Deconstructing RDMA-enabled Distributed Transaction Processing: Hybrid is Better!

Xingda Wei, Zhiyuan Dong, Rong Chen, Haibo Chen

Institute of Parallel and Distributed Systems (**IPADS**)
Shanghai Jiao Tong University

Remote Direct Memory Access (RDMA)

Kernel bypassing network

- Ultra **low** latency~(5us)
- Ultra **high** throughput

Offloading technology (one-sided)

- **Bypassing** CPU
- **Read/Write, CAS**[2] server's memory

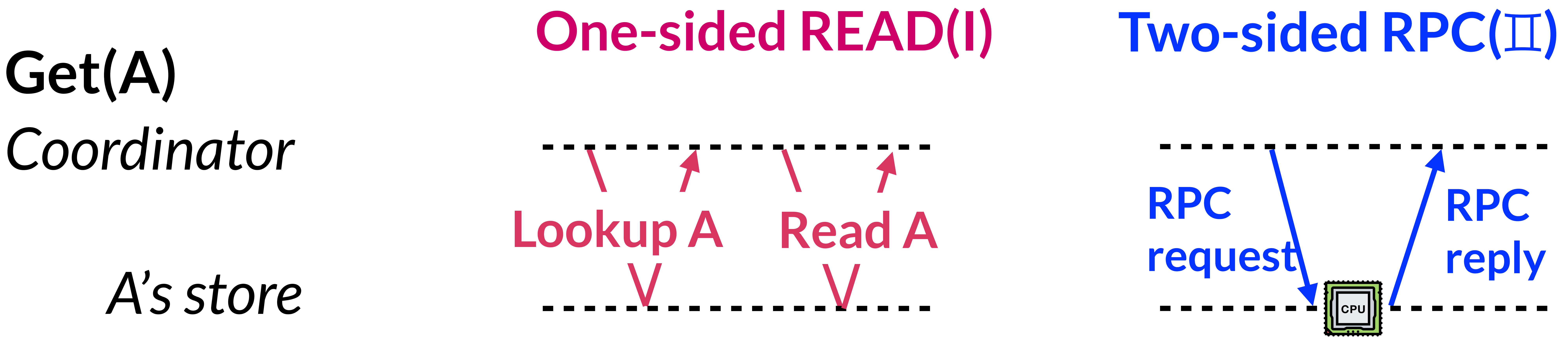
Gain interests from **Academia** & **Industry**

- **Orders of magnitude improvements** on distributed applications
- Available in the **public cloud**[1]

[1] https://azure.microsoft.com/en-us/blog/azure-linux-rdma-hpc-available/

[2] Atomic compare and swap

On-going debate over how to use RDMA for TXs

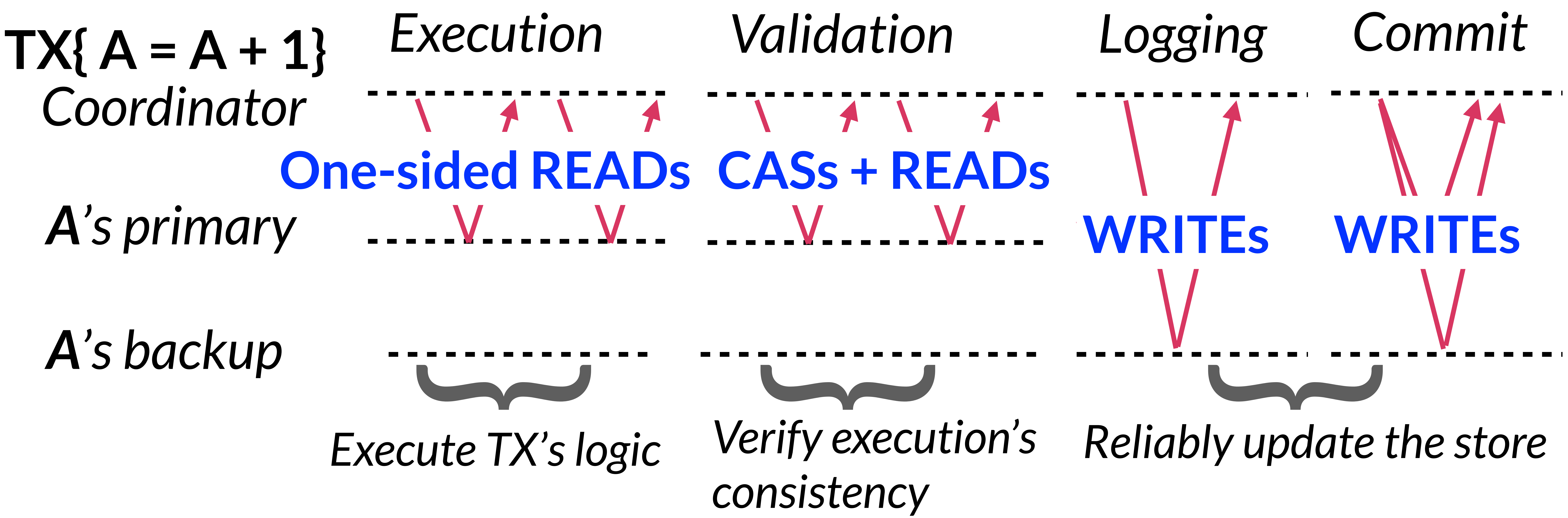


	One-sided READ(I)	Two-sided RPC(Ⅱ)
Performance	✓	✖
#Round-trips	>= 2	1

Transaction(TX)s are more complex

TX (e.g. OCC[1]) uses multiple **phases** for **serializability** & **availability**

- Each **can be offloaded** w one-sided primitive



[1] Optimistic concurrency control

Transaction(TX)s are more complex

Protocols

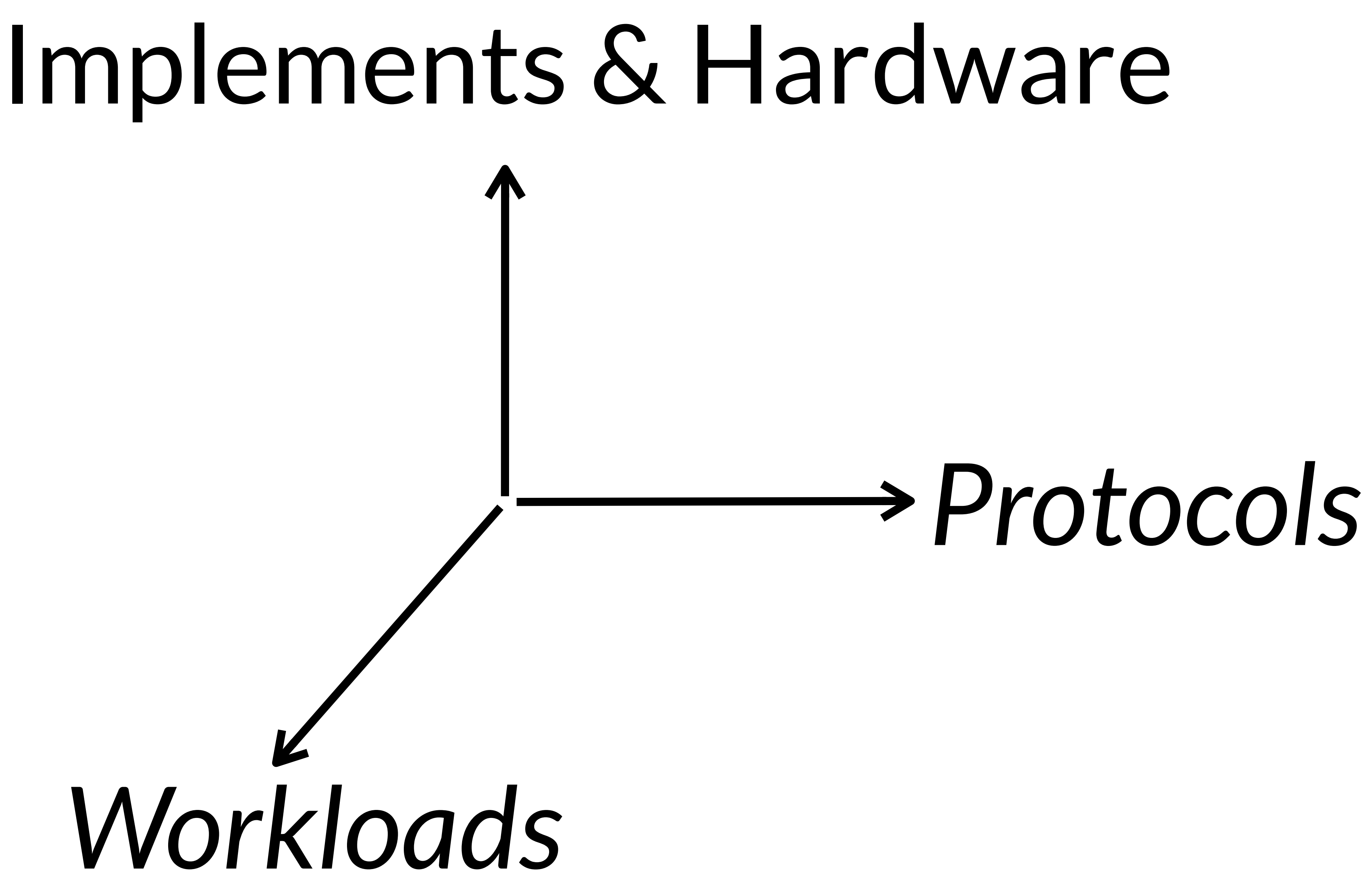
- OCC, 2PL, SI, ….

Impl on hardware devices

- One-sided vs. Two-sided, …
- CX3, CX4, CX5, ROCE

OLTP workloads

- TPC-C, TPC-E, TATP, Smallbank, …



This work: how to use RDMA for TXs

Focus on **OCC** in this work

- Use **phase-by-phase** approach

Well-tuned RDMA execution framework

- Representative RNICs (CX3 - CX5)

Representative OLTP workloads

- TPC-C, TPC-E, and Smallbank

Implements & Hardware

Workload

Optimistic Concurrency Control

Widely used in

Centralized

Silo[SOSP'13] Foedus[SIGMOD'15]

...

Distributed

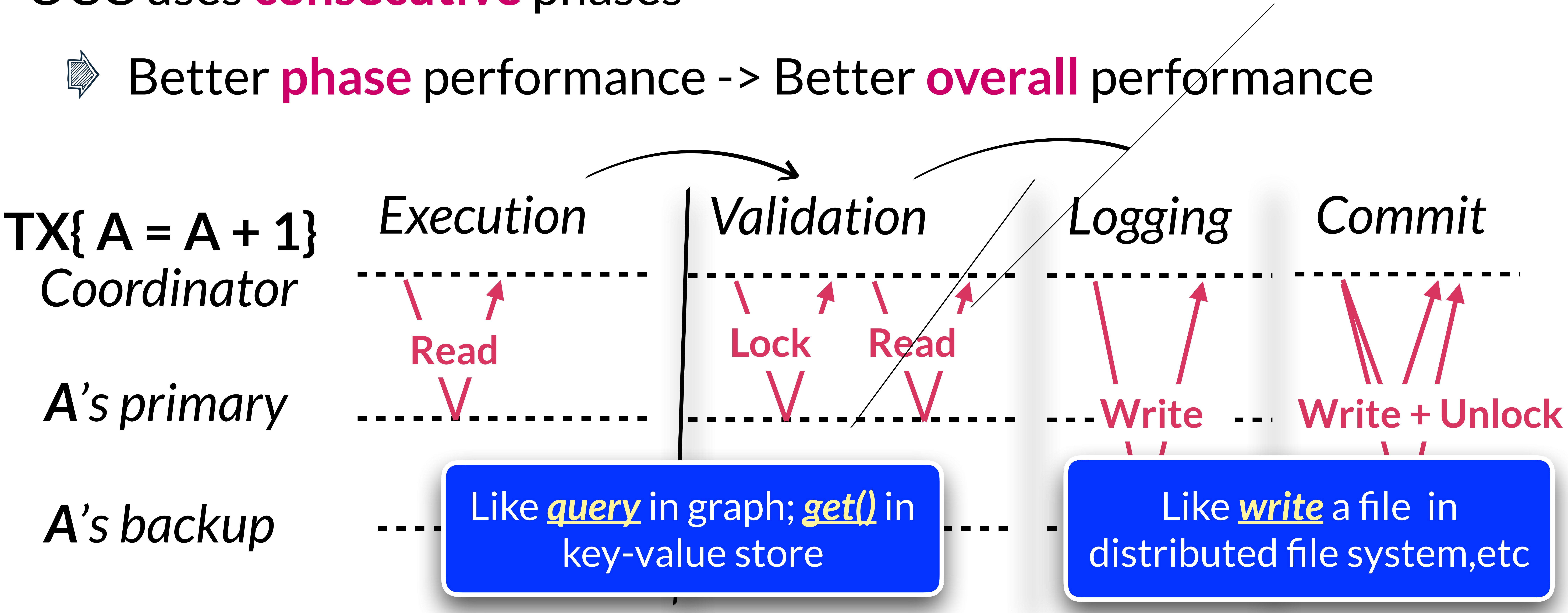
FaRM[SOSP'15] TAPIR[SOSP'15]

...

Phase-by-phase analysis is effective & useful

OCC uses **consecutive** phases

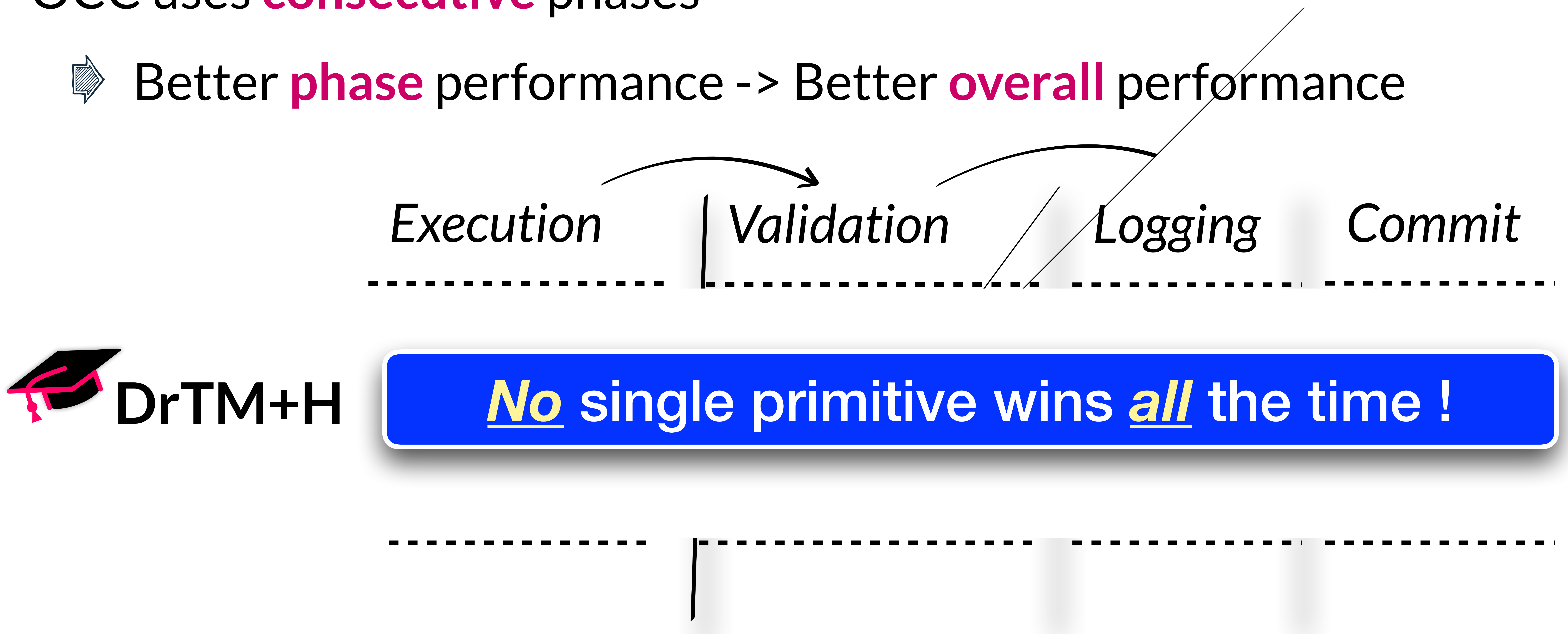
- Better **phase** performance -> Better **overall** performance



Deconstructing TX with phase-by-phase analysis

OCC uses **consecutive** phases

- Better **phase** performance -> Better **overall** performance



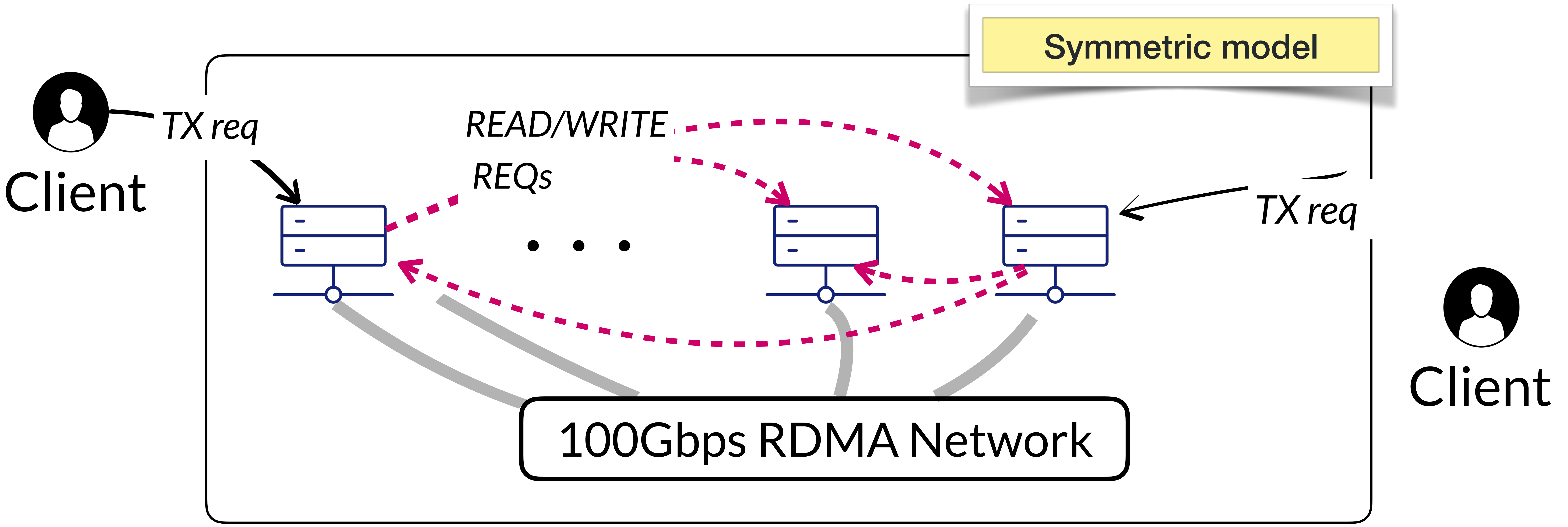
No **single primitive wins *all* the time !**

https://github.com/SJTU-IPADS/drtmh

Outline

- ☑ RDMA primitive-level analysis
- ☑ Phase-by-phase analysis for TX
- ☑ DrTM+H: Putting it all together

System model & evaluation setup

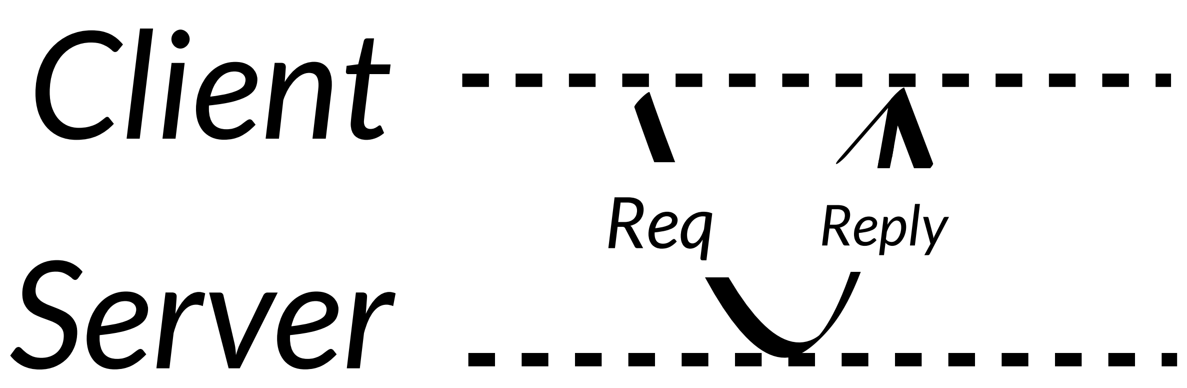


Evaluation setup

16 x

CPU	RNIC	Link layer
24 cores	2 * ConnectX-4 RNIC	Infiniband

Primitive analysis

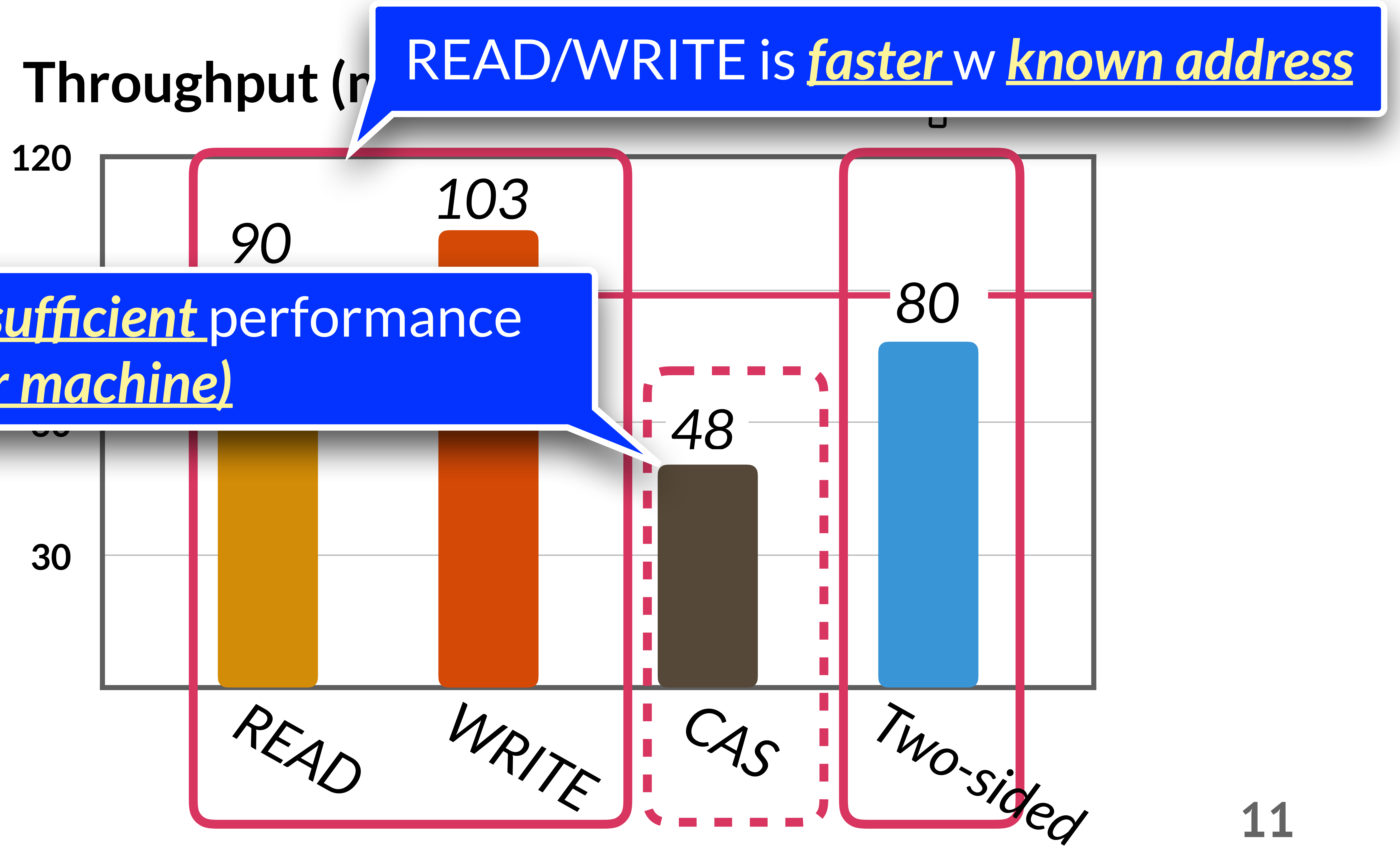


One-sided primitive

- Simple implementation (Native verbs API)
- Optimized event loop (Asyn

Two-sided (RPC)

- FaSST RPC [OSDI'16]
- Fastest in our setting



Passive ACK (PA)

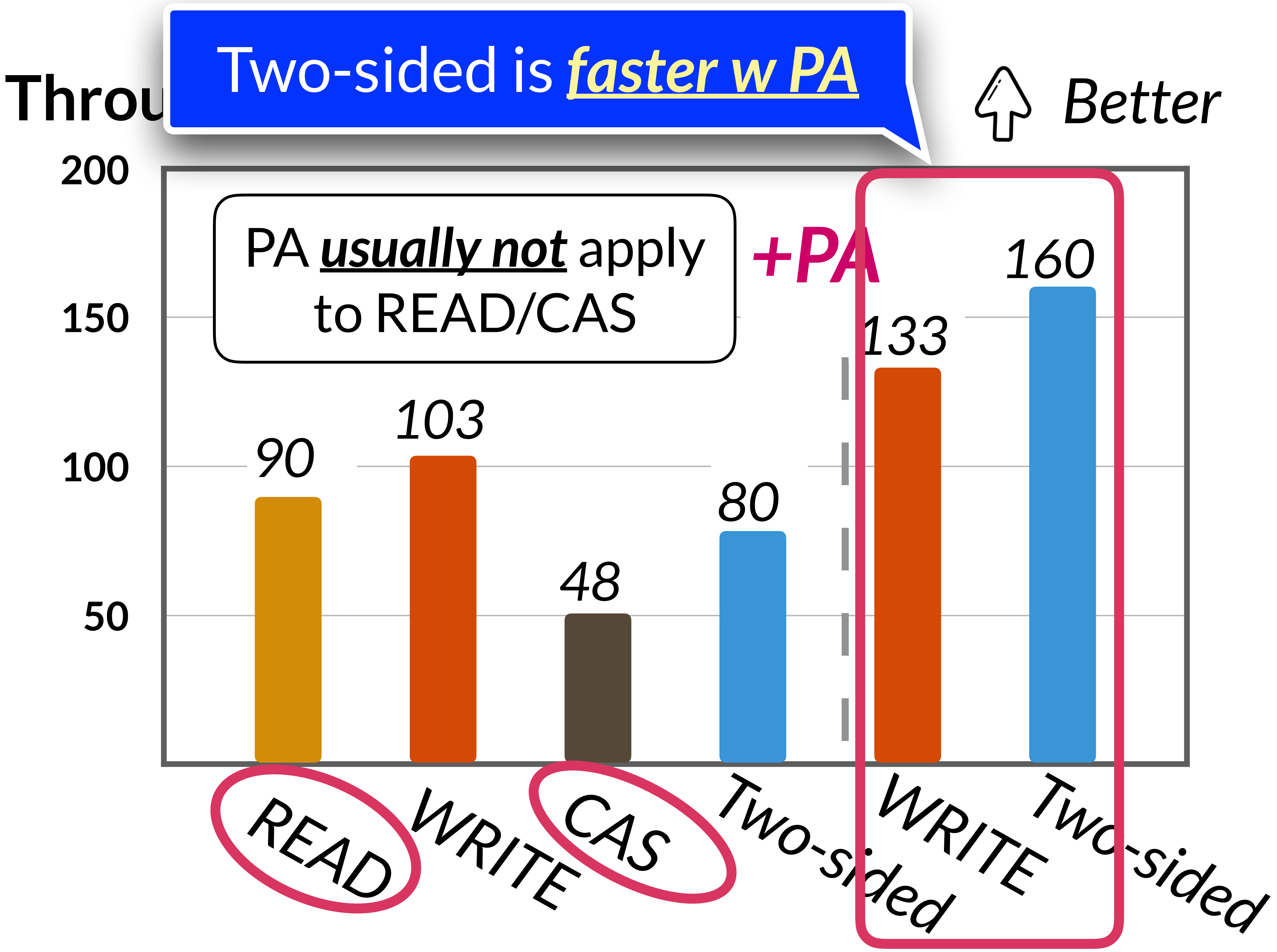
Opt: when the reply is **not on the critical path** of the execution

One-sided primitive

- **Unsignaled** requests

Two-sided primitive

- **Batch** replies (passively)



Towards phase-by-phase analysis

Transactional system

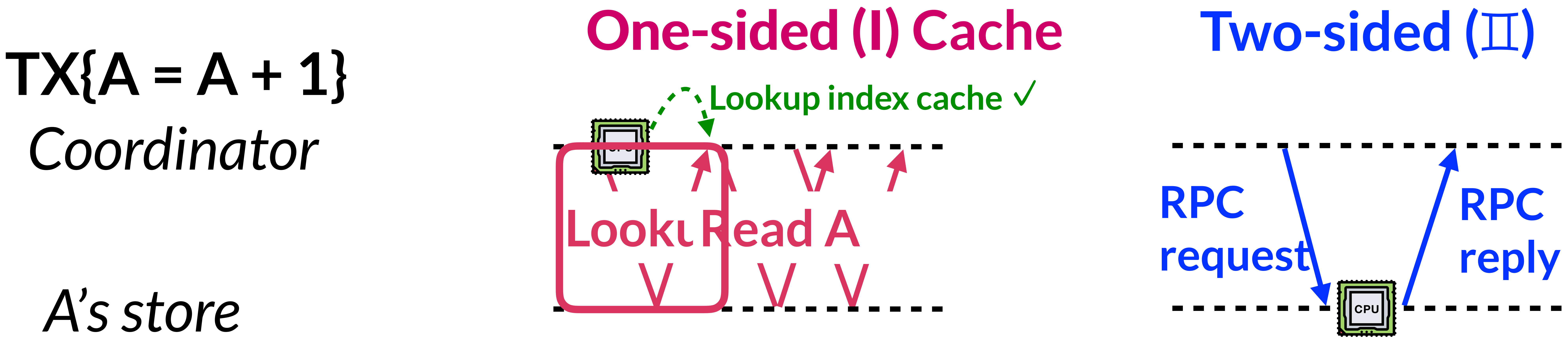
- Built atop of our well-tuned execution framework (primitive analysis)

Workloads

- TPC-C/no: new-order (distributed)
- Smallbank
- TPC-E/cp: custom-position

Execution = READs

Exe | *Val* | *Log* | *Commit*



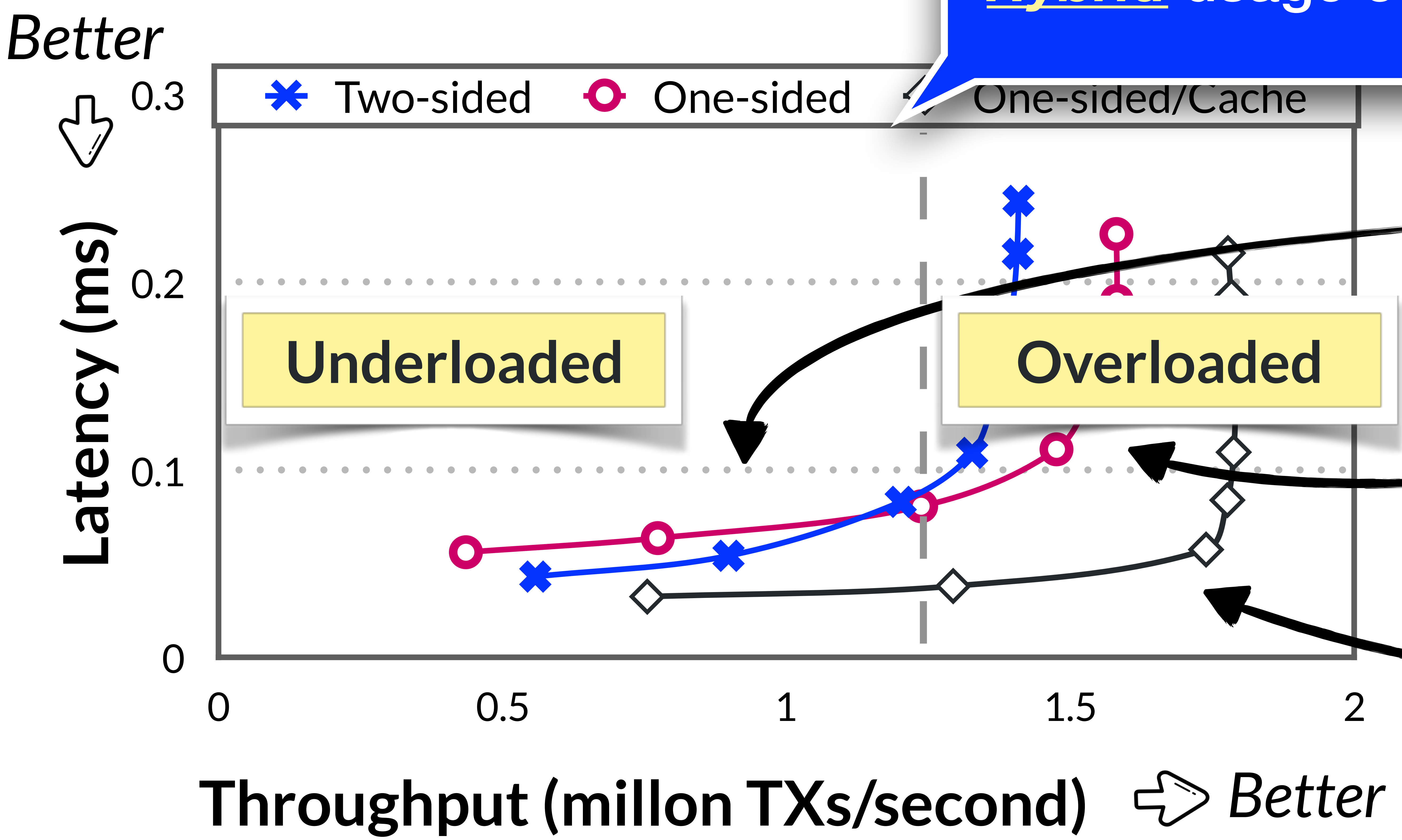
Optimization for one-sided primitive

- **RDMA friendly store** (e.g. DrTM-KV) -> ~**One-round** lookup
- **Index cache**, cache **hot** items address -> **One-round** (lookup + read)

Execution = READs

Exe | Val | Log | Commit

Hybrid usage of **ONE/Cache** + **Two-sided(miss)**



Two-sided **is faster w one round-trip**

One-sided **is faster w high CPU utilization**

READs are **better with one round-trip.**

Validation = LOCKs + READs

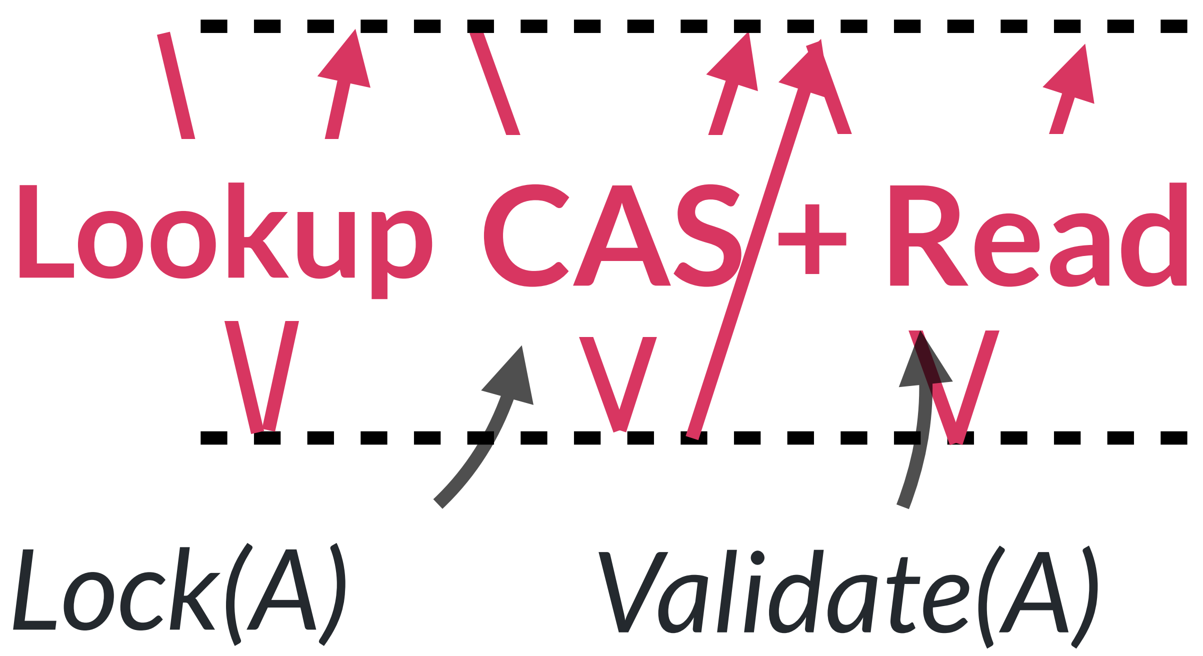
Exe | ***Val*** | *Log* | *Commit*

TX{A = A + 1}

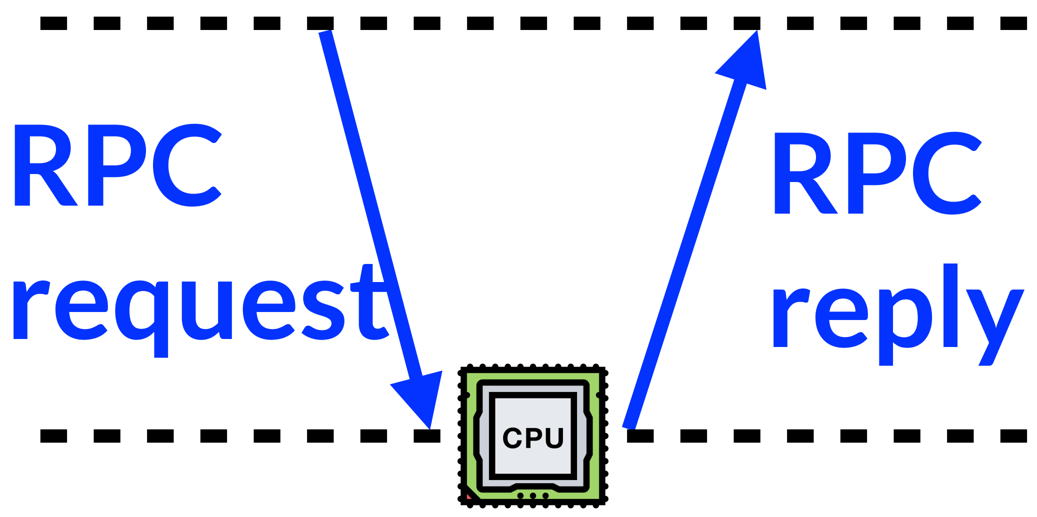
Coordinator

A's store

One-sided (I)



Two-sided (II)

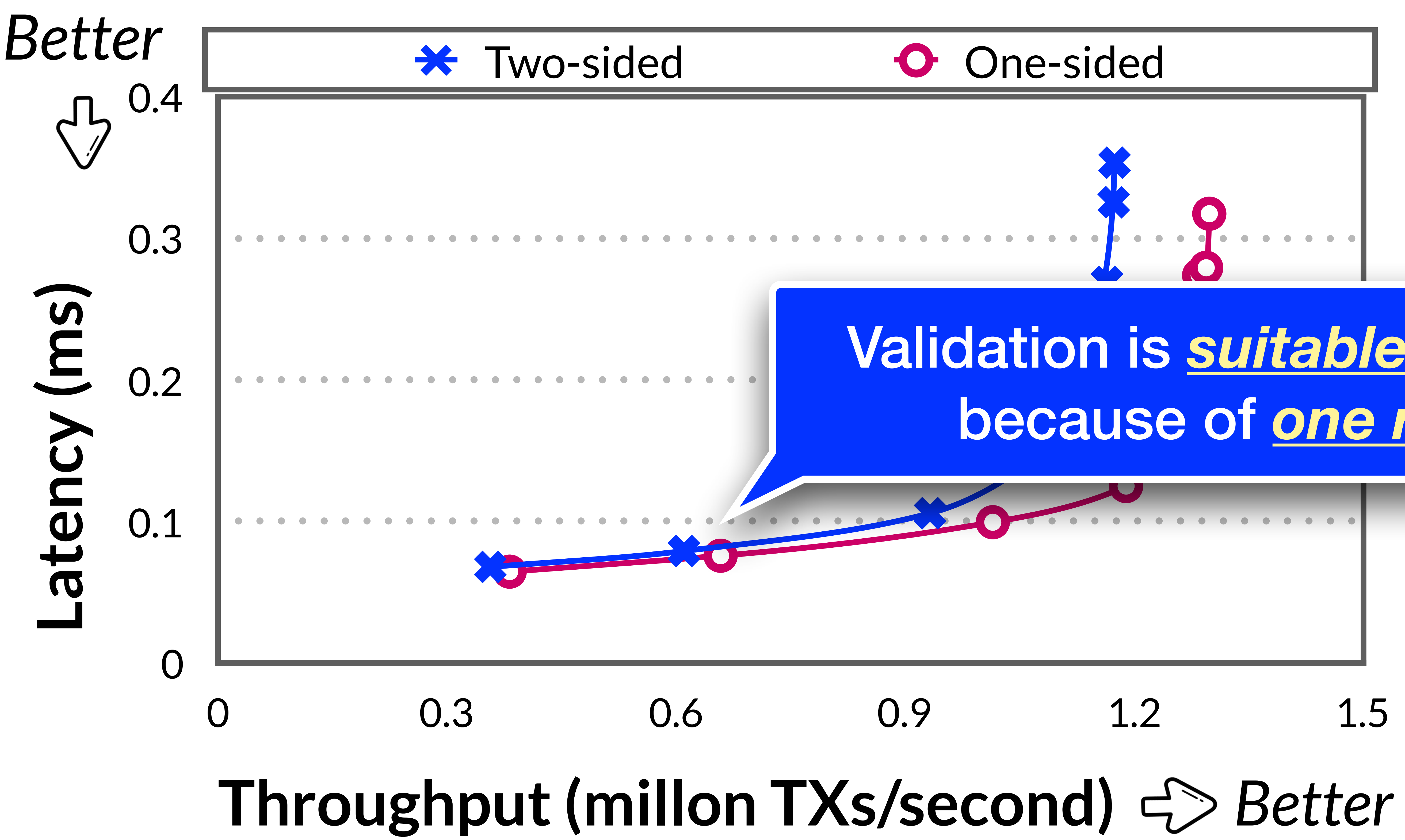


Optimization for one-sided primitive (for **one round-trip**)

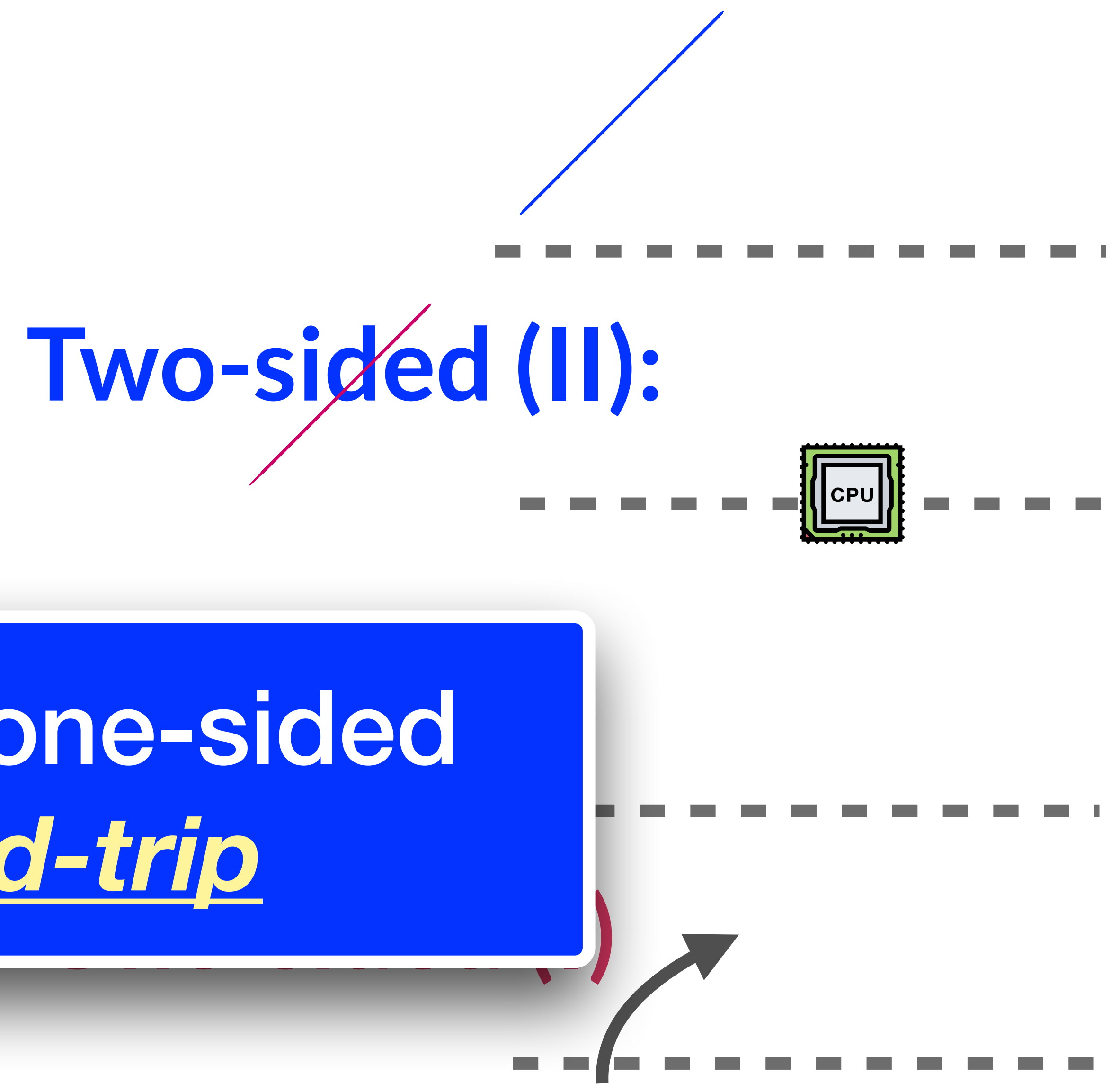
- Address **known w the execution phase** -> no need for lookup
- Locked value cannot be changed -> doorbell **batch READs w CASs**

Validation = LOCKs + READs

Exe | *Val* | *Log* | *Commit*



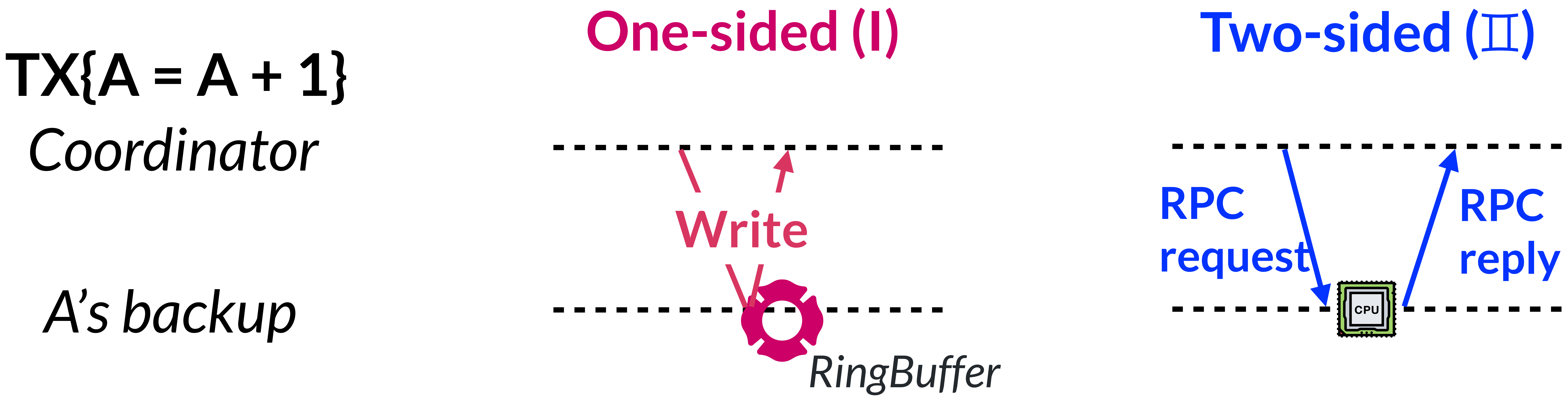
Validation is ***suitable*** for one-sided because of ***one round-trip***



CAS + Read batched together

Logging = WRITEs

Exe | *Val* | ***Log*** | *Commit*

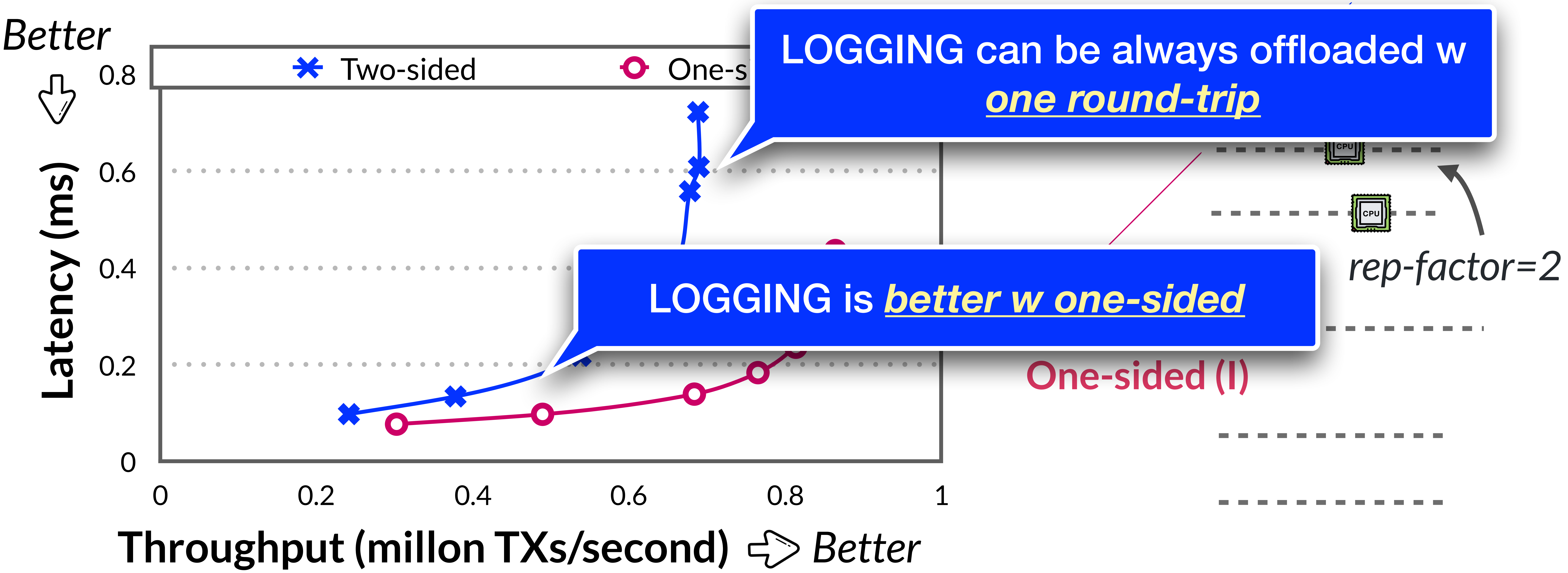


One round-trip for one-sided primitive

- **Ring buffer** based log management [FaRM@NSDI'14]
- **RNIC ack** -> logging succeed (Totally **bypassing** CPU)

Logging = WRITEs

Exe | *Val* | *Log* | *Commit*



Commit = WRITEs + UNLOCKs

Exe | *Val* | *Log* | ***Commit***

TX{A = A + 1}

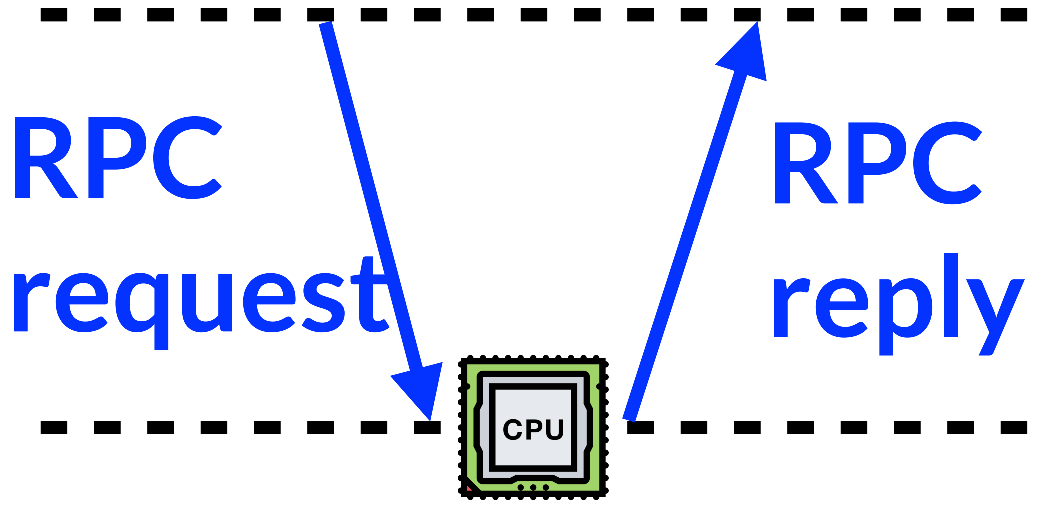
Coordinator

A's store

One-sided (I)

Unlocks implemented as WRITEs

Two-sided (Ⅱ)



One round-trip for one-sided primitive

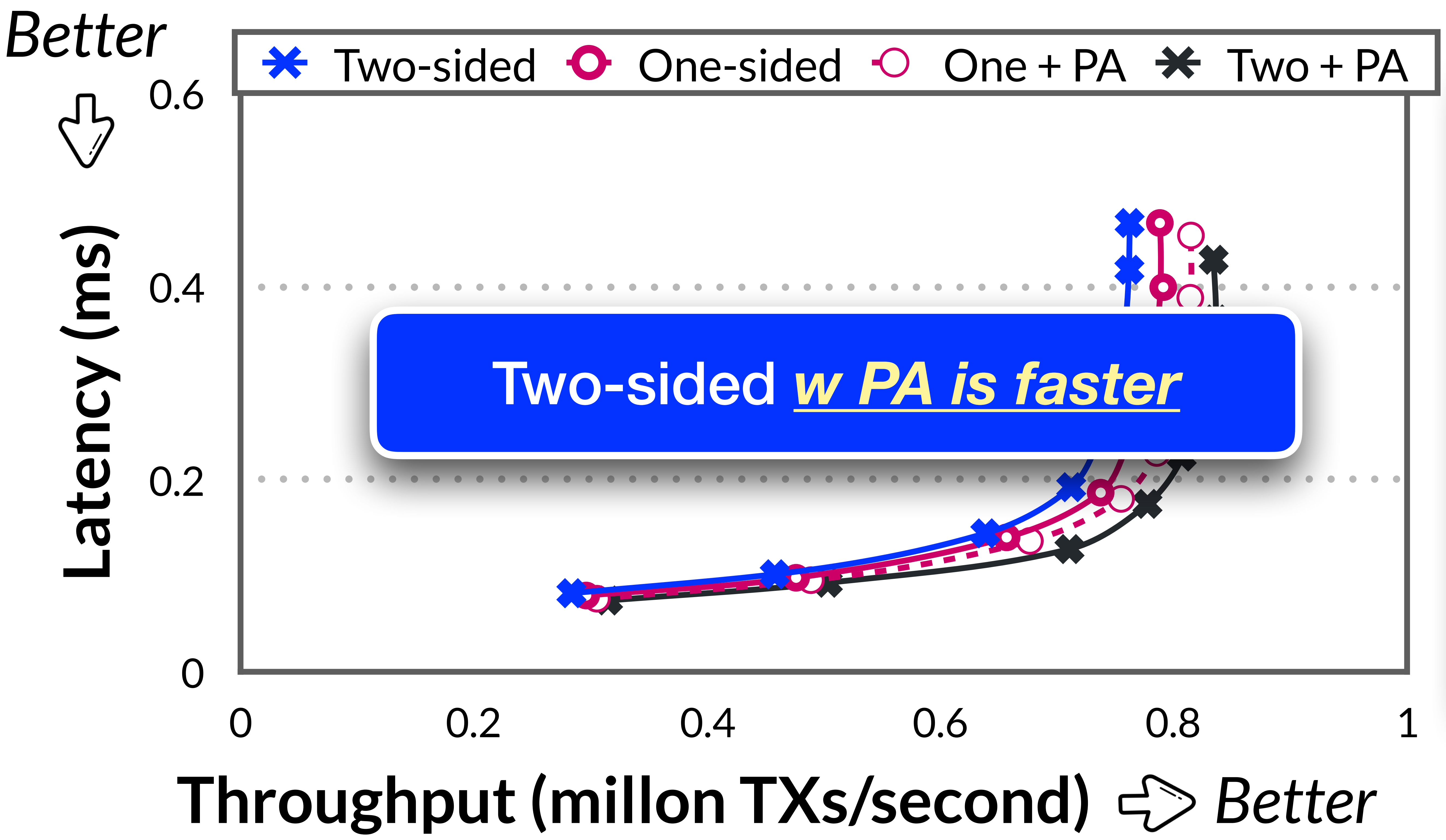
- Address **known w the execution phase** -> no need for lookup

Adding passive ACK to **both primitives**

- **Log succeed** indicates TX's **commit**

Commit = WRITEs + UNLOCKs

Exe | *Val* *Log* *Commit*



- Two-sided w PA has ***higher peak throughput***
- Commit RPC ***costs is small***
- Two-sided saves CPU ***at sender***

DrTM+H: Hybrid is better !

Hybrid system supports **serializability** & **high availability**

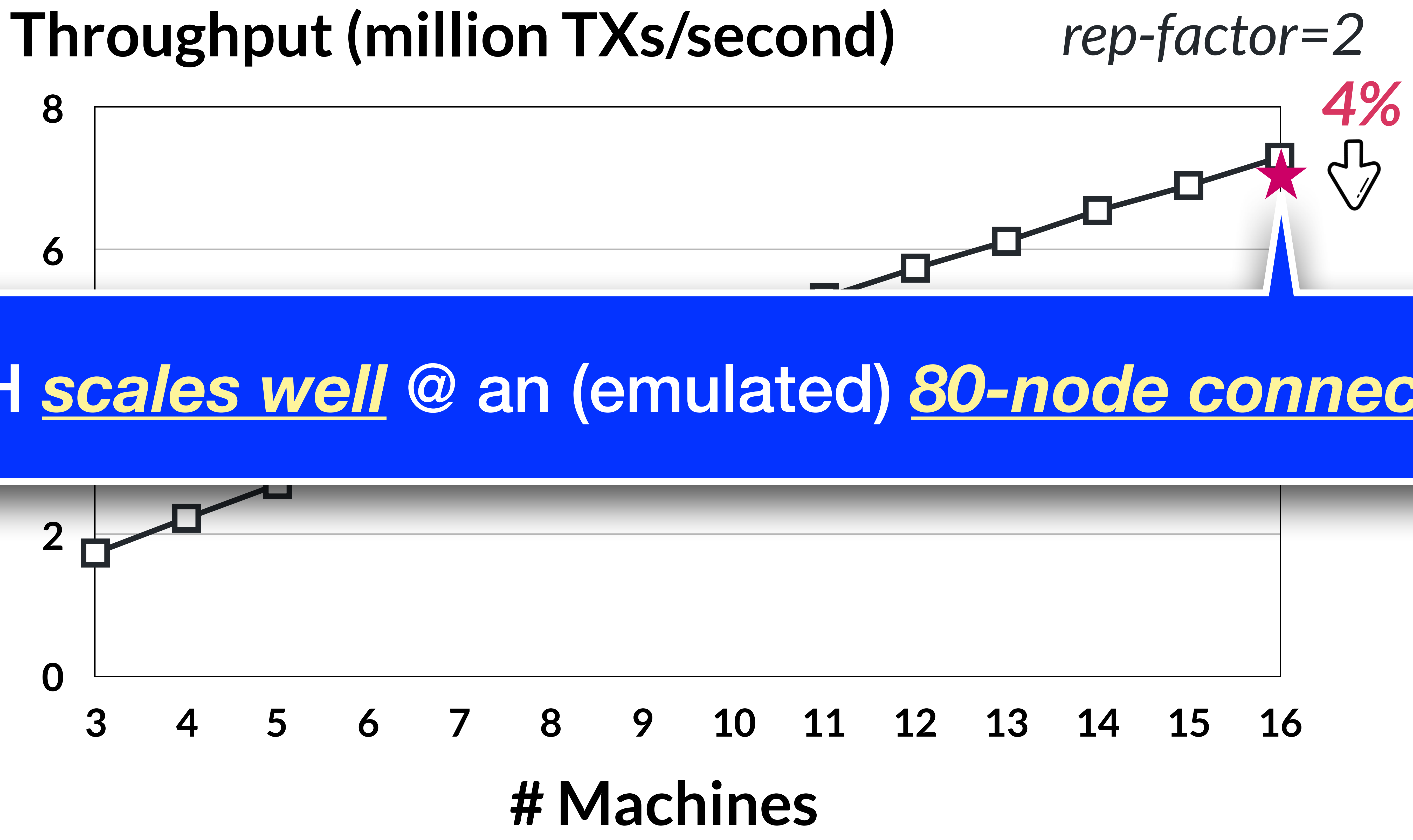
	Execution(E)	Validation(V)	Logging (L)	Commit (C)
DrTM+H	I + II	I	I	II

Hybrid choice

Specific optimizations

- **Passive ACK** to the commit phase (& log cleaning message)
- Speculative execution to send **outstanding requests (OR)** from one TX

Performance & scalability on TPC-C/no

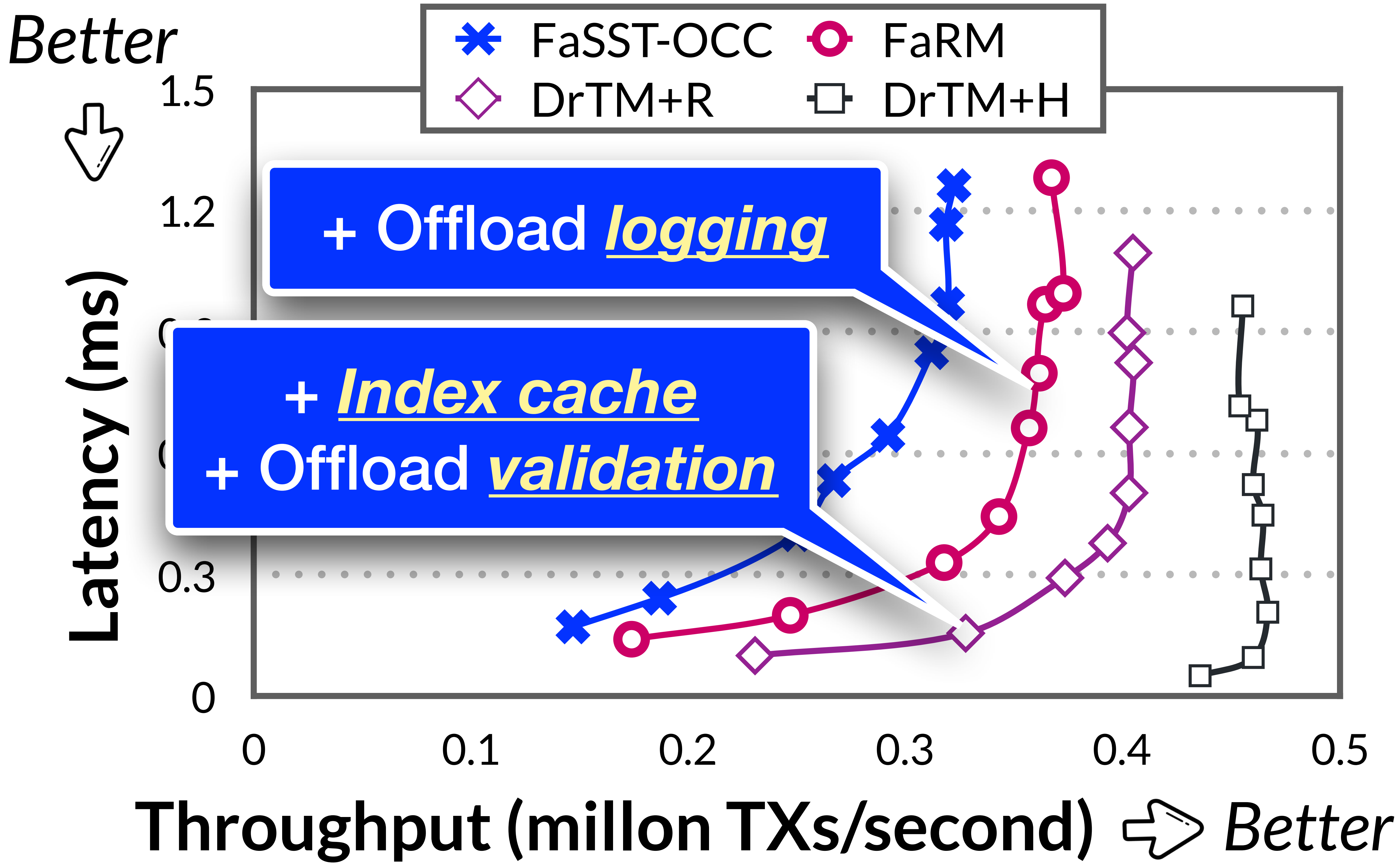


DrTM+H ***scales well*** @ an (emulated) ***80-node connections***

End-to-end comparison against prior designs

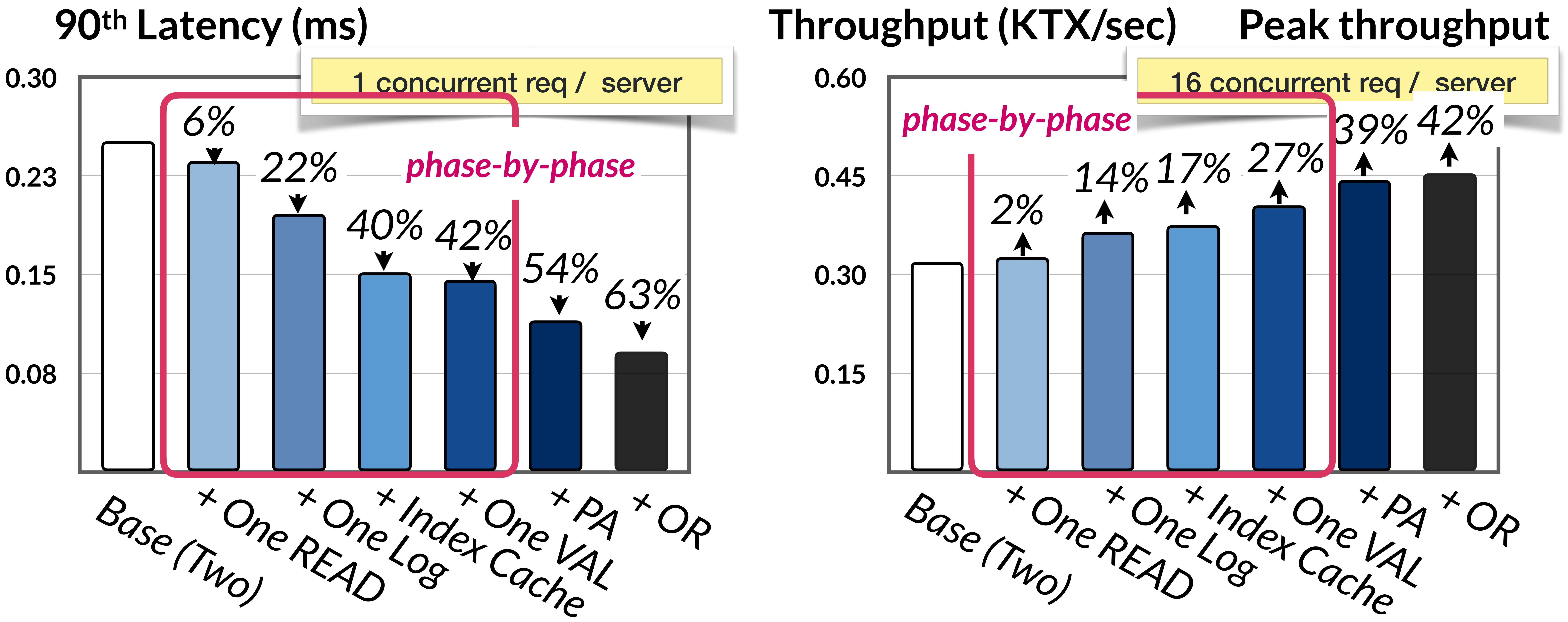
In the **same platform**, the **same protocol**, but w **different choices**

	E	V	L	C
FaSST-OCC[1]	Ⅱ	Ⅱ	Ⅱ	Ⅱ
DrTM+R	| [w cache]	|	|	|
FaRM	| [w/o cache]	| + Ⅱ	|	Ⅱ
DrTM+H	| + Ⅱ	|	|	Ⅱ



[1] FaSST uses a simplified OCC protocol compared to FaRM & DrTM+R.

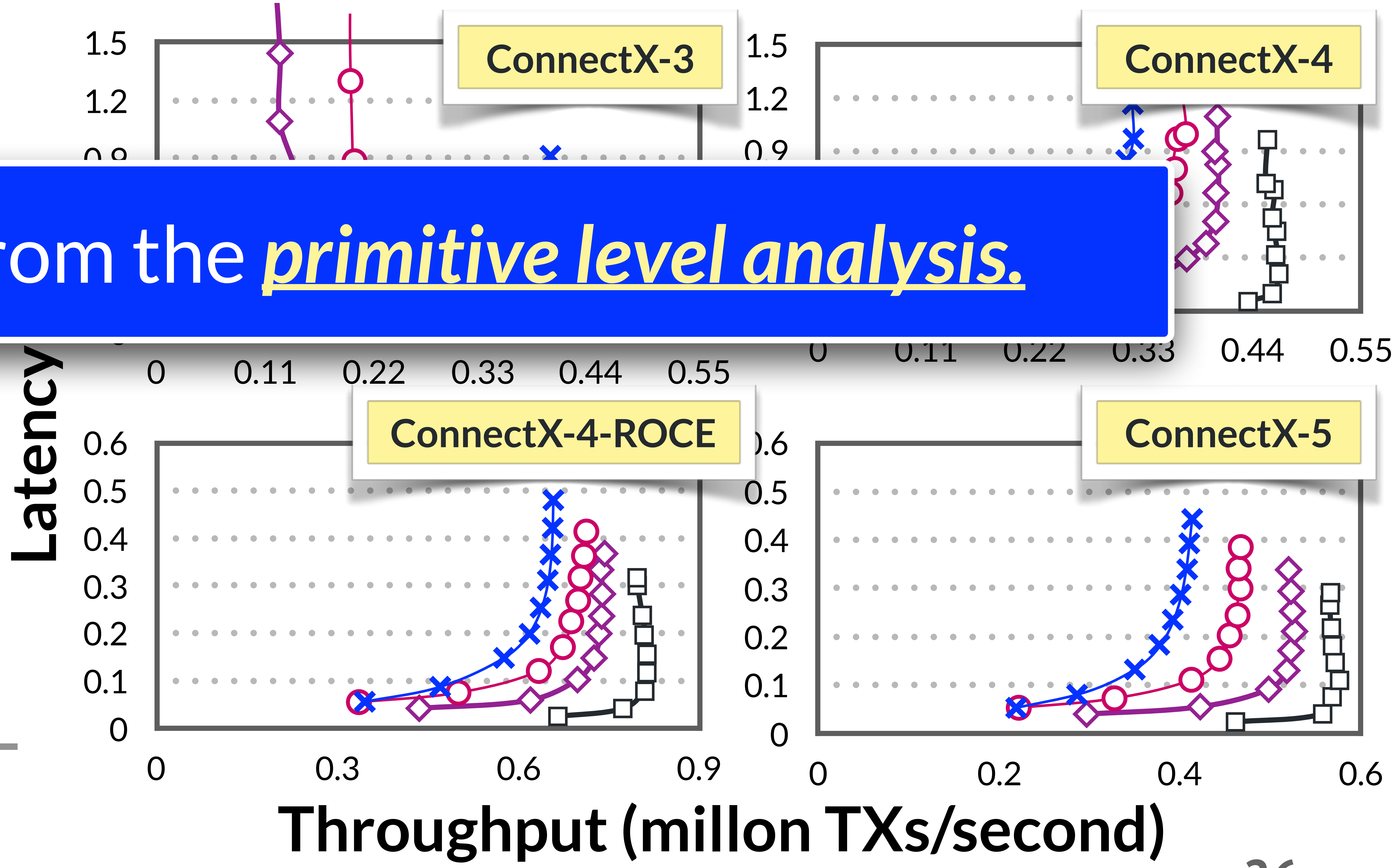
Where do the performance gains come from ?



Not a hard conclusion !

May depends on **RNIC's characteristic** & **network setting**

CX3[1]	Ⅱ	Ⅱ	Ⅱ	Ⅱ
CX4[2]	I + Ⅱ	I	I	Ⅱ
CX4-ROCE[1]	I + Ⅱ	I	I	Ⅱ
CX5[1][3]	I + Ⅱ	I	I	Ⅱ



The results start from the *primitive level analysis.*

[1]1-way replication used due to cluster limitation
[2] Main results in this talk
[3]1-RNIC per machine, others uses 2

Evaluation summary

Offloading w one-sided **improves the performance**

- Especially **w/o adding more round-trips**
- Less affected by **CPU load** at the server

One-sided primitive has **good scalability** on modern RNIC

- Especially when RNIC is **not the bottleneck** of the application
- Although one-sided primitive is restricted by hardware limitation

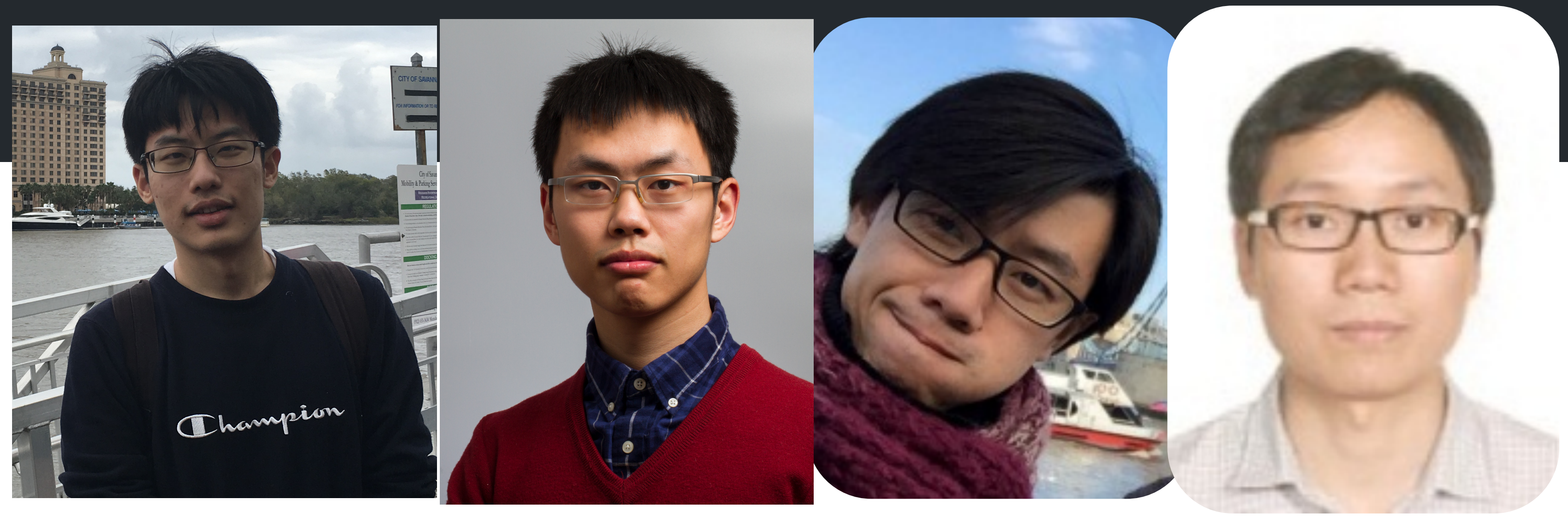
More: check our paper!

- ☑ Optimized execution framework
- ☑ Results of large scale

  Modern RNIC has good scalability for one-sided primitive
- ☑ Read-only Transactions

  A hybrid scheme also wins
- ☑ TPC-E, Smallbank

Conclusions



The **first systematic** study on

- How to use RDMA for OCC TXs

Thanks & QA

No single primitive is **better**!

- Depends on workload pattern & primitive analysis

Execution framework & DrTM+H are available @

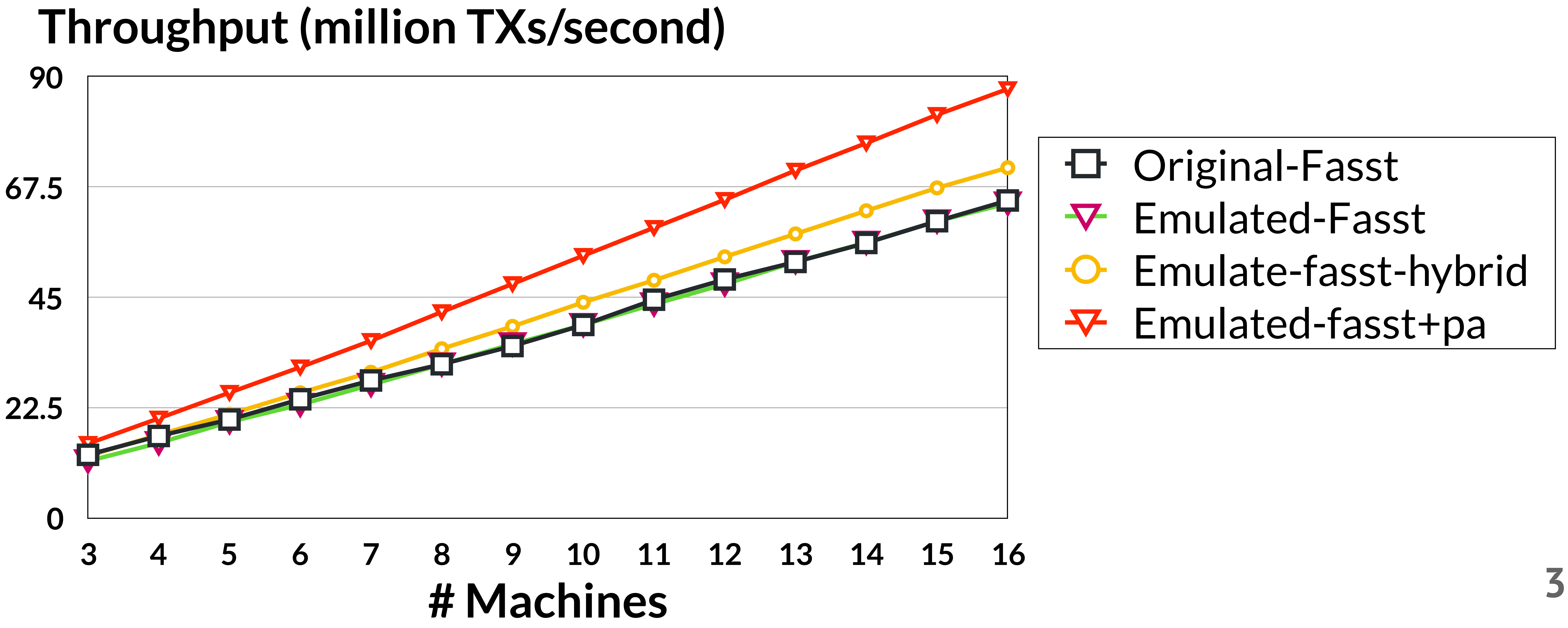
https://github.com/SJTU-IPADS/drtmh

Backups

Improved overall systems

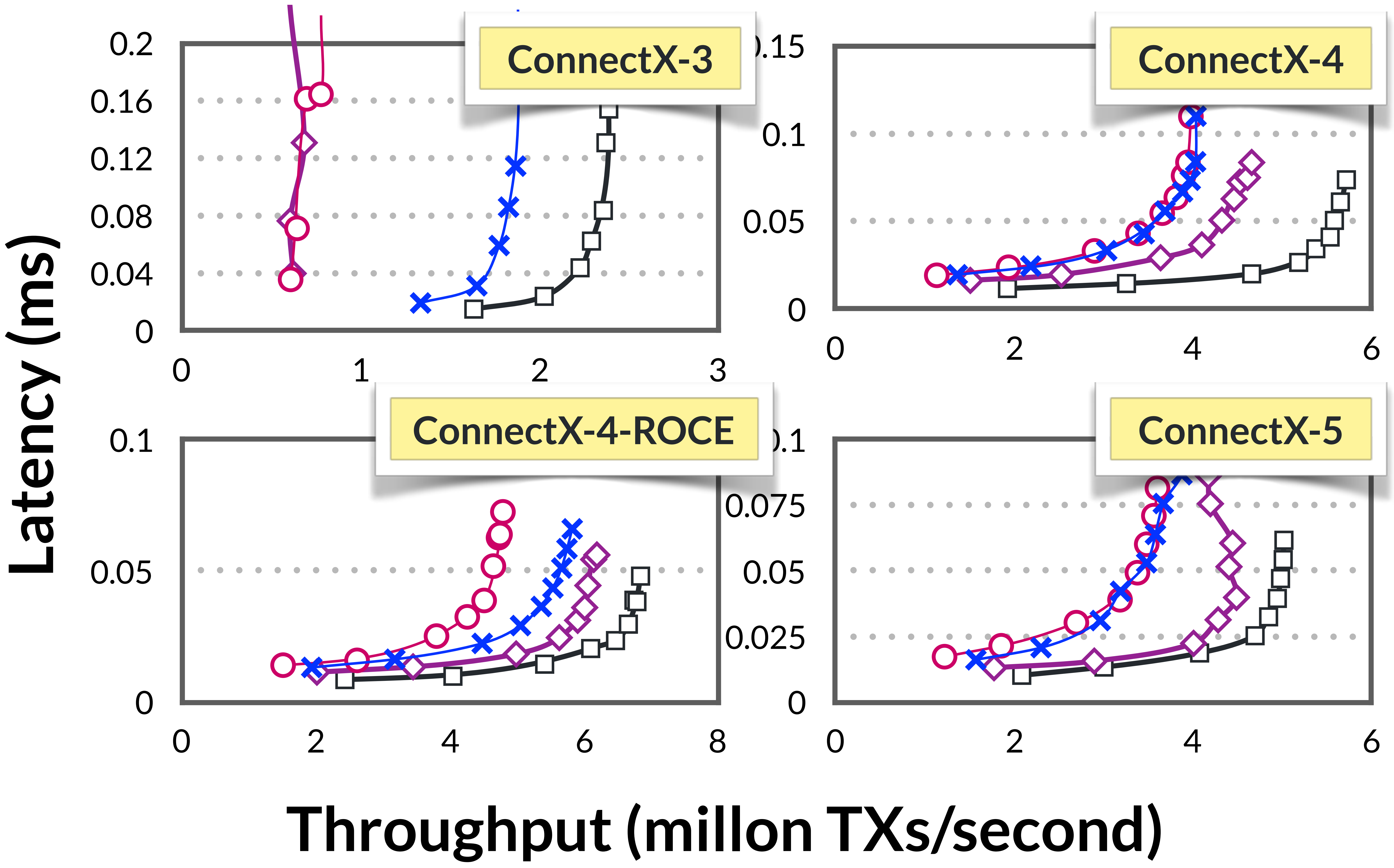
FaSST's simplified OCC protocol

Adding hybrid-schema for logging



Smallbank workloads

	E	V	L	C
CX3[1]	Ⅱ	Ⅱ	Ⅱ	Ⅱ
CX4	Ⅰ + Ⅱ	Ⅰ	Ⅰ	Ⅱ
CX4-ROCE[1]	Ⅰ + Ⅱ	Ⅰ	Ⅰ	Ⅱ
CX5[1][2]	Ⅰ + Ⅱ	Ⅰ	Ⅰ	Ⅱ



[1]1-way replication used due to cluster limitation

[2]1-RNIC per machine, others uses 2

RDMA based execution framework

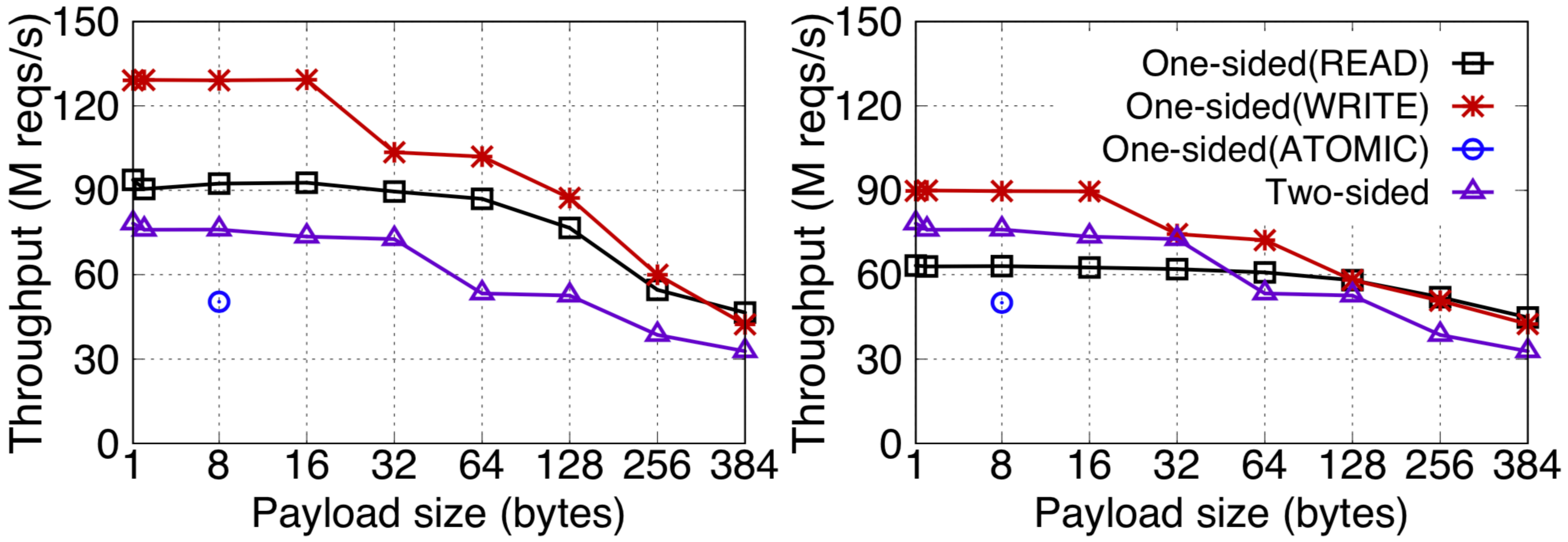
Applied & based RDMA optimizations

- FaRM [NSDI’14,SOSP’15]
- Herd [NSDI’14]
- RDMA guideline [ATC’16]
- FaSST [OSDI’16]

Others

- LITE [SOSP’2017] -> Further improve one-sided's scalability

Results using large connections

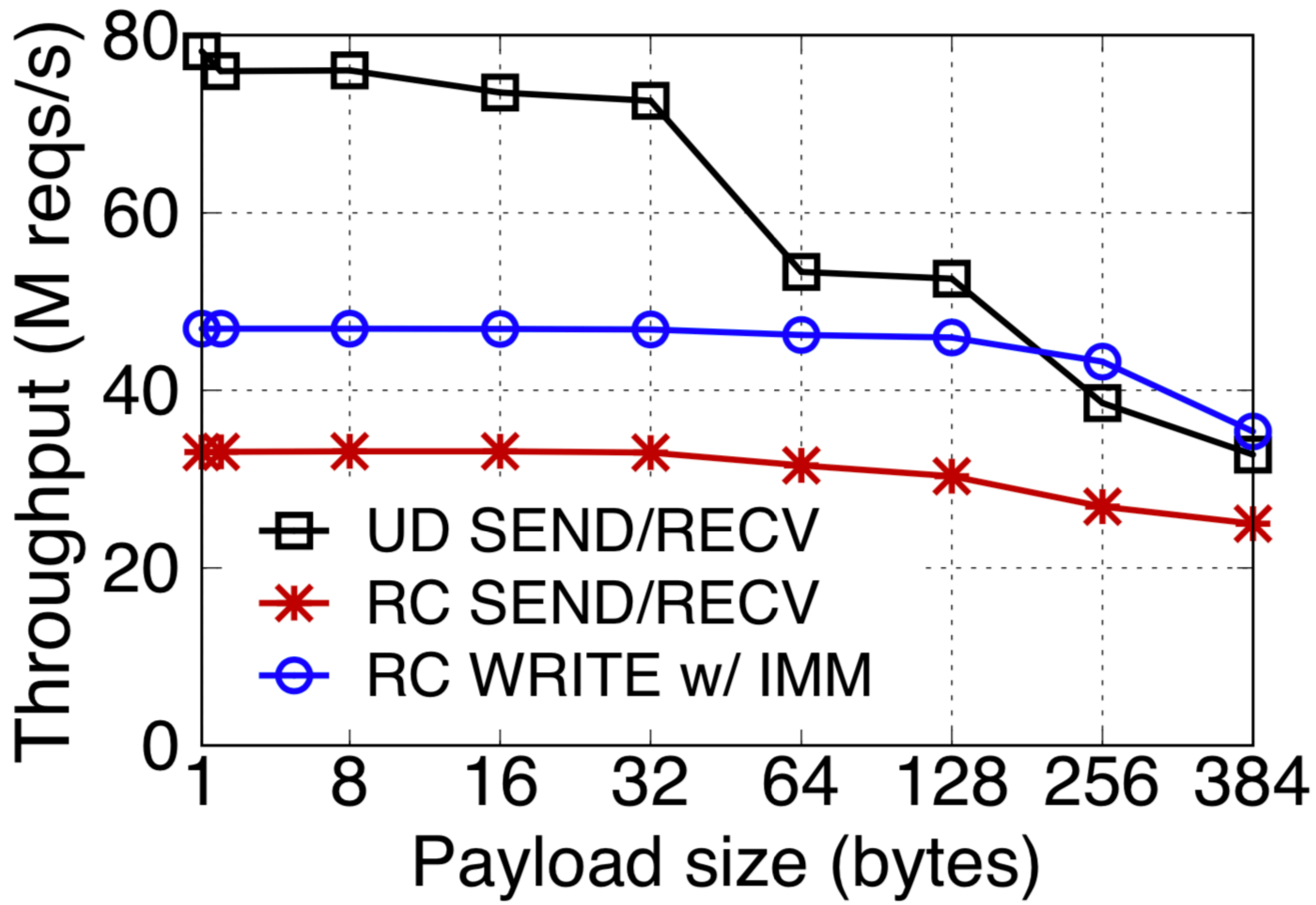


16 node

Emulate 80-node

Comparison of two-sided implementations

FaSST RPC uses UD SEND/RECV



RDMA enabled application

Load balance framework

Distributed TXs

Graph processing systems

Distributed file system