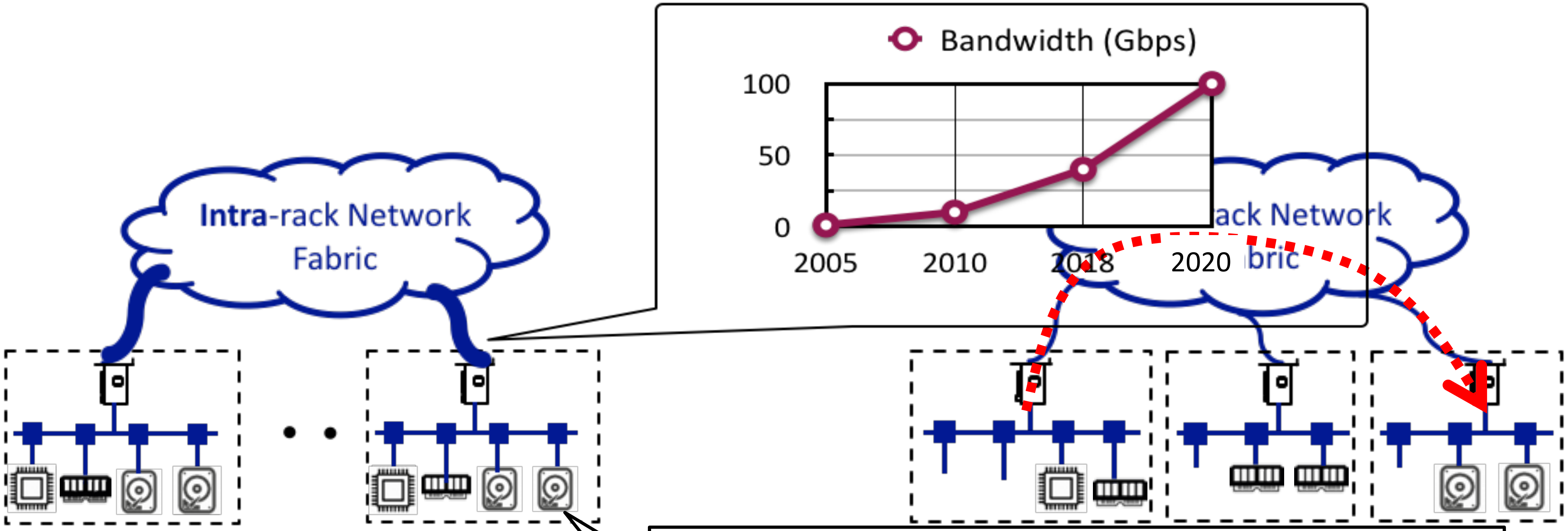
TCP≈RDMA: CPU-efficient Remote Storage Access with i10




Jaehyun Hwang Qizhe Cai Ao Tang Rachit Agarwal

Cornell University

i10 Motivation: Two trends in Remote I/O



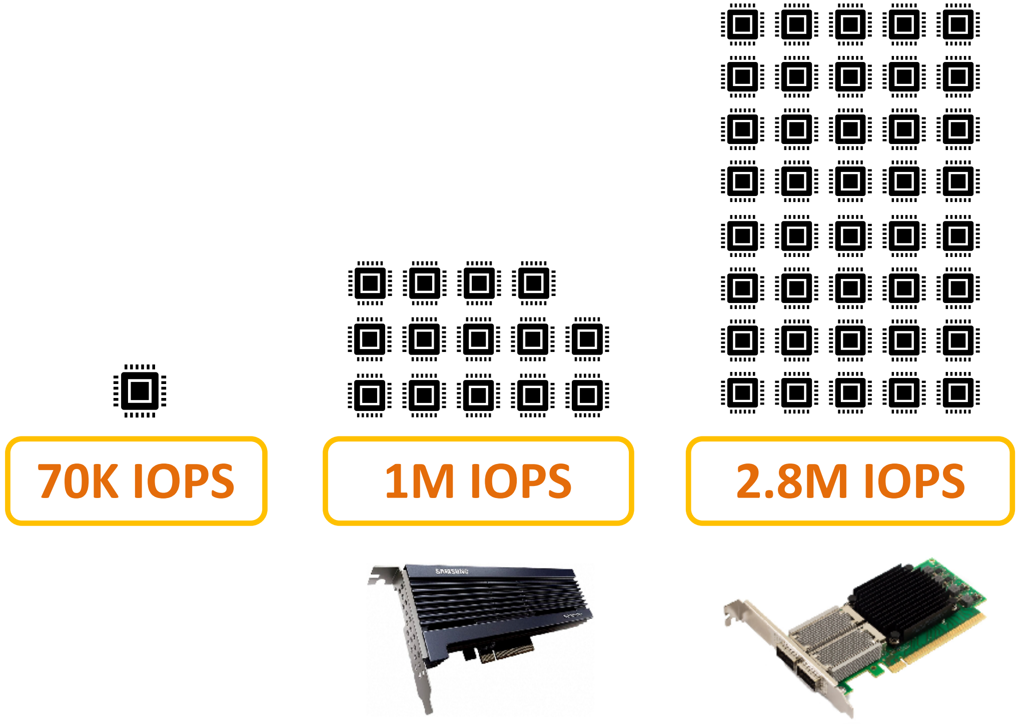
- Move from SCSI to NVMe
- Emergence of NVMe SSDs:
  - >1M IOPS (read), >400K IOPS (write)

1. High-throughput NVMe SSD, high-bandwidth access links

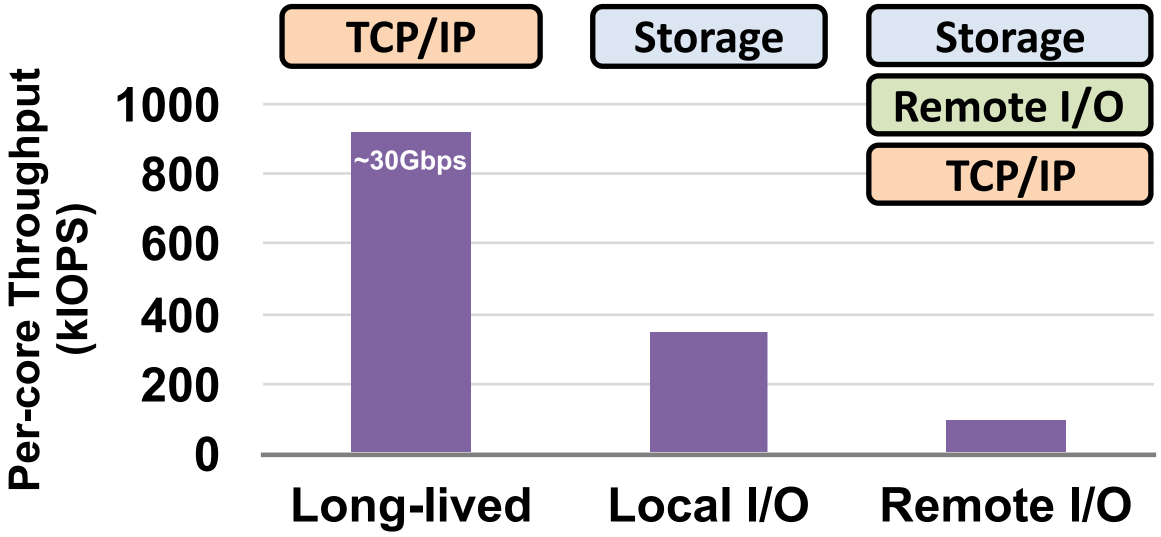
2. Apps use disaggregated storage via Remote I/O

What do these trends mean for Remote I/O?

- Software stack (iSCSI) performance



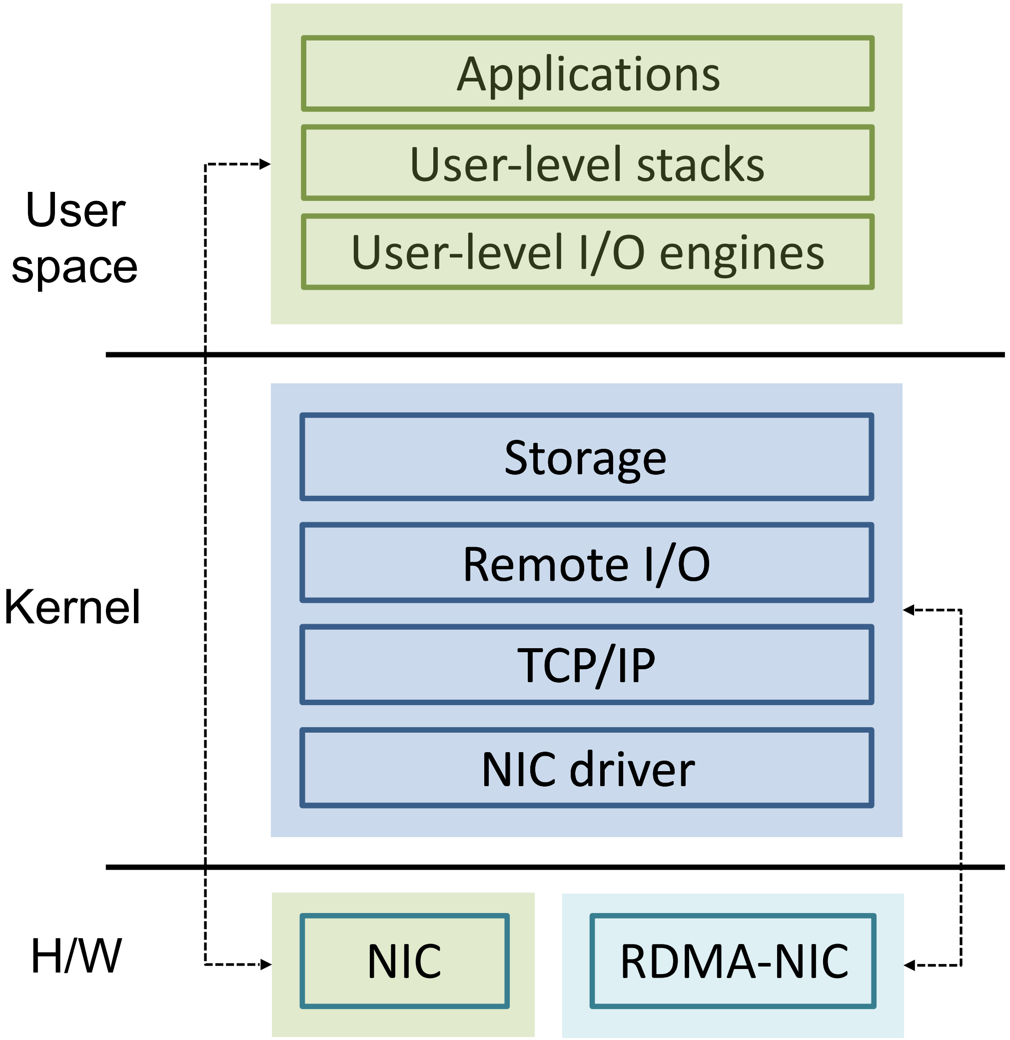
- Storage + network overlap



Bottlenecks pushed back to OS, software stack!

Bottlenecks at the boundary of storage and network stacks!

Previous Approaches



User-space storage and network stacks

– Storage + Remote I/O (user) + DPDK

– Performance: Good!

In-Kernel (NVMe-over-TCP)

– Storage + Remote I/O + TCP (all in the kernel)

– Performance: Not-so-good!

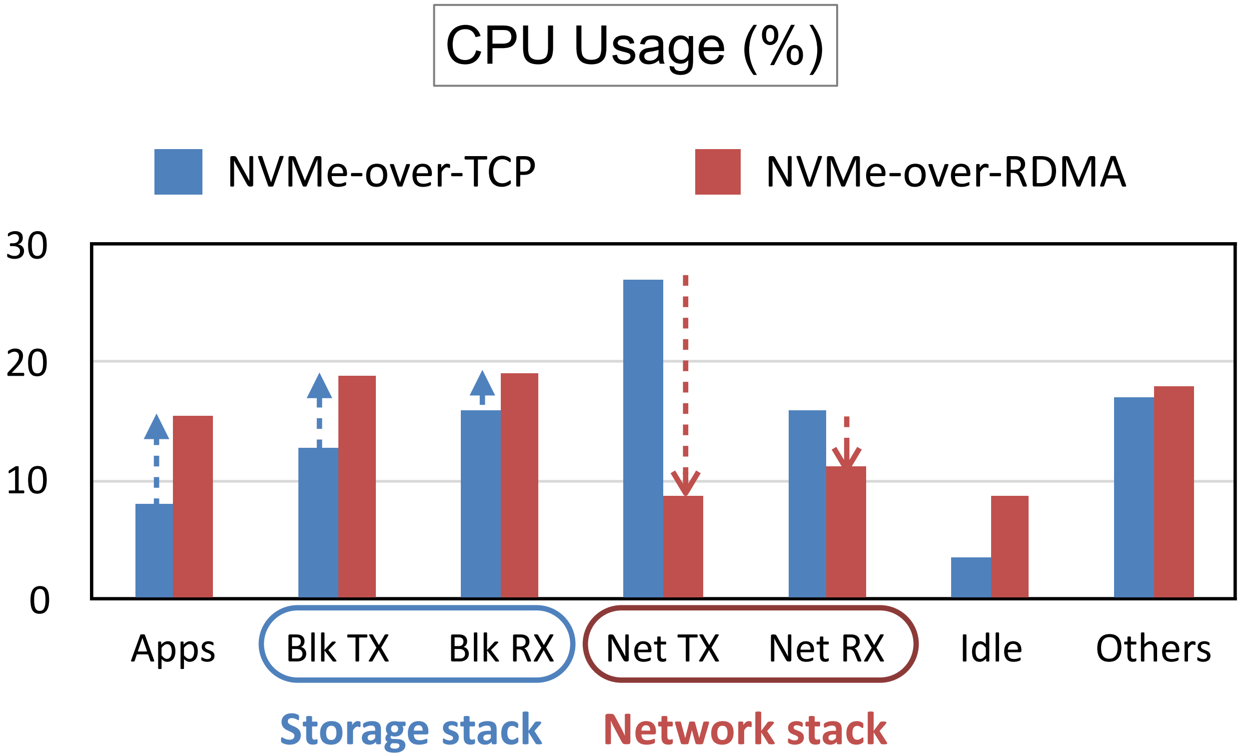
Fundamental?

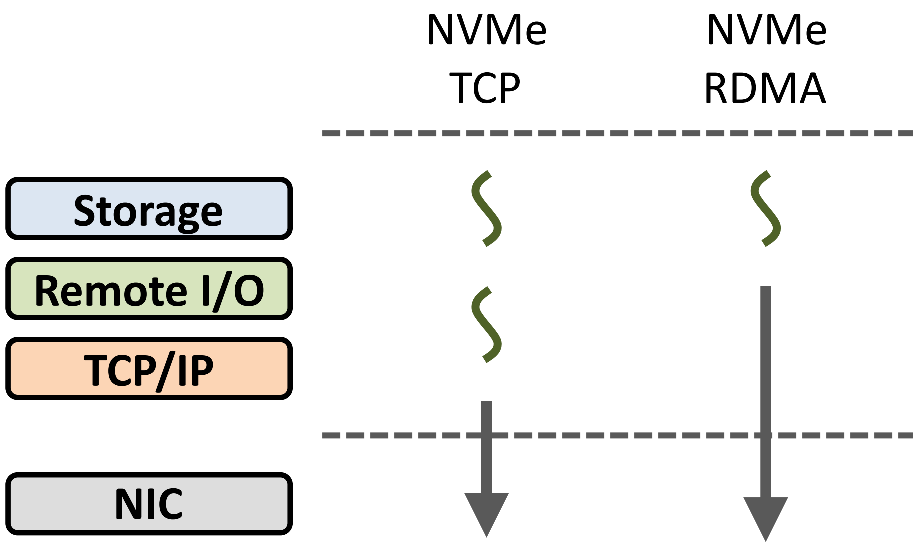
NVMe-over-RDMA

– Storage + Remote I/O (kernel) + RDMA

– Performance: Good!

Remote Storage Access Overheads: TCP vs. RDMA





Network processing overhead!

Context switching overhead!

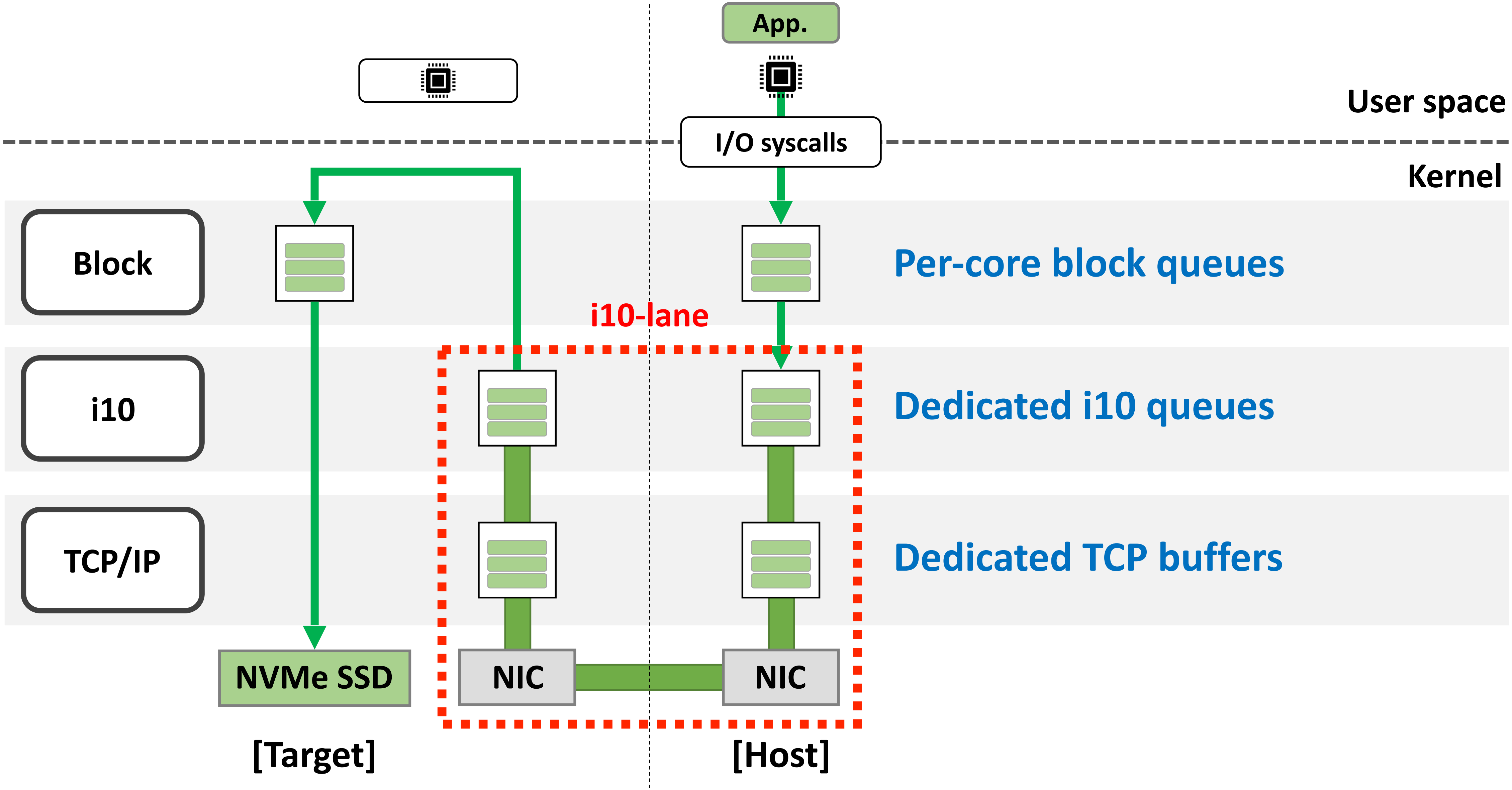
i10 Summary

- A new **remote I/O stack** implemented **entirely in the kernel**
  - No changes in apps, no changes in TCP/IP stack, no changes in hardware
- **Throughput-per-core similar to NVMe-over-RDMA**
  - Latency within 1.7x of RDMA (for SSD accesses)
- **Three *simple* ideas**
  - **i10-lane**: dedicated resources
  - **i10-caravans**: request and data batching
  - **Delayed doorbells**: Interrupt coalescing

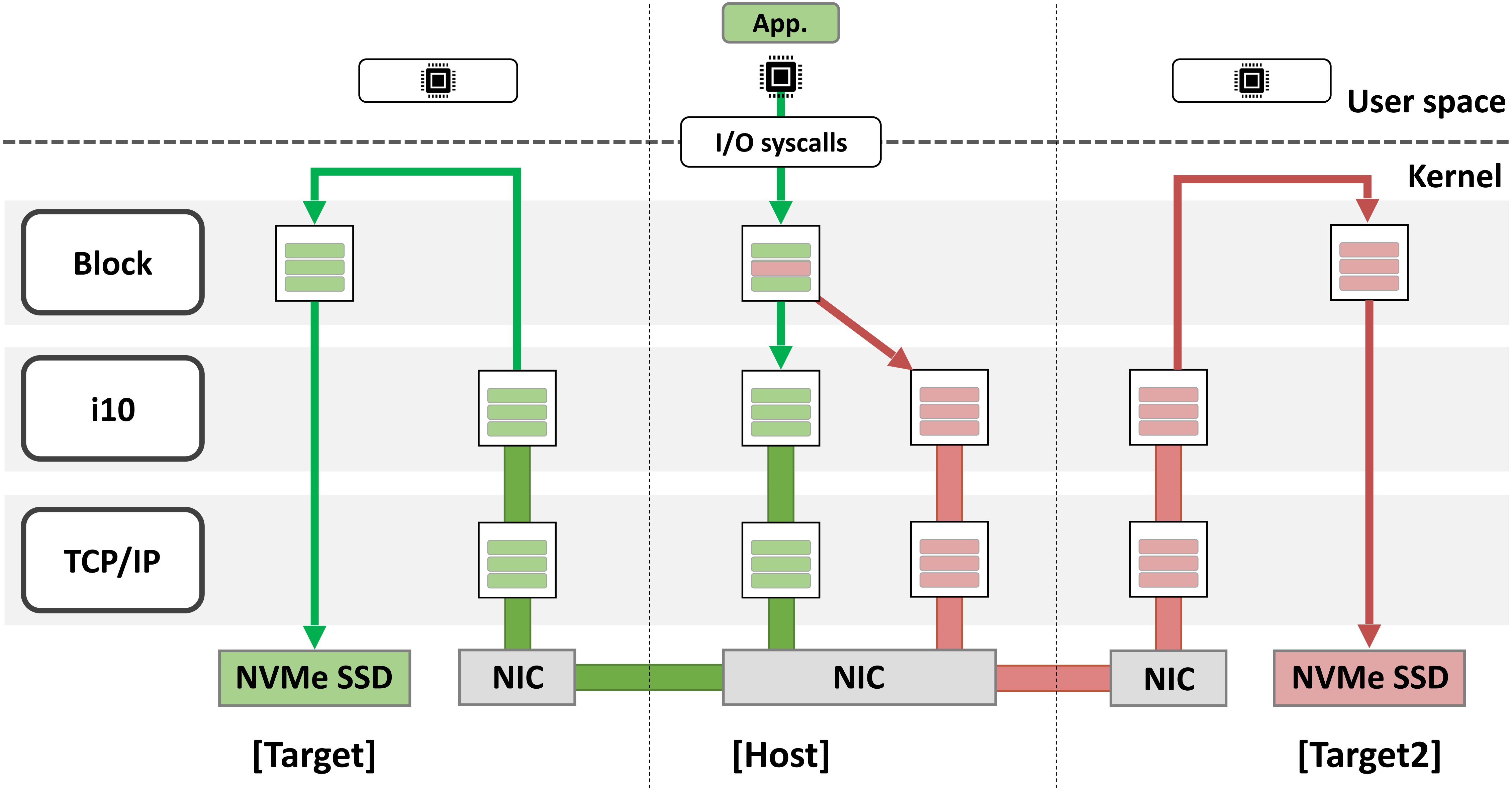
Minimize network processing

Minimize context switching

i10-lane: Dedicated per-(core, target) pair "pipe"

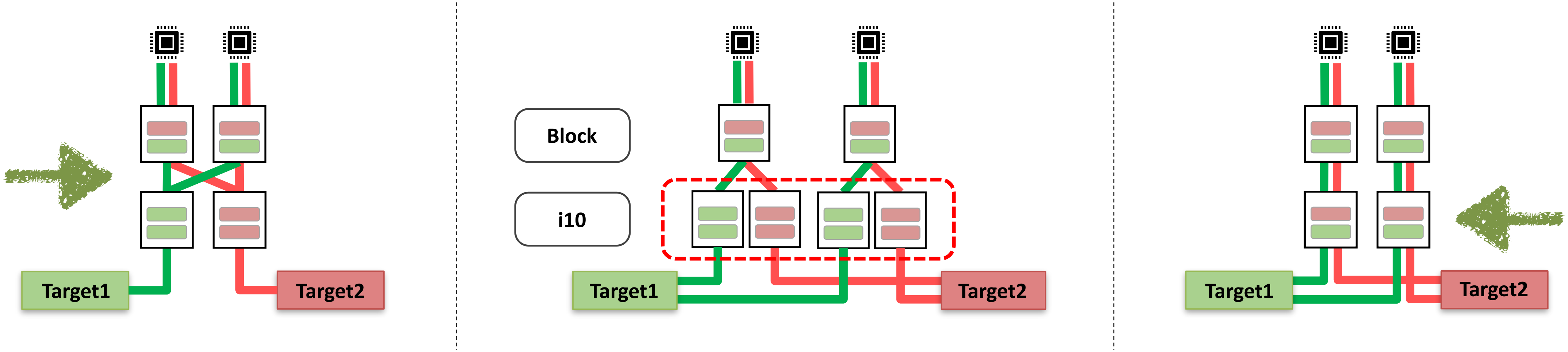


i10-lane: Dedicated per-(core, target) pair "pipe"



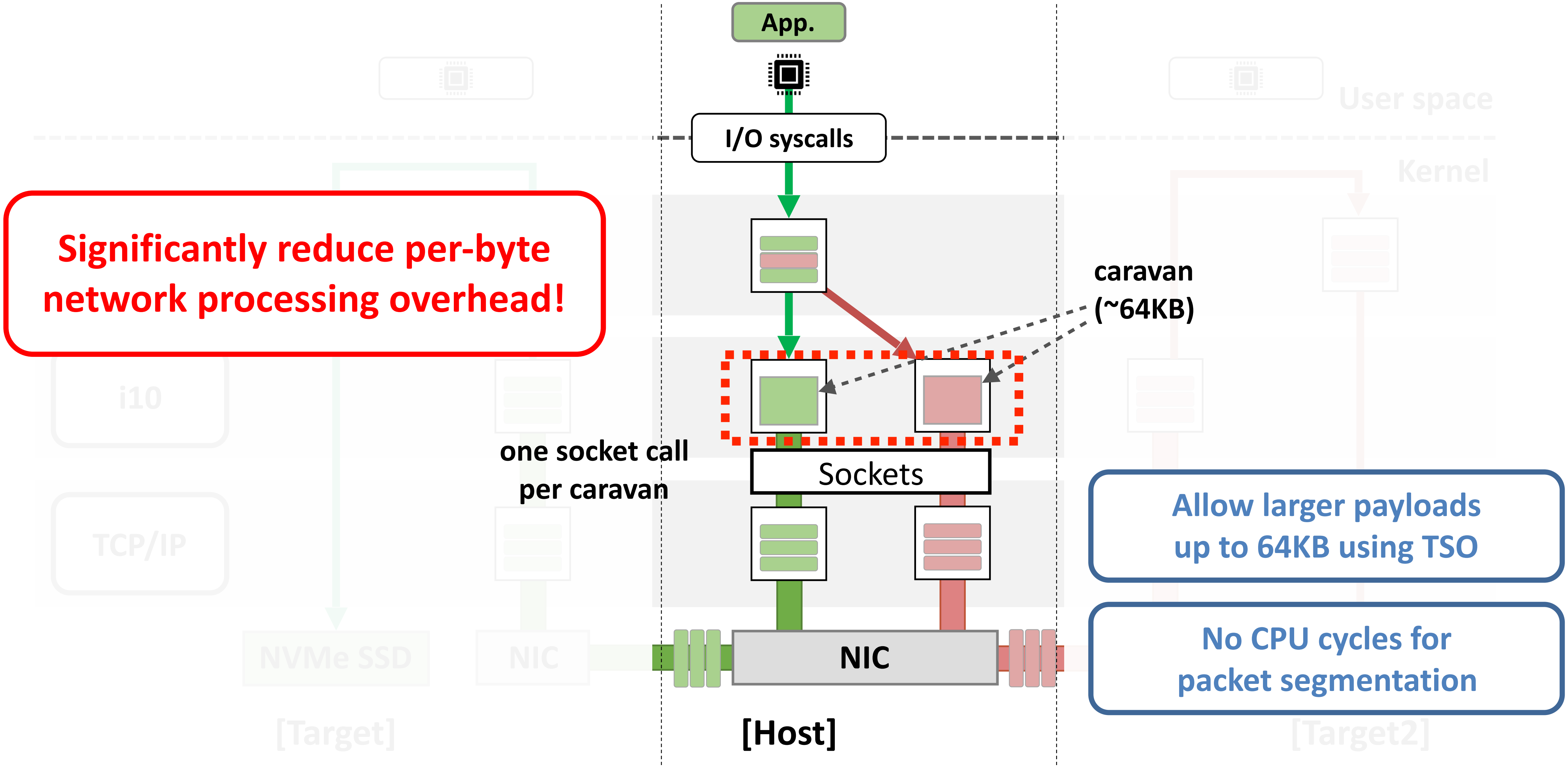
i10-lane: Why per-(core, target) pair lanes?

- **Per-target:** Too much contention
- **Per-core:** Fewer batching opportunities

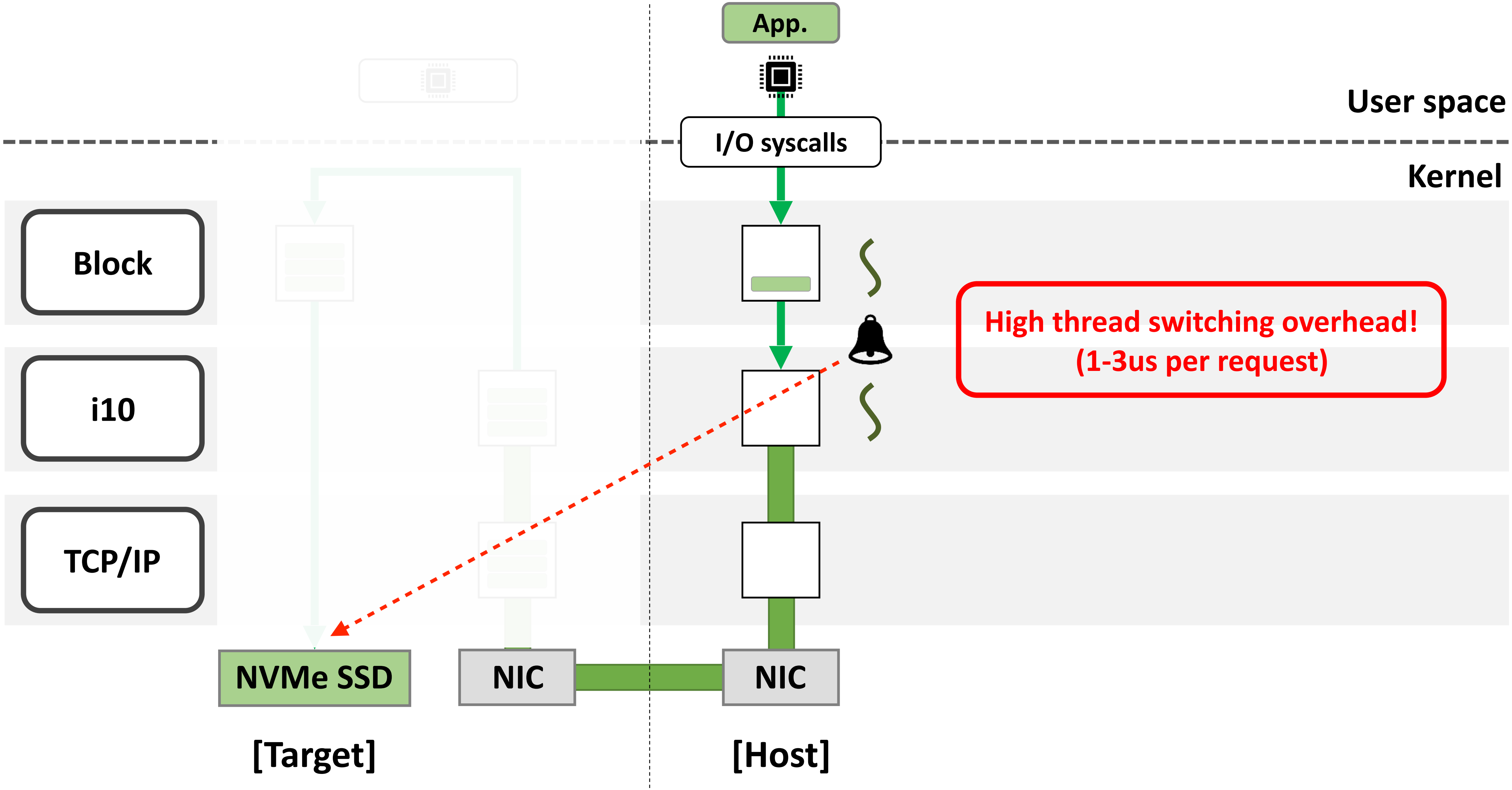


All requests in each i10 queue are destined to the same target over the same TCP connection

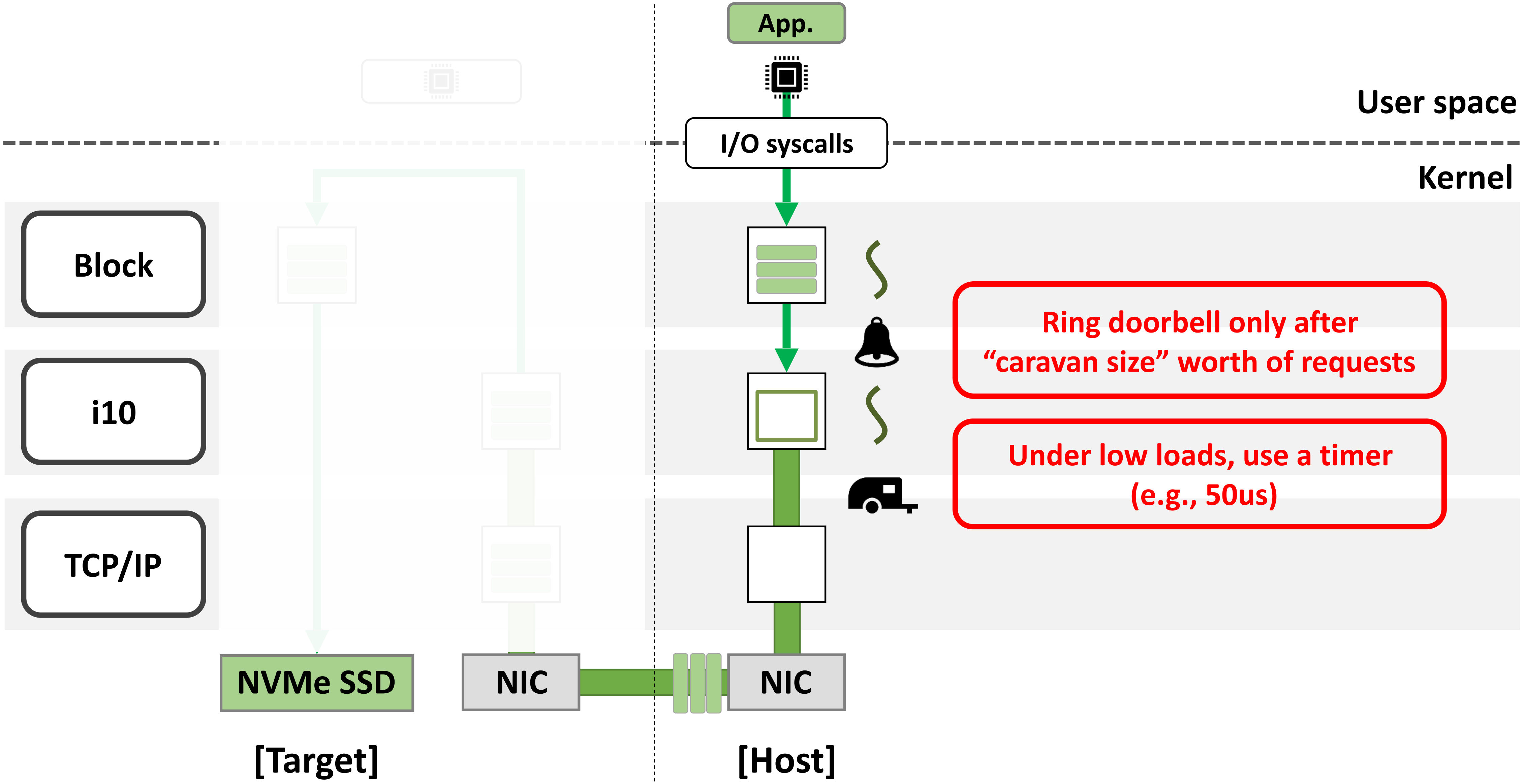
i10 Caravans: i10-lanes enable efficient batching



Context switching in Remote I/O (without i10)



Delayed doorbells: Minimizing context switching



i10 Evaluation Setup

- **Two 24-core servers connected directly**
  - **100Gbps Mellanox CX-5**
  - No switches in middle — ensure bottlenecks in the kernel
- **NVMe-device at both servers**
  - ~700k IOPS (read), ~400k IOPS (write)
  - ~100us read latency
- **No specialized hardware functionalities used in i10 evaluation**
  - For hardware and software configuration, see the paper.

i10 Evaluation: how does i10 performance …

- … compare to NVMe-over-RDMA?
  - Metrics of interest: throughput per core, average latency, tail latency
- … compare to user-space stacks?
- … vary with different workloads, hardware and applications?
  - read/write ratios
  - Delayed doorbell timeout
  - Aggregation size
  - I/O Request sizes
  - Storage device access latency
  - Real applications
  - Number of target devices
- … scale with number of cores?
- … depend on various design aspects (lanes, caravans, delayed doorbell)?

Please see our paper!

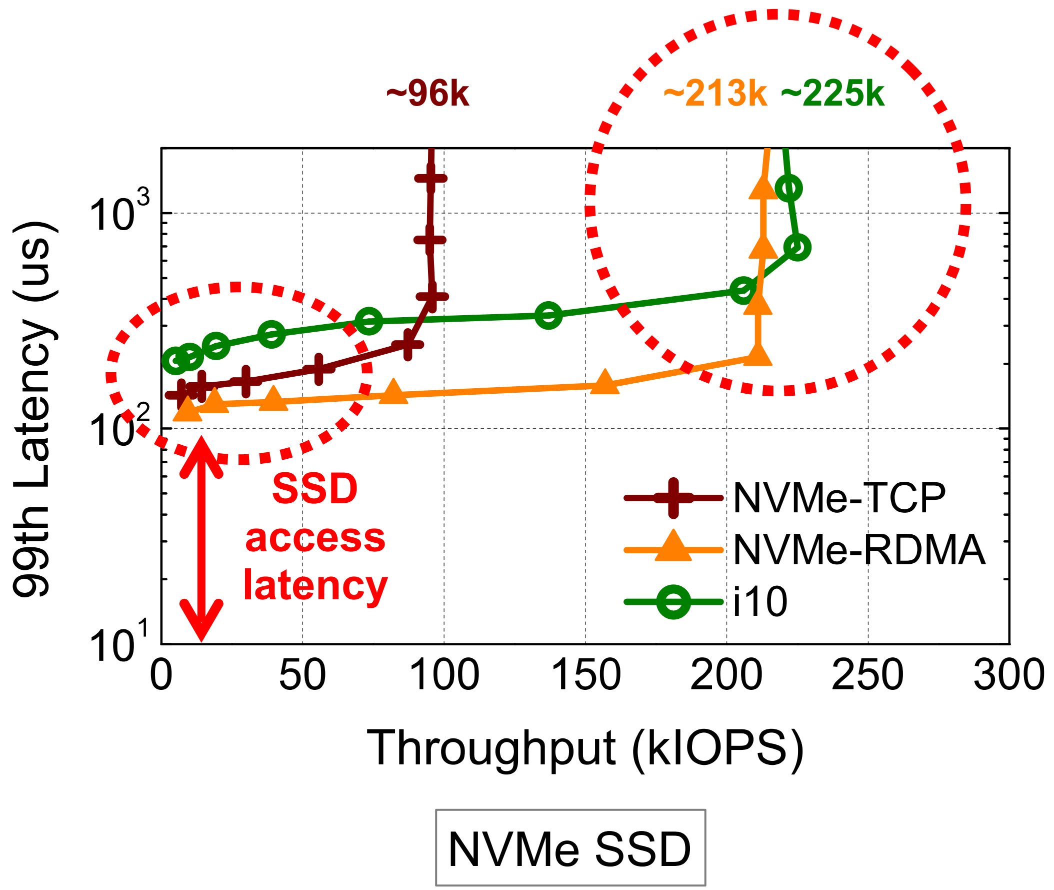
- **What do we expect from TCP ≈ RDMA ?**

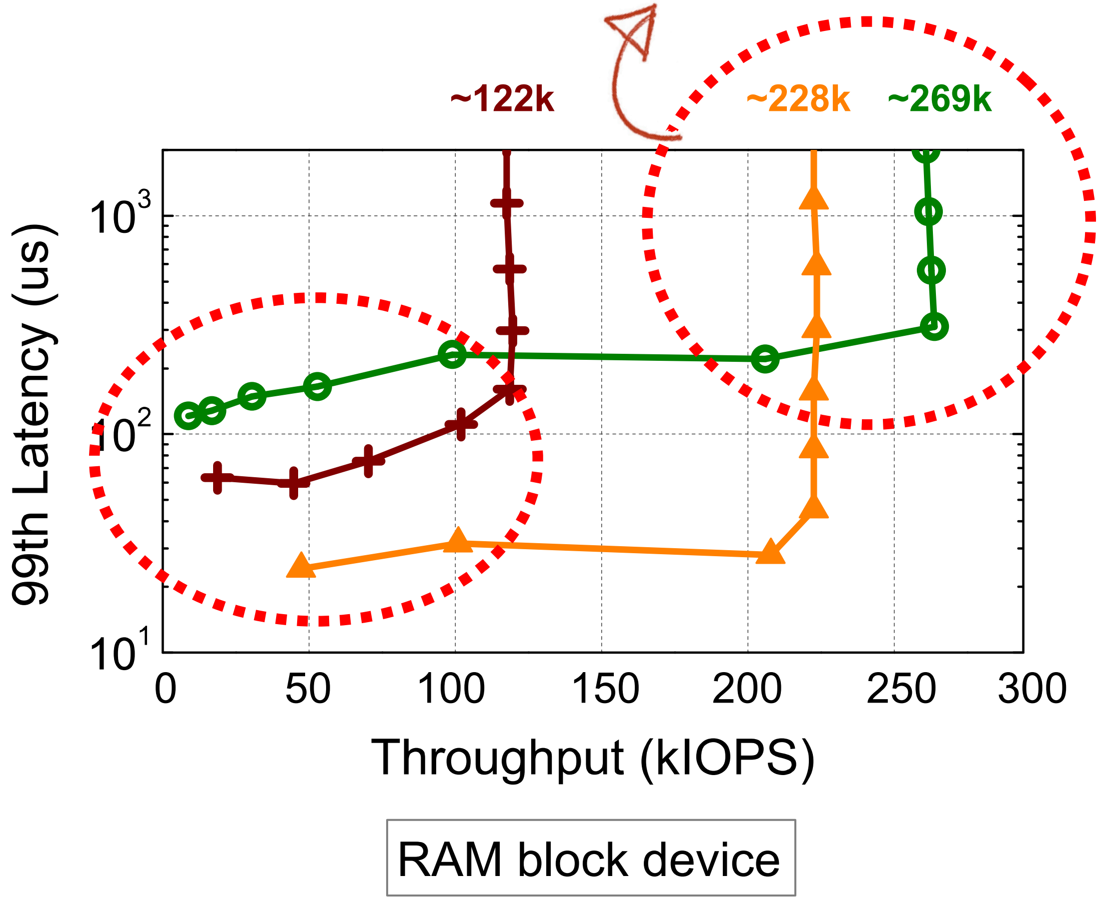
is the best case performance?

	Throughput	Latency
Answer:	Comparable (or better)	Not terrible (<1.7X)

Single core performance

High load latency: TCP ≈ RDMA





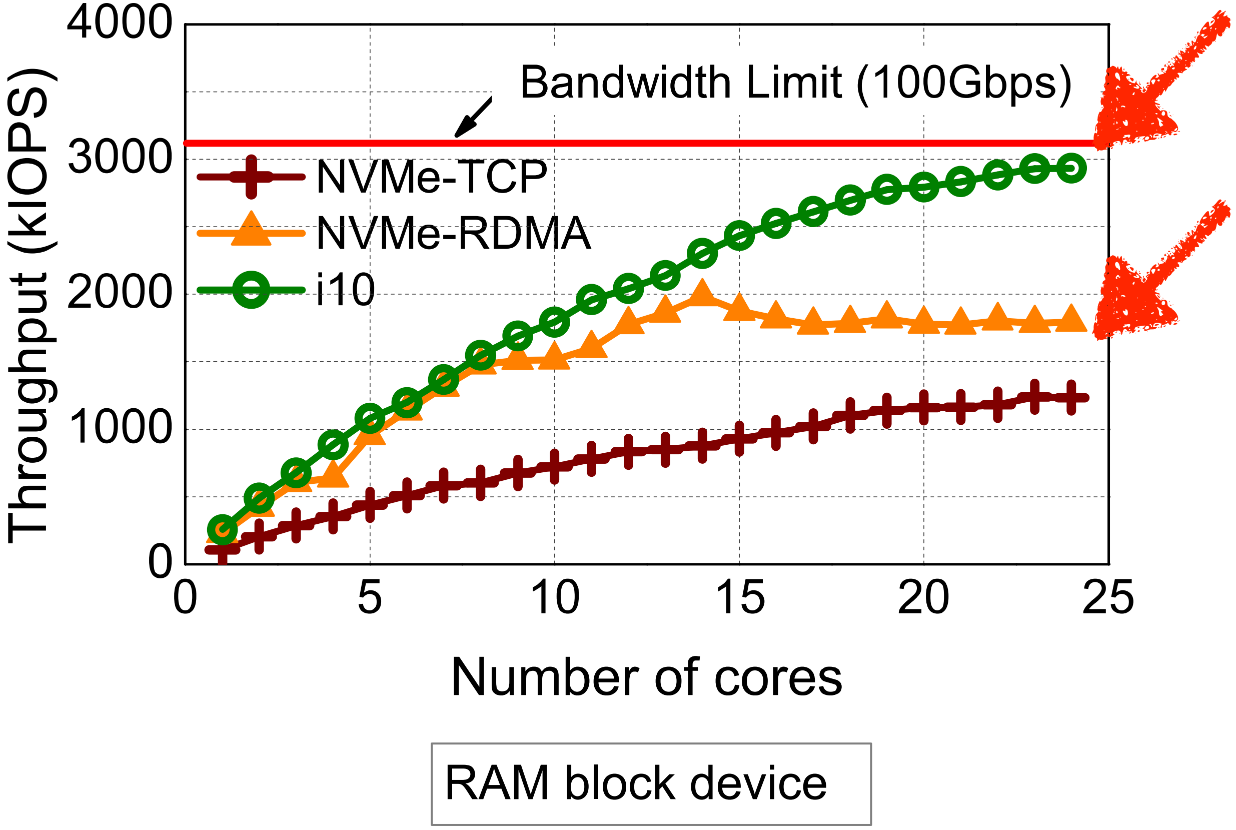
TCP ≈ RDMA:

- Throughput: **Comparable**
- Tail latency: **<1.7X**

TCP ≈ RDMA:

- Throughput: **Comparable (or better)**
- Tail latency: +97us

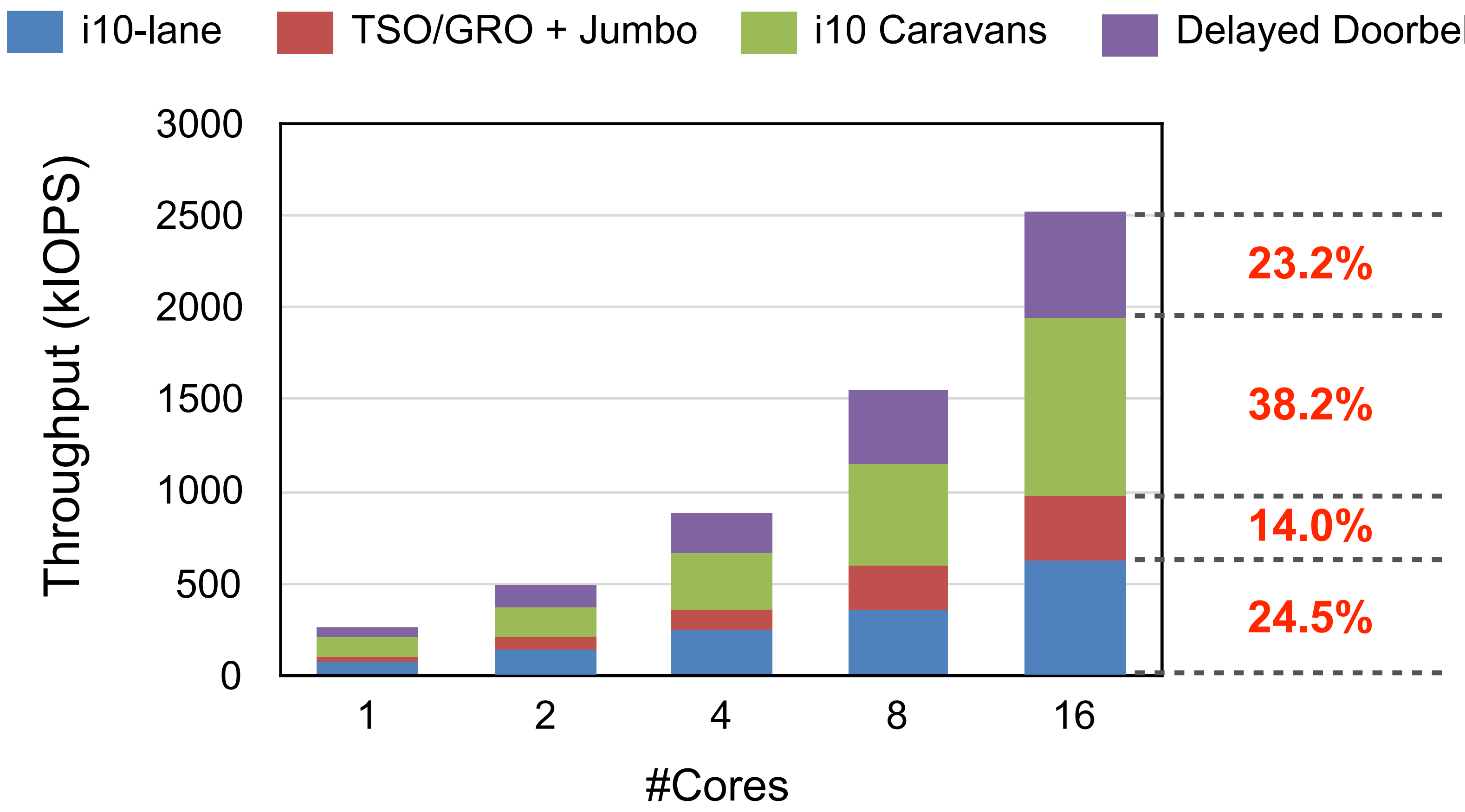
Scalability with number of cores



TCP ≈ RDMA:

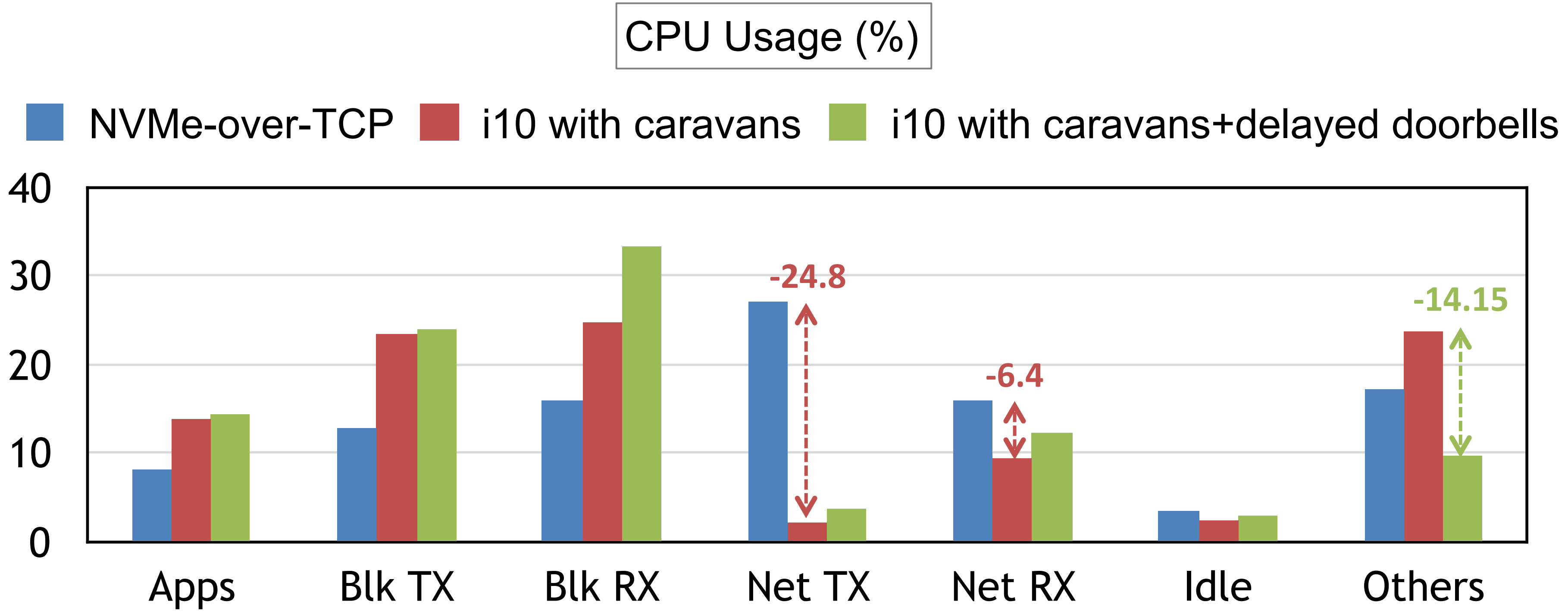
- Throughput: **Scales similar (~14 cores) or better**
- Seems related to hardware scalability

Benefits from individual design components



Each of the design component contributes to i10 performance

Understanding performance improvement

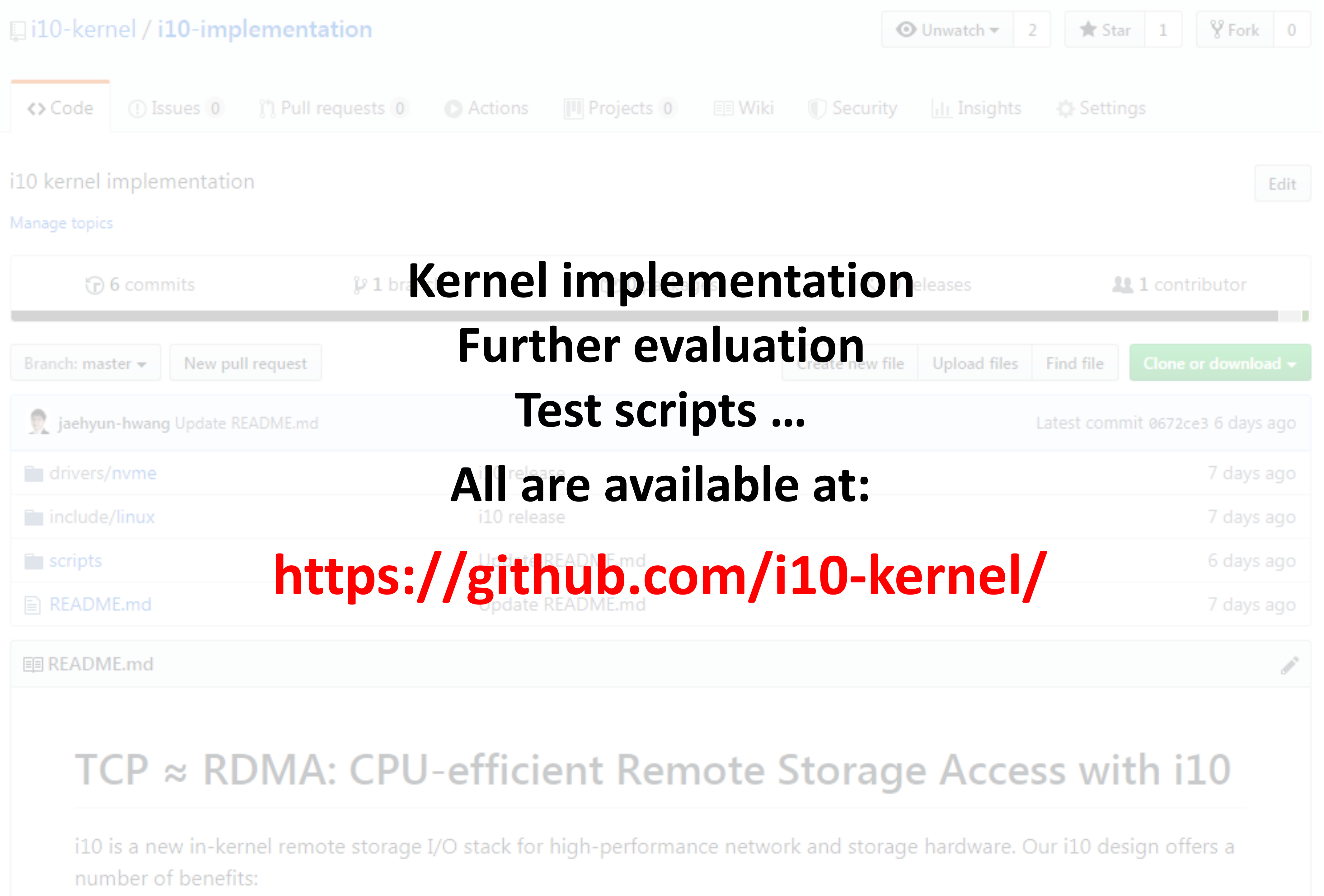


(scheduling overhead, etc.)

i10 improves over NVMe-over-TCP by using

Fewer cycles for network processing (Net Tx/Rx) and scheduling (Others)

More cycles for Applications, and block layer operations (Blk Tx/Rx)



Kernel implementation

Further evaluation

Test scripts …

All are available at:

https://github.com/i10-kernel/