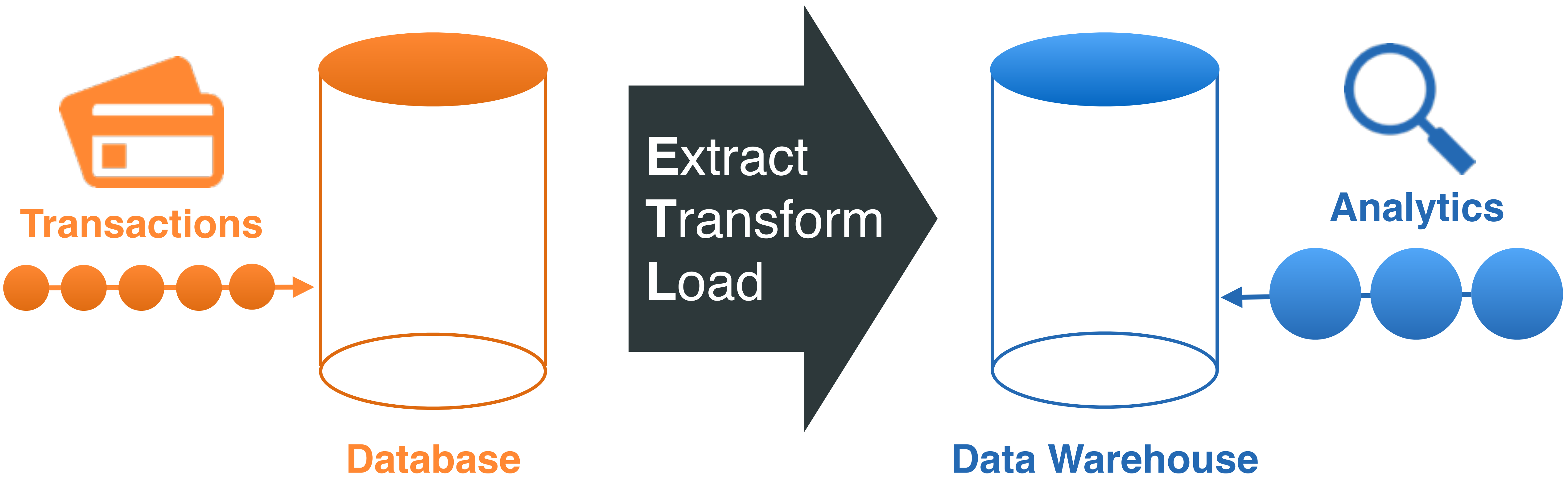
High-Speed Query Processing over High-Speed Networks

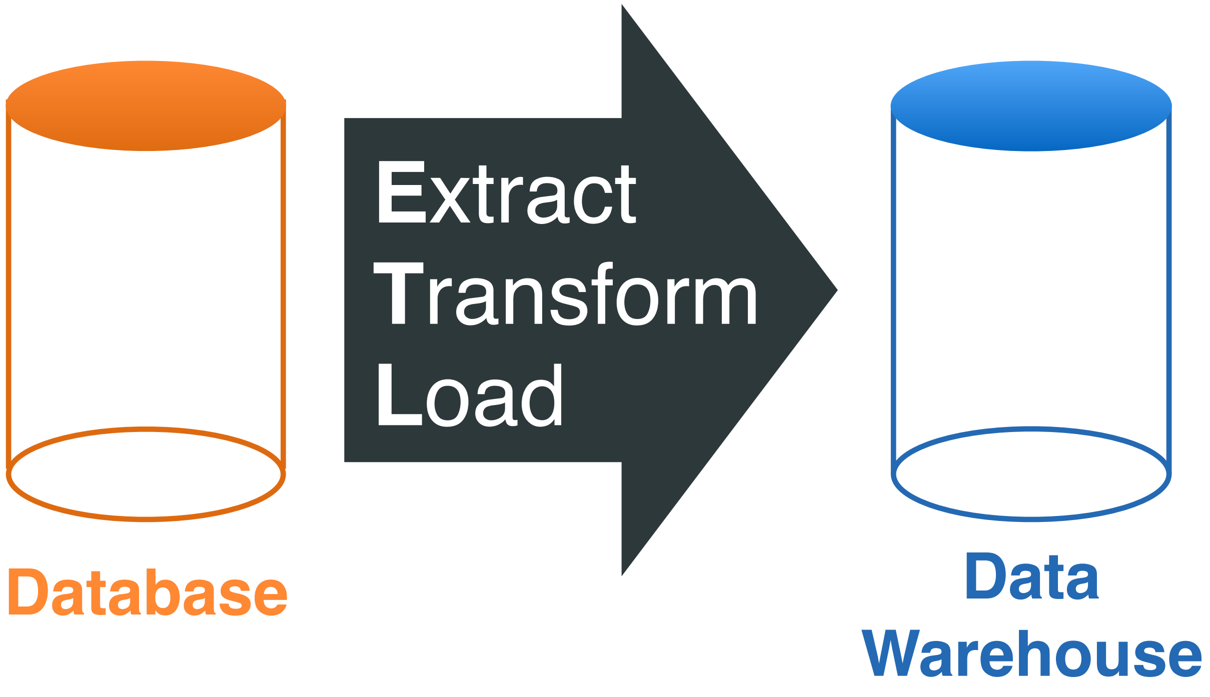
Wolf Rödiger, Tobias Mühlbauer, Alfons Kemper, Thomas Neumann

Traditional Data Warehouse



Traditional Data Warehouse

- Isolate business-critical **transactions** from analytical **queries**
- **ETL process** to update the data warehouse
- Periodic refresh leads to **data staleness**



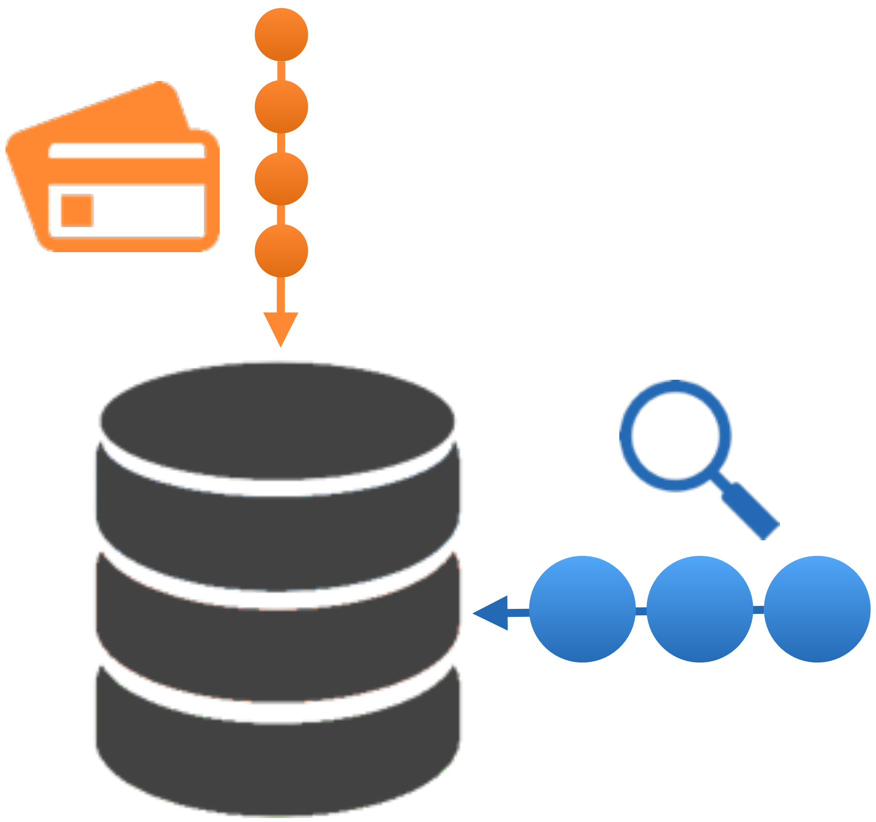
HyPer

- **Analytics**
  Excellent response times

- **Transactions**
  100k TPC-C transaction/s

- **Both workloads** on the same state in one system

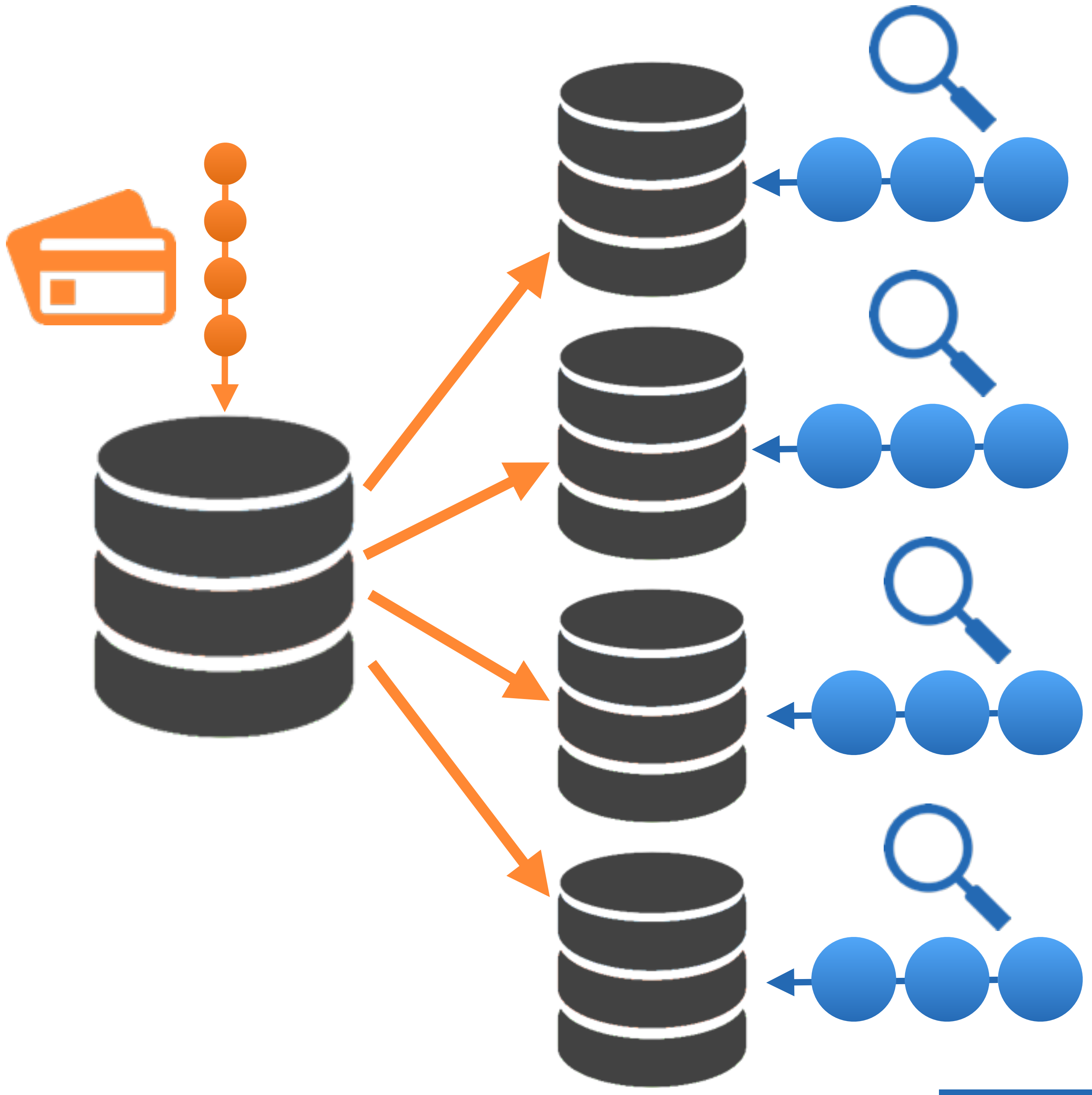
- Code generation, MVCC



Scale the **HyPer** main-memory database system to a **cluster** of machines

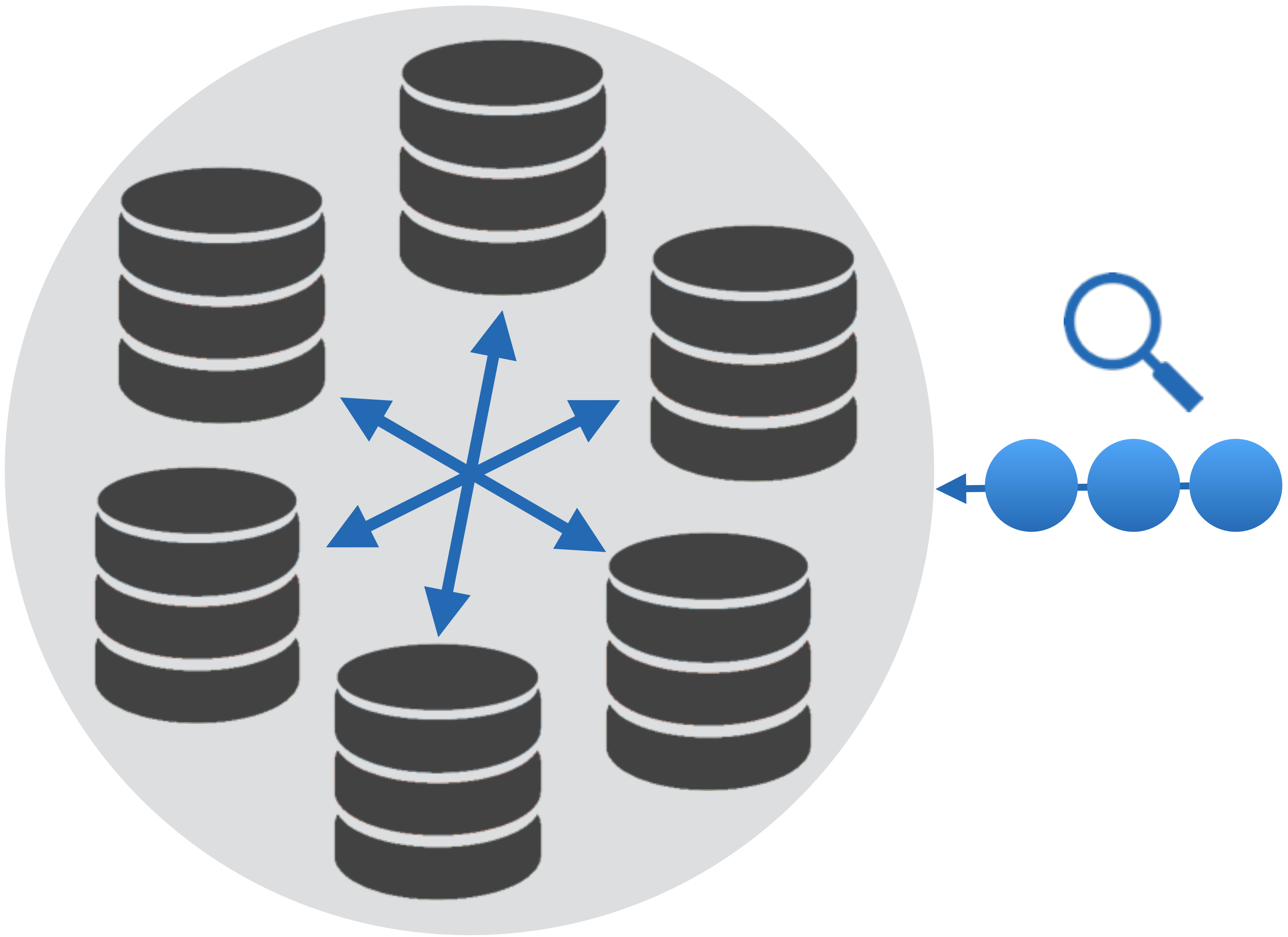
Full Replication

- **Replicate** the data from a primary server
- **Improved** query throughput
- **Same** main-memory, **same** response times as a single server

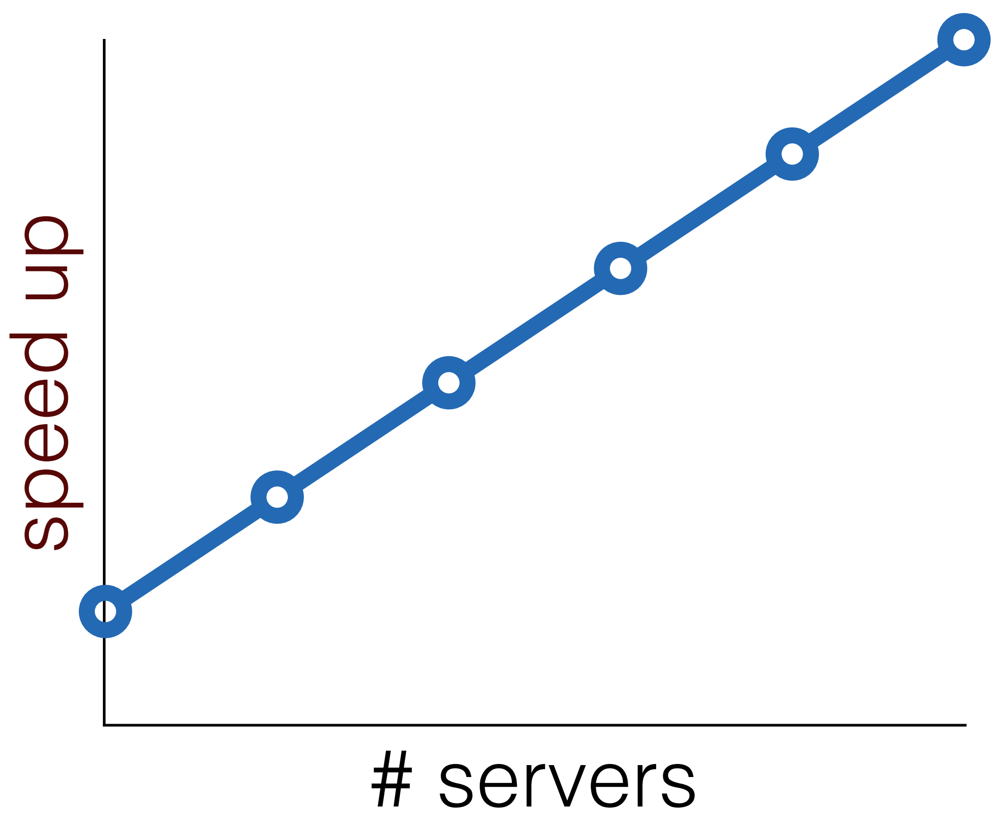


Horizontal Partitioning

- **Partition** the data across servers
- **Increases** main-memory capacity
- **But can we also speed up query processing?**

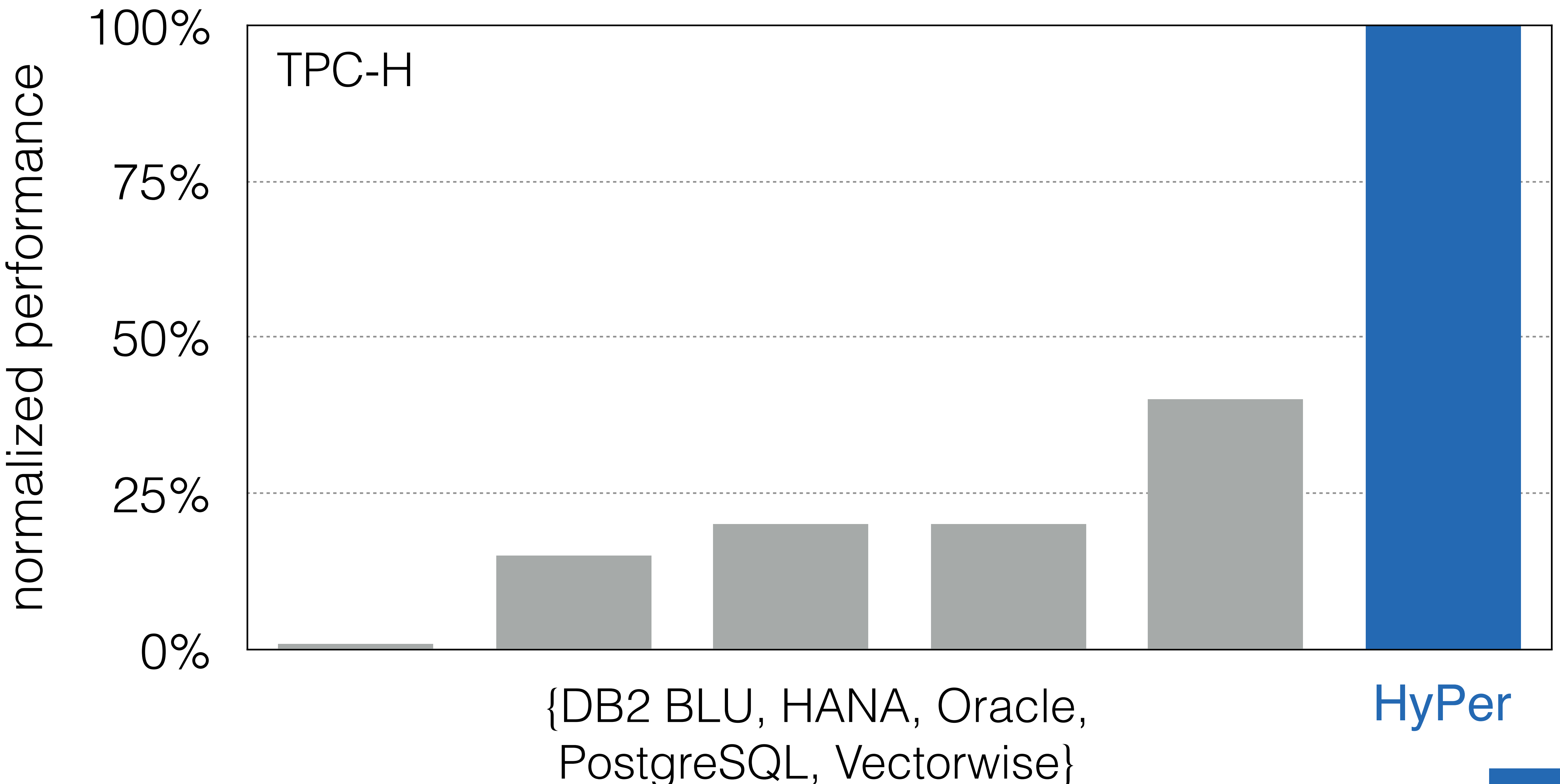


Speed-Up? Easy!



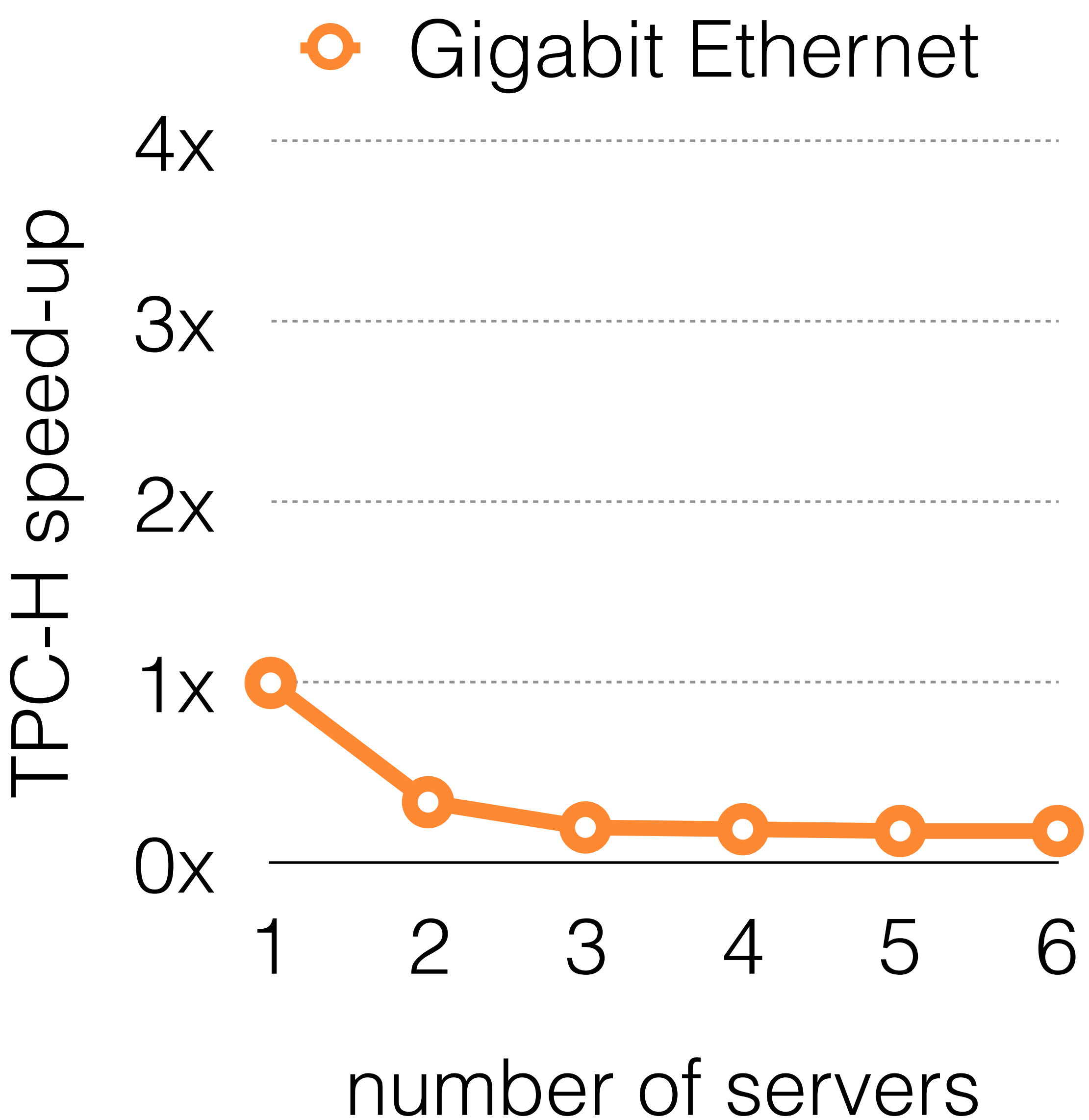
… as long as the system is **slow** on **one server.**

Unfortunately, *HyPer is Fast*



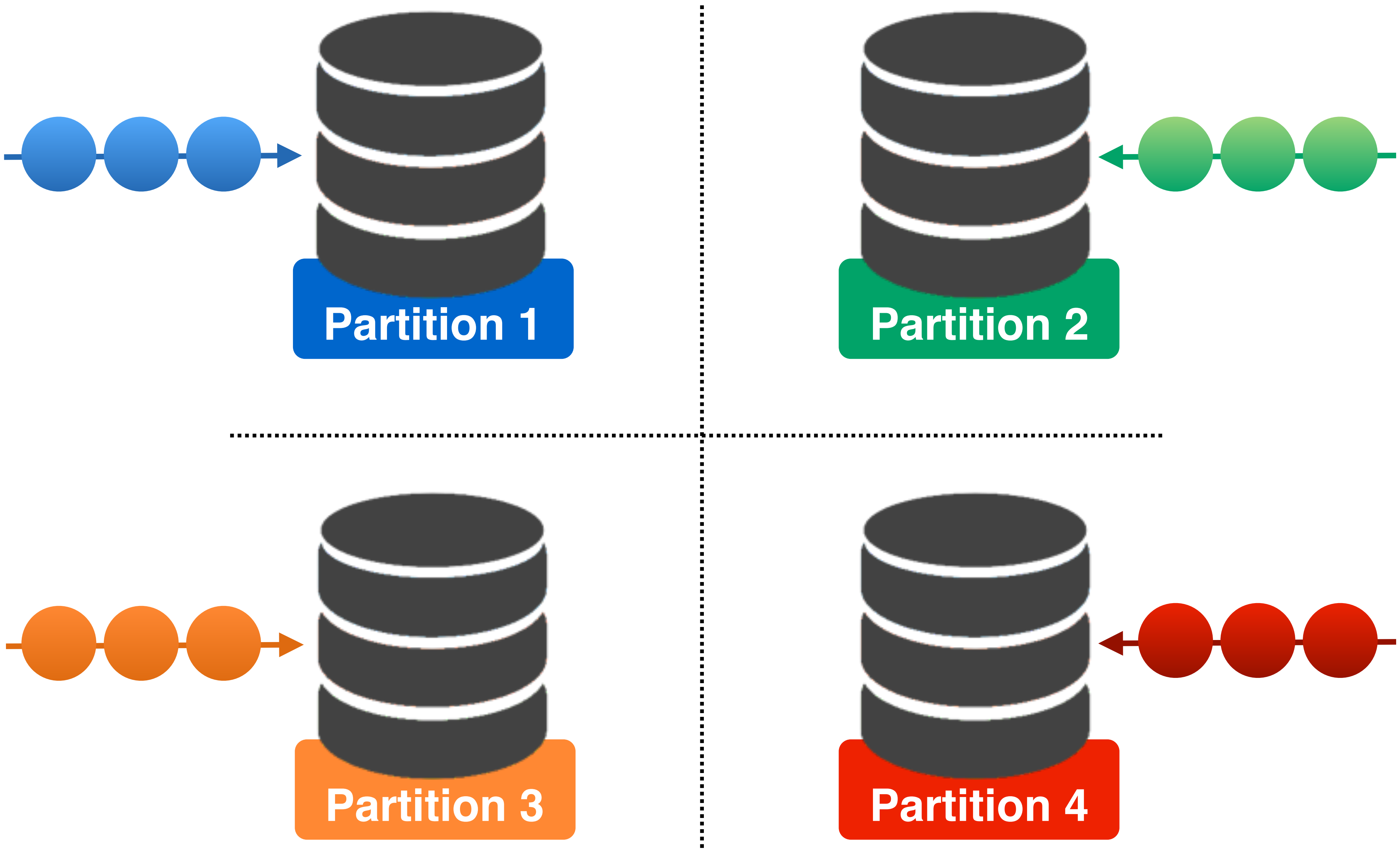
Negative Speed-Up

- Queries **shuffle data** for joins and aggregations
- **Low bandwidth** is main bottleneck
- More servers = **less performance**



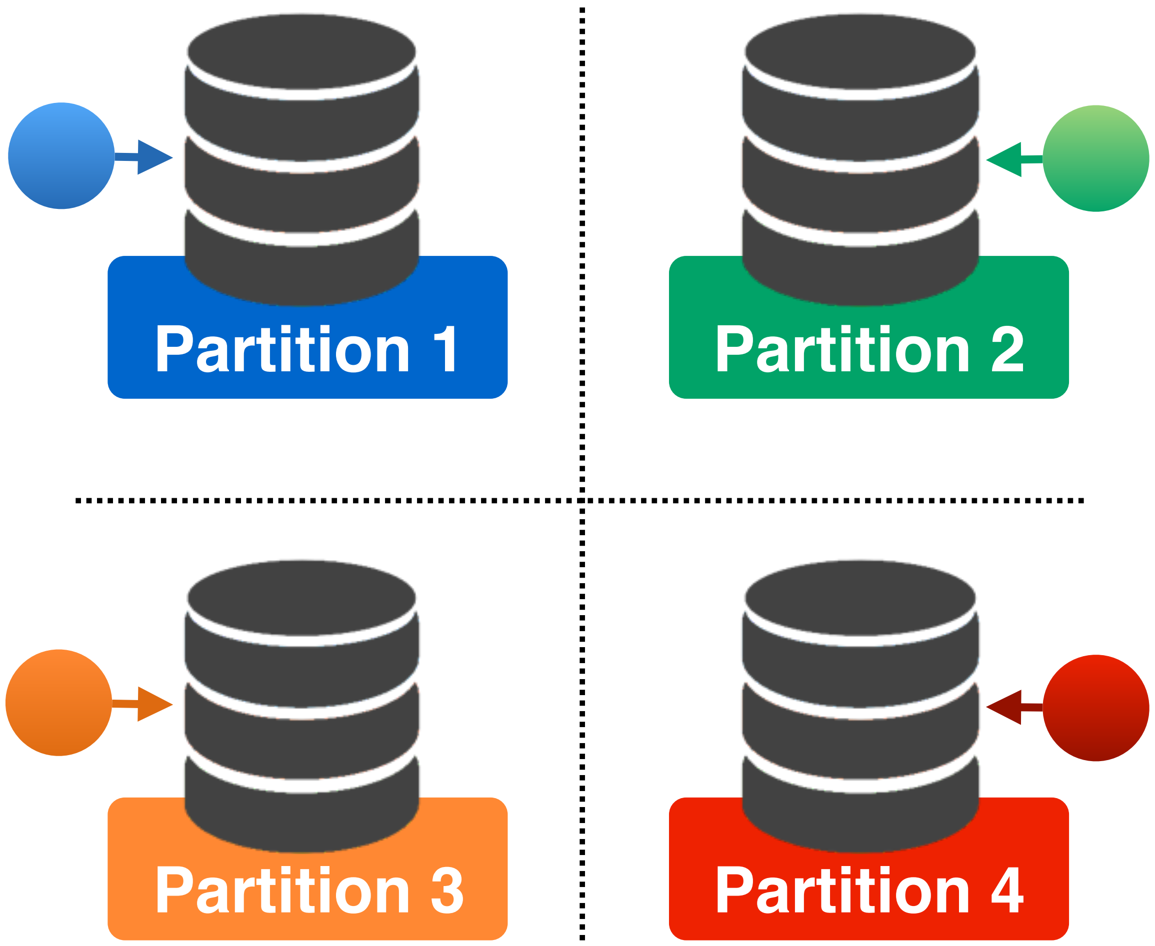
Scale HyPer to a cluster
and it should be **fast**

Can't we just avoid communication?



Can't we just avoid communication?

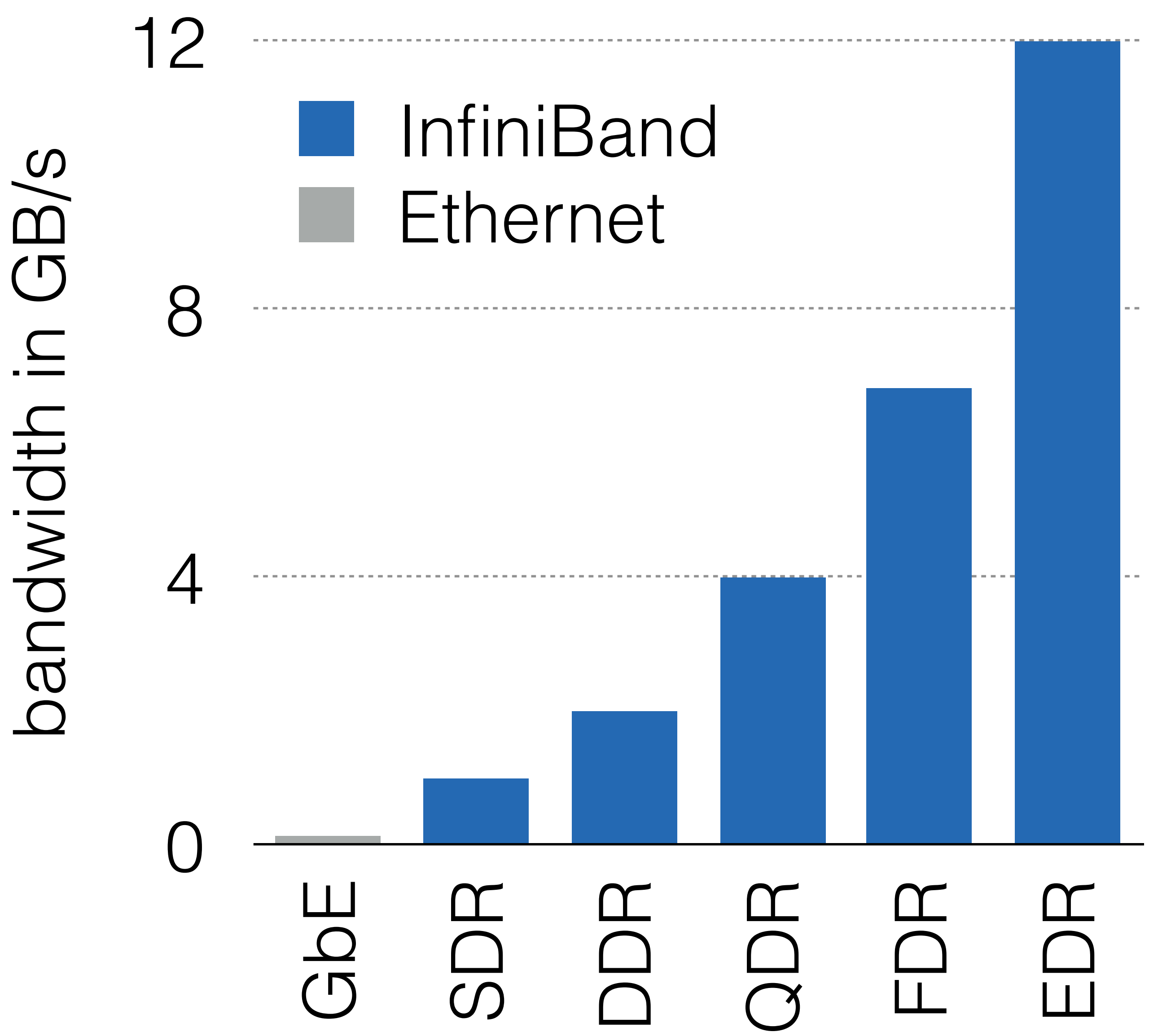
- **Partition** the data (H-Store/VoltDB do this for transactions)
- **Partition-crossing** queries problematic
- Partitioning depends on the **workload**



Can’t we just use faster network hardware?

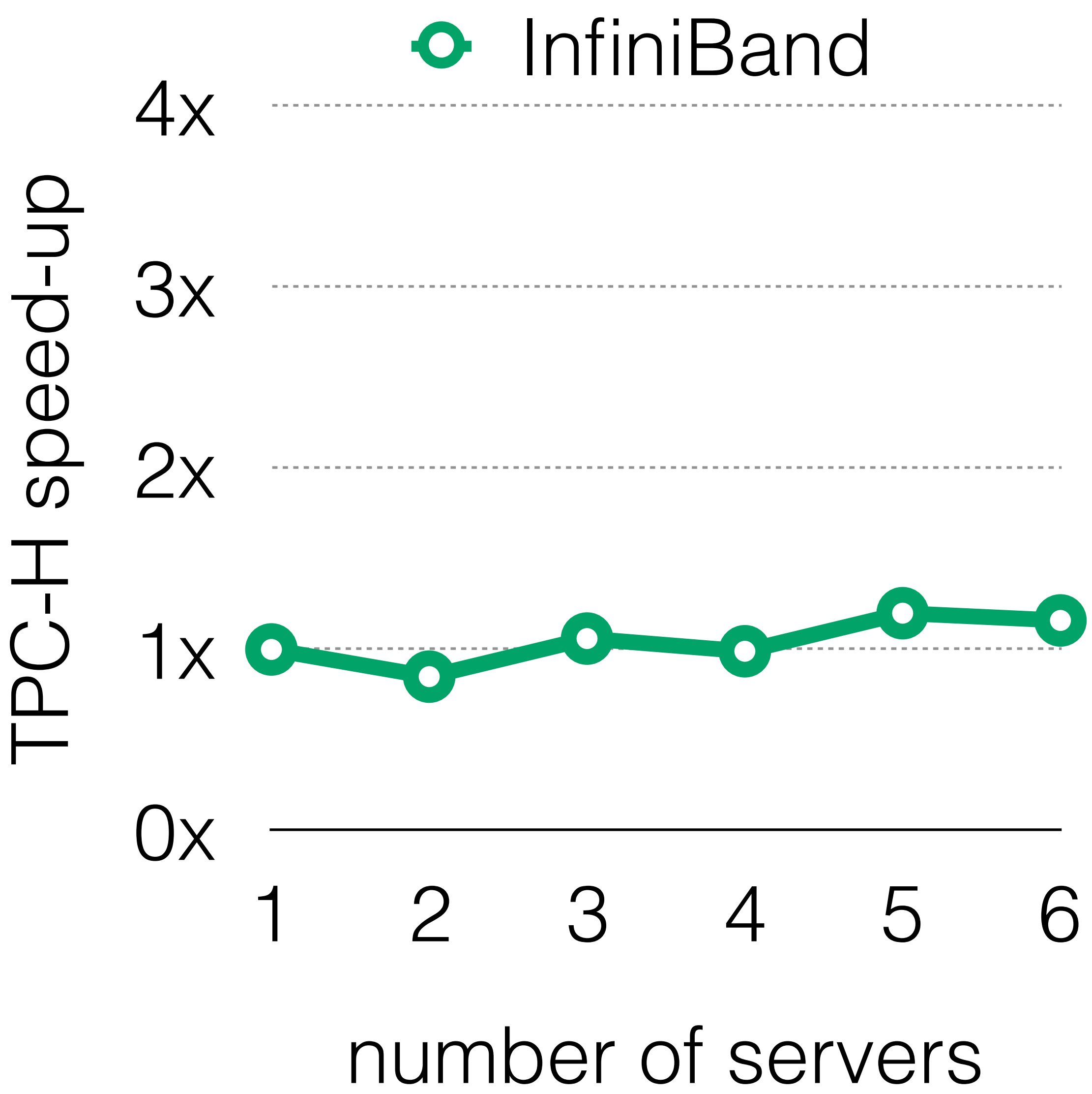
Can’t we just use faster network hardware?

- **Low bandwidth** is main bottleneck
- InfiniBand offers up to **100× the bandwidth**
- Existing software can use IPoIB **unchanged**



Can't we just use faster network hardware?

- **New bottlenecks:**
  - TCP/IP stack processing
  - Interrupts
  - Context switches
  - Multiple memory transfers

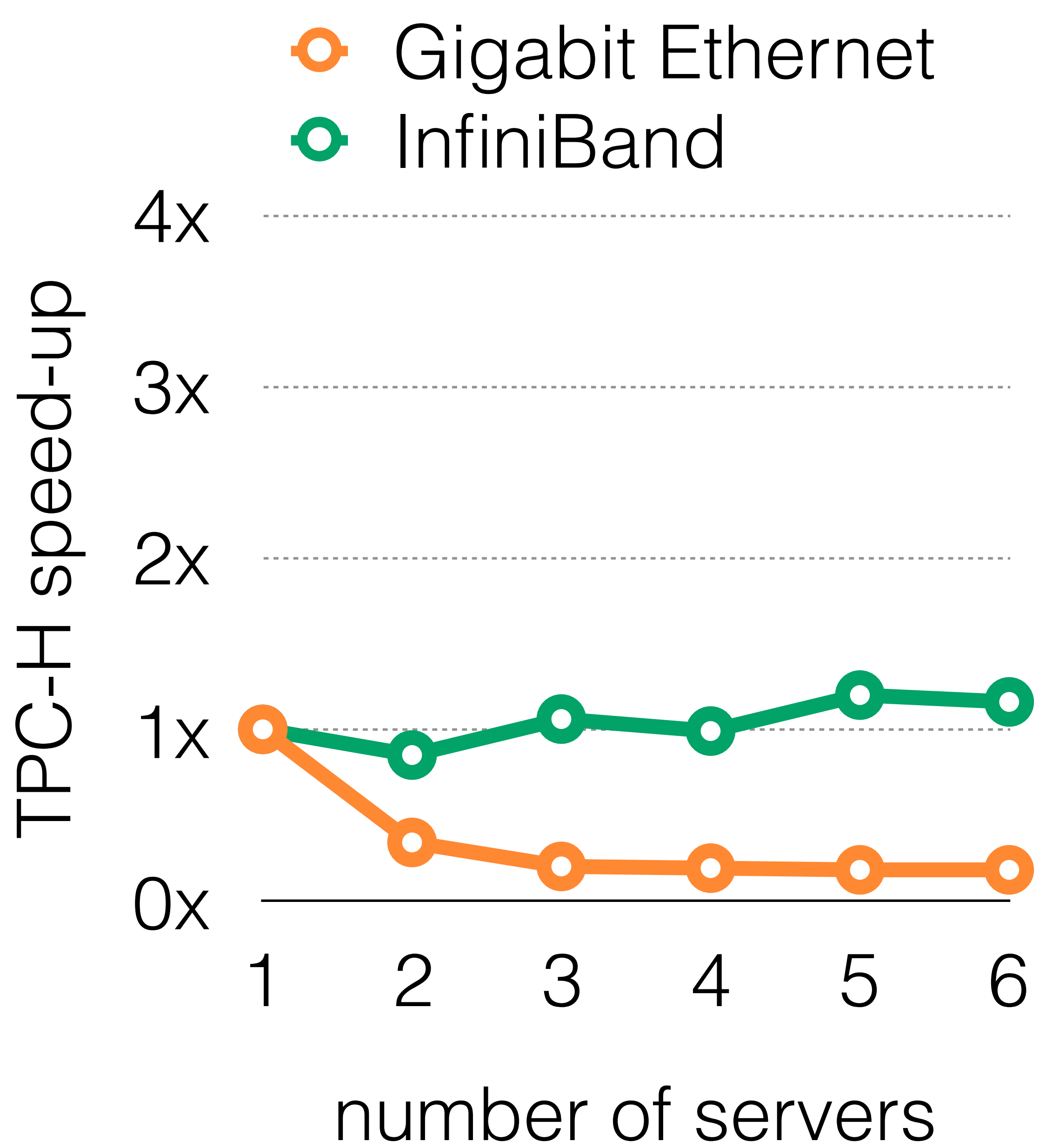


Software has to Change

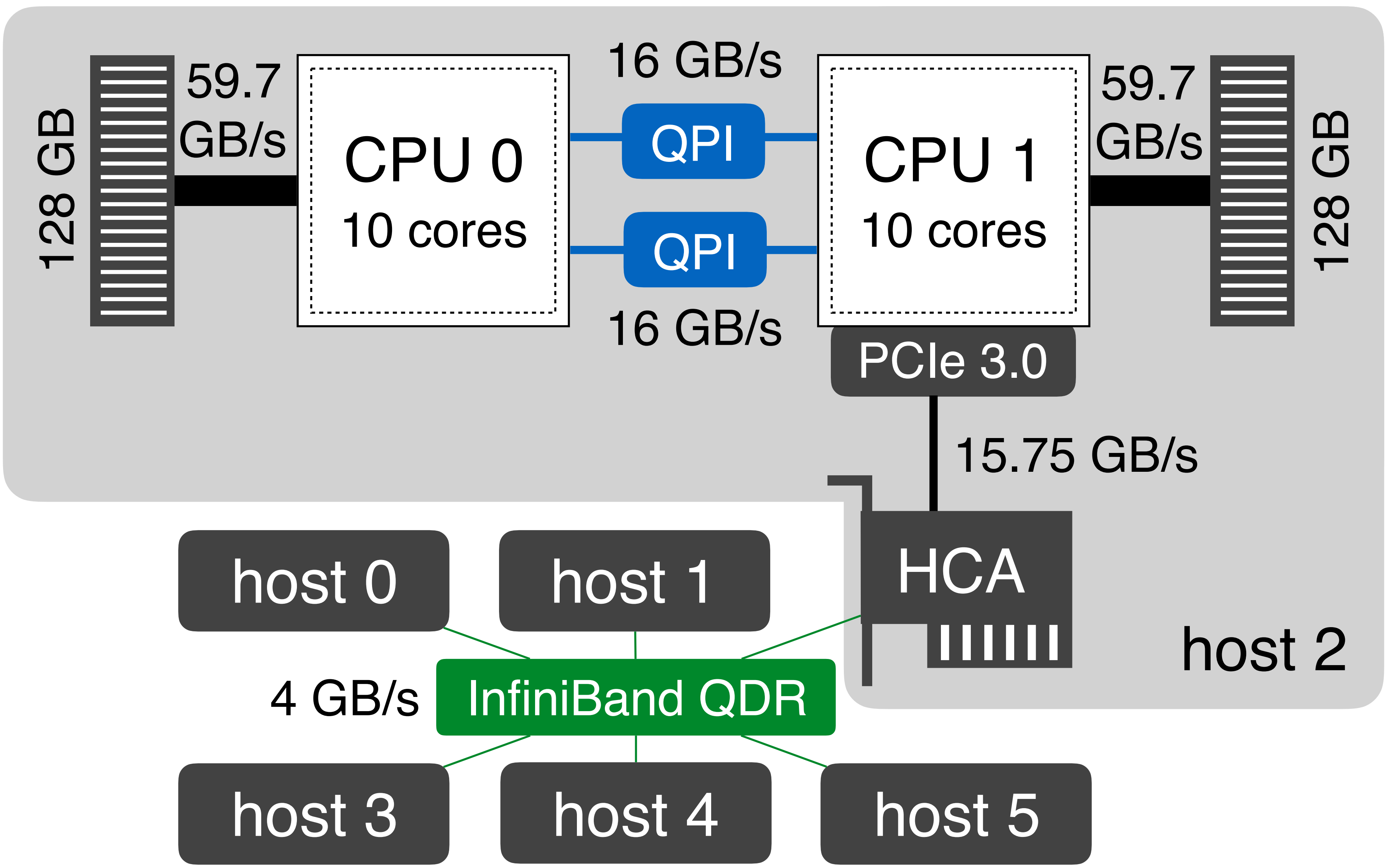
- For **slow** networks more servers = less performance

- New bottlenecks emerge for **faster** networks

- **Software** has to change as well



Two Types of Networks



Two Types of Networks

QPI

- Connects **NUMA** sockets in a server
- **32 GB/s** bandwidth
- **0.2 *μs*** latency
- **Cache-coherent**

InfiniBand QDR

- Connects **servers** in a cluster
- **4 GB/s** bandwidth
- **1.3 *μs*** latency
- **Not** cache-coherent

Hybrid Parallelism

On each server:*

- Use flexible worker threads instead of **exchange** operators
- **Work stealing** per CPU
- Work stealing across **NUMA** sockets

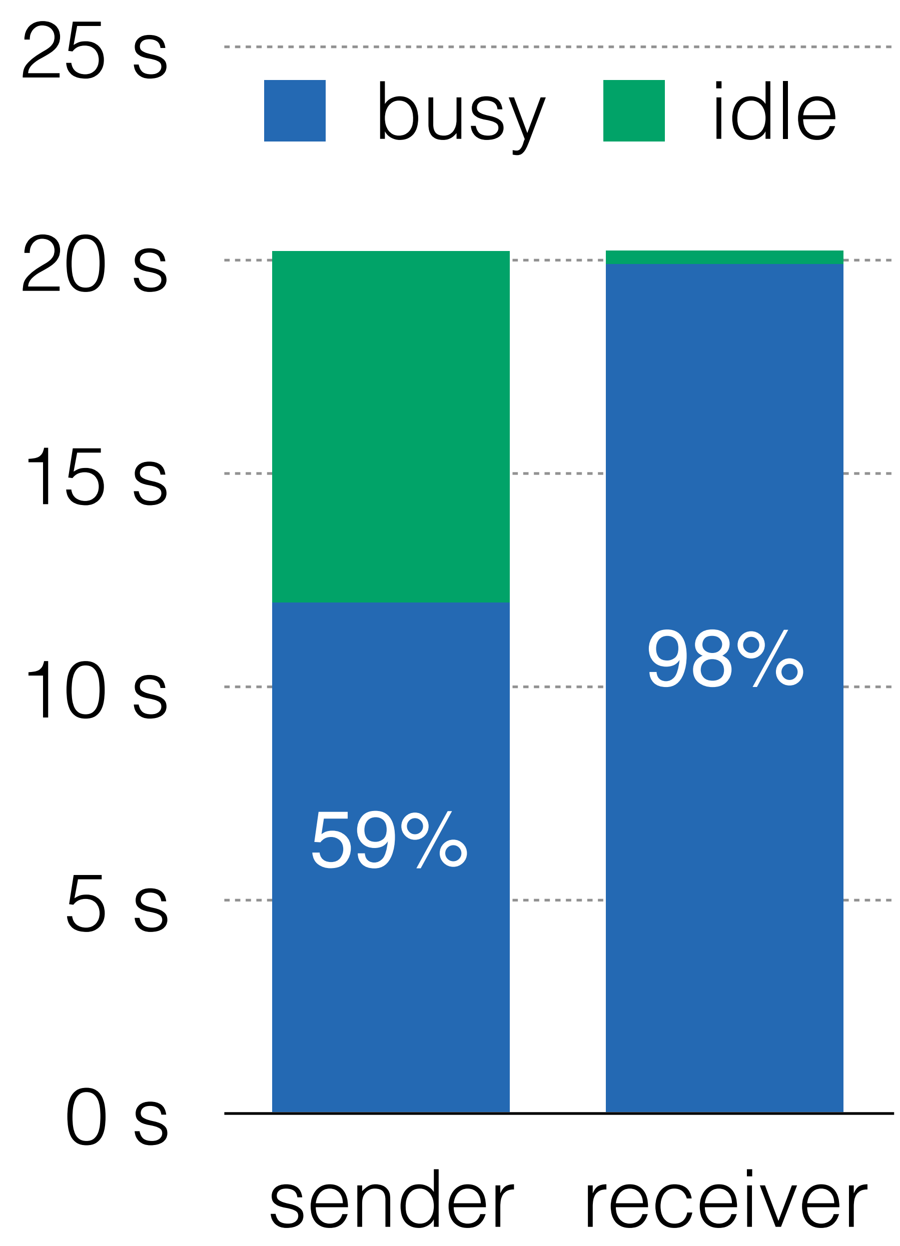
Between servers:

- Use Remote Direct Memory Access **(RDMA)** instead of TCP
- **Decoupled** exchange operators
- Network **scheduling**

* Leis et al., Morsel-driven parallelism, SIGMOD 2014

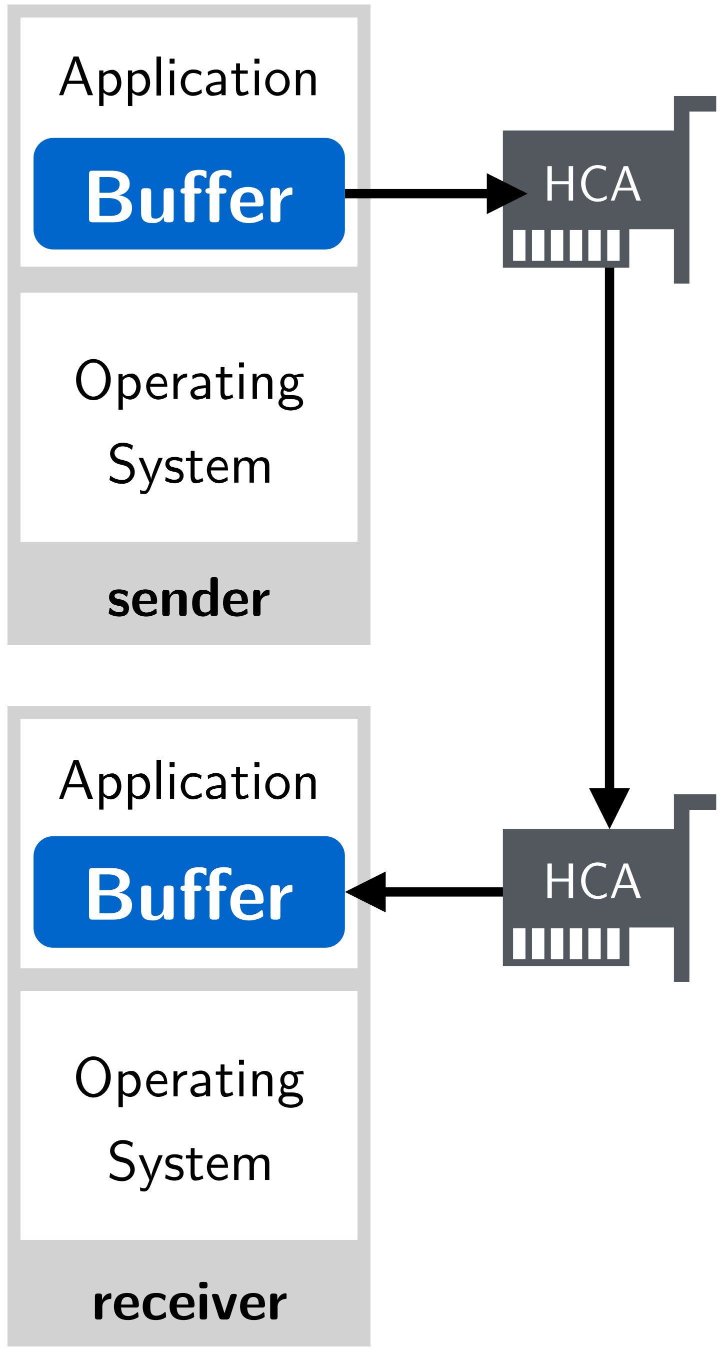
TCP over InfiniBand

- TCP is **compute-bound** at the receiver
- Even with **large MTUs** and **TCP offloading**
- Using a **separate core** for interrupts improves throughput by **53%**
- Still **compute-bound**

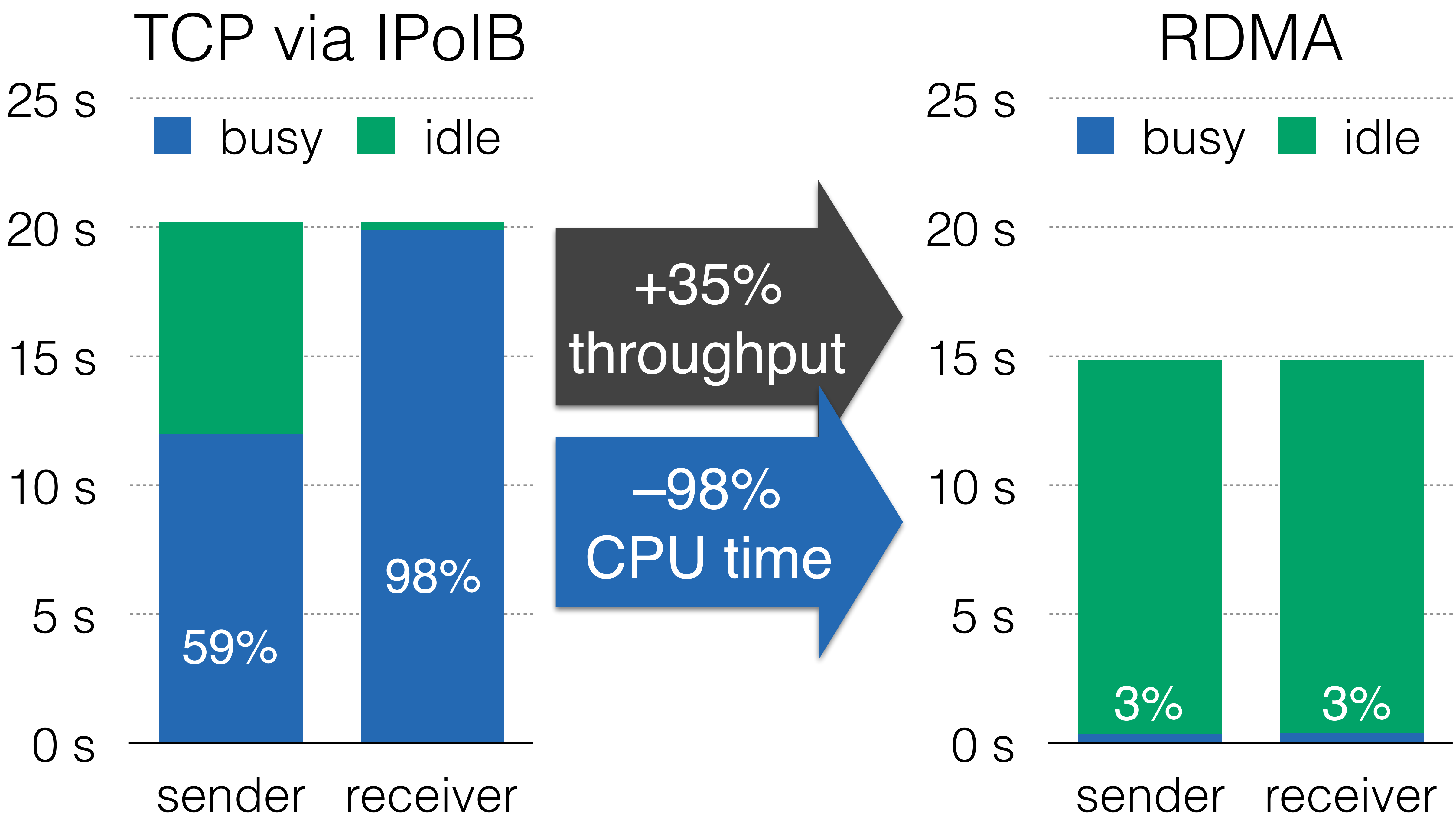


Remote Direct Memory Access

- **Bypasses** operating system and application
- **Zero-copy** network communication:
  - Achieves full **network** throughput
  - Almost no **CPU** cost
  - Less **data copying**

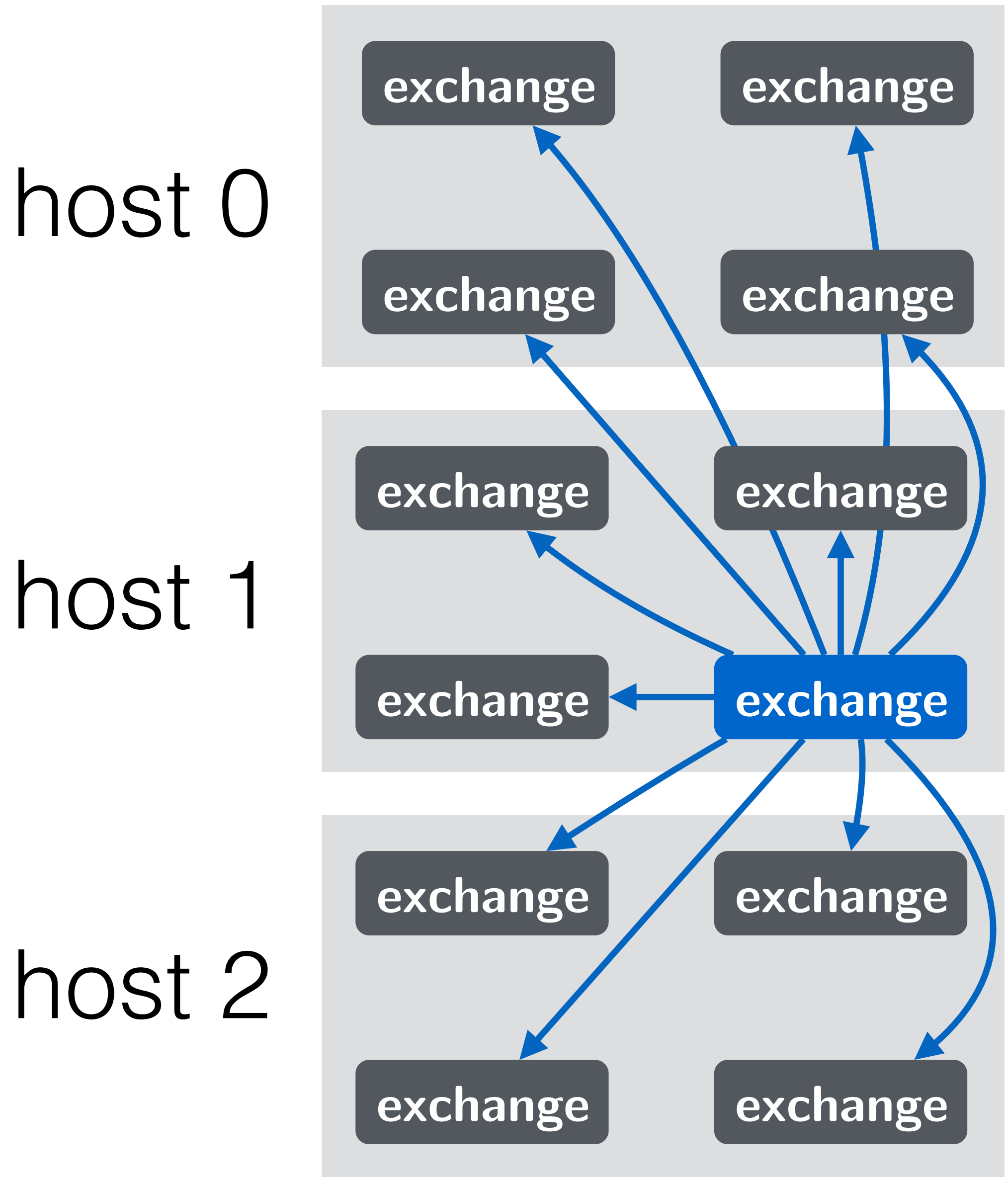


Remote Direct Memory Access



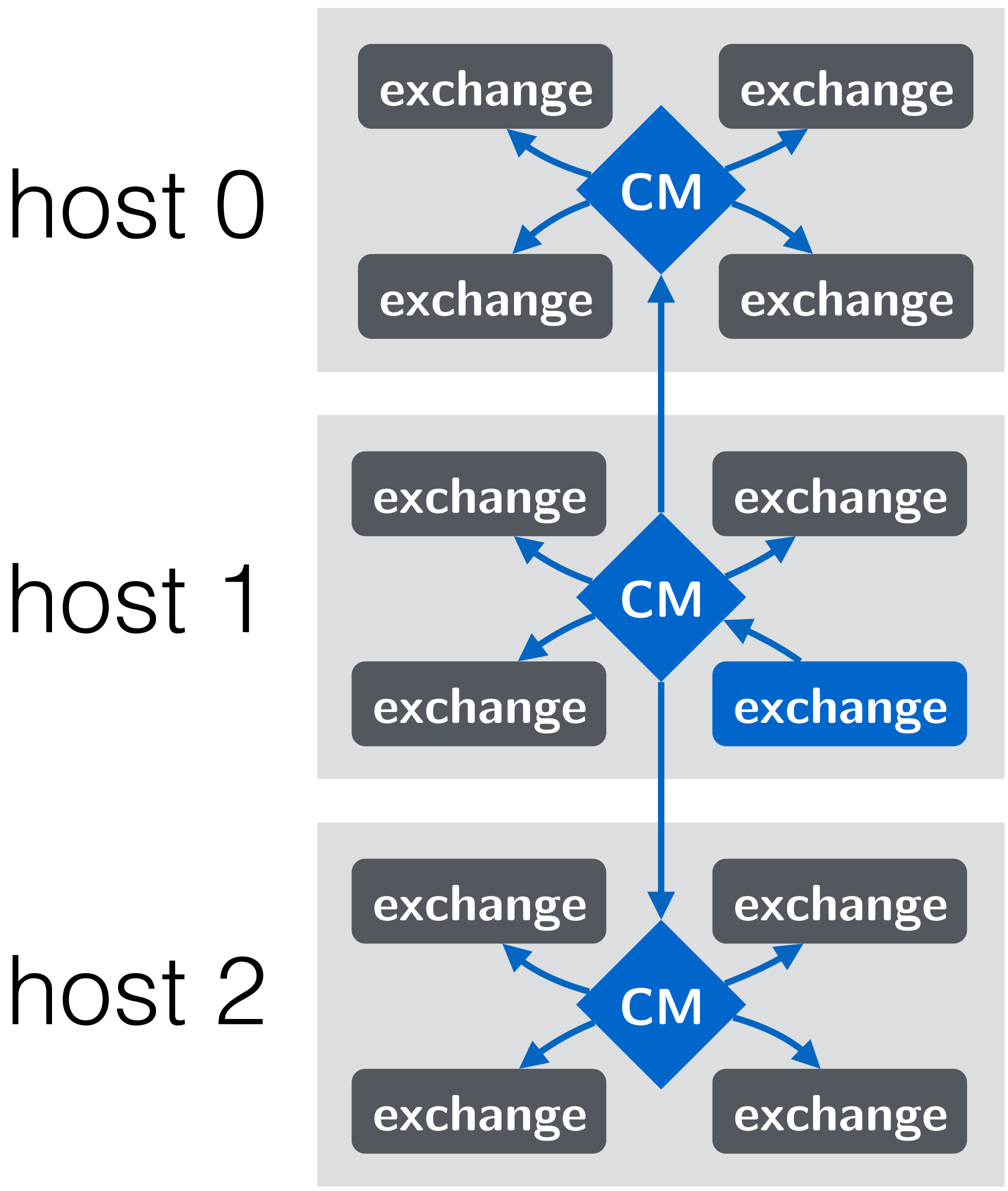
Classic Exchange

- A buffer **per exchange:**
  - # buffers per server = **servers** × **cores**^{2}
  - **1 GB**/server for 6 hosts and 20 cores
- **Skew** is huge problem:
  - Join key assigned to **fixed** exchange
  - **No work stealing**

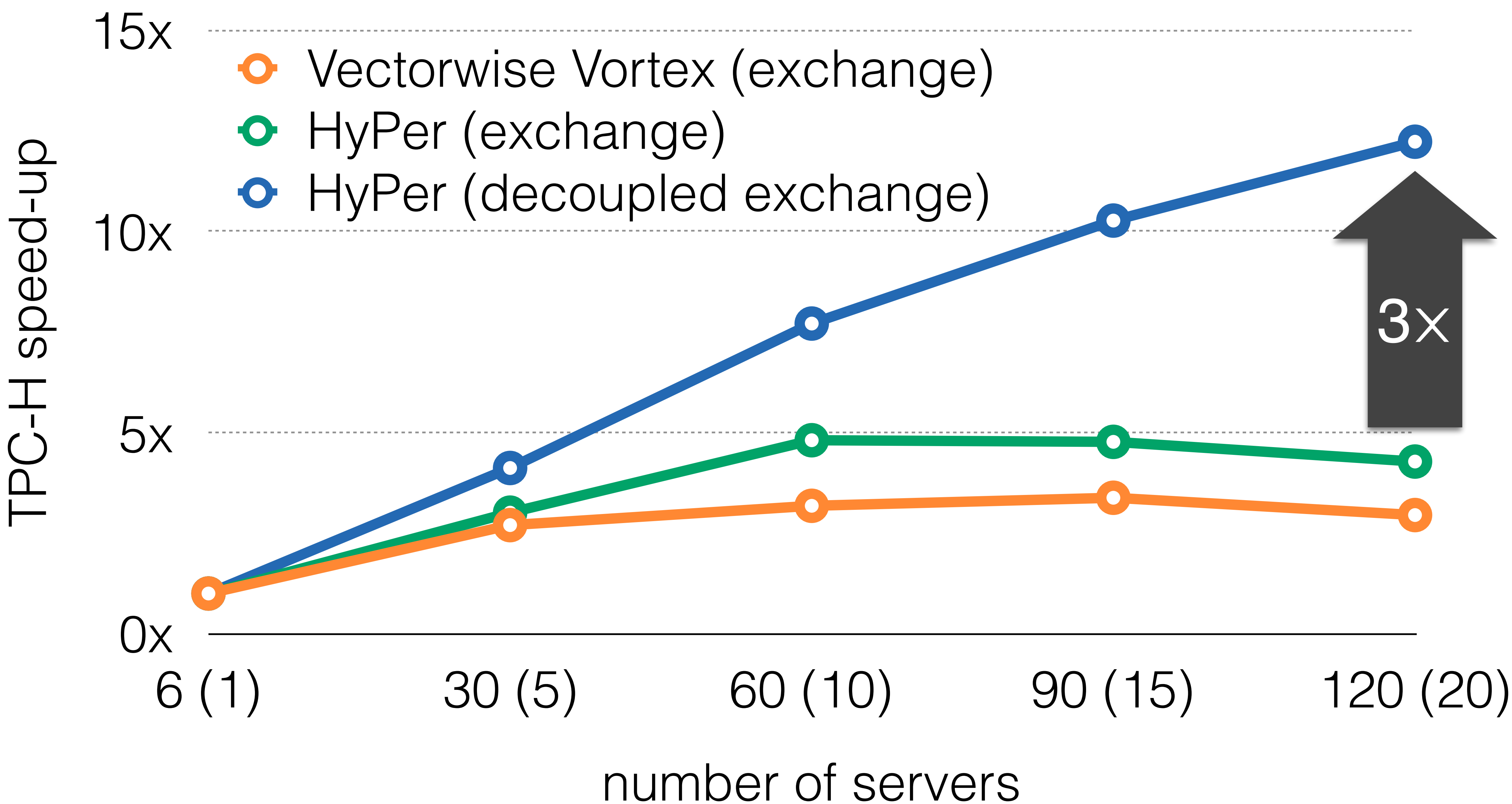


Decoupled Exchange

- Use **communication multiplexers** (CM)
- **Address servers** not individual cores:
  - Decreases **memory** consumption (2.5 MB instead of 1 GB)
  - Reduces negative impact of **skew**
  - Makes **broadcast** more applicable

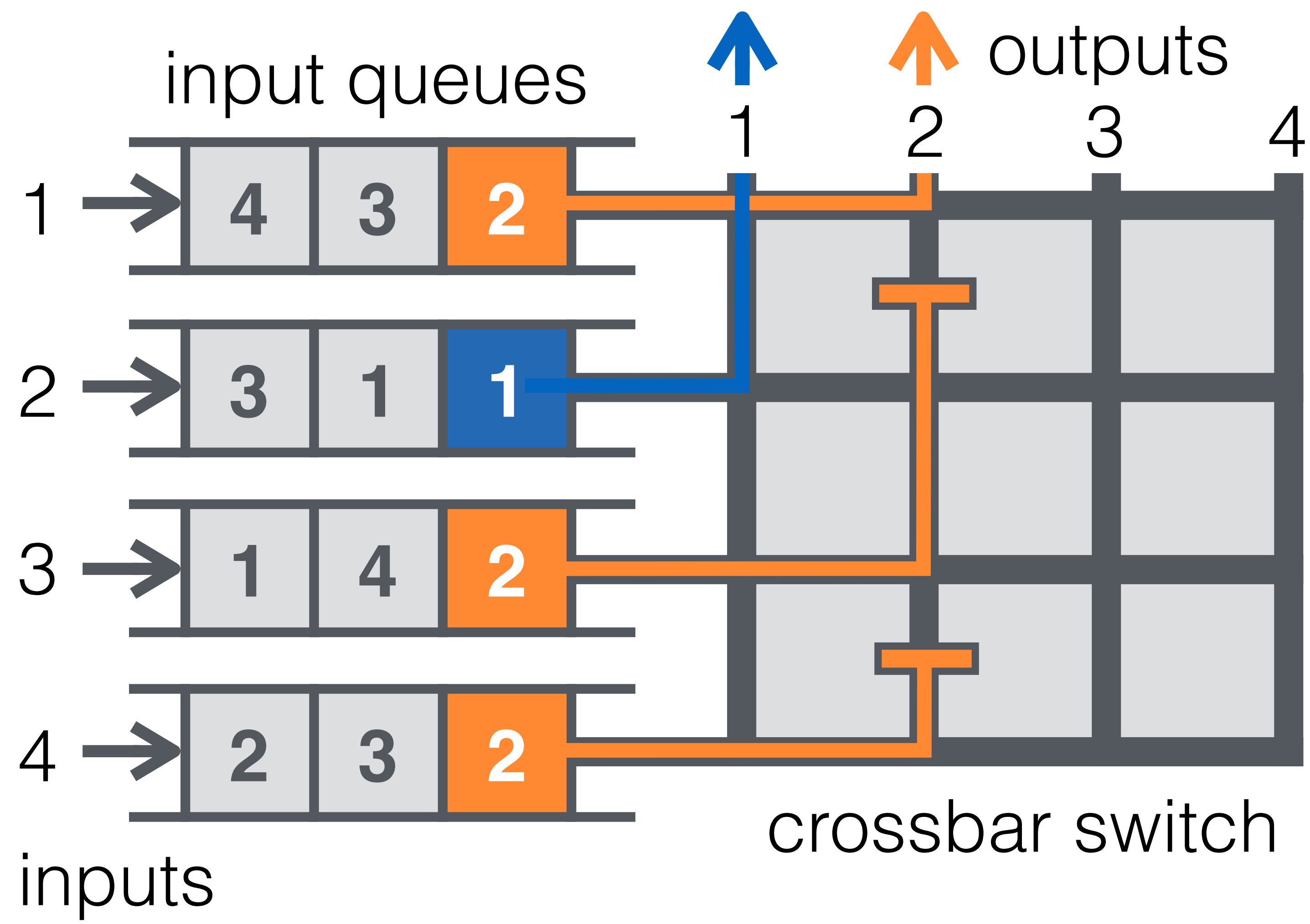


Decoupled Exchange

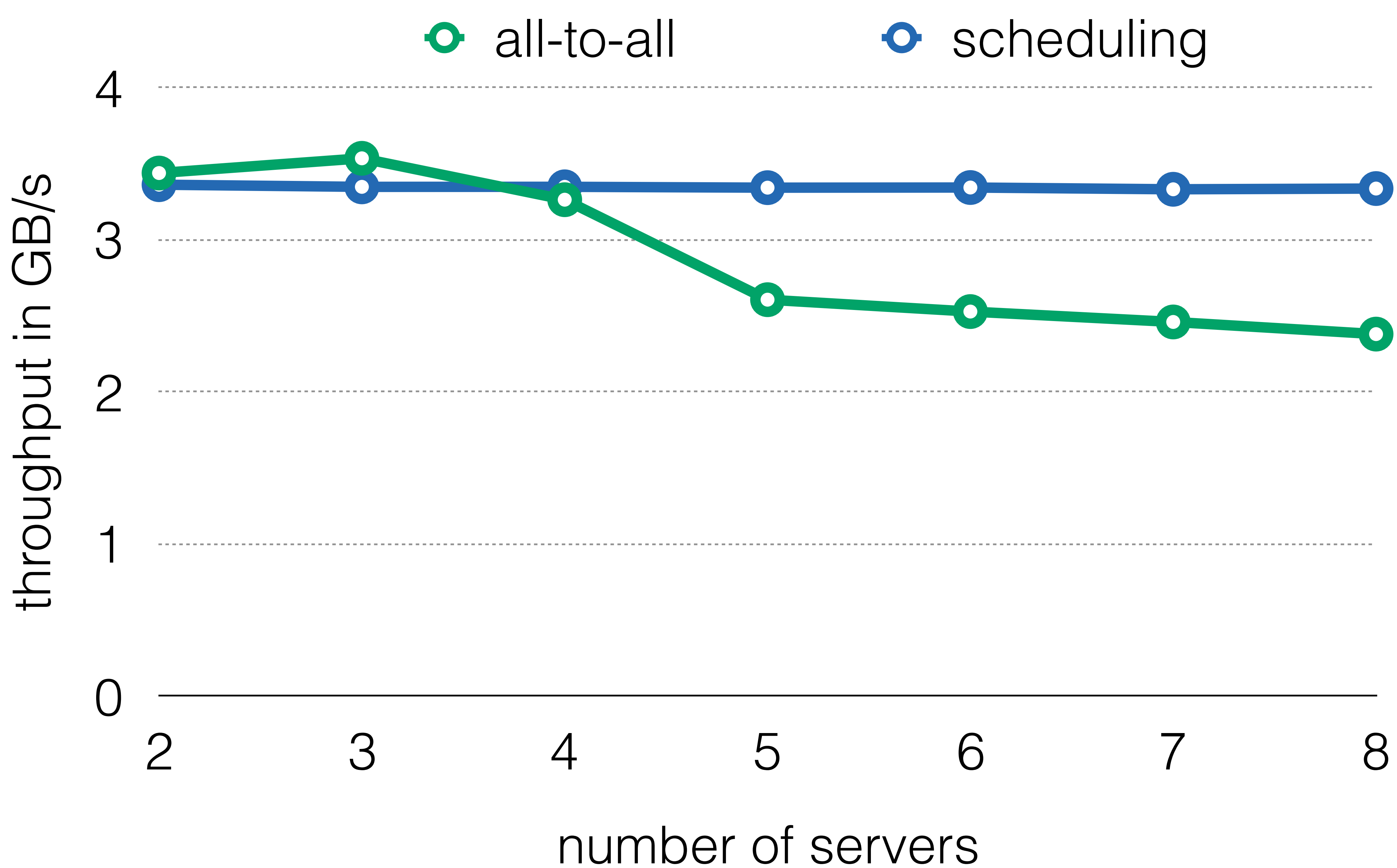


Network Scheduling

- Uncoordinated all-to-all transfers cause **switch contention**
- Make sure a server **sends** to at most one server
- **Low-latency** inline RDMA messages for network scheduling

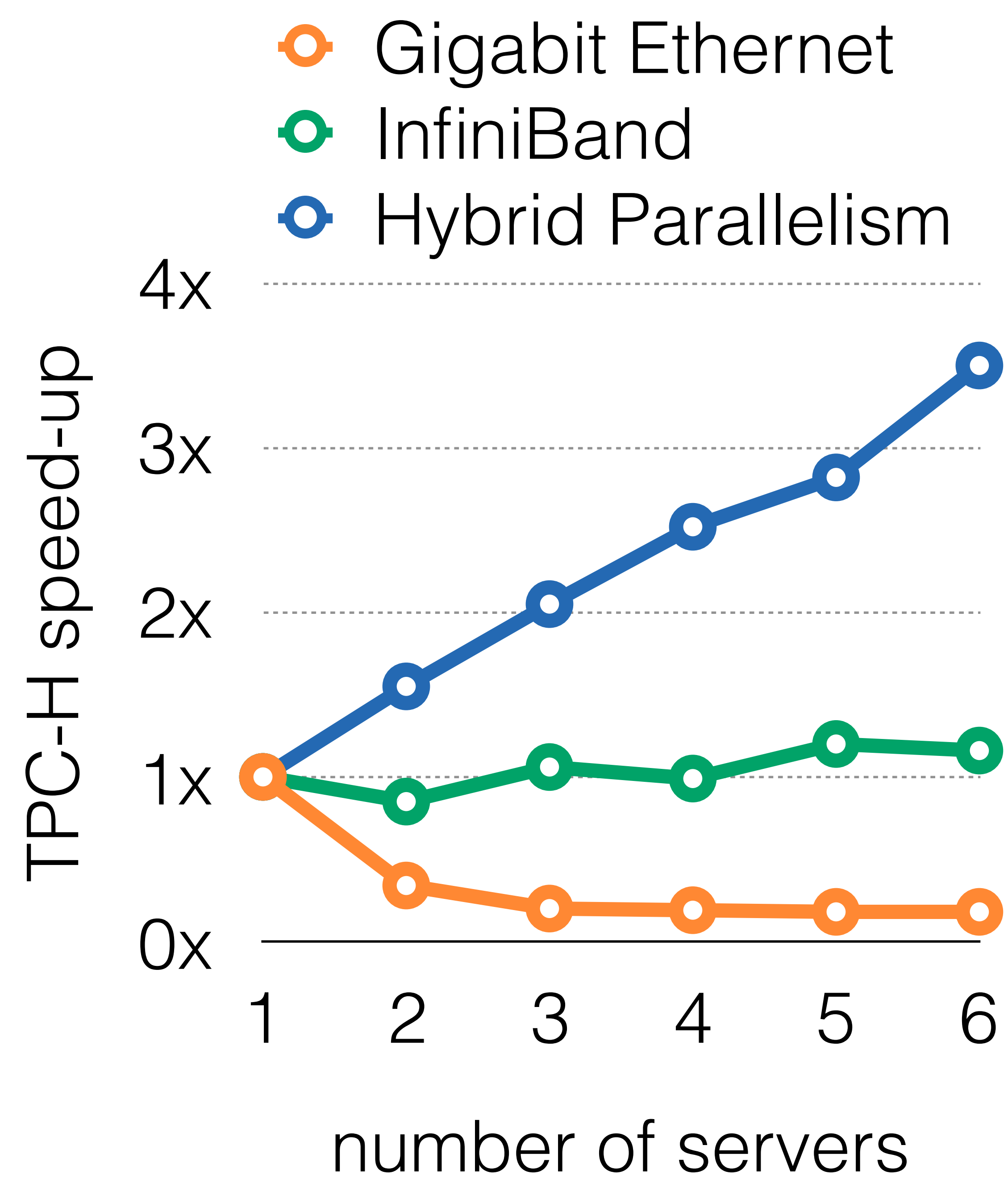


Network Scheduling

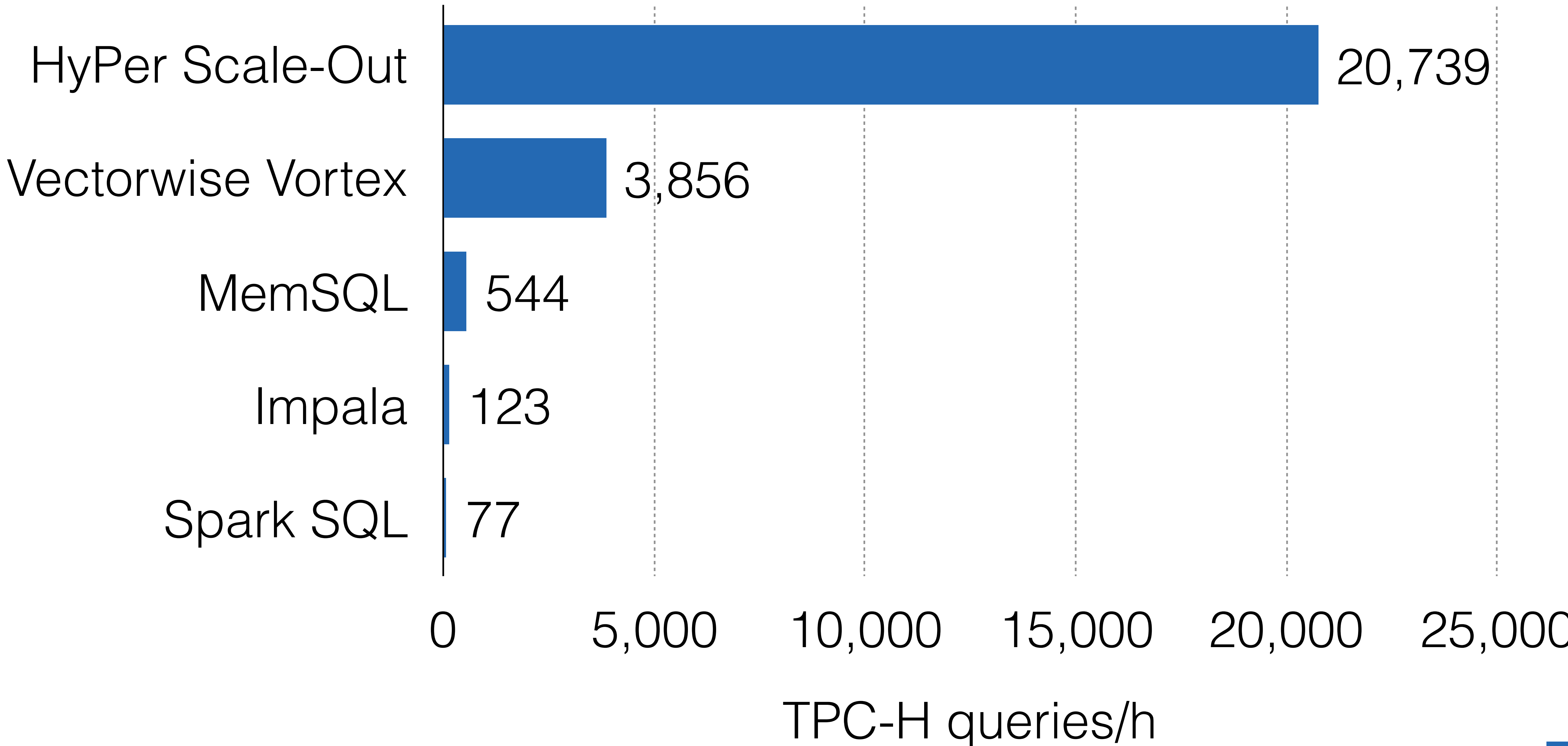


Summary

- For **slow** networks more servers = less performance
- New bottlenecks emerge for **faster** networks
- **Hybrid parallelism** optimizes for both types of networks



How do we compare?

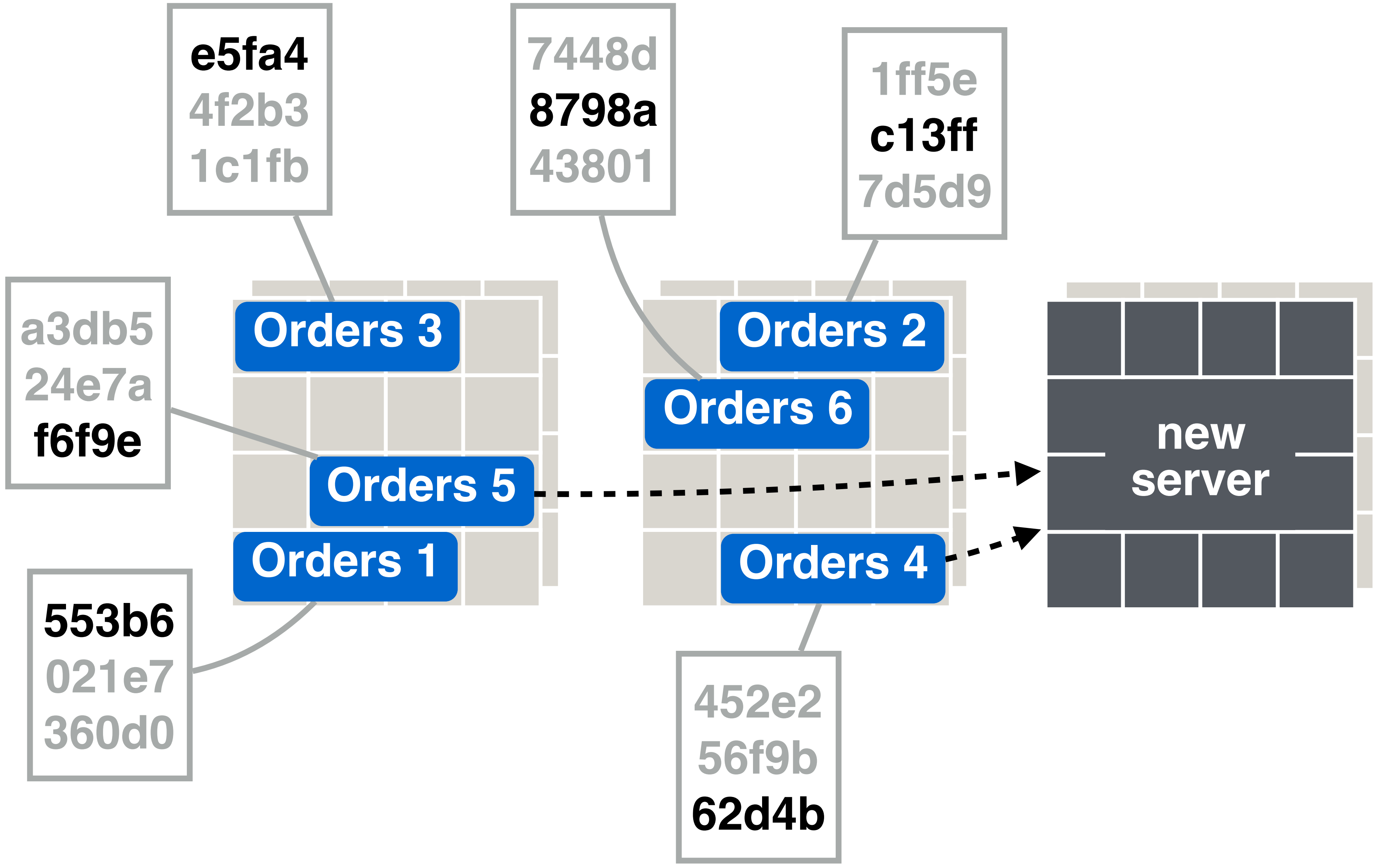


Future Work

What about low-latency distributed **transactions**?

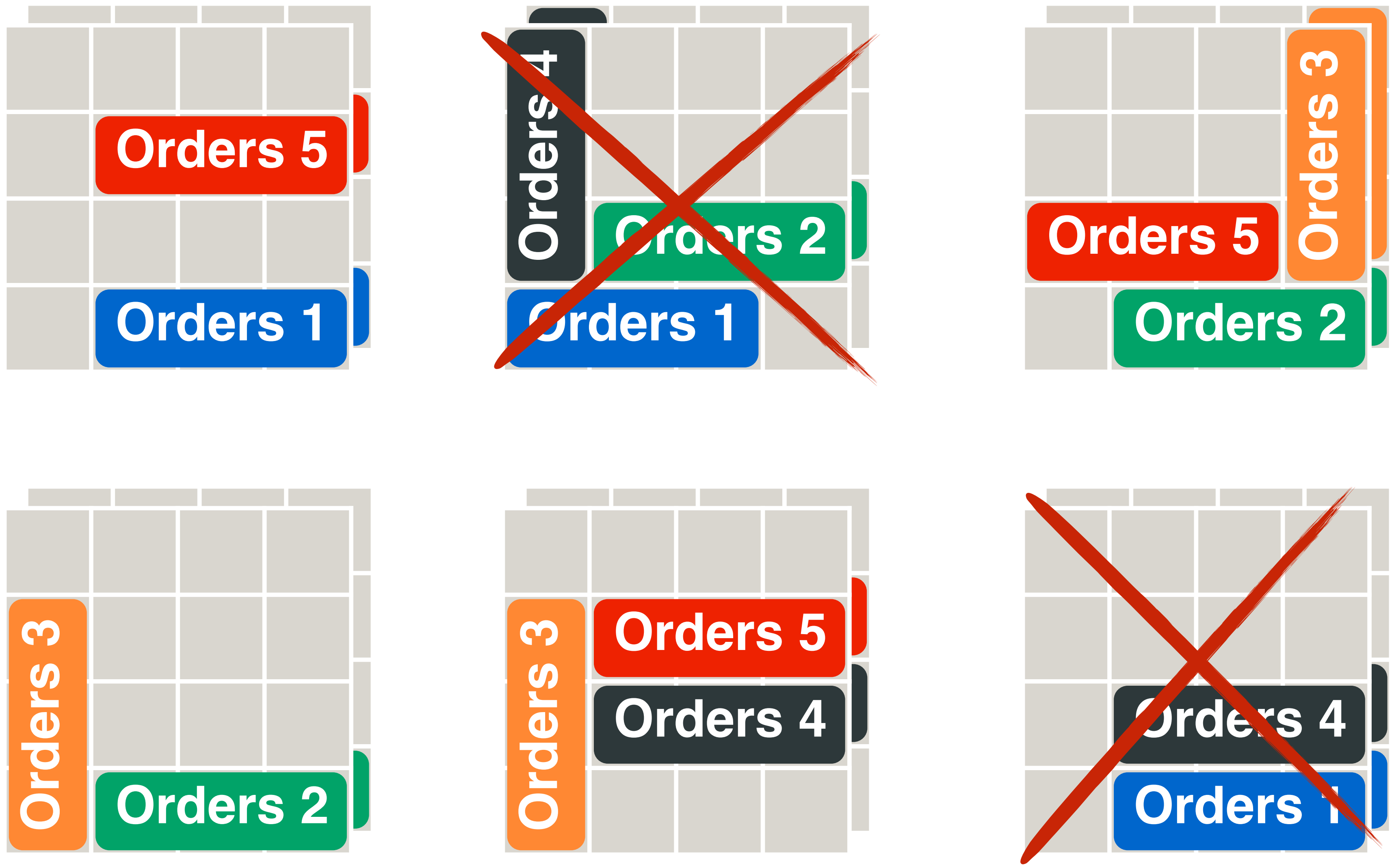
Backup

Elasticity

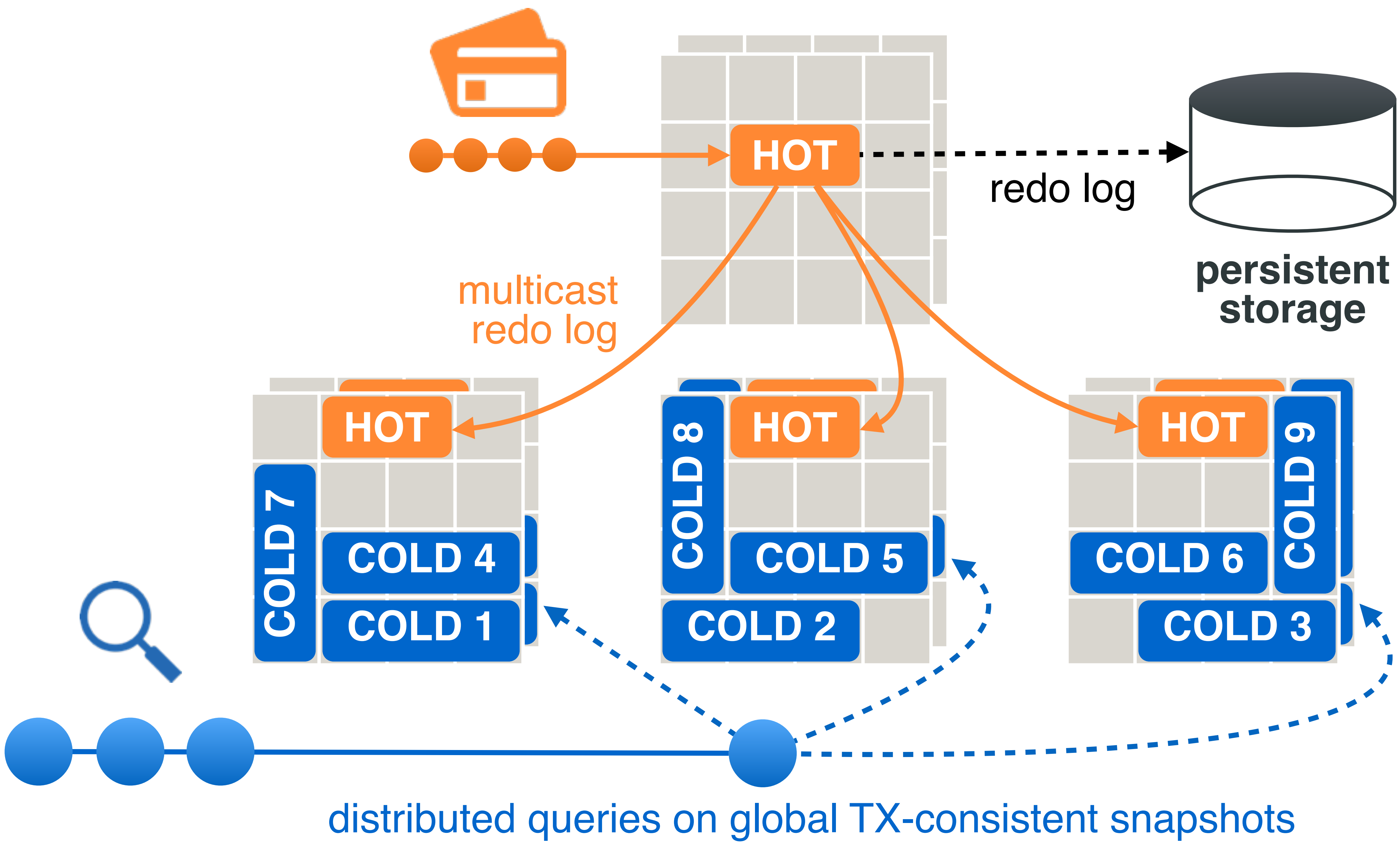


e5fa4
4f2b3
1c1fb
7448d
8798a
43801
1ff5e
c13ff
7d5d9
a3db5
24e7a
f6f9e
Orders 3
Orders 2
Orders 6
Orders 5
Orders 1
Orders 4
new server
553b6
021e7
360d0
452e2
56f9b
62d4b

High Availability



Hot/cold approach



Parallelism

theory

practice