WISCONSIN
UNIVERSITY OF WISCONSIN–MADISON

CS 839: Design the Next-Generation Database

Lecture 15: RDMA for DB

Xiangyao Yu

3/10/2020

Announcements

Upcoming deadlines:

- **Proposal due: Today**

Discussion Highlights

How to improve group-by aggregation performance?

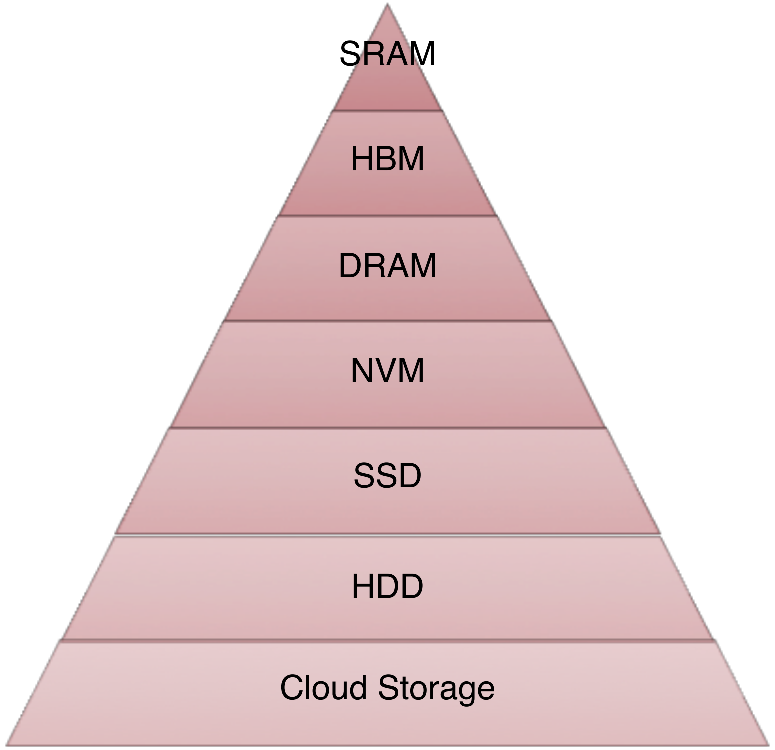
- Store aggregation values in registers
- Use buffers/caches to accelerate random accesses
- Keep hot data in registers and swap to memory as needed
- Sort data by groups in memory to convert random to sequential accesses

Smart SSD and PIM for transactions

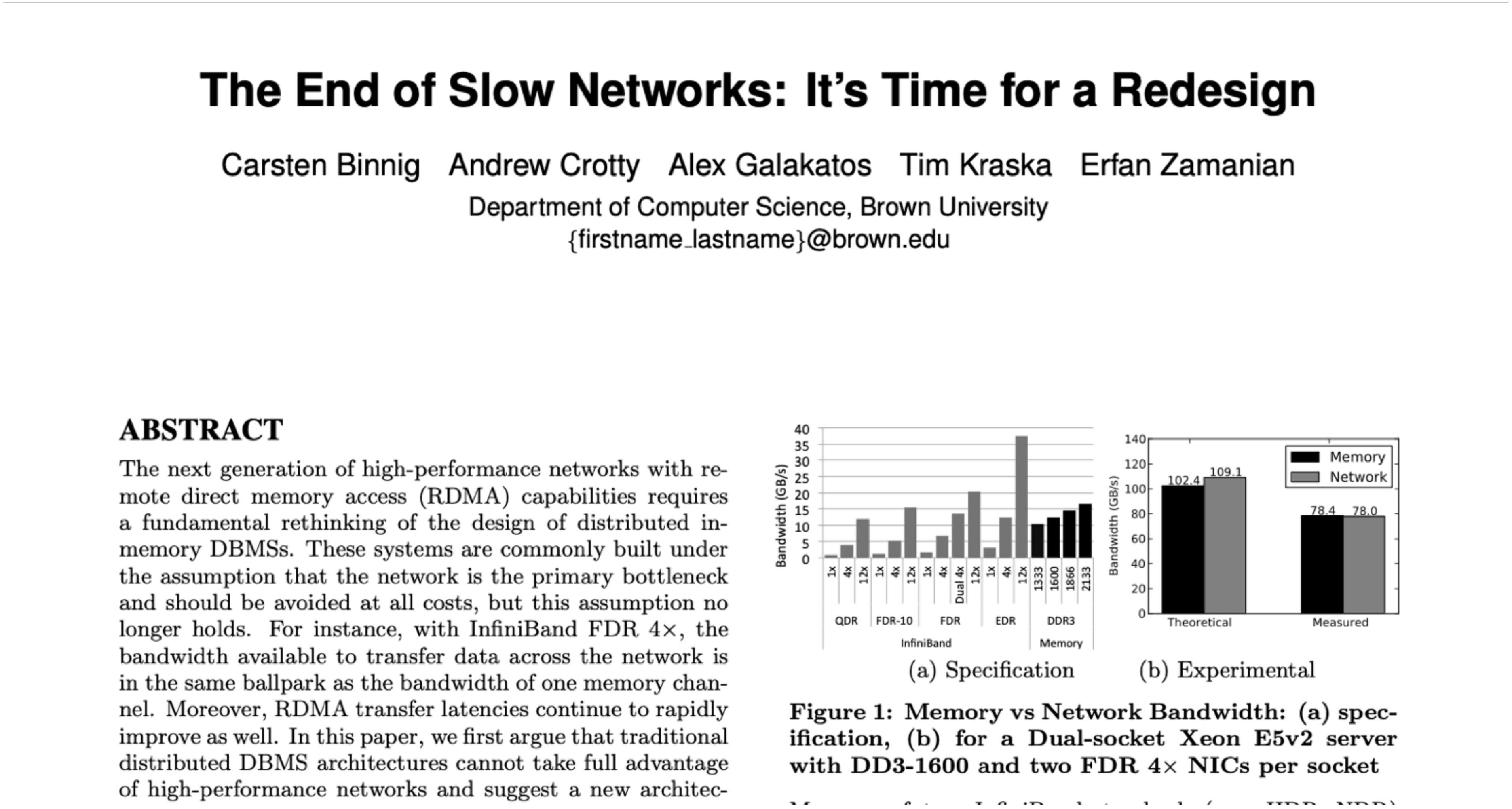
- Logging and garbage collection to Smart SSD
- Compression and decompression of column store
- Conflict detection
- Generating TID using SmartSSD/PIM

Where will PIM most likely to succeed in the storage hierarchy?

- NVM: persistency and byte-addressability
- HBM, DRAM, NVM, or SSD (no benefits for SRAM)
- Cloud storage obviously (e.g., PushdownDB ☺) then move up the stack SSD/NVM and may stop at DRAM due to limited benefits and added complexity
- Three Tiered DB: admission/eviction can be pushed down to NVM or SSD



Today’s Paper



The End of Slow Networks: It’s Time for a Redesign

Carsten Binnig Andrew Crotty Alex Galakatos Tim Kraska Erfan Zamanian
Department of Computer Science, Brown University
{firstname_lastname}@brown.edu

ABSTRACT

The next generation of high-performance networks with remote direct memory access (RDMA) capabilities requires a fundamental rethinking of the design of distributed in-memory DBMSs. These systems are commonly built under the assumption that the network is the primary bottleneck and should be avoided at all costs, but this assumption no longer holds. For instance, with InfiniBand FDR 4×, the bandwidth available to transfer data across the network is in the same ballpark as the bandwidth of one memory channel. Moreover, RDMA transfer latencies continue to rapidly improve as well. In this paper, we first argue that traditional distributed DBMS architectures cannot take full advantage of high-performance networks and suggest a new architec-

(a) Specification (b) Experimental

Figure 1: Memory vs Network Bandwidth: (a) specification, (b) for a Dual-socket Xeon E5v2 server with DD3-1600 and two FDR 4× NICs per socket

VLDB 2016

Today’s Agenda

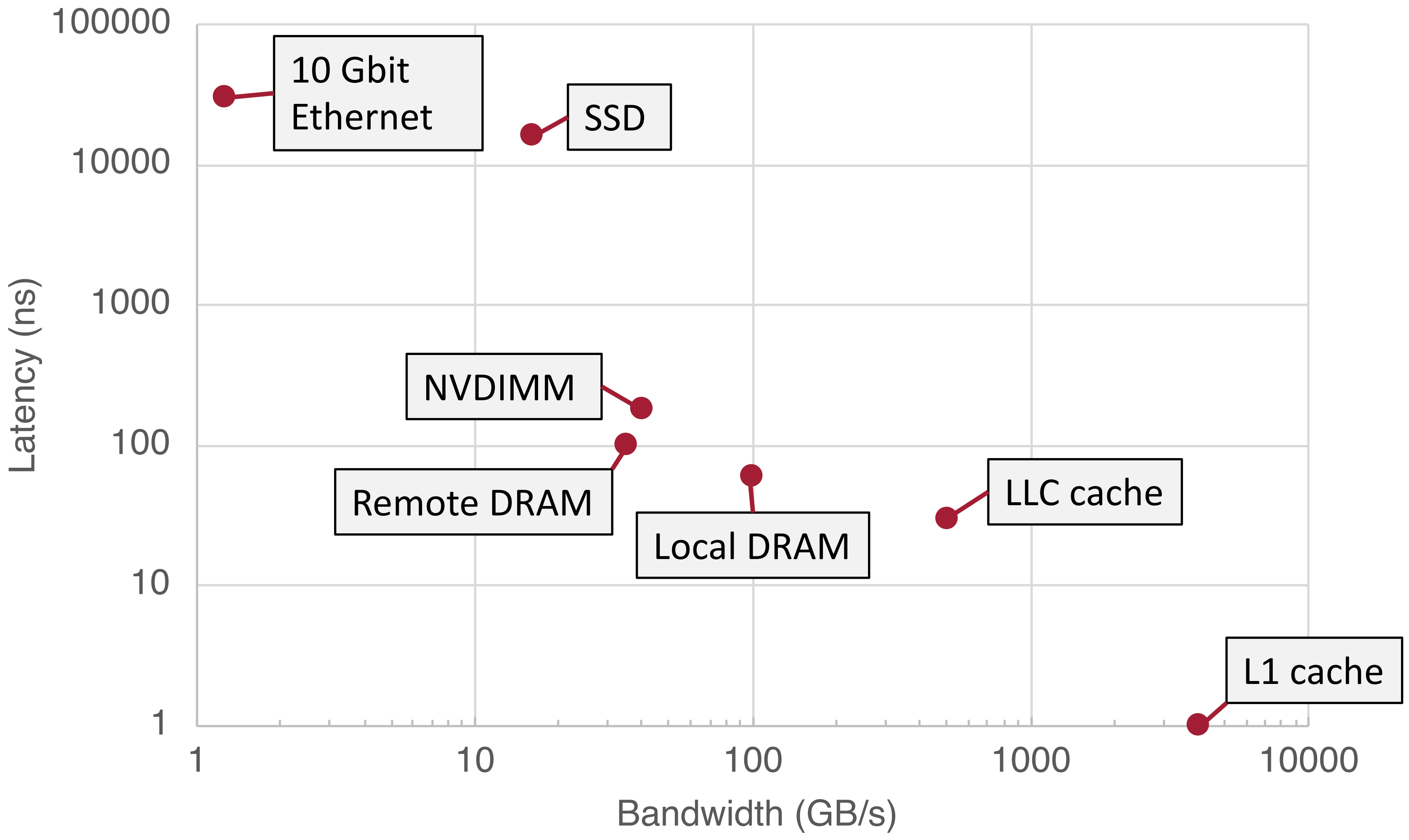
InfiniBand and RDMA

NAM architecture

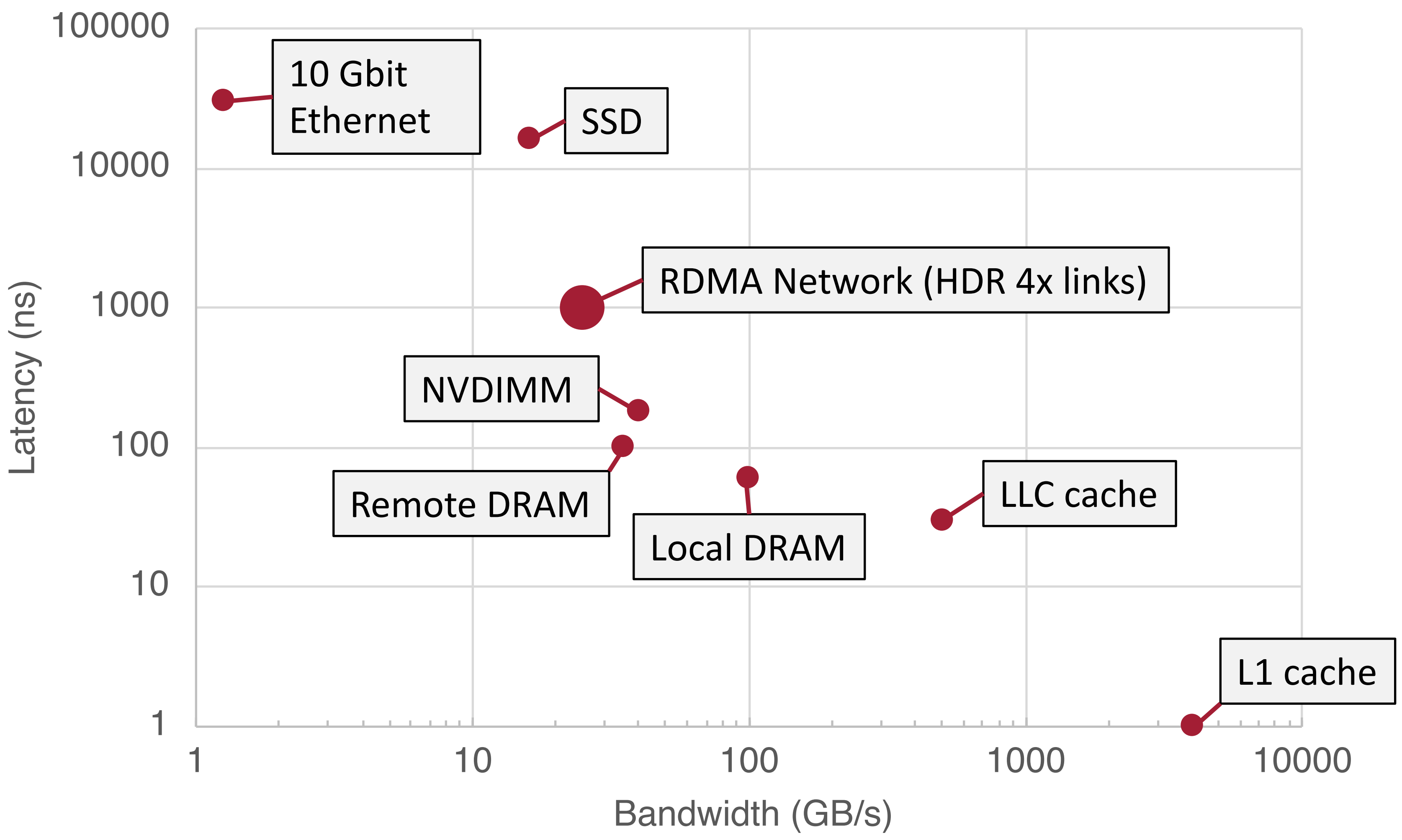
RDMA for OLTP

RDMA for OLAP

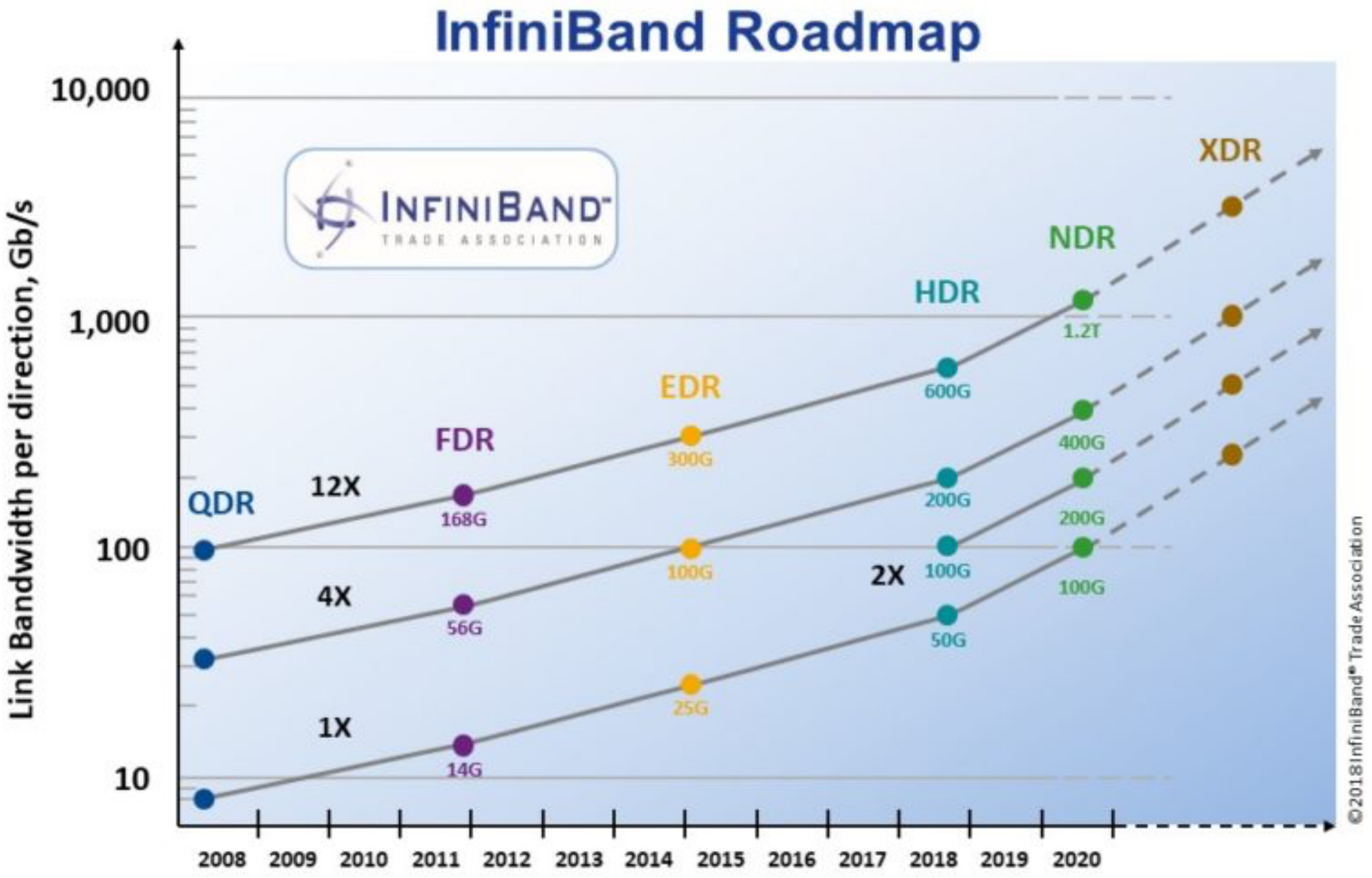
Bandwidth and Latency



Bandwidth and Latency



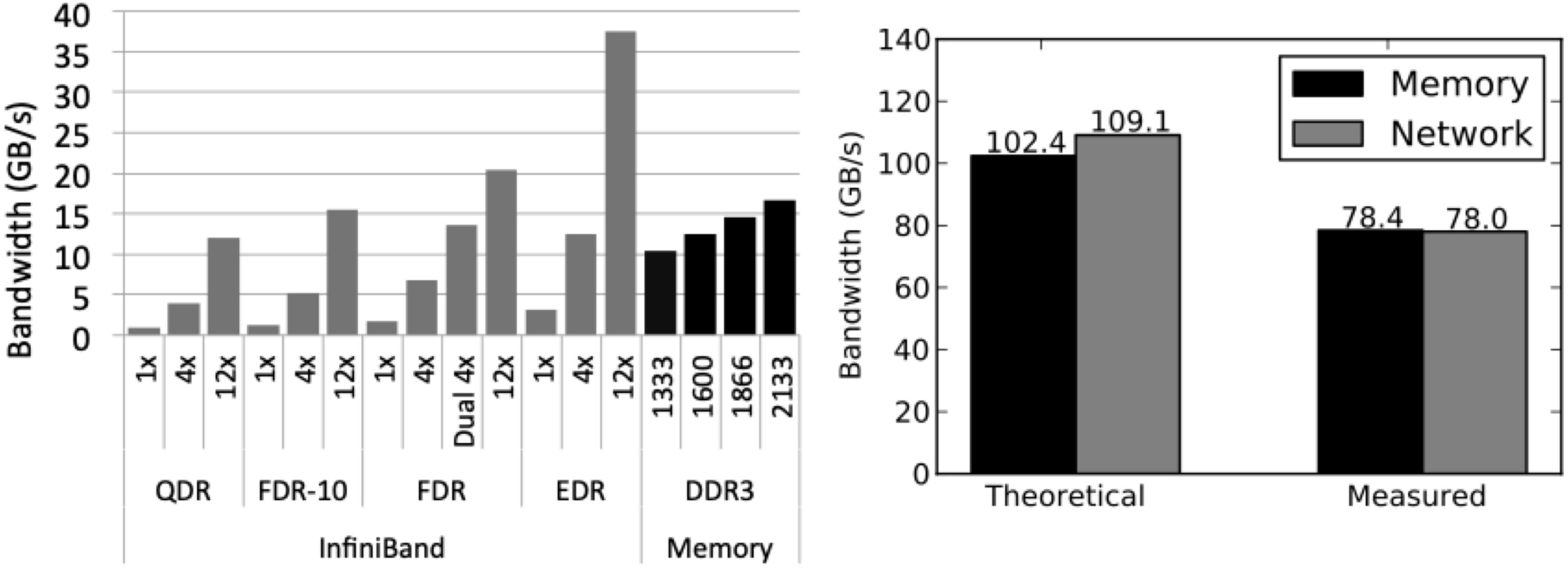
InfiniBand Roadmap



- SDR (single data rate): 2.5Gb/s
- DDR (double data rate): 5 Gb/s
- QDR (quad data rate): 10 Gb/s
- FDR (fourteen data rate): 14 Gb/s
- EDR (enhanced data rate): 25.8 Gb/s
- HDR (high data rate): 50 Gb/s
- NDR (next data rate): 100 Gb/s

Source: https://www.infinibandta.org/infiniband-roadmap/

Network vs. Memory Bandwidth



(a) Specification (b) Experimental

Network bandwidth is comparable to main memory bandwidth (assuming that PCIe is not a bottleneck)

InfiniBand and RDMA

RDMA: Remote direct memory access

IPoIB (IP over InfiniBand)

- Classic TCP/IP stack

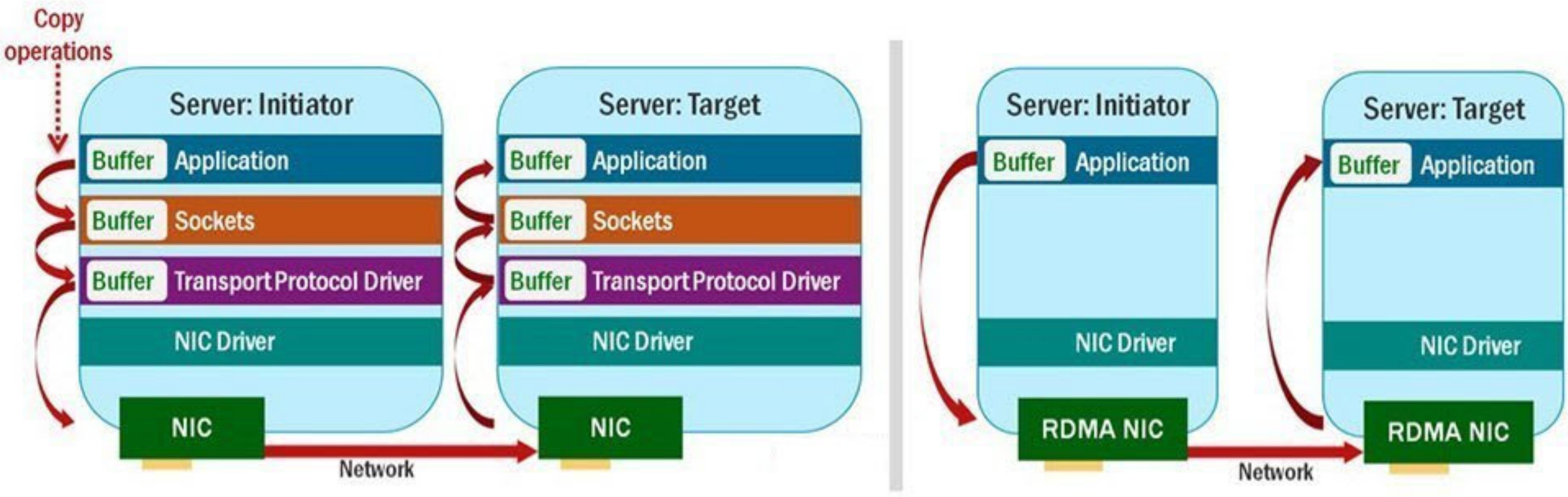
RDMA 1-sided verbs

- Access remote memory through read, write, or atomic operation

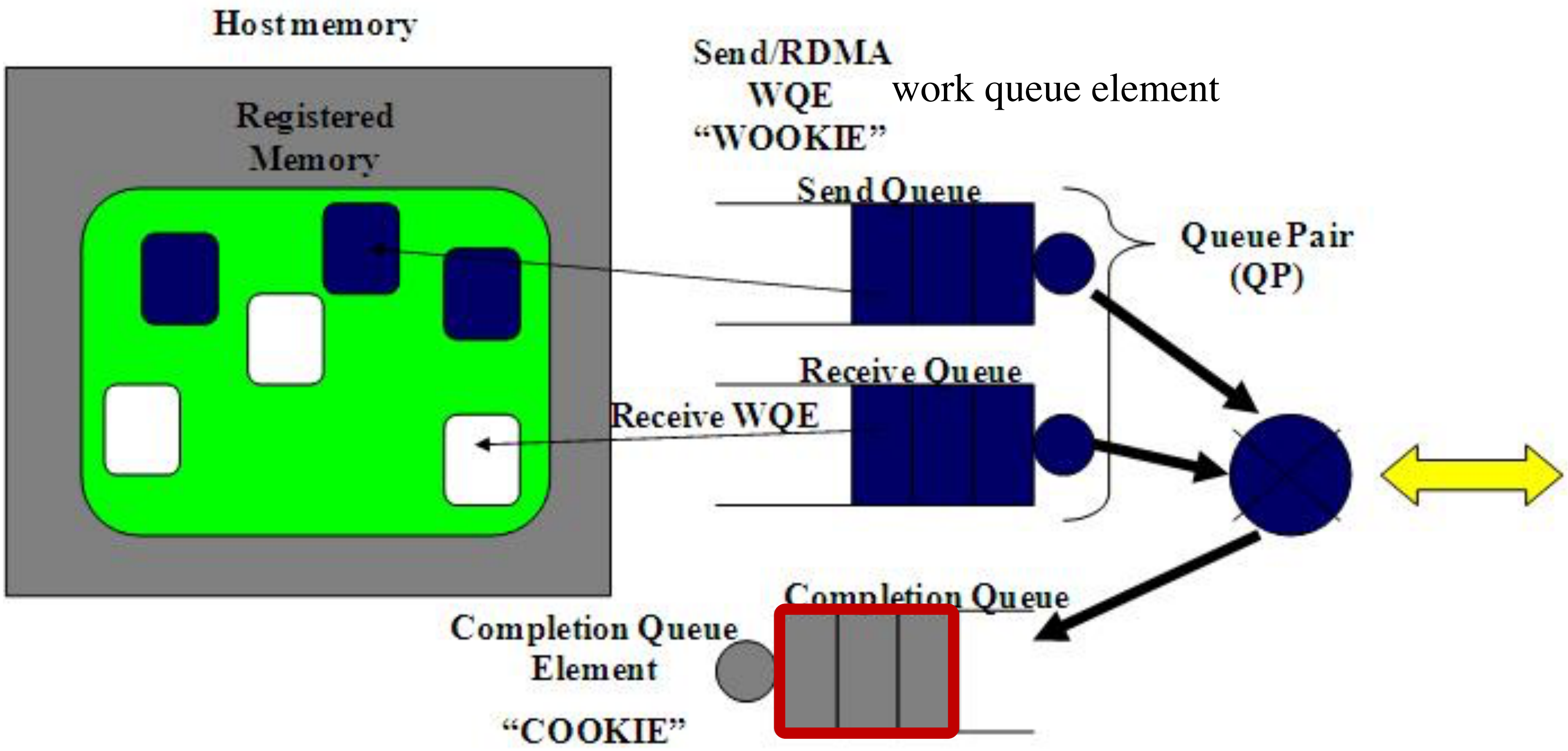
RDMA 2-sided verbs

- Send and Receive

TCP vs. RDMA


Copy operations
Server: Initiator
Buffer
Application
Buffer
Sockets
Buffer
Transport Protocol Driver
NIC Driver
NIC
Network
Server: Target
Buffer
Application
Buffer
Sockets
Buffer
Transport Protocol Driver
NIC Driver
NIC
Server: Initiator
Buffer
Application
NIC Driver
RDMA NIC
Network
Server: Target
Buffer
Application
NIC Driver
RDMA NIC

Queue Pairs



An application allocates a "memory region"

Queue pair: send queue and receive queue

Signaled vs. unsignaled and selective signaling

RDMA Transports

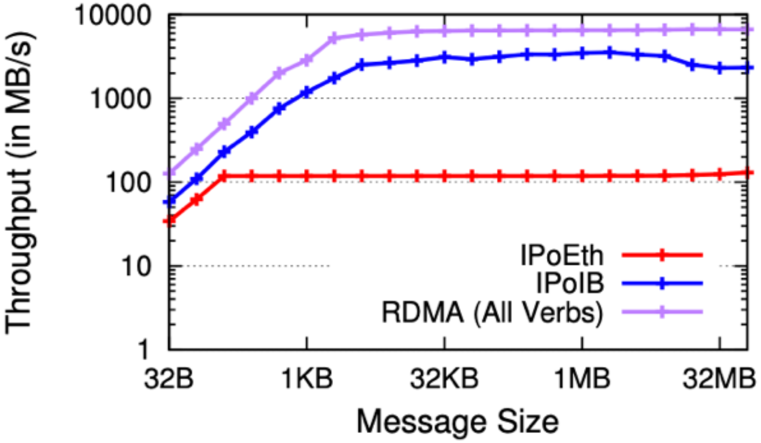
Connected: one QP send/receive with exactly one QP
Unconnected (datagram): one QP send/receive with any QP

Reliable: Messages delivered at most once, in order, and without corruption
Unreliable: No guarantees of delivery nor the order

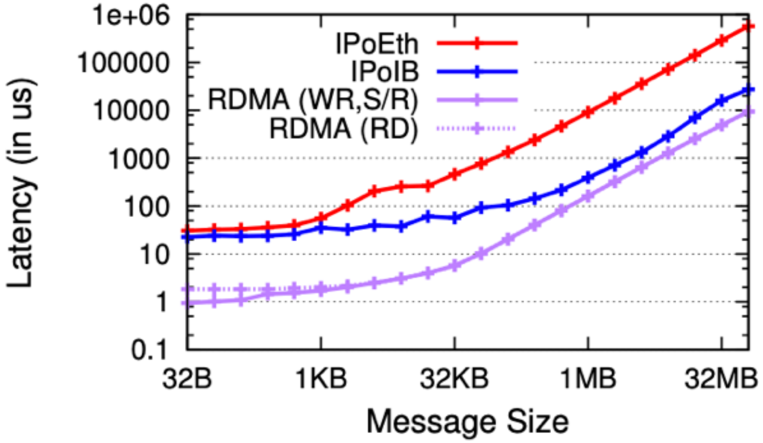
Reliable Connected (similar to TPC)	
Unreliable Connected	Unreliable Datagram (similar to UDP)

Microbenchmarks

Network throughput and latency



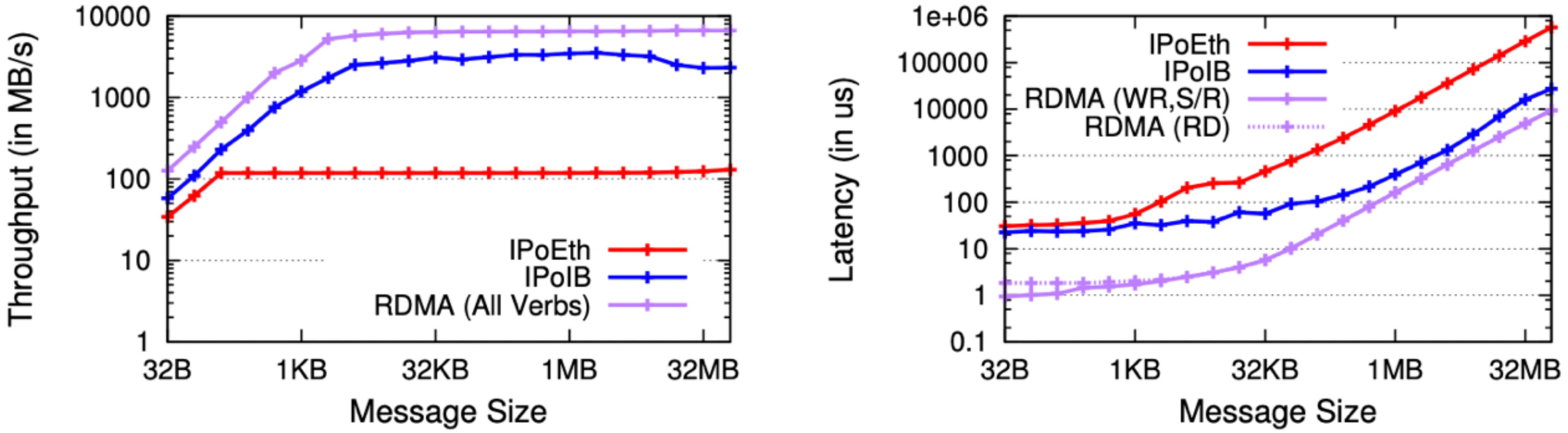
(a) Throughput



(b) Latency

Microbenchmarks

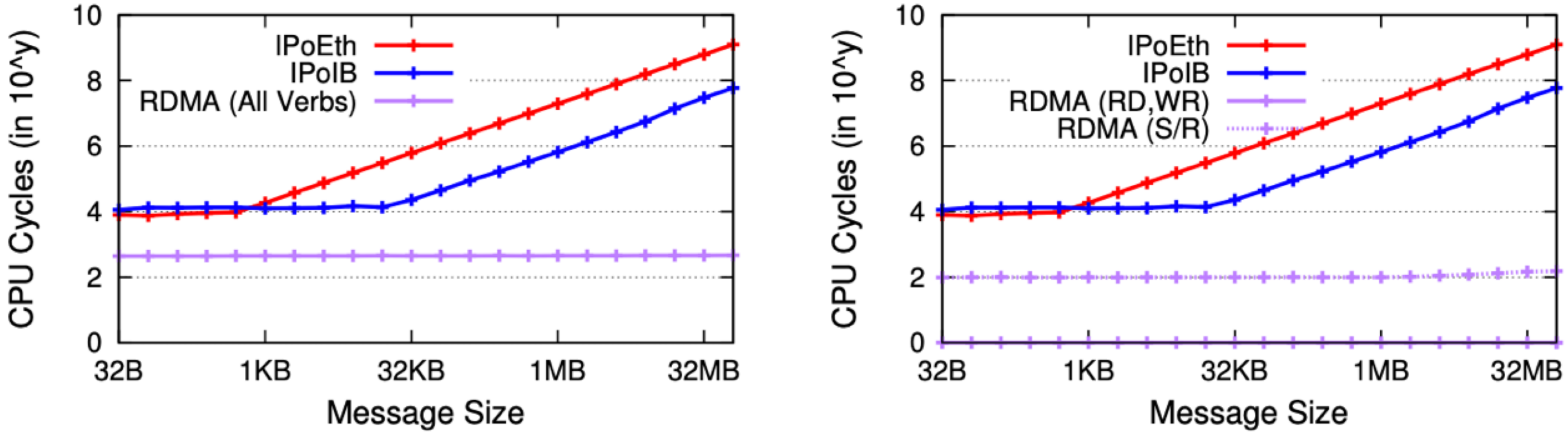
Network throughput and latency



(a) Throughput

(b) Latency

CPU overhead

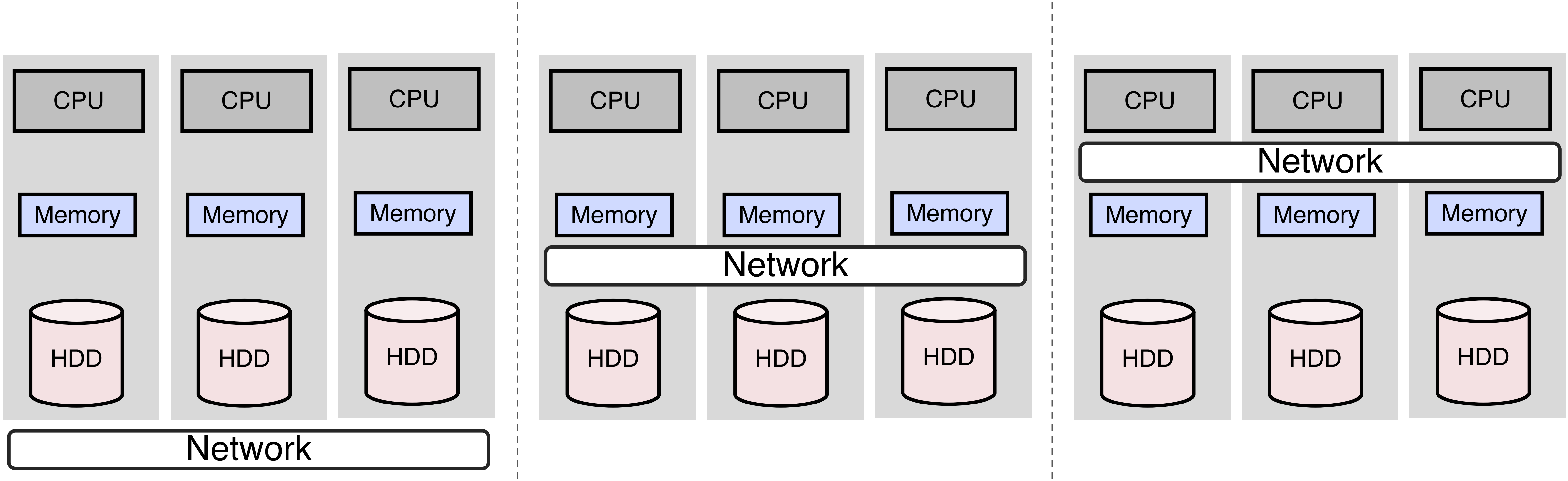


(a) Client

(b) Server

Network Attached Memory (NAM)

Shared-Nothing vs. Shared-Memory

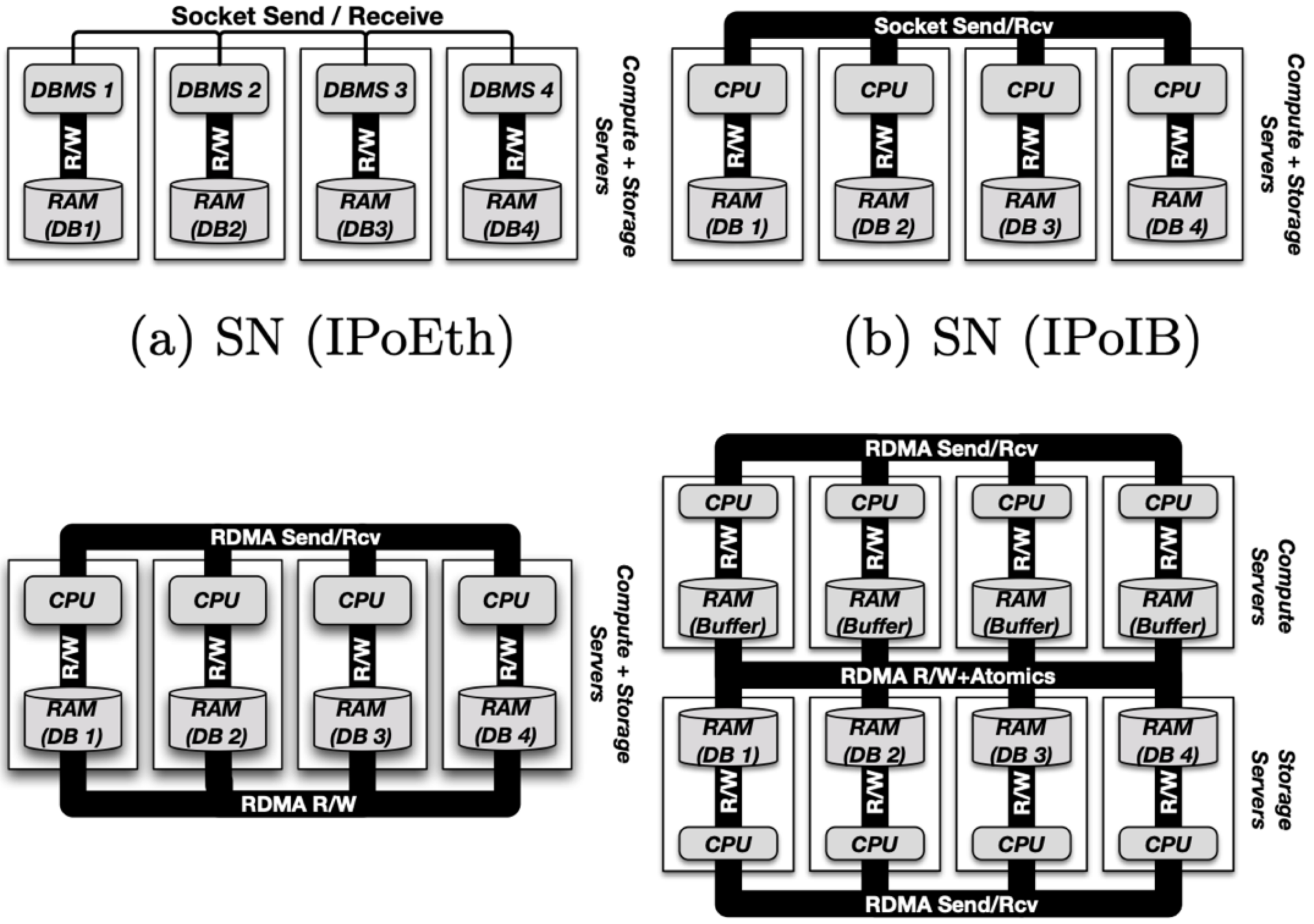


Shared Nothing

Shared Disk

Shared Memory

Shared-Nothing vs. Shared-Memory



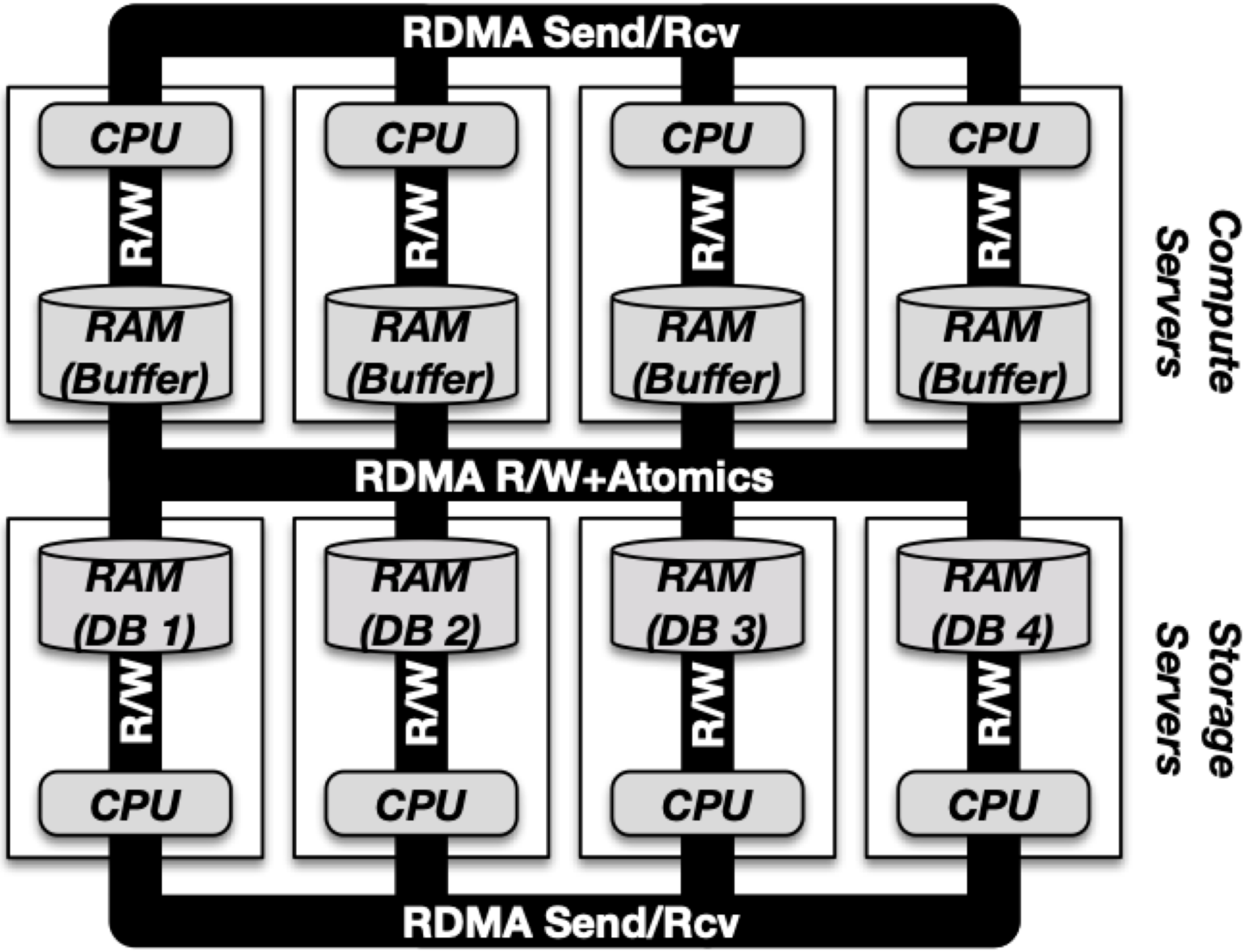
(a) SN (IPoEth)

(b) SN (IPoIB)

(c) SM (RDMA)

(d) NAM (RDMA)

Network-Attached Memory



Shared memory + memory disaggregation

- Storage nodes scale independently of compute nodes
- Efficiently handle data imbalance

RDMA for OLTP

Snapshot Isolation

- **Snapshot isolation (SI)**: All reads see a consistent snapshot of the database that contains **last committed** values at the time the transaction started, no updates conflict with any concurrent updates made since that snapshot.

Initailly checking.balance = 1000

```
If checking.balance > 100
     bal = checking.balance
     bal = bal – 100
     checking.balance = bal
```

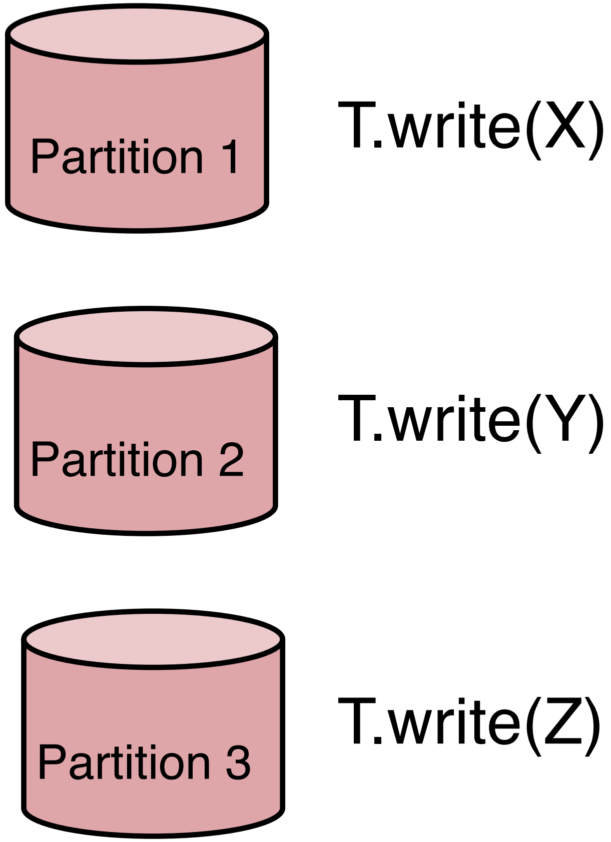
```
If checking.balance > 100
     bal = checking.balance
     bal = bal – 100
     checking.balance = bal
```

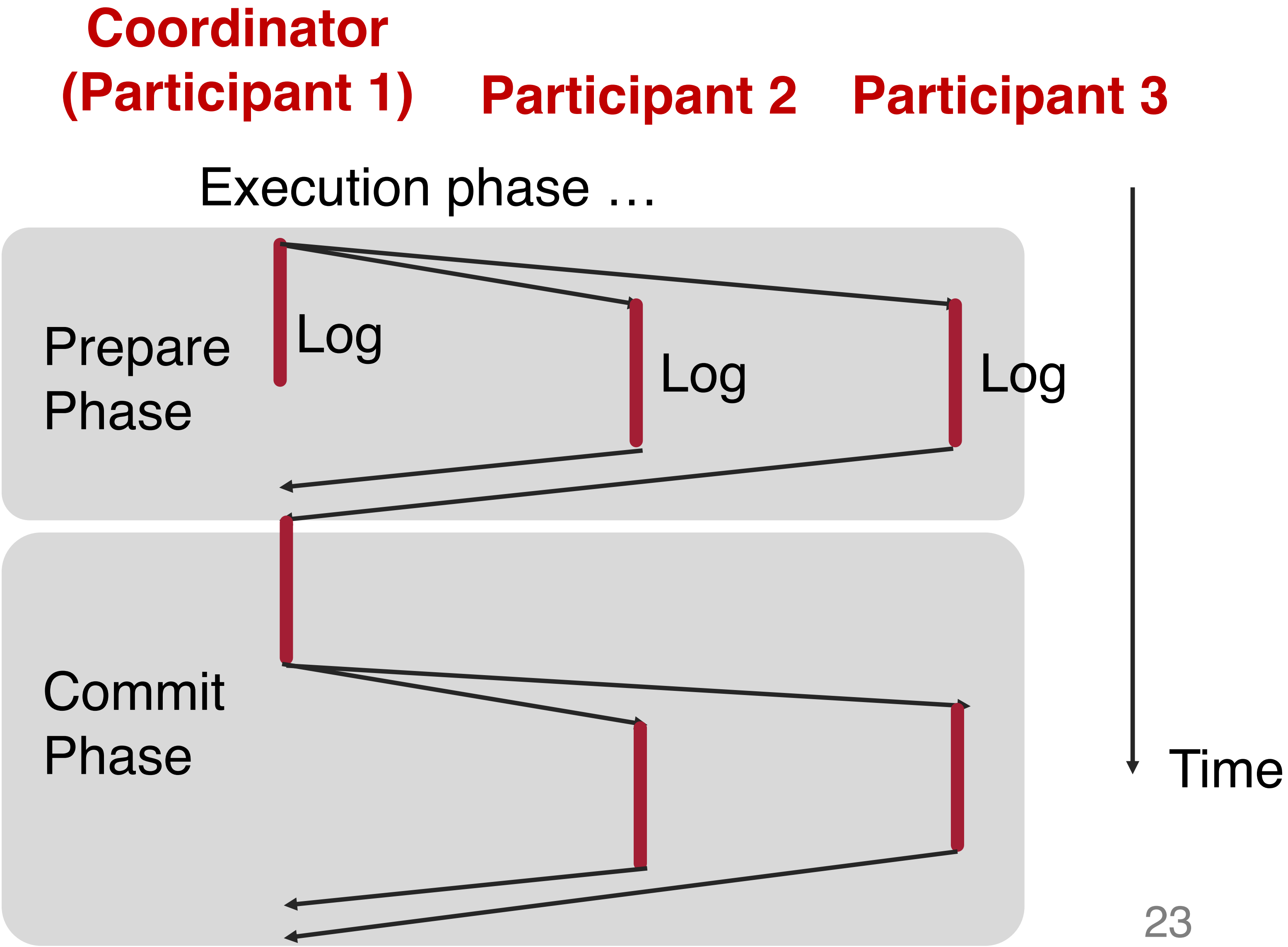
Generalized Snapshot Isolation

- **Snapshot Isolation (SI)**: All reads see a consistent snapshot of the database that contains **last committed** values at the time the transaction started, no updates conflict with any concurrent updates made since that snapshot.

- Generalized Snapshot Isolation (GSI): SI + transaction need not observe the “latest” snapshot.

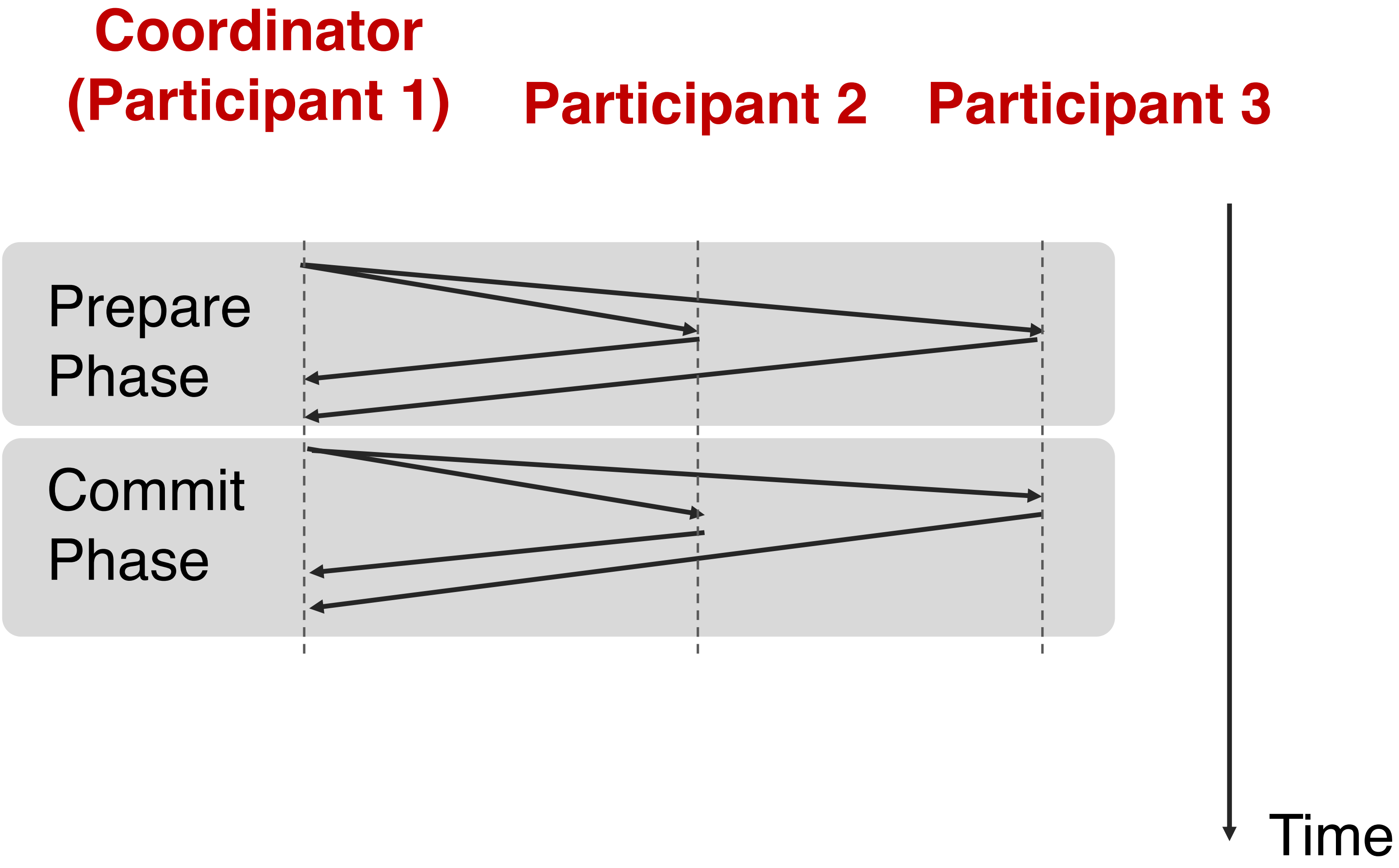
Recap: Two-Phase Commit (2PC)



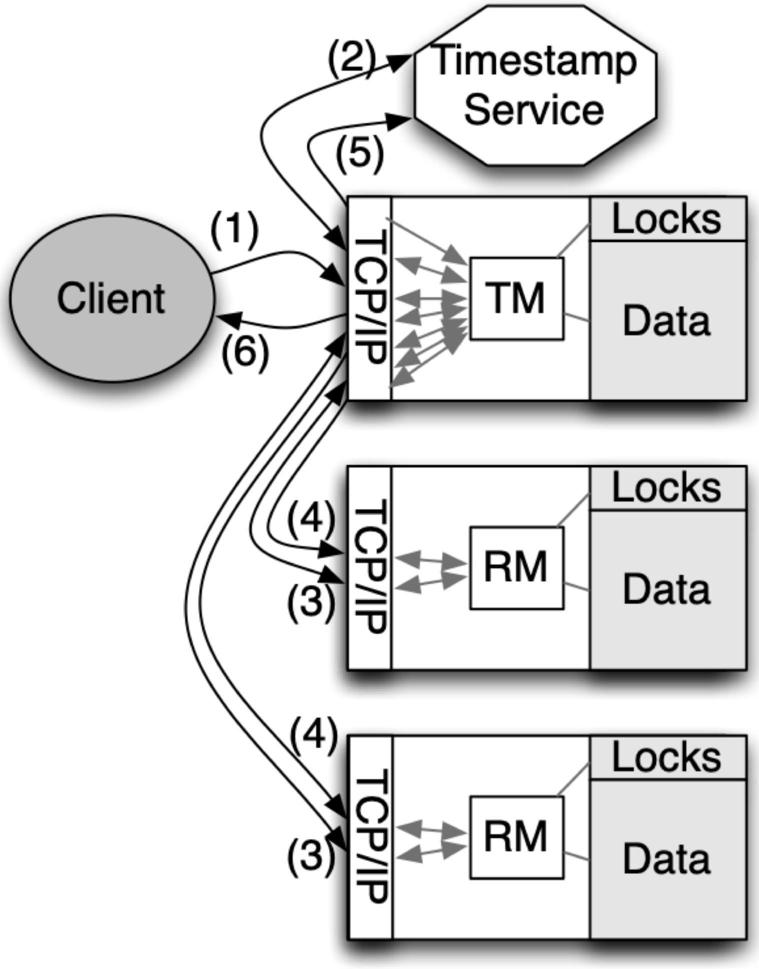


2PC is expensive

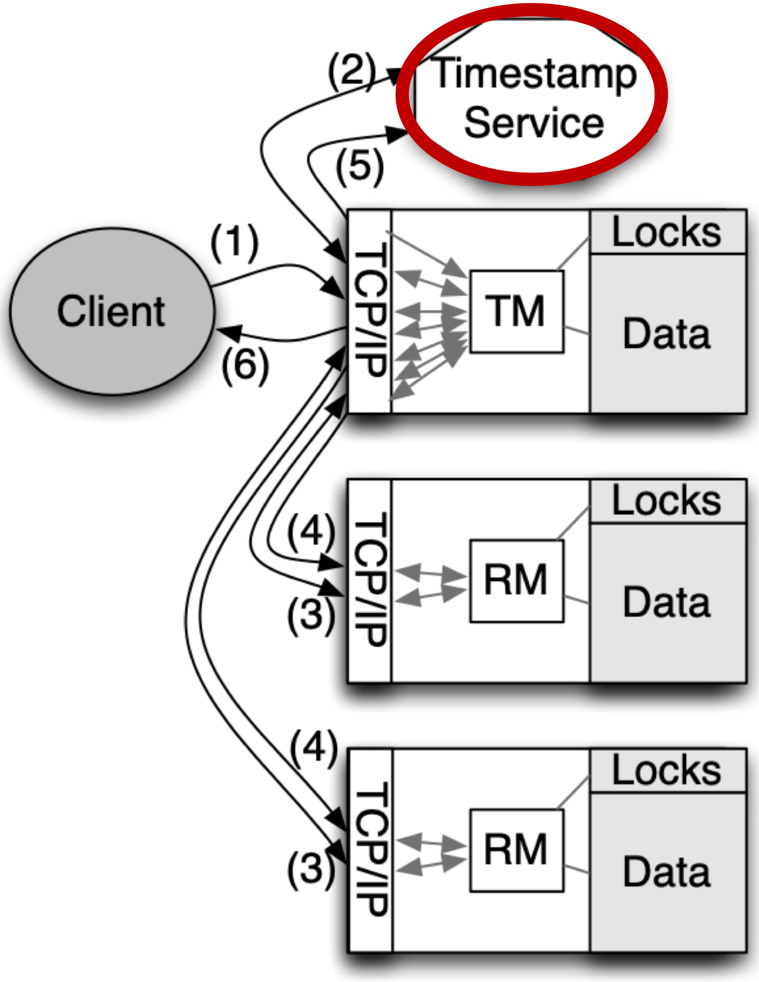
Two-Phase Commit (2PC) w/o Logging



Cost of Traditional 2PC with GSI

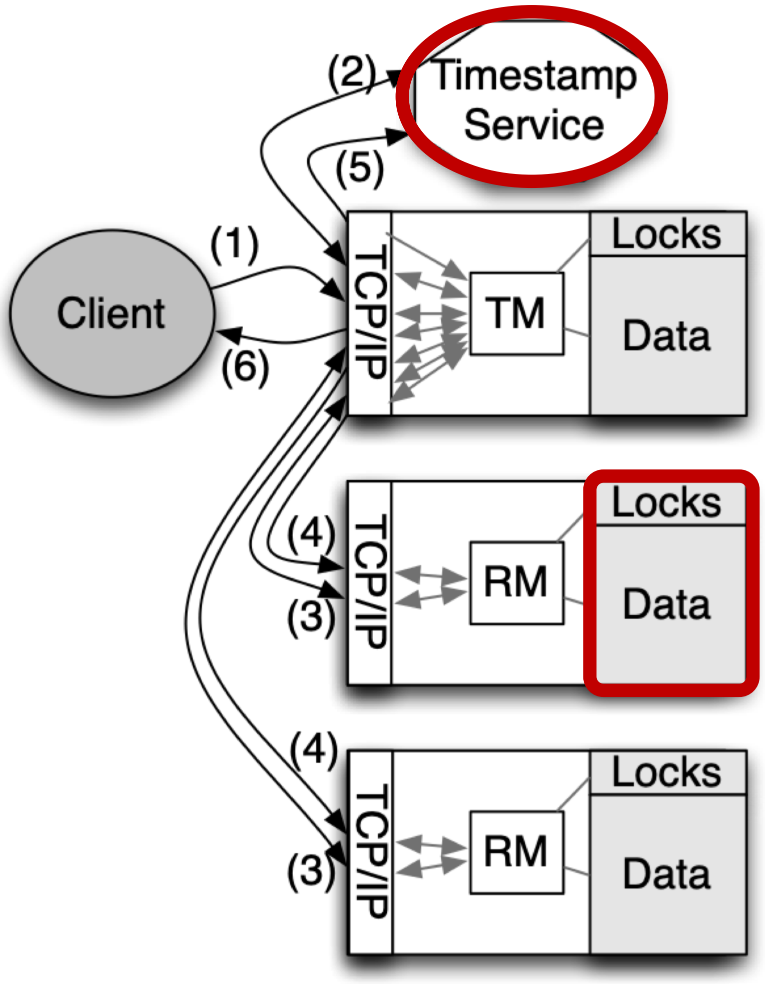


Cost of Traditional 2PC with GSI



Task 1: Find the highest committed timestamp

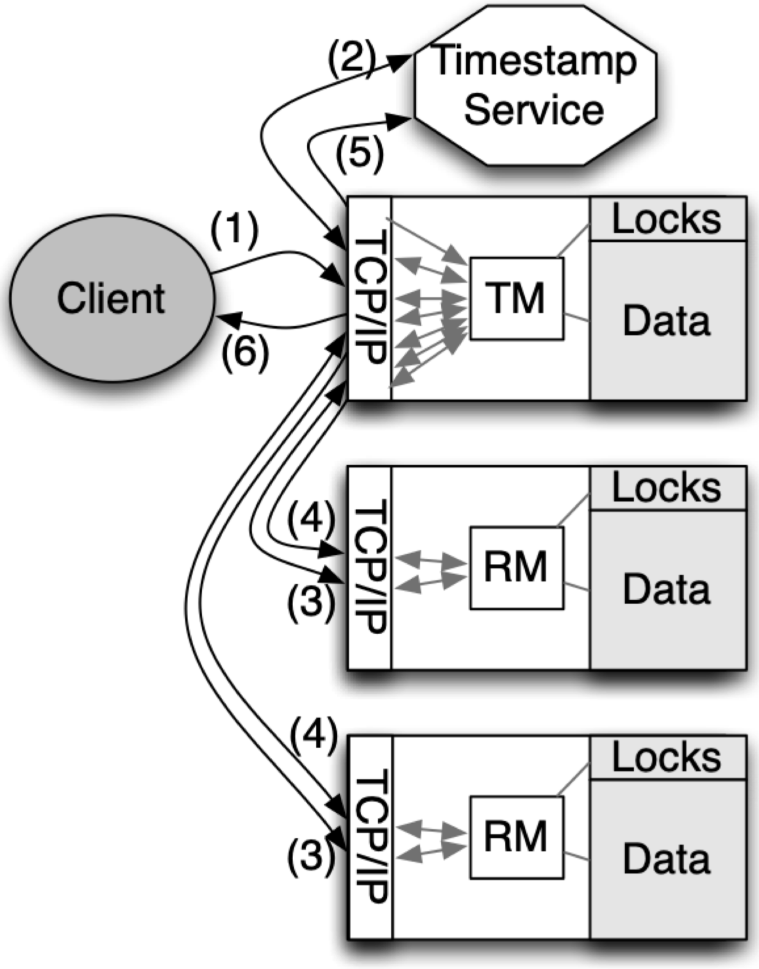
Cost of Traditional 2PC with GSI



Task 1: Find the highest committed timestamp

Task 2: Validate the transaction in each RM (resource managers)

Cost of Traditional 2PC with GSI



With *n* RMs, need to send *2 + 4n* messages and receive *3 + 4n* messages

Throughput upper bound:

$$c \cdot cycles_c \cdot (n+1)/(5+8n) \cdot cycles_m$$

c: core count

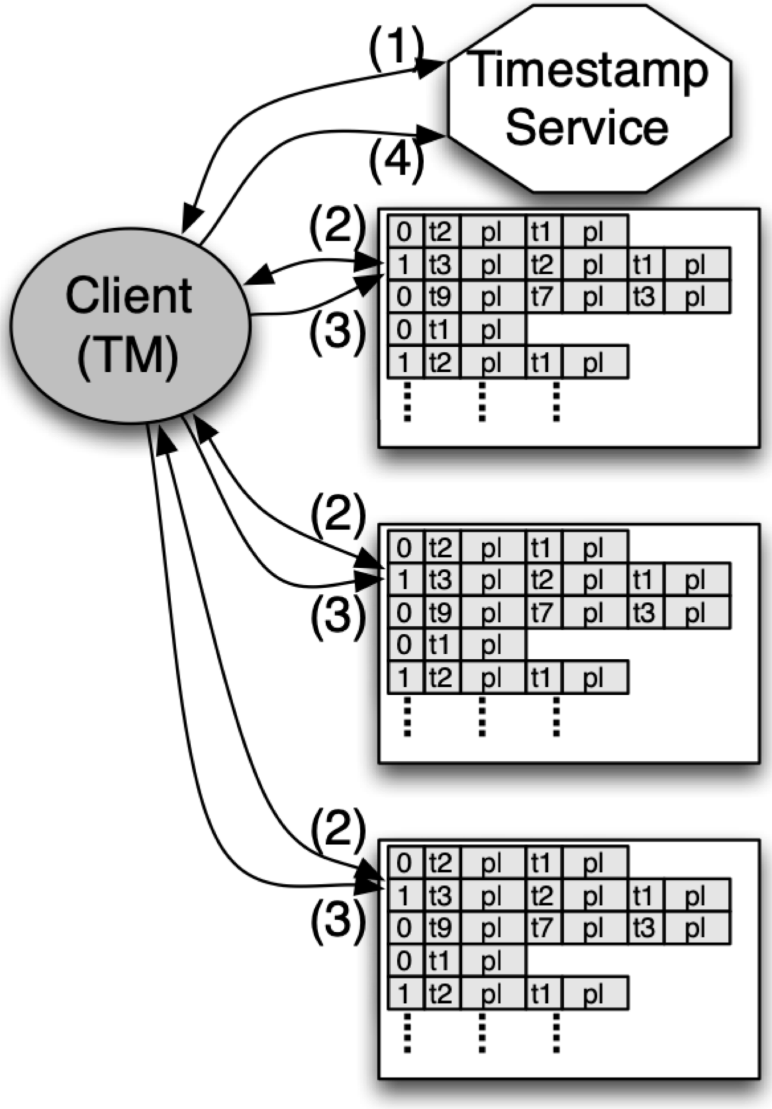
$cycles_c$: a core executes $cycles_c$ per second

$cycles_m$: a message costs $cycles_m$

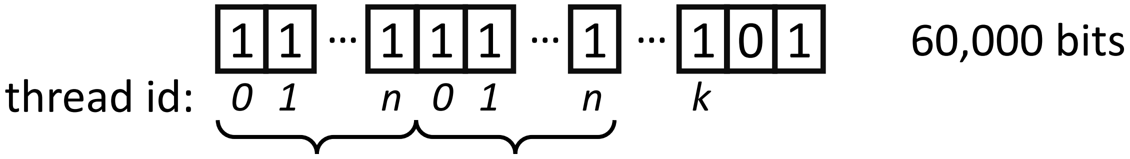
⇒ **647,000 transactions / second**

(All transactions access all nodes)

RDMA-based SI (RSI)

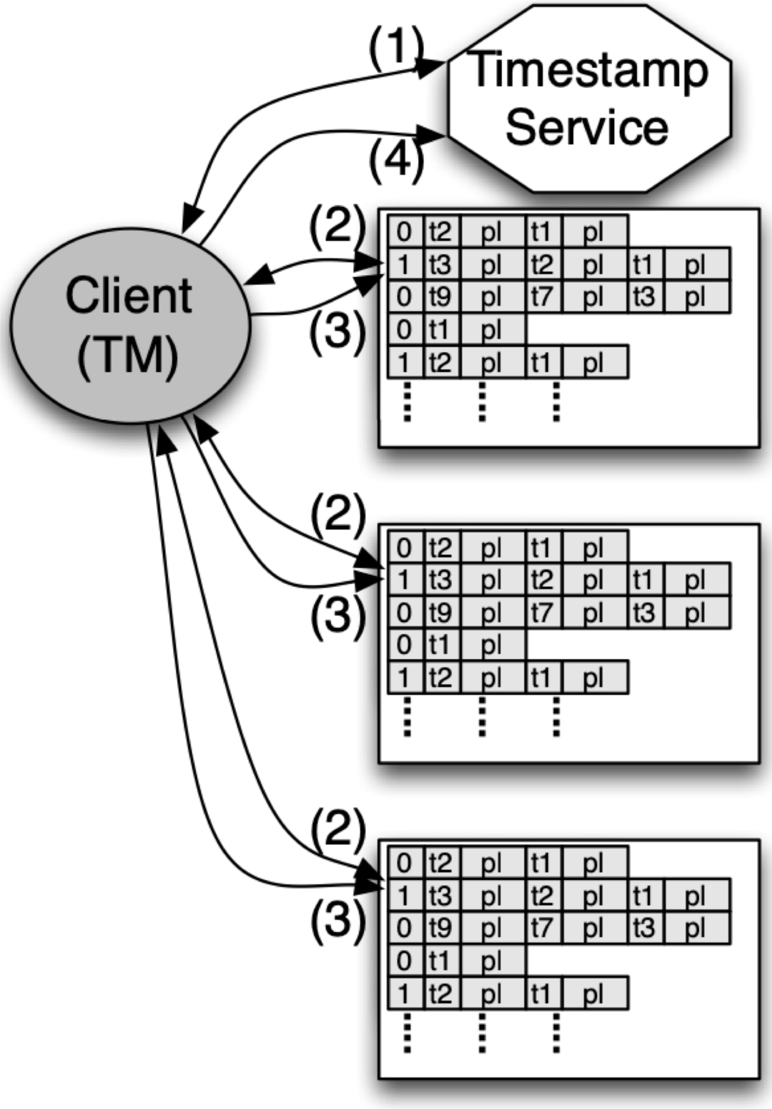


Task 1: Find the highest committed timestamp

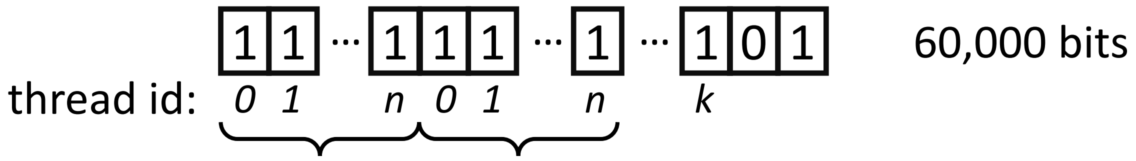


- Wrap around when all bits are set
- assume all threads make progress at the same rate

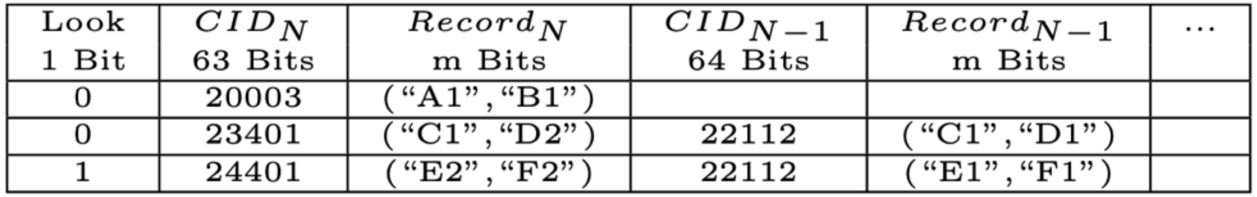
RDMA-based SI (RSI)



Task 1: Find the highest committed timestamp



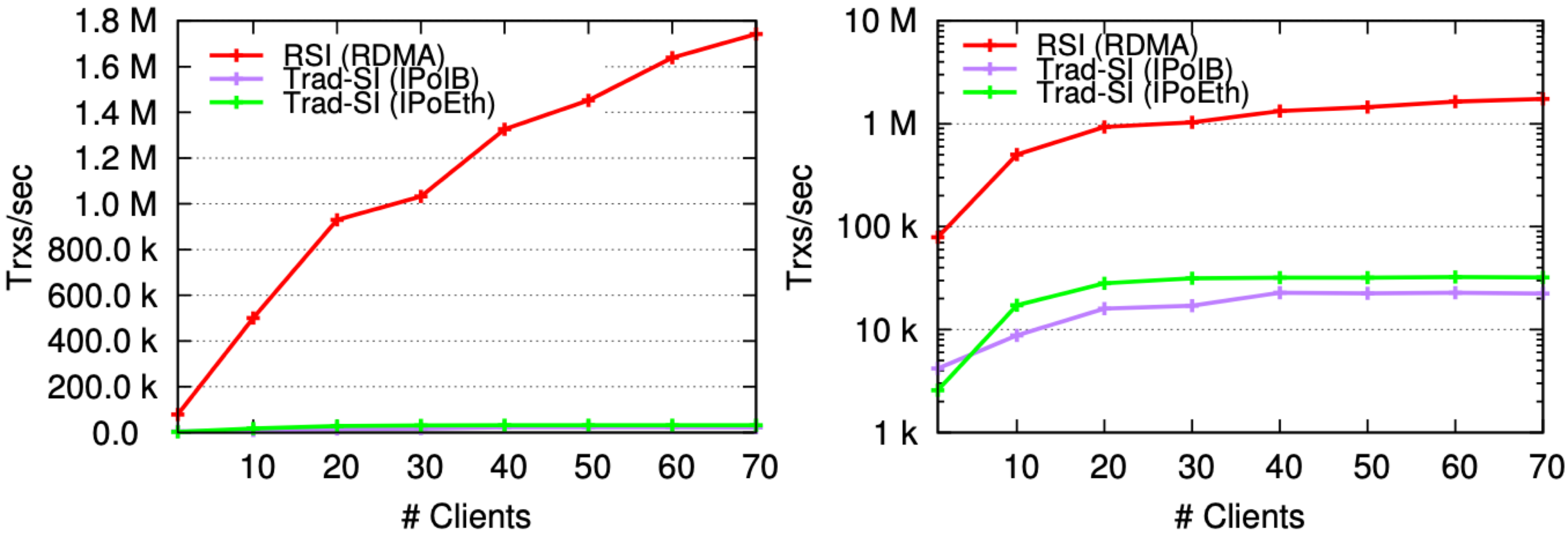
Task 2: Validate the transaction in each RM (resource managers)



Look 1 Bit	CID_N 63 Bits	$Record_N$ m Bits	CID_{N-1} 64 Bits	$Record_{N-1}$ m Bits	...
0	20003	(“A1”,“B1”)			
0	23401	(“C1”,“D2”)	22112	(“C1”,“D1”)	
1	24401	(“E2”,“F2”)	22112	(“E1”,“F1”)	

- Lock and validation with a single remote compare-and-swap

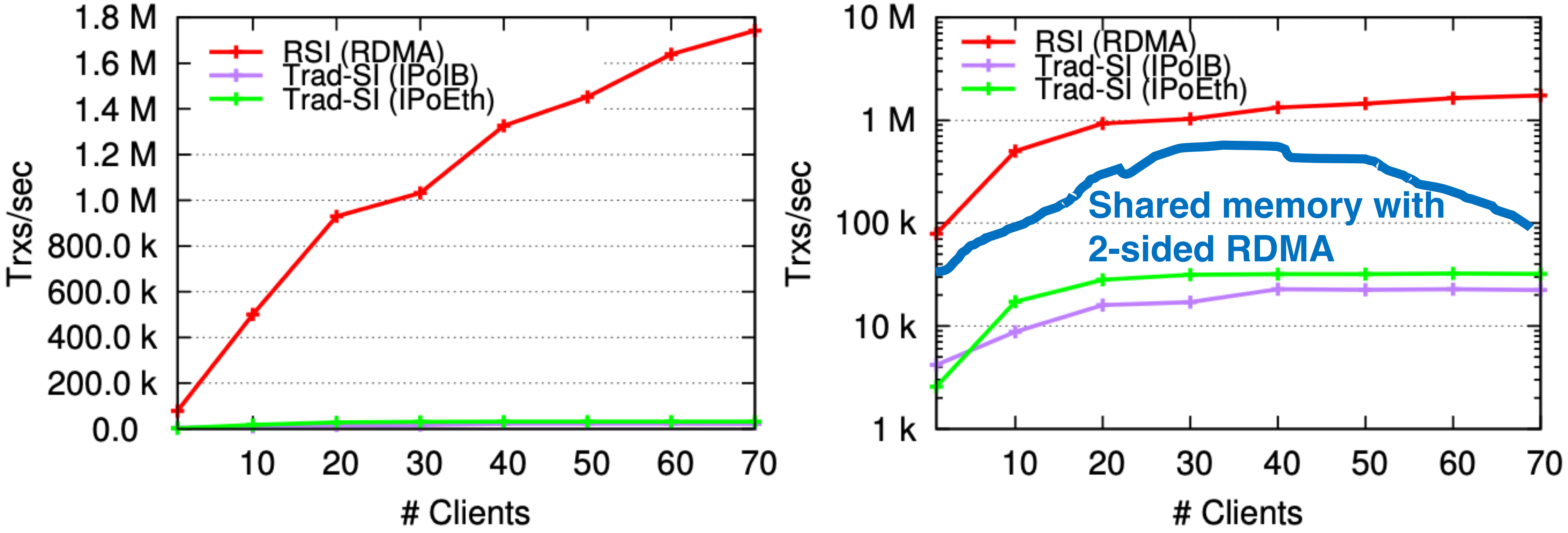
OLTP Evaluation



(a) Linear-Scale

(b) Log-Scale

OLTP Evaluation



Shared memory with 2-sided RDMA

(a) Linear-Scale

(b) Log-Scale

RDMA for OLAP

Grace Hash Join (GHJ)

Phase 1: Partitioning R and S using the join key

Phase 2: Local join

Grace Hash Join (GHJ)

Phase 1: Partitioning R and S using the join key

1. Read data on the sender
2. Transfer data over network
3. Materialize data on the receiver

$$
\begin{aligned}
T_{part}(R) &= T_{mem}(R) + T_{net}(R) + T_{mem}(R) \\
&= w_r|R|c_{mem} + w_r|R|c_{net} + w_r|R|c_{mem} \\
&= w_r\ (2\ c_{mem}|R| + c_{net}|R|)
\end{aligned}
$$

w_r: width of a tuple

c_{net}/c_{mem} : cost of accessing a tuple over the network/memory

Grace Hash Join (GHJ)

Phase 1: Partitioning R and S using the join key

Phase 2: Local join (parallel radix join)

$$T_{join}(R,S) = \underbrace{(T_{mem}(R) + T_{mem}(S))}_{\text{Radix Phase 1}} + \underbrace{(T_{mem}(R) + T_{mem}(S))}_{\text{Radix Phase 2}}$$

$$= 2 \cdot c_{mem} \cdot (w_r \cdot |R| + w_s \cdot |S|)$$

$$T_{part}(R) = w_r \; (2 \, c_{mem} |R| + c_{net} |R|)$$

$$T_{part}(S) = w_r \; (2 \, c_{mem} |S| + c_{net} |S|)$$

$$T_{GHJ} = T_{part}(R) + T_{part}(S) + T_{join}(R,S)$$

$$= (w_r|R| + w_s|S|) \cdot \boxed{(4 \cdot c_{mem} + c_{net})}$$

Minimizing network traffic

Semi-Reduction using Bloom Filters

Phase 1: create bloom filters for R and S on the join key

Phase 2: filter R and S using the bloom filter and partition

Phase 3: Local join (parallel radix join)

$$\begin{aligned} T_{join+bloom} =& (w_r|R| + w_s|S|) \cdot \\ & (c_{mem} + 4 \cdot sel \cdot c_{mem} + sel \cdot c_{net}) \end{aligned}$$

Semi-Reduction using Bloom Filters

Phase 1: create bloom filters for R and S on the join key

Phase 2: filter R and S using the bloom filter and partition

Phase 3: Local join (parallel radix join)

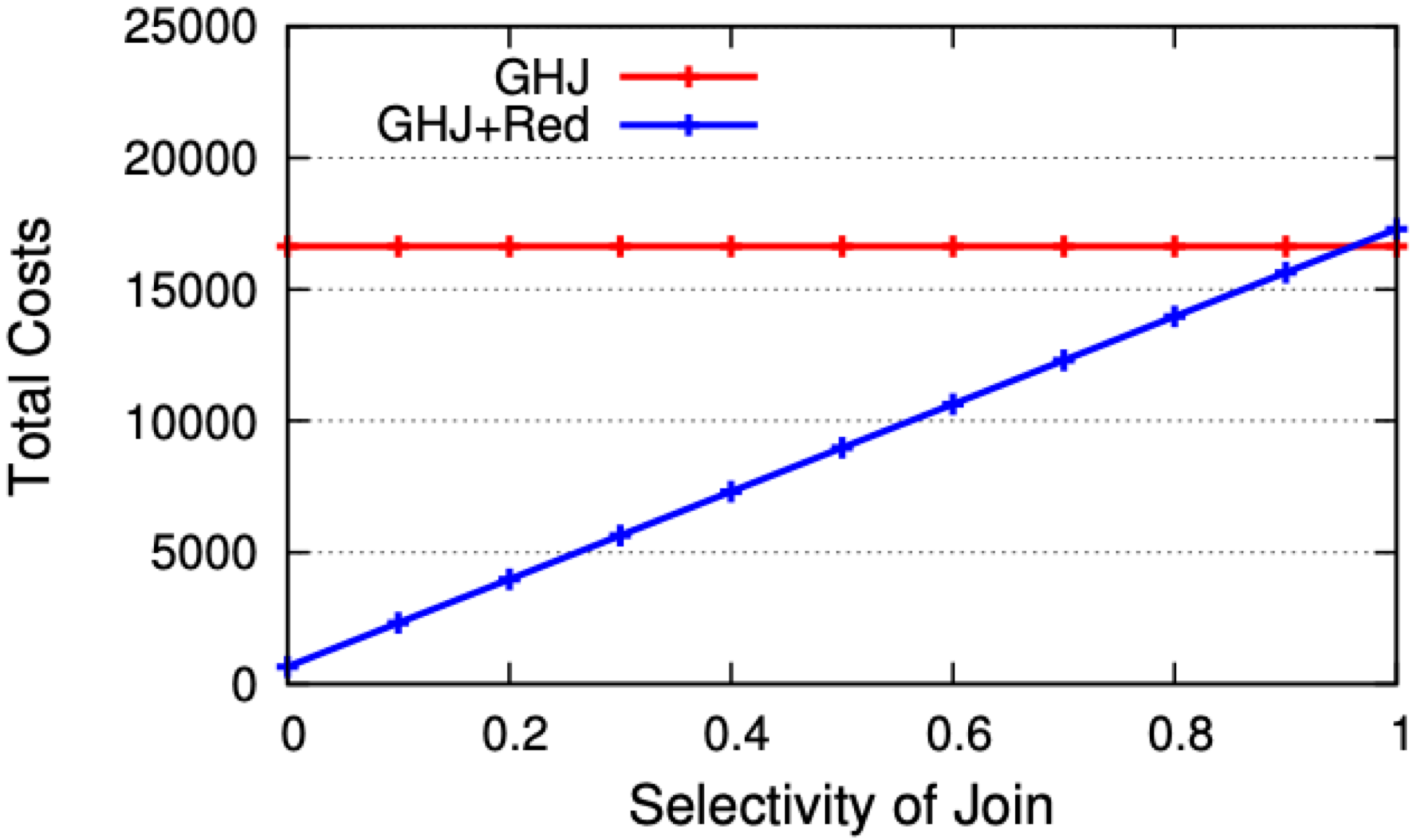
$$\begin{aligned} T_{join+bloom} =& (w_r|R| + w_s|S|)\cdot \\ & (c_{mem} + 4 \cdot sel \cdot c_{mem} + \boxed{sel \cdot c_{net})} \end{aligned}$$

Further reducing network traffic

$$\begin{aligned} T_{GHJ} &= T_{part}(R) + T_{part}(S) + T_{join}(R, S) \\ &= (w_r|R| + w_s|S|) \cdot (4 \cdot c_{mem} + c_{net}) \end{aligned}$$

GHJ and Reduction

Join cost with IPoEth



GHJ with RDMA

For the partitioning phase, use 1-sided RDMA to directly write to remote memory

Phase 1: Partitioning R and S using the join key

1. Read data on the sender
2. Transfer data over network
3. ~~Materialize data on the receiver~~

$$T_{part}(R) = T_{mem}(R) + T_{net}(R) + \cancel{T_{mem}(R)}$$

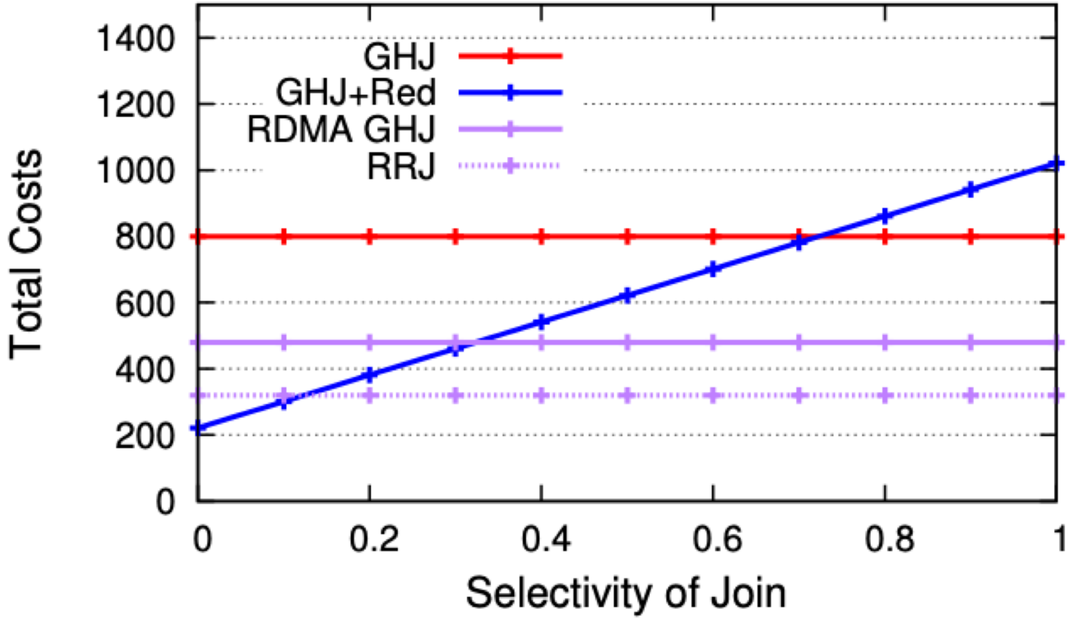
RDMA Radix Join (RRJ)

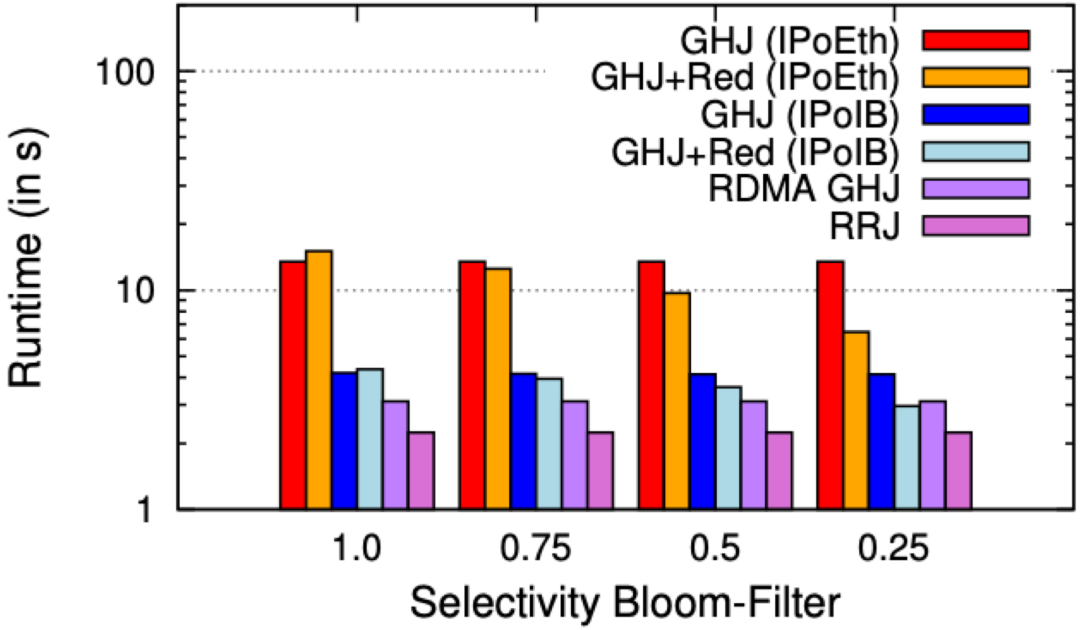
Extend in-memory radix join to leverage RDMA directly

$$\begin{aligned} T_{join}(R, S) &= \underbrace{(T_{mem}(R) + T_{mem}(S))}_{\text{Radix Phase 1}} + \underbrace{(T_{mem}(R) + T_{mem}(S))}_{\text{Radix Phase 2}} \\ &= 2 \cdot c_{mem} \cdot (w_r \cdot |R| + w_s \cdot |S|) \end{aligned}$$

(assuming network cost is similar to memory cost)

Evaluation of Joins





RDMA-DB – Q/A

No comparison to the state-of-the-art

Research space relatively new?

Evolvement of RDMA changing the design space

RDMA vs. InfiniBand

PCIe bandwidth bottleneck

Open challenges of RDMA databases?

RDMA vs. DMA

Scalability analysis missing

- RDMA does not scale well for large number of concurrent connections due to cache thrashing

Other network technologies?

Group Discussion

What are the opportunities of using RDMA in the execution phase of transaction processing? (e.g., locking, indexing, etc.)

Name a few components in a database (and distributed systems in general) that may be significantly affected by a faster network.

NAM architecture disaggregates computation and memory. What in your opinion are the opportunities and challenges of memory disaggregation?

Before Next Lecture

Submit discussion summary to https://wisc-cs839-ngdb20.hotcrp.com

- **Deadline: Wednesday 11:59pm**

Submit project proposal today

Submit review for

- Rethinking Database High Availability with RDMA Networks
- [Optional] Query Fresh: Log Shipping on Steroids