplifying encryption key rotation
  - Frequent key rotation accelerates detection of root-level compromises
  - Hardware supports frequent encryption key rotation with low availability disruption

1RMA: Rekeys

- Specialized Rekey operation lets user cheaply install new region keys, simplifying encryption key rotation
  - Frequent key rotation accelerates detection of root-level compromises
  - Hardware supports frequent encryption key rotation with low availability disruption
- No more expensive than an RMA write operation

1RMA Software

- Software abstracts large transfers & does congestion control

1RMA Software Stack

- Layer 1 (lowest layer)
  - Managed by CommandPortal object
  - Provides command/completion queue construct
  - Handles mmap()

1RMA Software Stack

- Layer 1 (lowest layer)
  - Managed by CommandPortal object
  - Provides command/completion queue construct
  - Handles mmap()
- Layer 2
  - Managed by CommandExecutor object
  - Handles chunking, pacing, and congestion control

1RMA Software Stack

- Layer 1 (lowest layer)
  - Managed by CommandPortal object
  - Provides command/completion queue construct
  - Handles mmap()
- Layer 2
  - Managed by CommandExecutor object
  - Handles chunking, pacing, and congestion control
- Layer 3+
  - Application software

1RMA Software Stack (cont.)

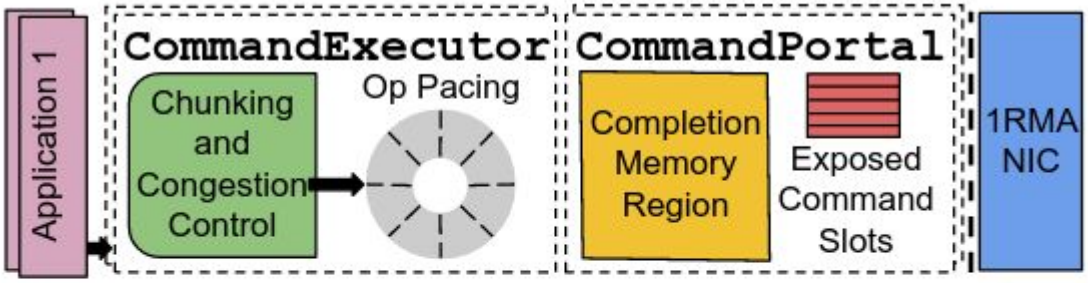


Figure 4: 1RMA Software consists of two components: (1) `CommandPortal` provides a familiar command/completion queue construct and (2) `CommandExecutor` provides support for arbitrary large transfers and congestion management.

Discussion Questions

- Do you think 1RMA meets all of its goals? Why or why not?

Discussion Questions

- Do you think 1RMA meets all of its goals? Why or why not?
- How would you evaluate 1RMA? In what case(s) do you think 1RMA performs the best? The worst?

Evaluation: 1RMA's Claims

- Software handles congestion control
  - Prevents apps from monopolizing network
  - Converges to fair bandwidth shares

Evaluation: 1RMA's Claims

- Software handles congestion control
  - Prevents apps from monopolizing network
  - Converges to fair bandwidth shares
- Apps handle failure recovery and inter-operation ordering
  - Solicitation rules prevent goodput loss in all but worst cases
  - Shed load eagerly instead of risking wasted bandwidth

Evaluation: 1RMA's Claims

- Software handles congestion control
  - Prevents apps from monopolizing network
  - Converges to fair bandwidth shares
- Apps handle failure recovery and inter-operation ordering
  - Solicitation rules prevent goodput loss in all but worst cases
  - Shed load eagerly instead of risking wasted bandwidth
- Encryption
  - At line rate, with minimal client-observable disruption
    - Line rate: 100Gbps and 100M ops/sec
  - Hardware supported key rotation

Evaluation: Baselines

- Standard RDMA

Evaluation: Baselines

- Standard RDMA
- Pony
  - Google's software-defined NIC
  - State-of-the-art datacenter networking alternative
  - Most similar to 1RMA in its objectives

Evaluation: Baselines

- Standard RDMA
- Pony
  - Google's software-defined NIC
  - State-of-the-art datacenter networking alternative
  - Most similar to 1RMA in its objectives
- Note: In all experiments, network stacks limited to at most 1 CPU
  - Paper states that Pony performs much better with larger CPU allocations

Evaluation: Performance

- Low latency in common cases

Evaluation: Performance

- Low latency in common cases
- Latency is stable and predictable under less common failure cases

Evaluation: Performance

- Low latency in common cases
- Latency is stable and predictable under less common failure cases
- Doesn't work as well when there are many small independent ops
    - 1RMA chunks large transfers, so smaller ops experience delay similar to that of a 4KB chunk

Evaluation: Performance

- Low latency in common cases
- Latency is stable and predictable under less common failure cases
- Doesn't work as well when there are many small independent ops
  - 1RMA chunks large transfers, so smaller ops experience delay similar to that of a 4KB chunk
- Has high bandwidth and op rates
  - 1RMA is able to converge to and maintain 100Gbps within ~8 RTTs (with idle RTT of 5 microseconds and a CWND of 15 outstanding operations)

Latency

- 1RMA has a lower latency than Pony and standard RDMA (when core limited)

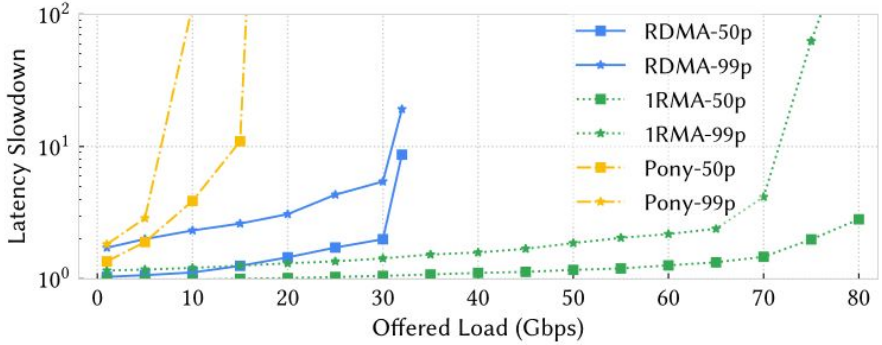


Figure 10: Latency vs. offered load for a uniform random traffic pattern.

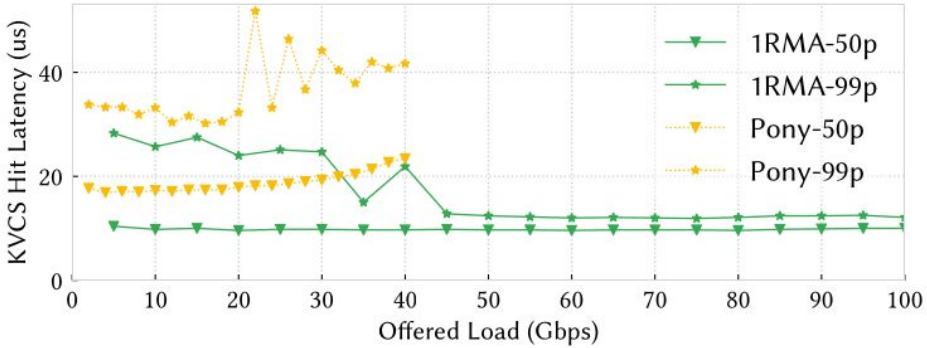


Figure 8: Latency vs. offered load for KVCS.

Latency (cont.)

- Figure 12: Plots latency slowdown of small ops as we vary the size of the competing background op

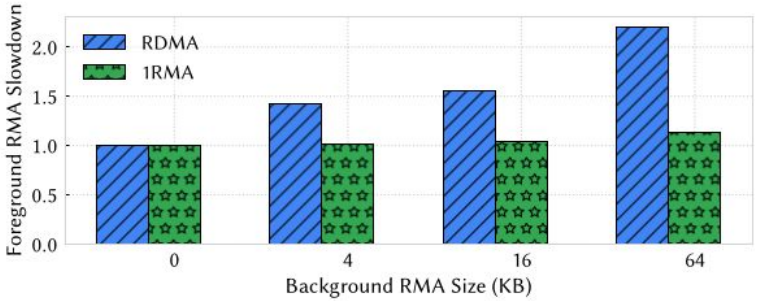


Figure 12: Impact of head-of-line blocking for a point-to-point workload.

Latency (cont.)

- Figure 12: Plots latency slowdown of small ops as we vary the size of the competing background op
- 1RMA
  - Since software chunks large transfers, smaller ops experience a delay comparable to the service time of a single 4KB chunk

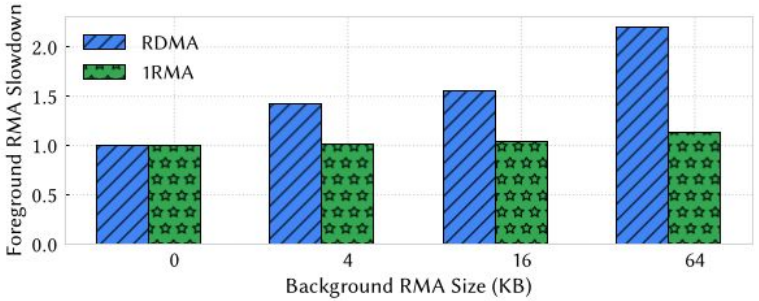


Figure 12: Impact of head-of-line blocking for a point-to-point workload.

Latency (cont.)

- Figure 12: Plots latency slowdown of small ops as we vary the size of the competing background op

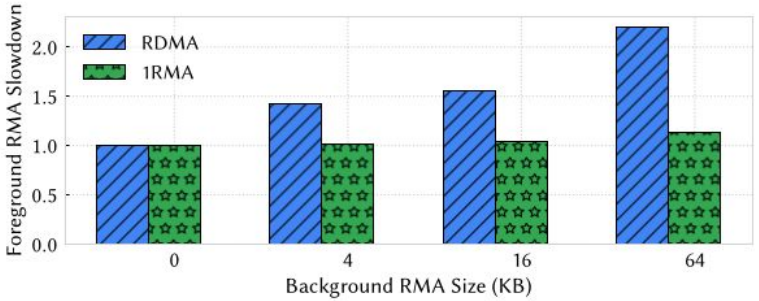


Figure 12: Impact of head-of-line blocking for a point-to-point workload.

- 1RMA
  - Since software chunks large transfers, smaller ops experience a delay comparable to the service time of a single 4KB chunk
- Standard RDMA
  - RNIC delay is proportional to the service time of a background op, and potentially severe
  - Caused by RDMA's induced ordering

Latency (cont.)

- Figure 13: Plots slowdown experienced by small ops in a heavy-tailed workload

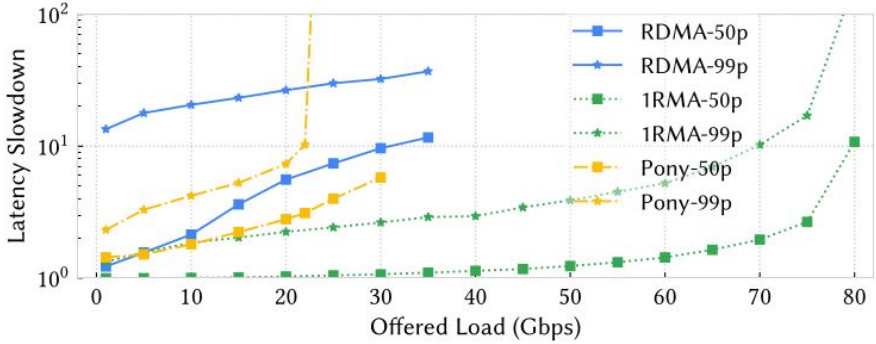


Figure 13: Impact of head-of-line blocking for a uniform random traffic pattern.

Latency (cont.)

- Figure 13: Plots slowdown experienced by small ops in a heavy-tailed workload

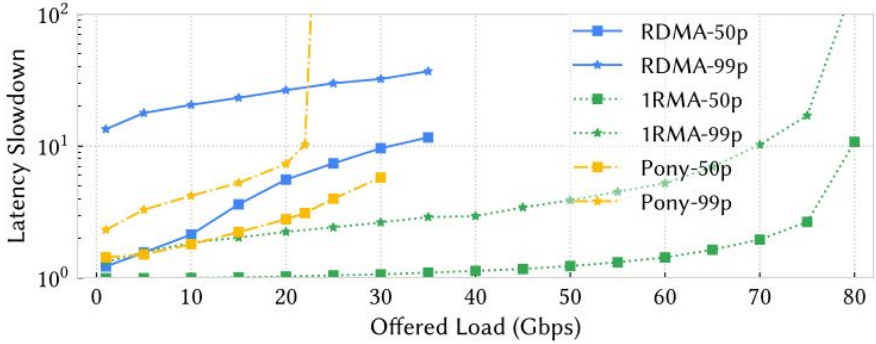


Figure 13: Impact of head-of-line blocking for a uniform random traffic pattern.

- Compared to baseline, 1RMA slows down smaller ops minimally
- However, 1RMA's median and tail slowdowns are 6-10 times smaller than baselines

Congestion Control

- 1RMA quickly converges to fair bandwidth share (within ~8 RTTs)

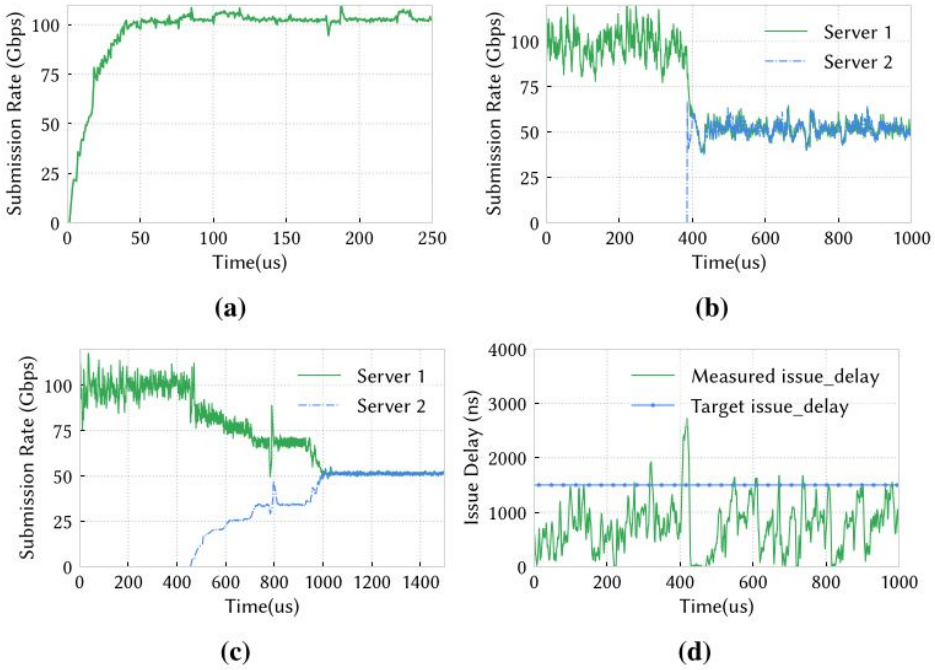


Figure 15: (a) Fast convergence of 1RMA CC - 0 to 100Gbps in 40us; (b) Fast convergence from single flow at full rate to two flows at fair-share rates; (c) Slower convergence when we configure 1RMA to not separate local congestion from remote; (d) `issue_delay` over time as reported by the NIC for (b).

Congestion Control

- 1RMA quickly converges to fair bandwidth share (within ~8 RTTs)
- Without separate measurements for local and remote congestion, applications converge to a fair share of bandwidth 20x slower

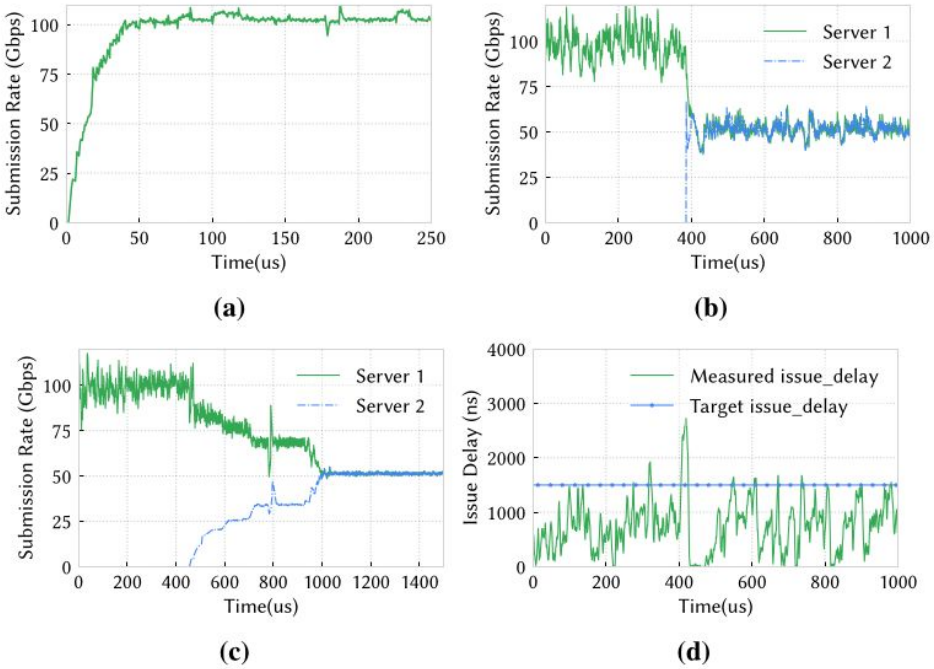


Figure 15: (a) Fast convergence of 1RMA CC - 0 to 100Gbps in 40us; (b) Fast convergence from single flow at full rate to two flows at fair-share rates; (c) Slower convergence when we configure 1RMA to not separate local congestion from remote; (d) `issue_delay` over time as reported by the NIC for (b).

Discussion Questions

- The authors of the paper stated, “In all our experiments, we limit each networking stack to (at most) a single host CPU for network transport processing… With larger CPU allocations, Pony performance scales commensurately.”
- With larger CPU allocations, how would Pony compare to 1RMA in terms of performance?
- Do you think the authors of the paper should have included this benchmark?

Works Cited

- 1RMA: Re-envisioning Remote Memory Access for Multi-tenant Datacenters. *<https://abelay.github.io/6828seminar/papers/singhvi:1rma.pdf>*.
- 1RMA Slides. *<https://pages.cs.wisc.edu/~asinghvi/slides/1rma.pdf>*.
- SIGCOMM 2020: Session 12: 1RMA: Re-envisioning Remote Memory Access for Multi tenant Datacenters. *<https://www.youtube.com/watch?v=sxiUi-bui_U>*.

SIGCOMM 2020 Questions

A connectionless offload seems to trade-off the performance of a hardware implemented connected transport for scalability and hardware simplicity, somewhat similar to the trade-off of tightly vs. loosely coupled systems. Do you see cases where you might still prefer connected offloads?

SIGCOMM 2020 Questions

A connectionless offload seems to trade-off the performance of a hardware implemented connected transport for scalability and hardware simplicity, somewhat similar to the trade-off of tightly vs. loosely coupled systems. Do you see cases where you might still prefer connected offloads?

I would imagine that a connection oriented connection transport would be still applicable in cases where the workload is not large scale or corresponds to a few tenants of a small scale workload, because there you wouldn't go into the issue of your hardware connection being exhausted.

SIGCOMM 2020 Questions (cont.)

Several of RDMA's benefits come from offloading tasks to hardware (e.g. segmentation), but 1RMA moves some of these back to software. Is there a fundamental sweet spot for the division of labor, or does it depend on the workloads and use cases and keeps evolving?

SIGCOMM 2020 Questions (cont.)

Several of RDMA's benefits come from offloading tasks to hardware (e.g. segmentation), but 1RMA moves some of these back to software. Is there a fundamental sweet spot for the division of labor, or does it depend on the workloads and use cases and keeps evolving?

Guiding principal: Things that can be done solely in hardware (solicitation, DMA-ing packet metadata, etc.) are done in hardware. Things like failure recovery, ordering, and congestion (which require and/or benefit from app intervention) are done in software.

SIGCOMM 2020 Questions (cont.)

One main benefit of RDMA is low CPU overhead. Onloading many RDMA features to software increases CPU utilizations and increases the cost of the cloud. What is the CPU overhead?

SIGCOMM 2020 Questions (cont.)

One main benefit of RDMA is low CPU overhead. Onloading many RDMA features to software increases CPU utilizations and increases the cost of the cloud. What is the CPU overhead?

Our evaluation shows that we are able to drive at 100 Gbps using half a CPU core, and we believe that that is an acceptable trade-off. Original requirement: supporting workloads in a multi-tenant environment. With many transient transfers, hardware segmentation will not work well, so pushed segmentation to software. Failure recovery, loss recovery, and ordering are application dependent, so pushed to software.