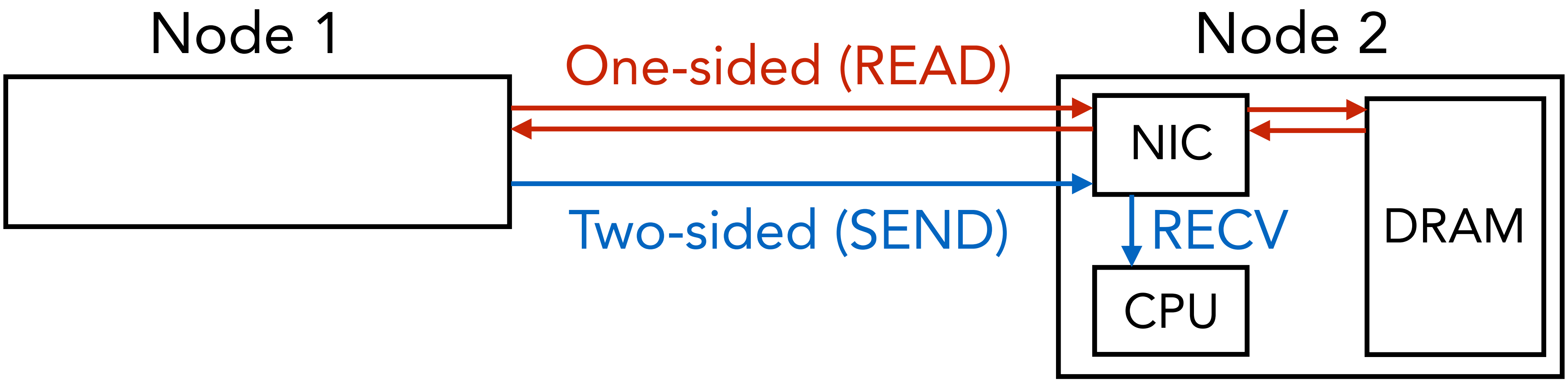
# FaSST: Fast, Scalable, and Simple Distributed Transactions
with
# Two-Sided (RDMA) Datagram RPCs

**Anuj Kalia (CMU)**
Michael Kaminsky (Intel Labs)
David Andersen (CMU)

# One-slide summary




**Existing systems**

Use one-sided RDMA (READs and WRITEs) for transactions

**FaSST**

- Uses RPCs over two-sided ops
- ~2x faster than existing systems
- **F**ast, **s**calable, **s**imple

# In-memory distributed transactions

Distributed ACID transactions can be fast in datacenters

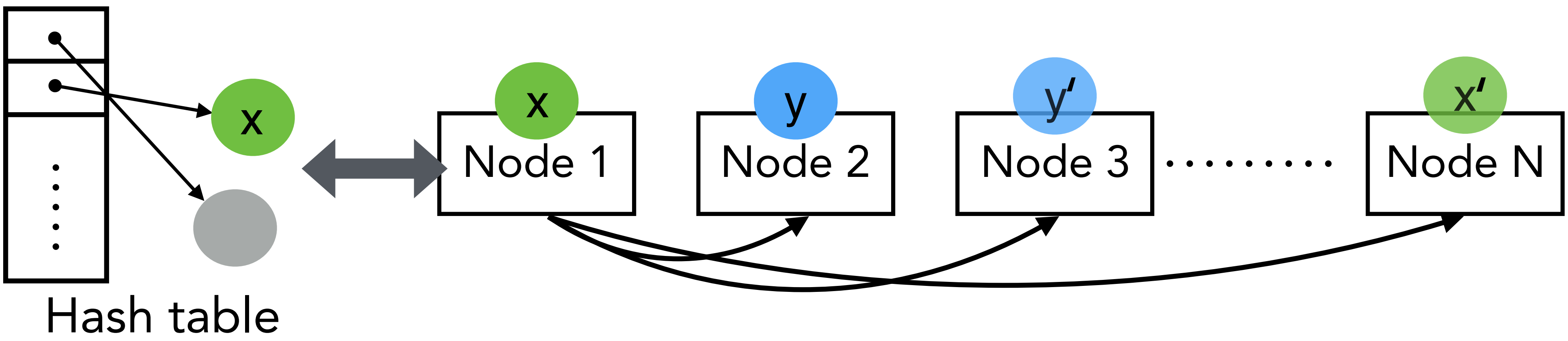
FaRM *[SOSP 15, NSDI 14]*, DrTM *[SOSP 15, EuroSys 15]*, RSI *[VLDB 16]*

**Enablers:**

1. **Cheap DRAM, NVRAM**: No slow components on critical path

2. **Fast networks**: Low communication overhead

# Transaction environment

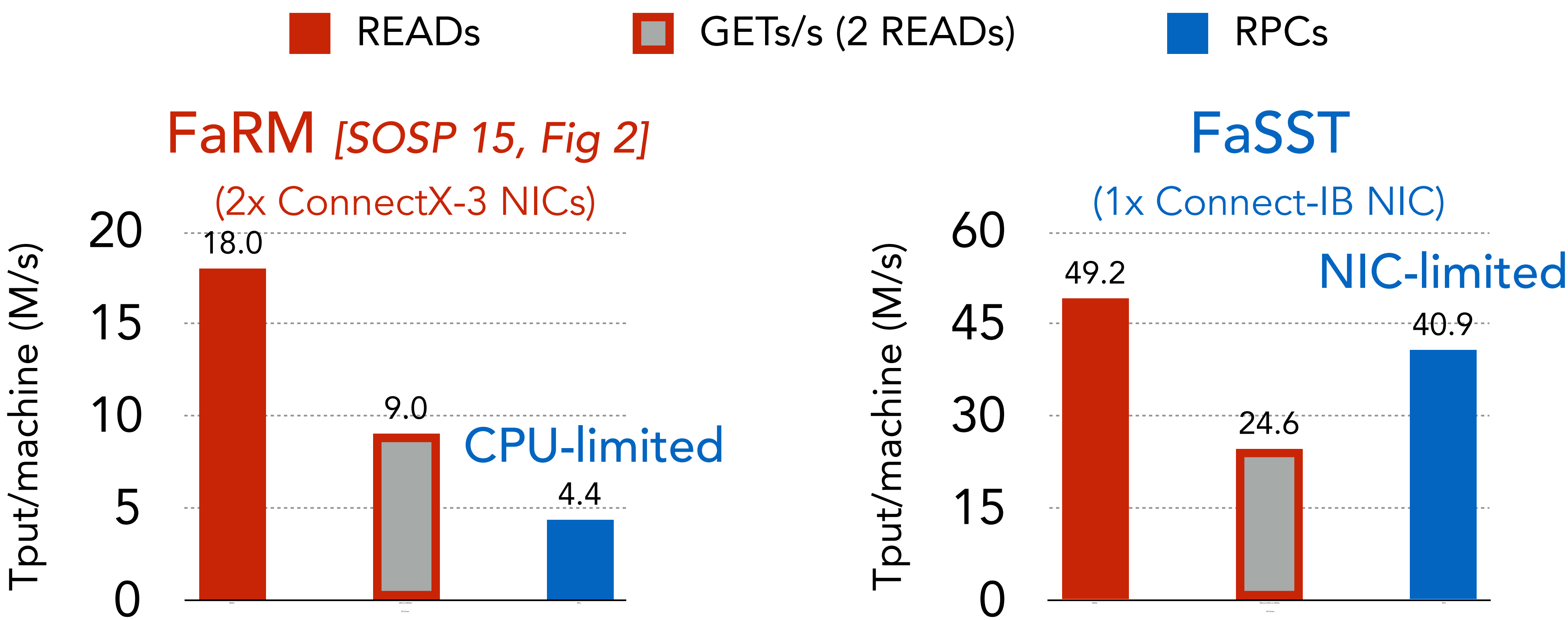



**How to access remote data structures?**

| | Existing systems | FaSST |
|---|---|---|
| Method | One-sided READs | Two-sided RPCs |
| Round trips | ≧2 | 1 |

# RPC v/s READs microbenchmark

**FaSST RPCs make transactions *faster***



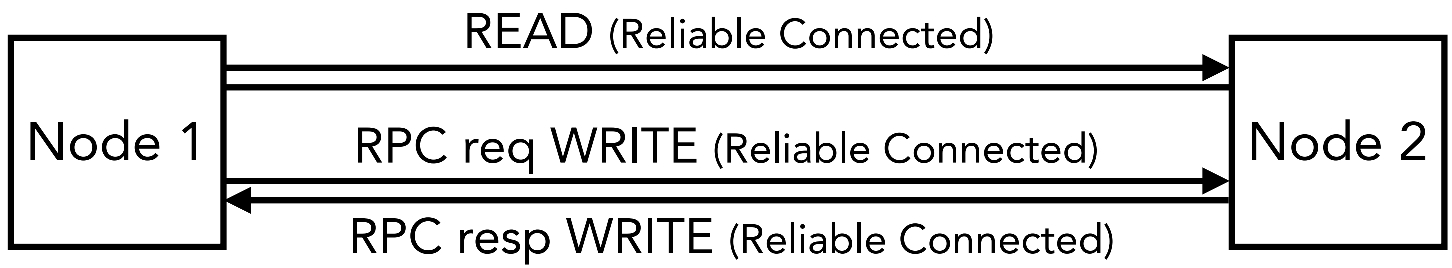
# Reasons for slow RPCS

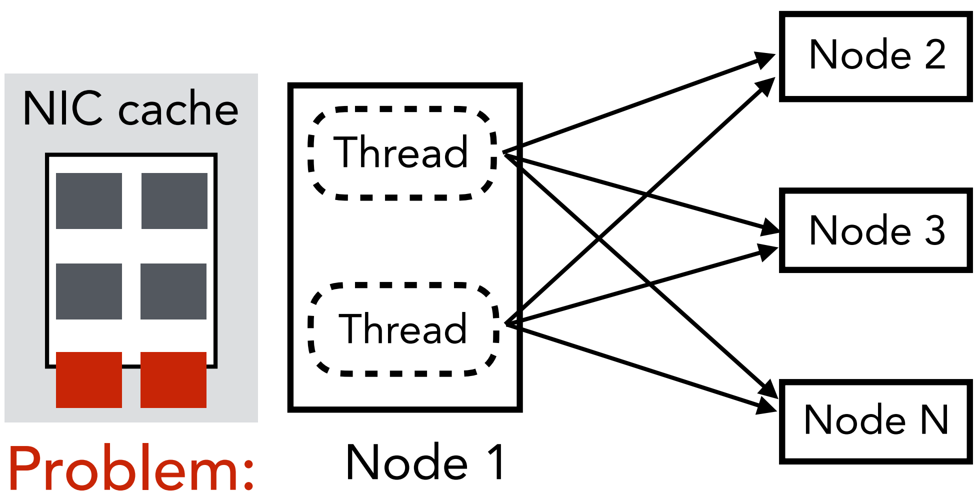
| | Existing systems | FaSST |
|---|---|---|
| Method | One-sided READs | Two-sided RPCs |
| Round trips | ≧2 | 1 |
| Scalable transport | ✗ | ✓ |
| | Effect: NIC cache misses | |
| Lock-free I/O | ✗ | ✓ |
| | Effect: Low per-thread tput | |

# One-sided RDMA does not scale

READs & WRITEs must use a connected transport layer

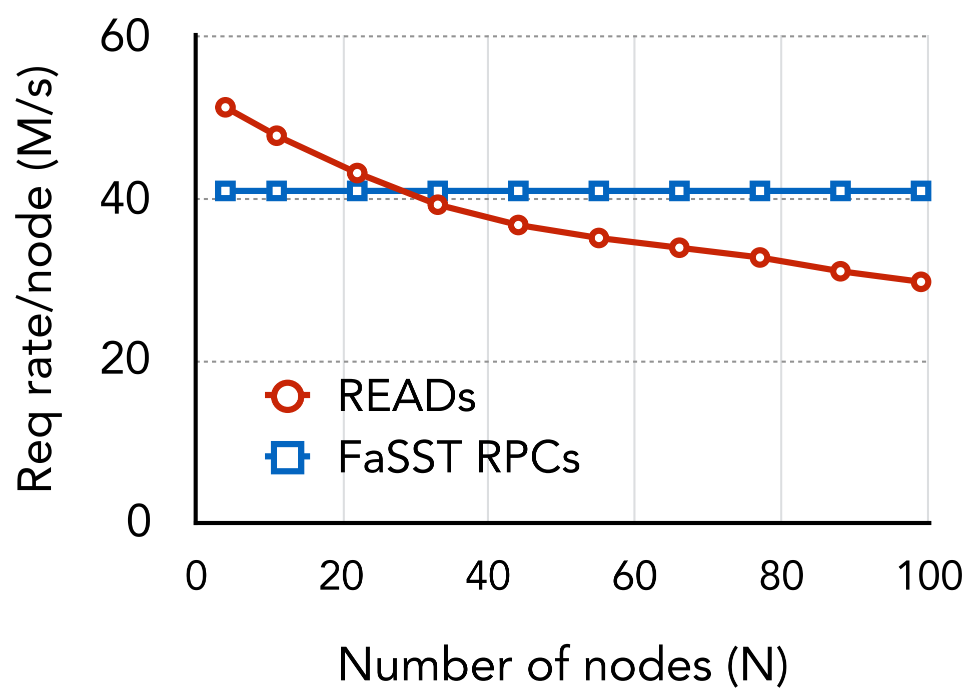
One-sided systems



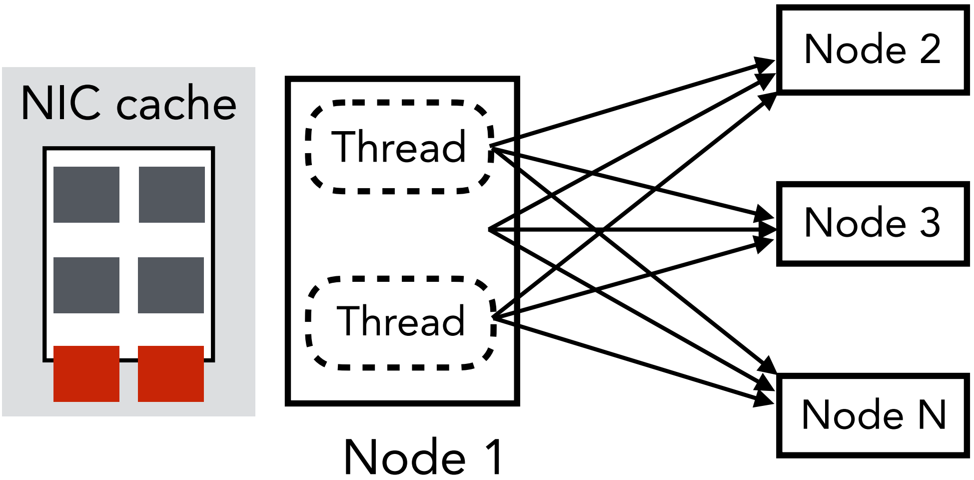




Problem:

Cache overflow

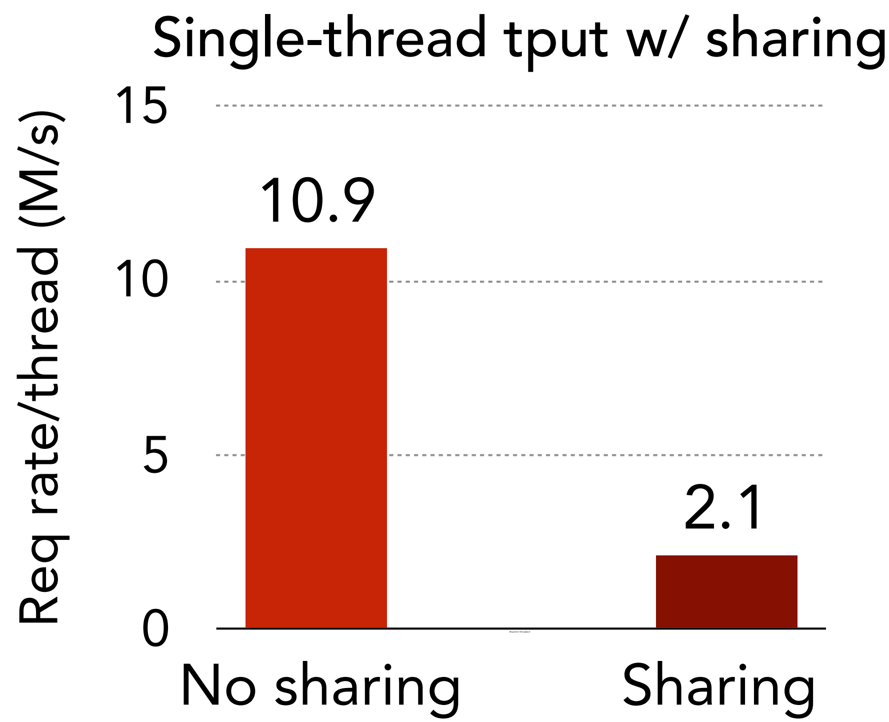


# CPU overhead of connection sharing




~~Problem:~~
~~Cache overflow~~

Problem:
Connection sharing




Local overhead of remote bypass = 5x

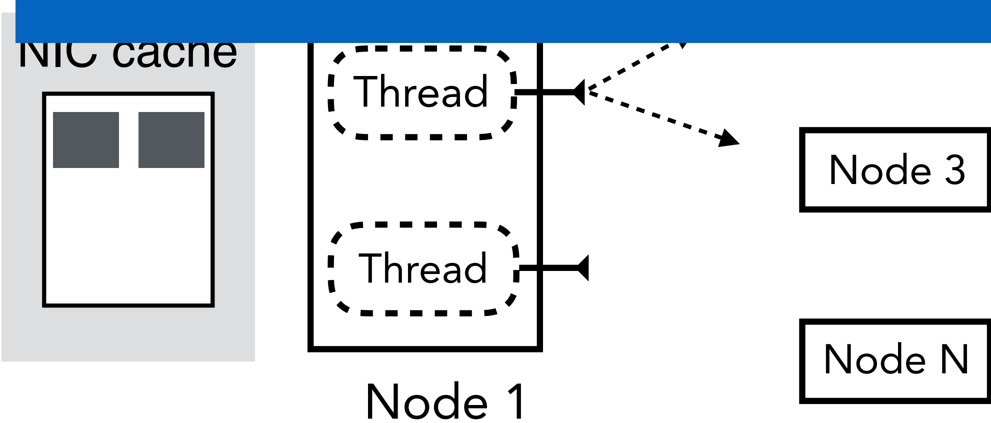
# Connectionless transport scales

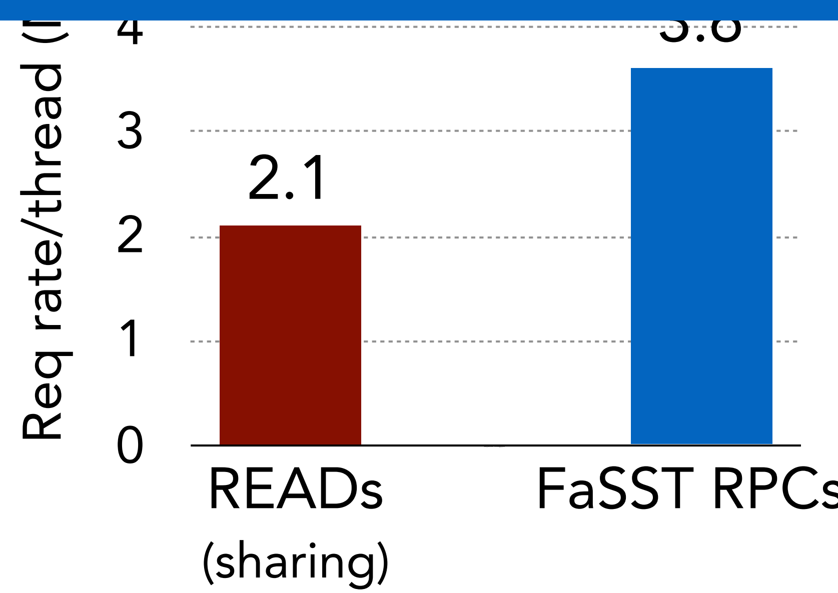
But it supports only two-sided (SEND/RECV) operations

**READs don’t use fewer CPU cycles than RPCs!**

Local overhead offsets remote gains

**FaSST RPCs make transactions *scalable***



# FaSST RPCs make transactions ***<u>Simpler</u>***

## Remote bypassing designs are complex

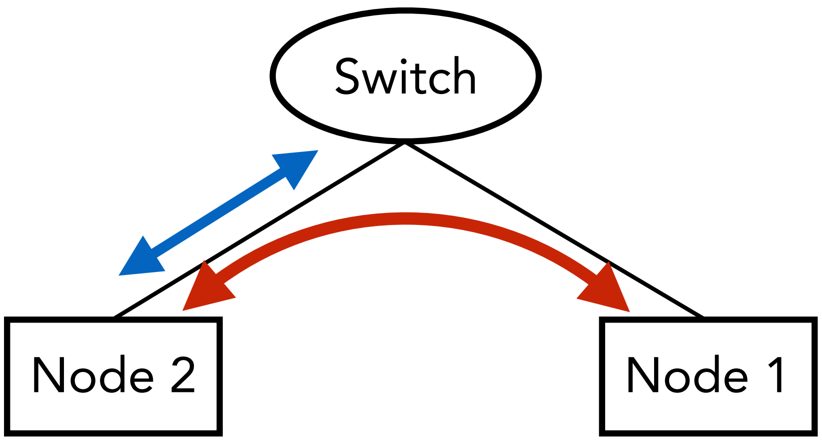
- Redesign and rewrite data stores
- Hash table [FaRM-KV, *NSDI 14*], B-Tree [Cell, *ATC 15*]

## RPC-based designs are simple

- Reuse existing data stores
- Hash table [MICA, *NSDI 14*], B-Tree [Masstree, *EuroSys 12*]

# UD does not provide reliability.

## But the link layer does!




- No end-to-end reliability

+ Link layer flow control

+ Link layer retransmission

No packet loss in

- 69 nodes, 46 hours
- 100 trillion packets
- 50 PB transferred

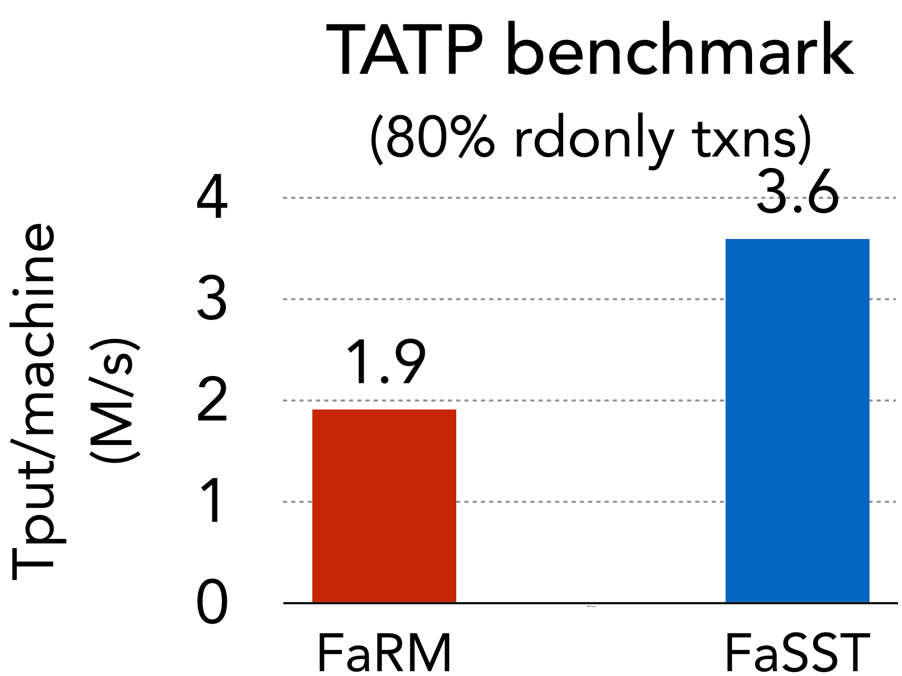
Handle packet loss similar to machine failure: See paper

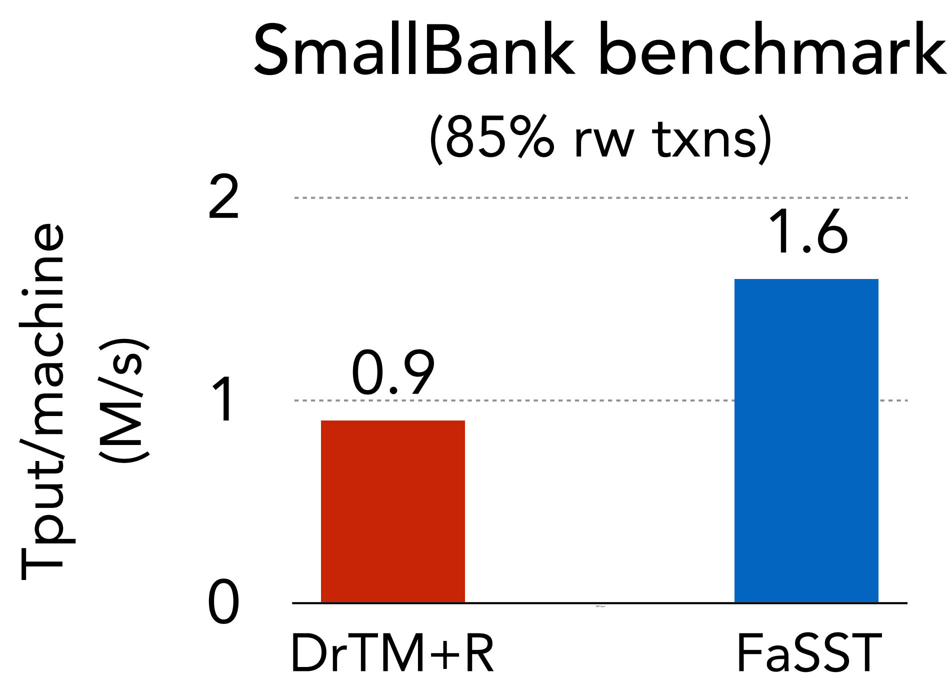
# Performance comparison

| | Nodes | NICs | Cores |
|---|---|---|---|
| FaRM | 50 | 2x ConnectX-3 | 16 |
| DrTM+R | 6 | 1x ConnectX-3 | 10 |
| **FaSST** | 50 | 1x ConnectX-3 | 8 |

**vs FaRM:** FaSST uses 50% fewer h/w resources

**vs DrTM+R:** FaSST makes no data locality assumptions



# Conclusion

Transactions with one-sided RDMA are:

1. **Slow:** Data access requires multiple round trips
2. **Non-scalable:** Connected transports
3. **Complex:** Redesign data stores

Transactions with two-sided datagram RPCs are:

1. **Fast:** One round trip
2. **Scalable:** Datagram transport + link layer reliability
3. **Simple:** Re-use existing data stores

Code: https://github.com/efficient/fasst