Remote TCP Connection Offload and Applications

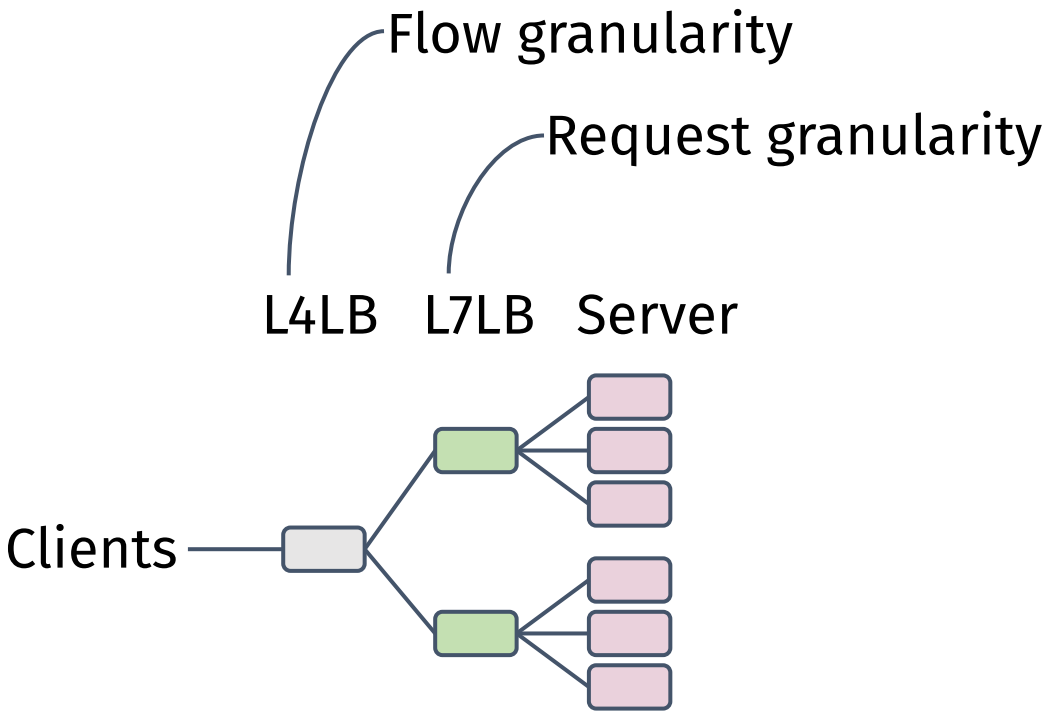
Shuo Li*, Steven W. D. Chien*, Tianyi Gao, Michio Honda
University of Edinburgh
4 May 2026, USENIX NSDI, Renton, WA

THE UNIVERSITY *of* EDINBURGH
informatics

*Joint first authors

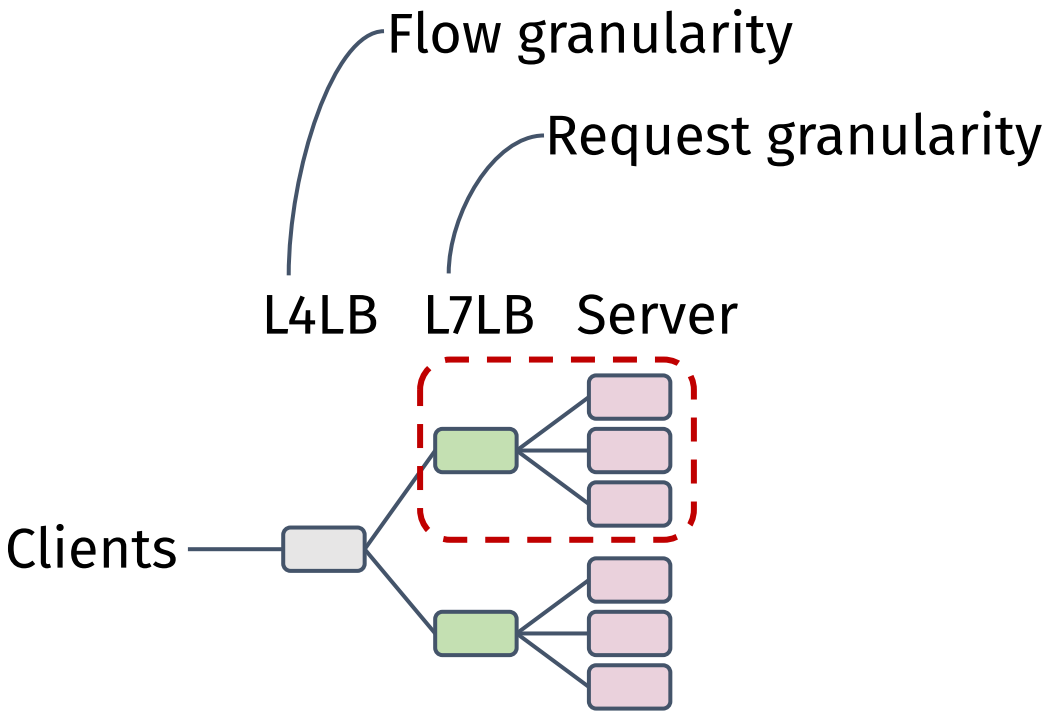
Overview

- Load balancing in datacenters



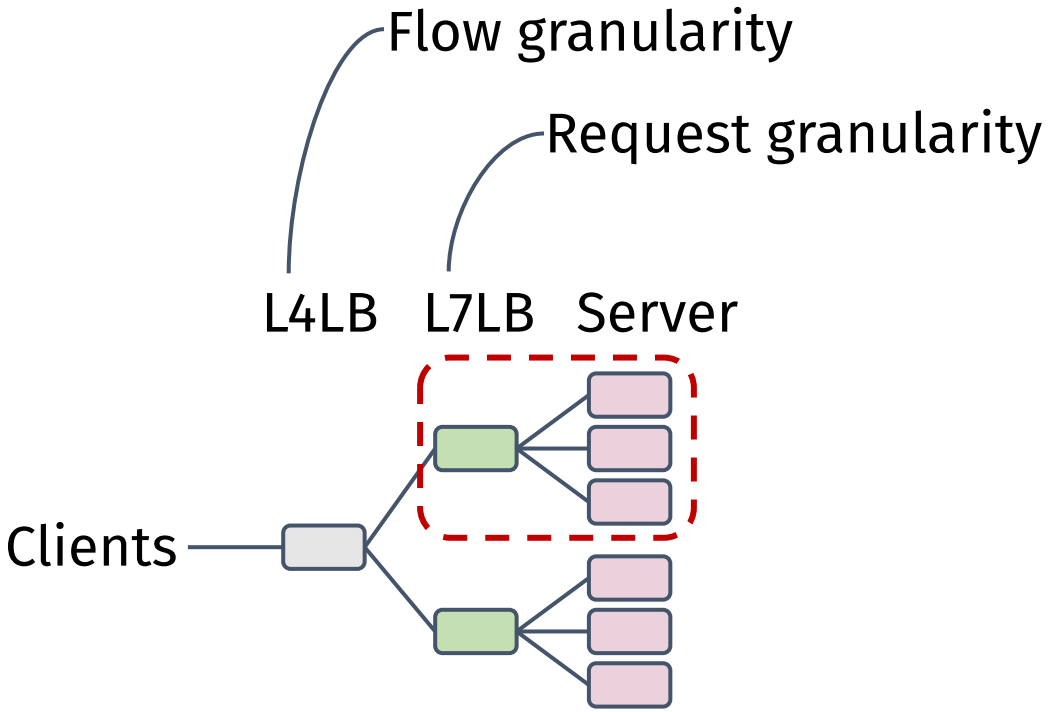
Overview

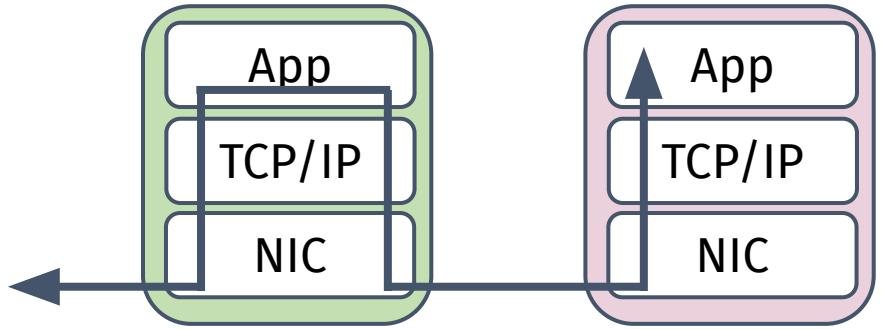
- Load balancing in datacenters



Overview

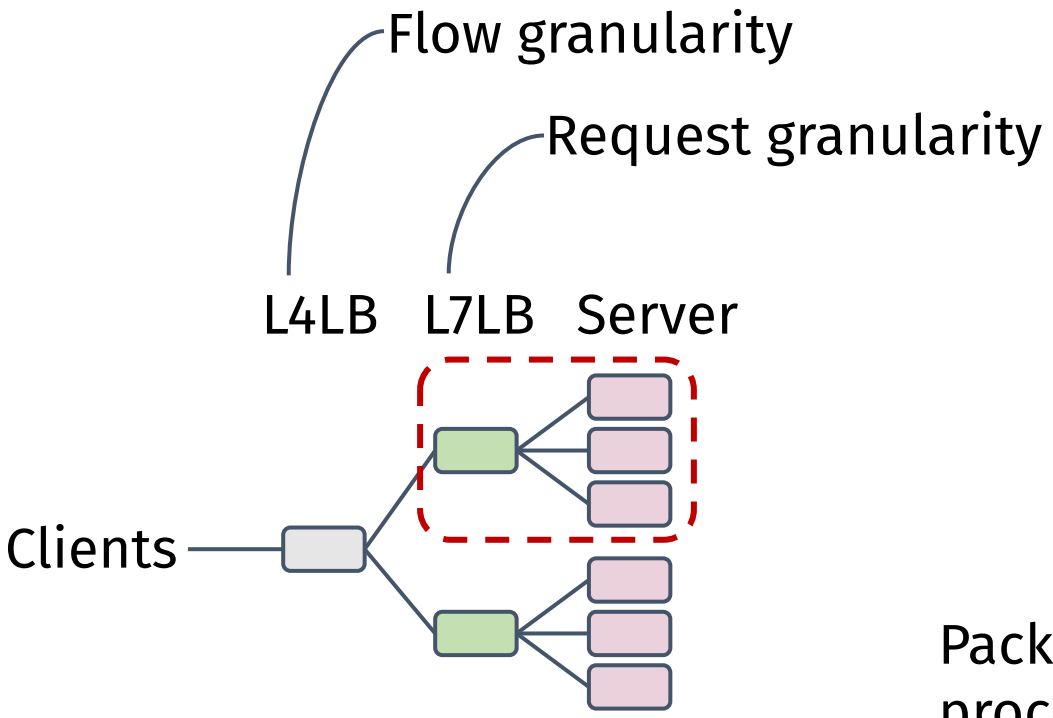
- Load balancing in datacenters
- L7LB datapath



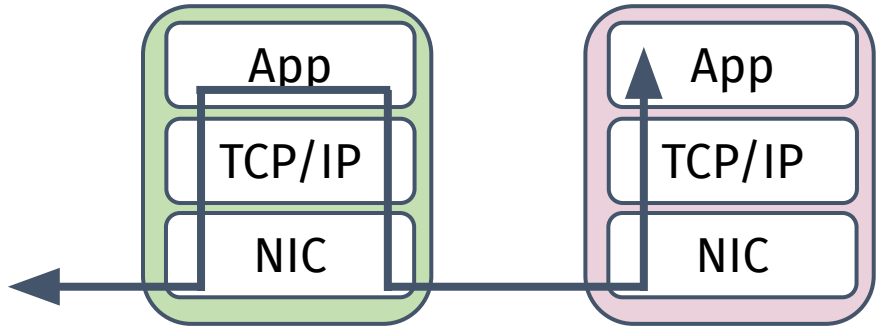


Overview

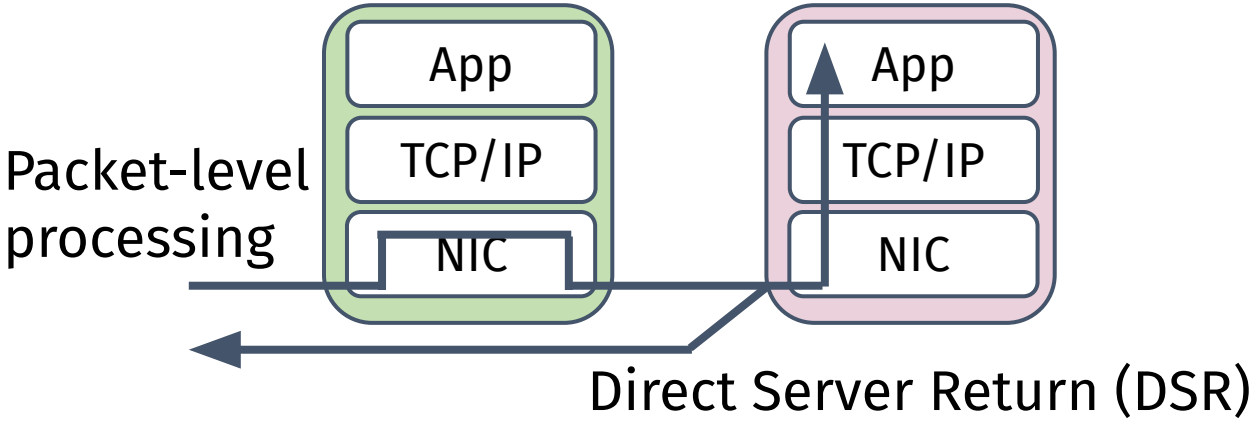
- Load balancing in datacenters



- L7LB datapath



- **XO** datapath



Existing L7LB enhancements

- Flow steering at a programmable switch [1, 2]
  - Hard to deploy

[1] Prism, NSDI'21
[2] Capybara, APSys'23

Existing L7LB enhancements

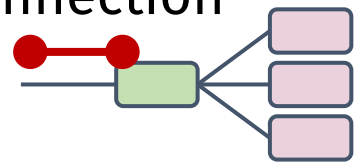
- Flow steering at a programmable switch [1, 2]
  - Hard to deploy
- No Direct Server Retern (DSR) [3]
  - Inefficient
- No TLS support [4]
  - Impractical
- No real app integration [1, 2, 3]
  - Hard to use

[1] Prism, NSDI'21
[2] Capybara, APSys'23
[3] DISC, NSDI'25
[4] Miresga, WWW'25

XO in-action

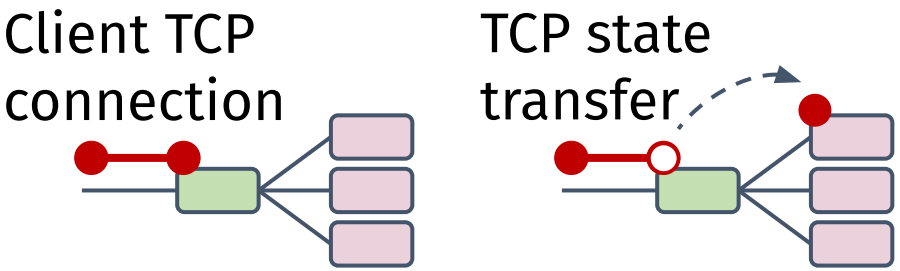
- L7LB accepts a client connection
- Read the client request

Client TCP
connection



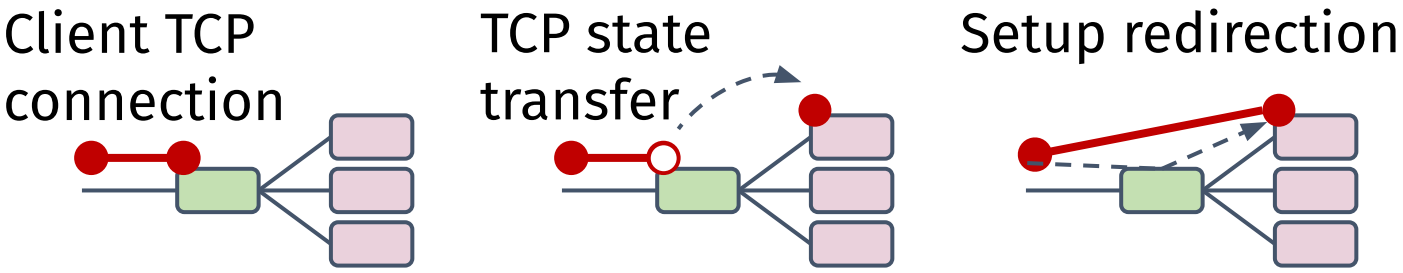
XO in-action

- L7LB accepts a client connection
- Read the client request
- Choose the backend server
- L7LB migrates the connection to the server



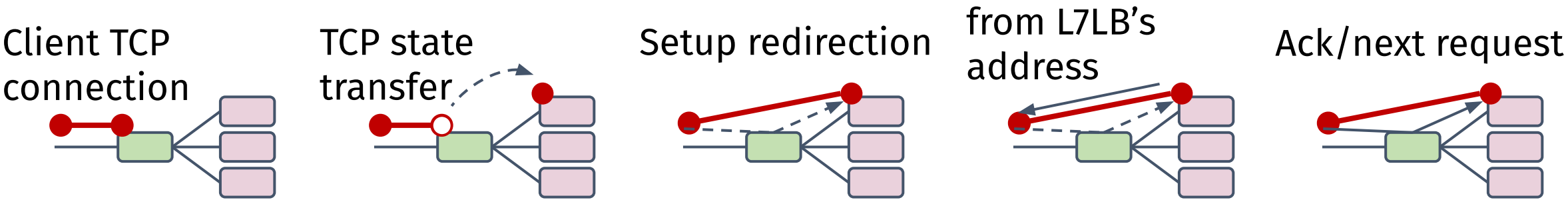
XO in-action

- L7LB accepts a client connection
- Read the client request
- Choose the backend server
- L7LB migrates the connection to the server
- L7LB's NIC or packet I/O layer redirects the ingress packets



XO in-action

- L7LB accepts a client connection
- Read the client request
- Choose the backend server
- L7LB migrates the connection to the server
- L7LB's NIC or packet I/O layer redirects the ingress packets
- The server modifies the source address



SW/HW Hybrid packet redirection

- eBPF is fast to setup (good for fast conn. migration) but inefficient
- tc-flower is slow but efficient after migration due to hw offload

	Operation (µs)		Rate (Mpps)		Latency (µs)	
	Insert	Remove	64B	1500B	64B	1500B
eBPF (tc)	4.01	3.77	0.79	0.78	21.06	22.42
eBPF (XDP)	38.31	7.41	6.65	2.07	16.52	18.45
tc (CX5)	476	404	33.01	2.07	8.26	9.89
tc (CX7)	2143	1134	33.08	2.07	8.41	9.97
tc (Agilio)	68	65	22.12	2.07	19.77	20.58

SW/HW Hybrid packet redirection

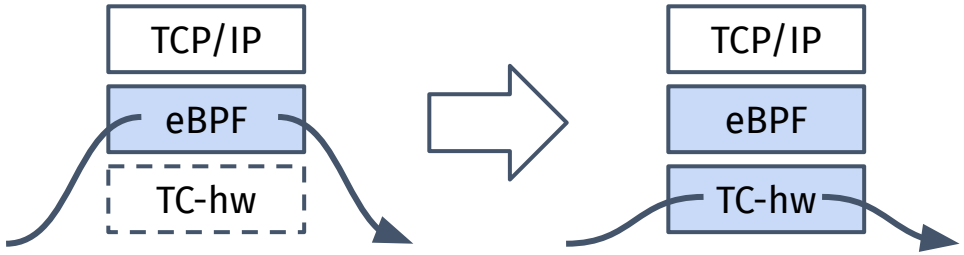
- eBPF is fast to setup (good for fast conn. migration) but inefficient
- tc-flower is slow but efficient after migration due to hw offload

	Operation (µs)		Rate (Mpps)		Latency (µs)	
	Insert	Remove	64B	1500B	64B	1500B
eBPF (tc)	4.01	3.77	0.79	0.78	21.06	22.42
eBPF (XDP)	38.31	7.41	6.65	2.07	16.52	18.45
tc (CX5)	476	404	33.01	2.07	8.26	9.89
tc (CX7)	2143	1134	33.08	2.07	8.41	9.97
tc (Agilio)	68	65	22.12	2.07	19.77	20.58

SW/HW Hybrid packet redirection

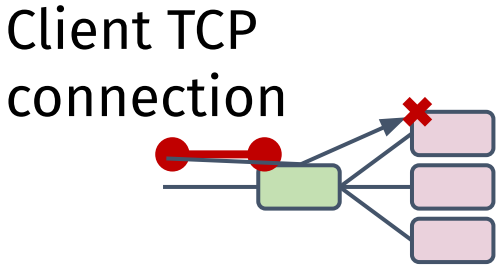
- eBPF is fast to setup (good for fast conn. migration) but inefficient
- tc-flower is slow but efficient after migration due to hw offload
- We do hybrid: *SW until HW activated*
  - Install the equivalent flow rule with both eBPF and tc-flower

	Operation (µs)		Rate (Mpps)		Latency (µs)	
	Insert	Remove	64B	1500B	64B	1500B
`eBPF (tc)`	4.01	3.77	0.79	0.78	21.06	22.42
`eBPF (XDP)`	38.31	7.41	6.65	2.07	16.52	18.45
`tc (CX5)`	476	404	33.01	2.07	8.26	9.89
`tc (CX7)`	2143	1134	33.08	2.07	8.41	9.97
`tc (Agilio)`	68	65	22.12	2.07	19.77	20.58



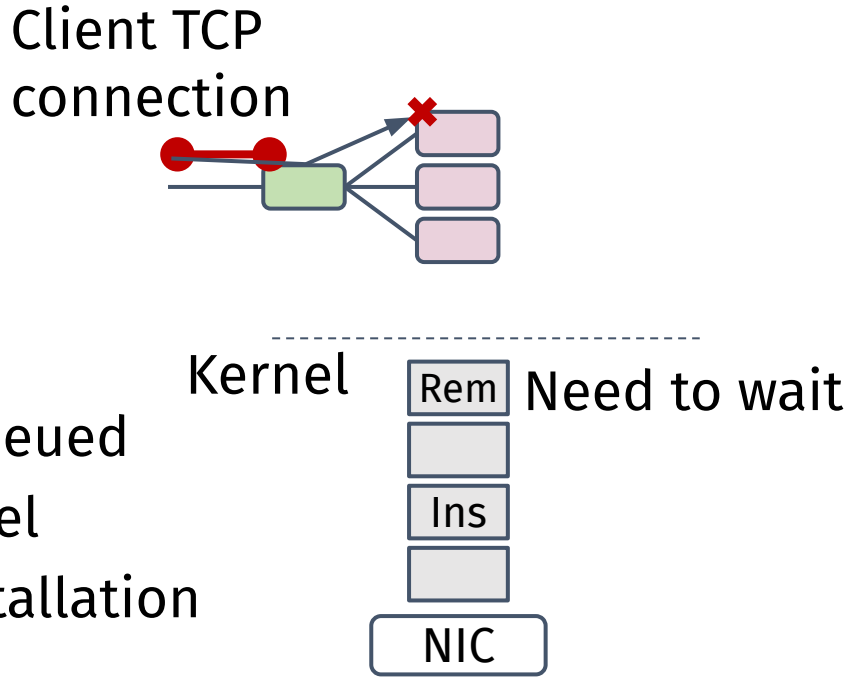
Cancellable queue

- Returning the connection to L7LB needs both SW/HW rule withdrawal



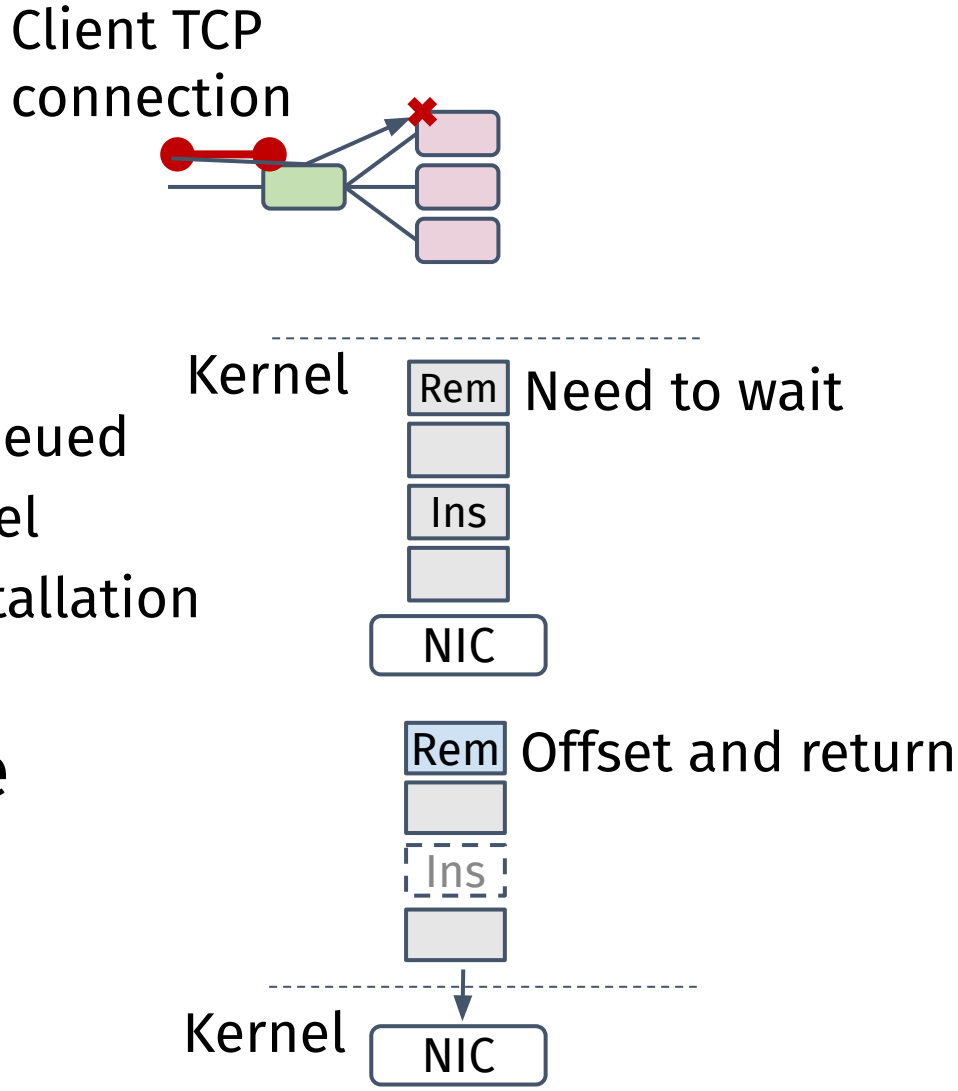
Cancellable queue

- Returning the connection to L7LB needs both SW/HW rule withdrawal
- HW rule withdrawal can take long
  - Rule commands are globally serialized and queued
  - They cannot be cancelled once enter the kernel
  - HW rule withdrawal can happen before its installation (e.g., small single response)



Cancellable queue

- Returning the connection to L7LB needs both SW/HW rule withdrawal
- HW rule withdrawal can take long
  - Rule commands are globally serialized and queued
  - They cannot be cancelled once enter the kernel
  - HW rule withdrawal can happen before its installation (e.g., small single response)
- We move the queue to the user space
  - Offset the pending installation rule

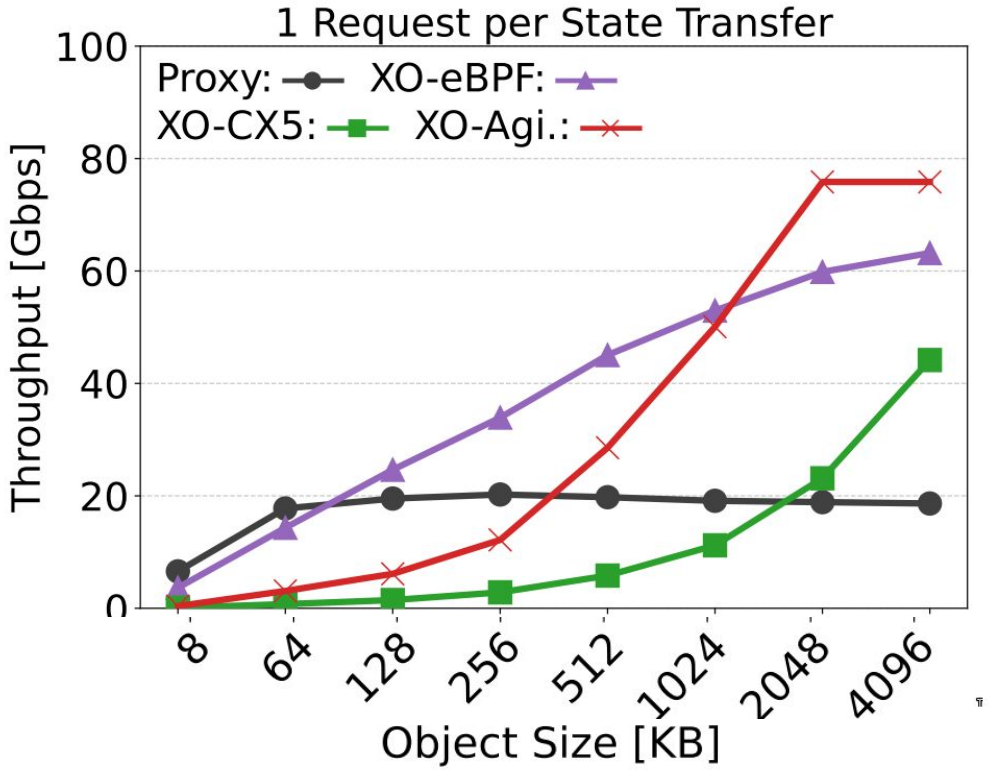


Application porting experience

- XO (or any connection migration-based approaches) needs both L7LB and server app modification - how hard it is?
- Nginx (replicated backend) and Ceph (sharded backend)
- The use of kernel stack eased porting significantly
  - Full file descriptor semantics, threading model etc
  - It’d have been much harder if we opted for kernel-bypass stacks

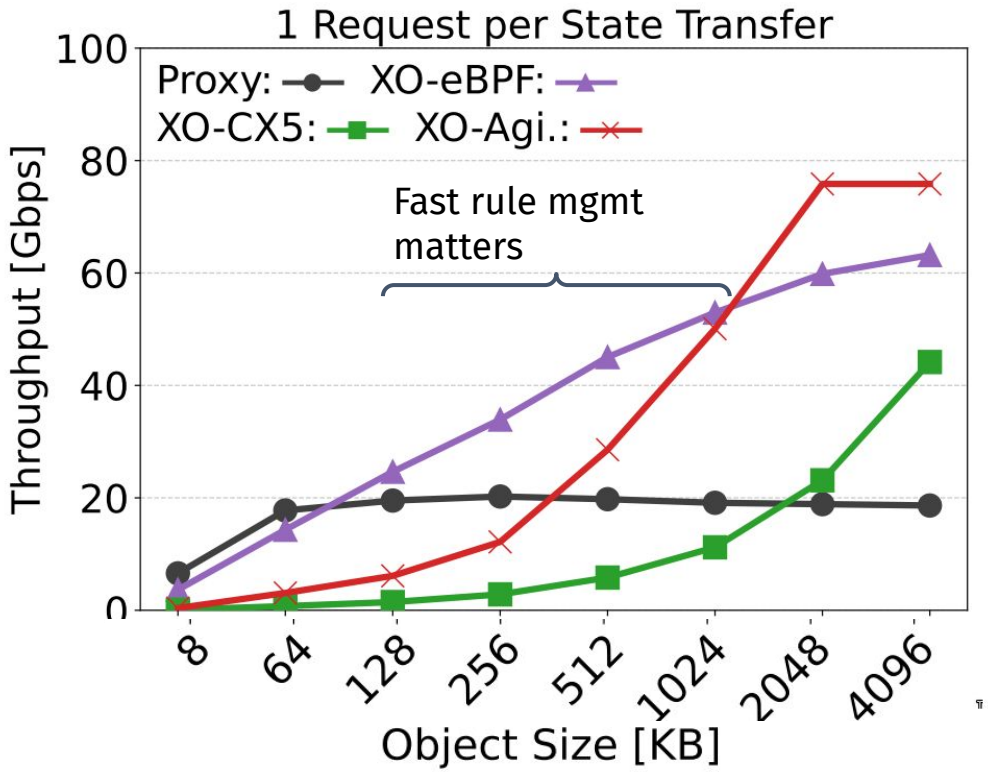
XO throughput

- 25Gbps L7LB and server links
  - Larger objects or less frequent migrations amortize the connection migration costs



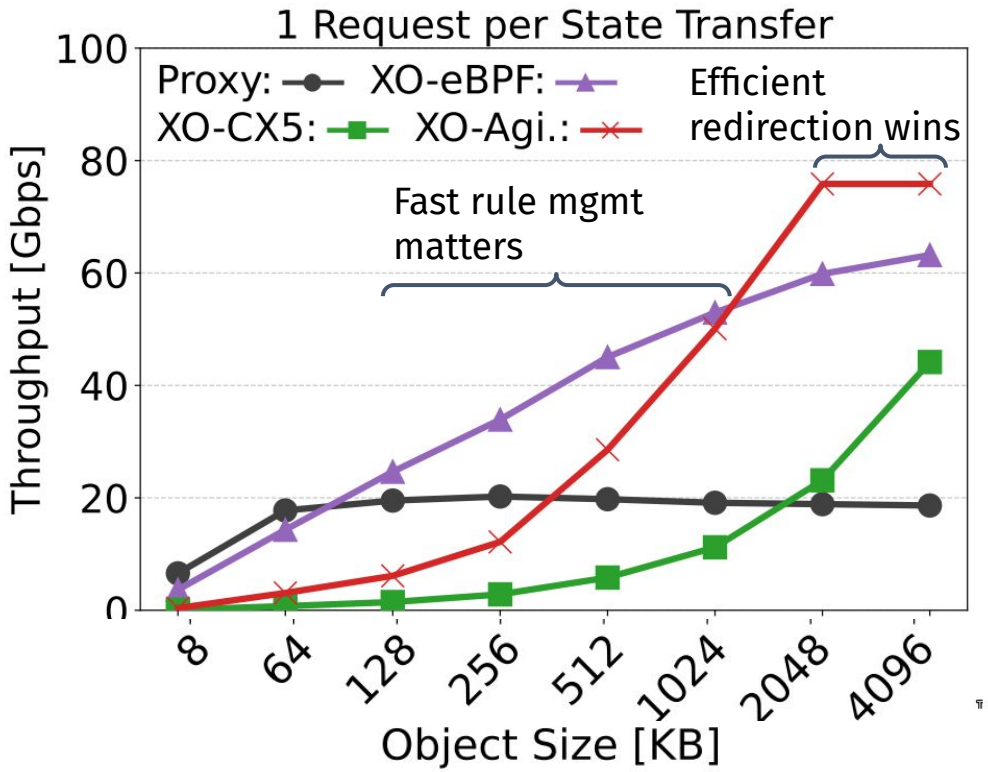
XO throughput

- 25Gbps L7LB and server links
  - Larger objects or less frequent migrations amortize the connection migration costs



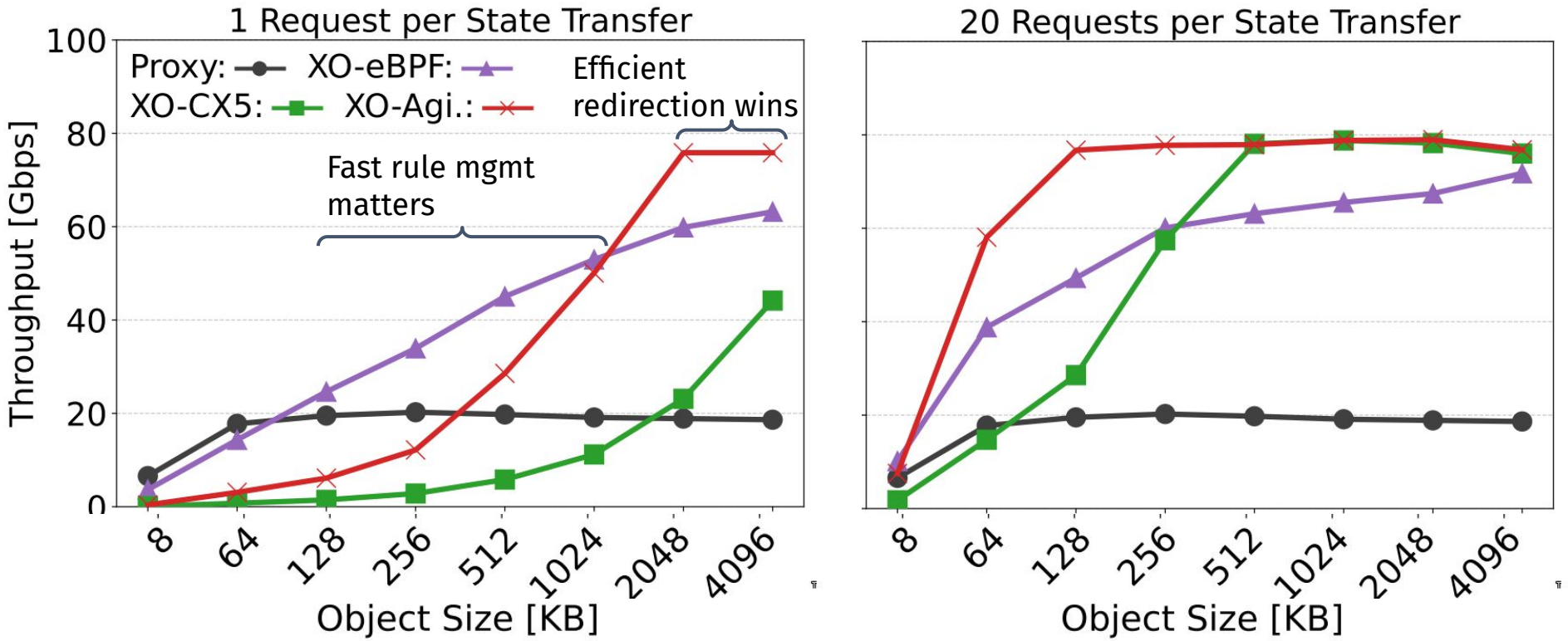
XO throughput

- 25Gbps L7LB and server links
  - Larger objects or less frequent migrations amortize the connection migration costs



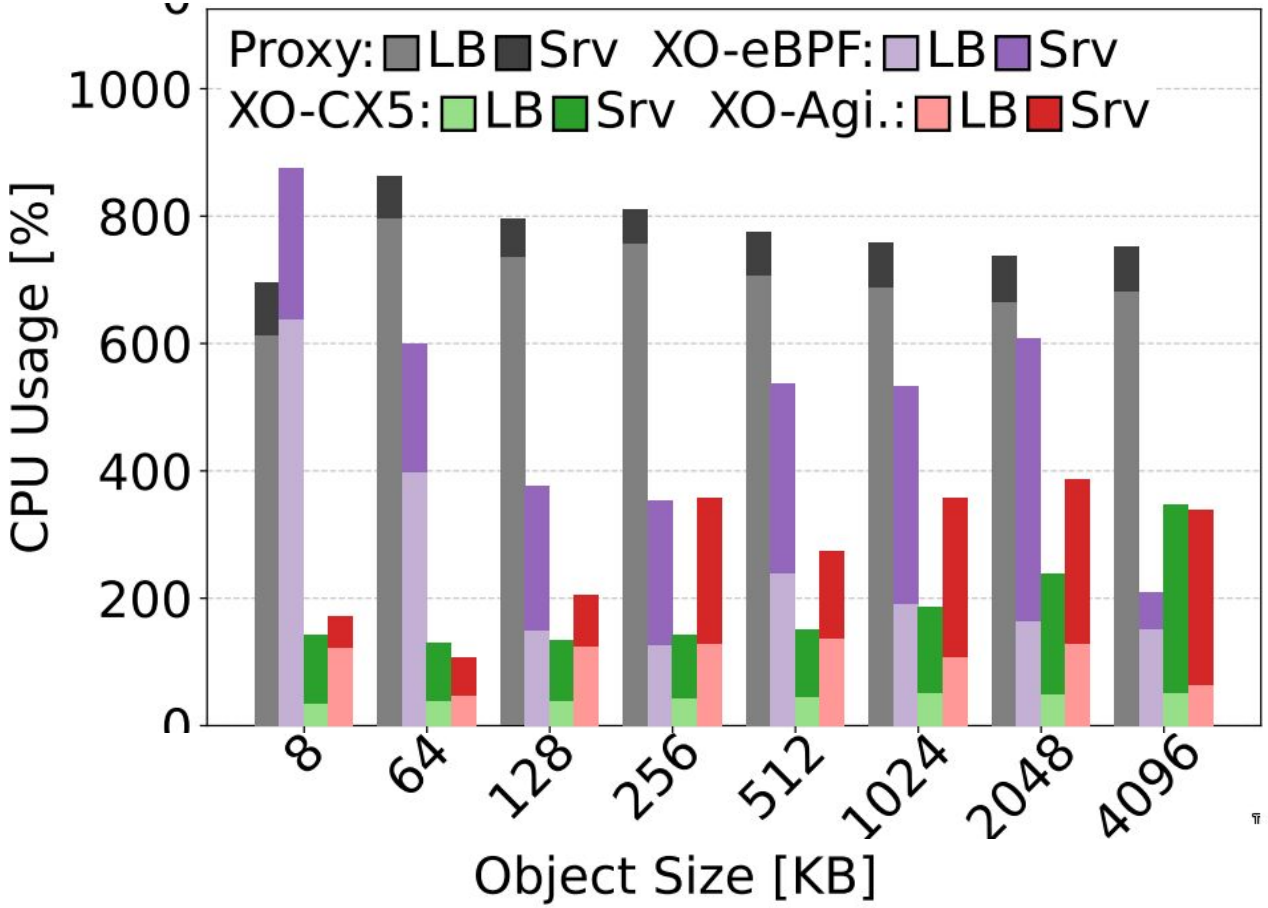
XO throughput

- 25Gbps L7LB and server links
  - Larger objects or less frequent migrations amortize the connection migration costs



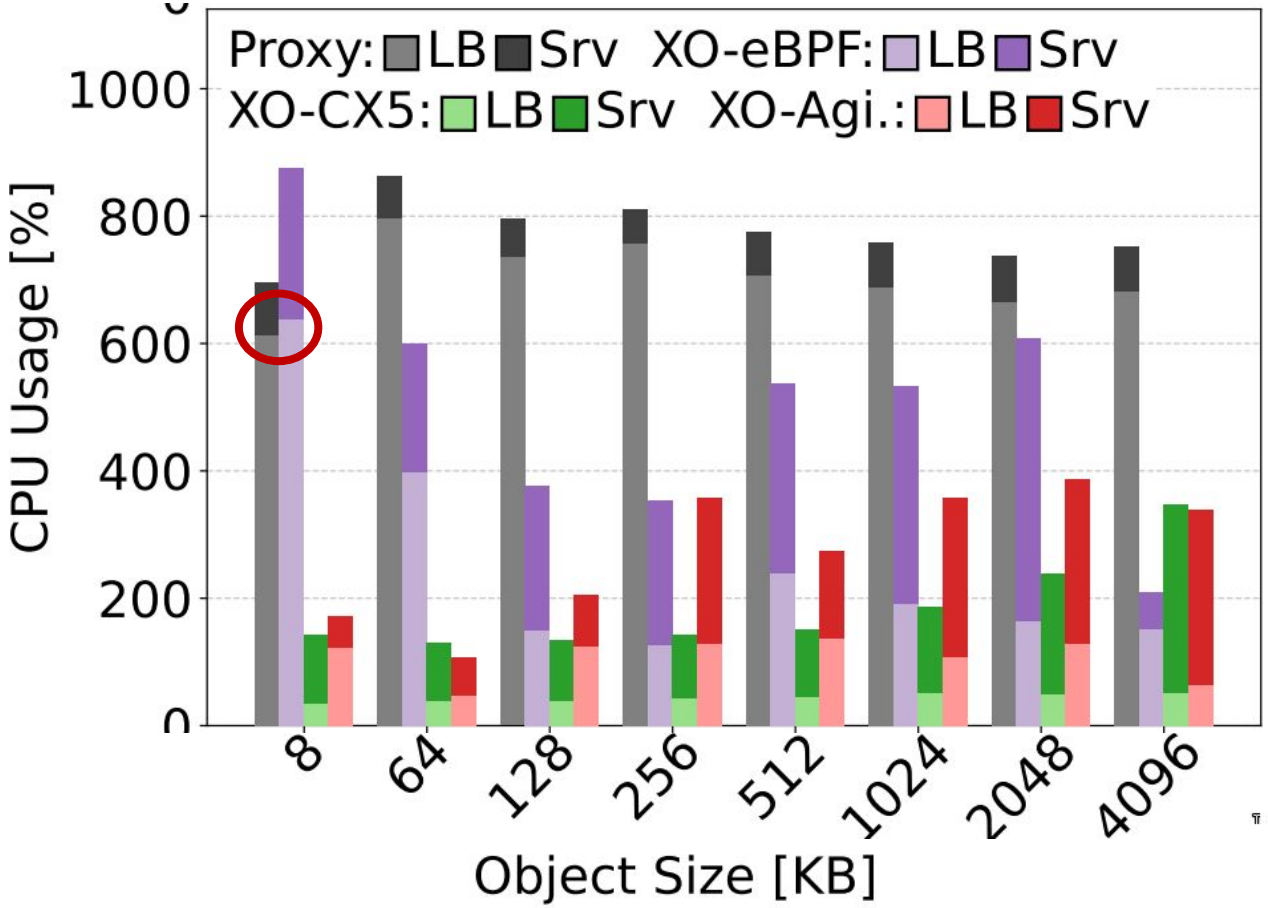
XO CPU usage

- 25Gbps L7LB and server links
  - L7LB CPU usage decreases despite higher throughput



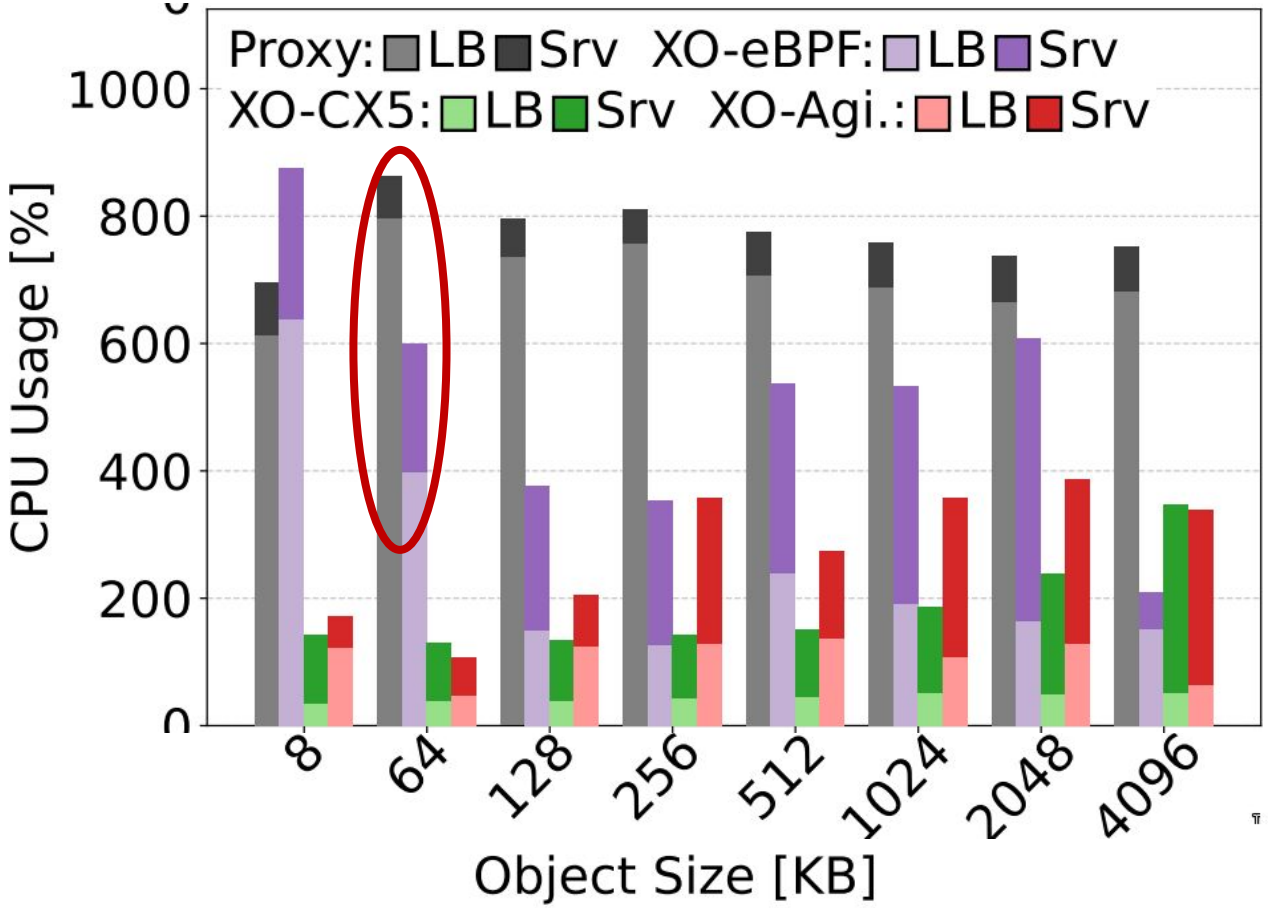
XO CPU usage

- 25Gbps L7LB and server links
  - L7LB CPU usage decreases despite higher throughput



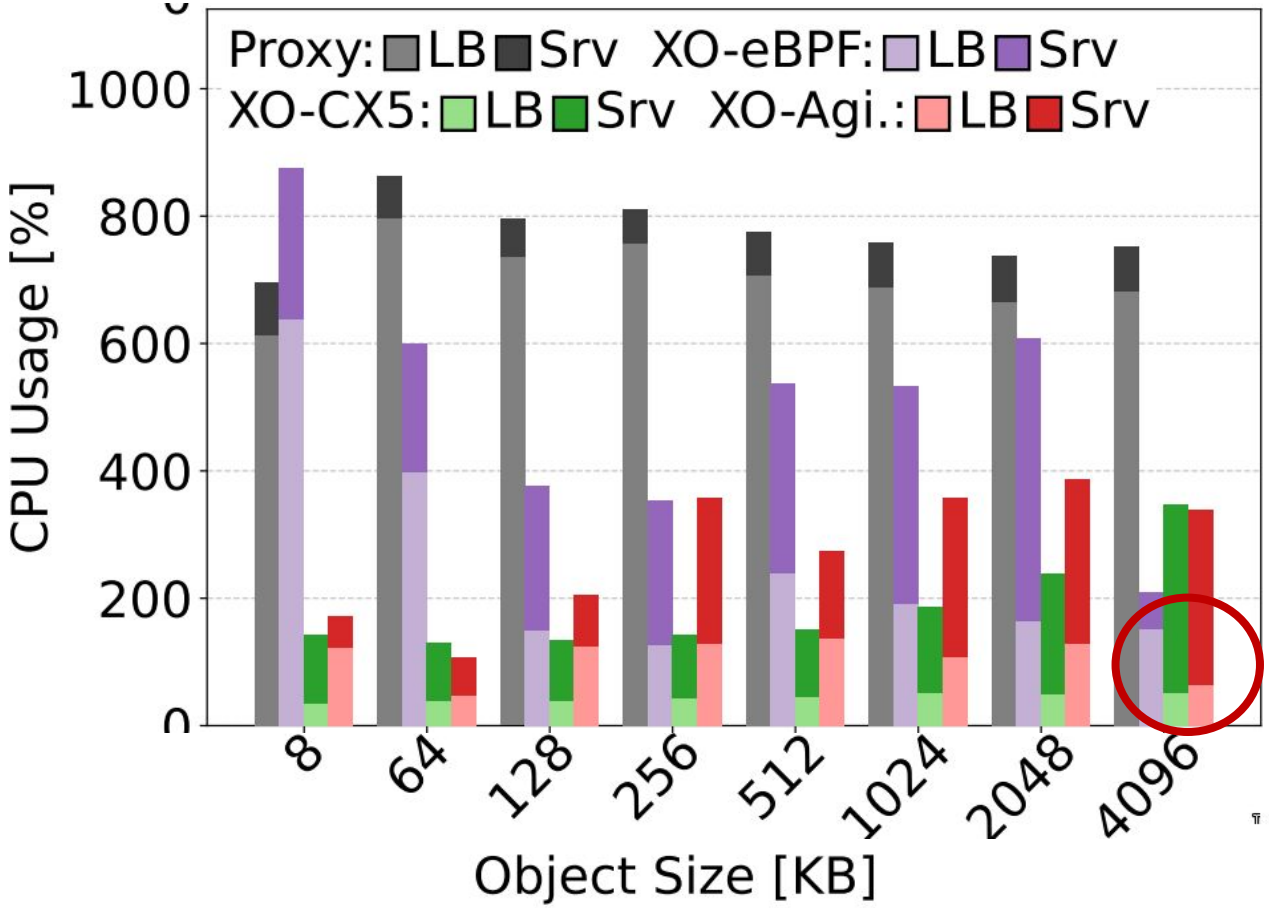
XO CPU usage

- 25Gbps L7LB and server links
  - L7LB CPU usage decreases despite higher throughput



XO CPU usage

- 25Gbps L7LB and server links
  - L7LB CPU usage decreases despite higher throughput



Connection migration latency

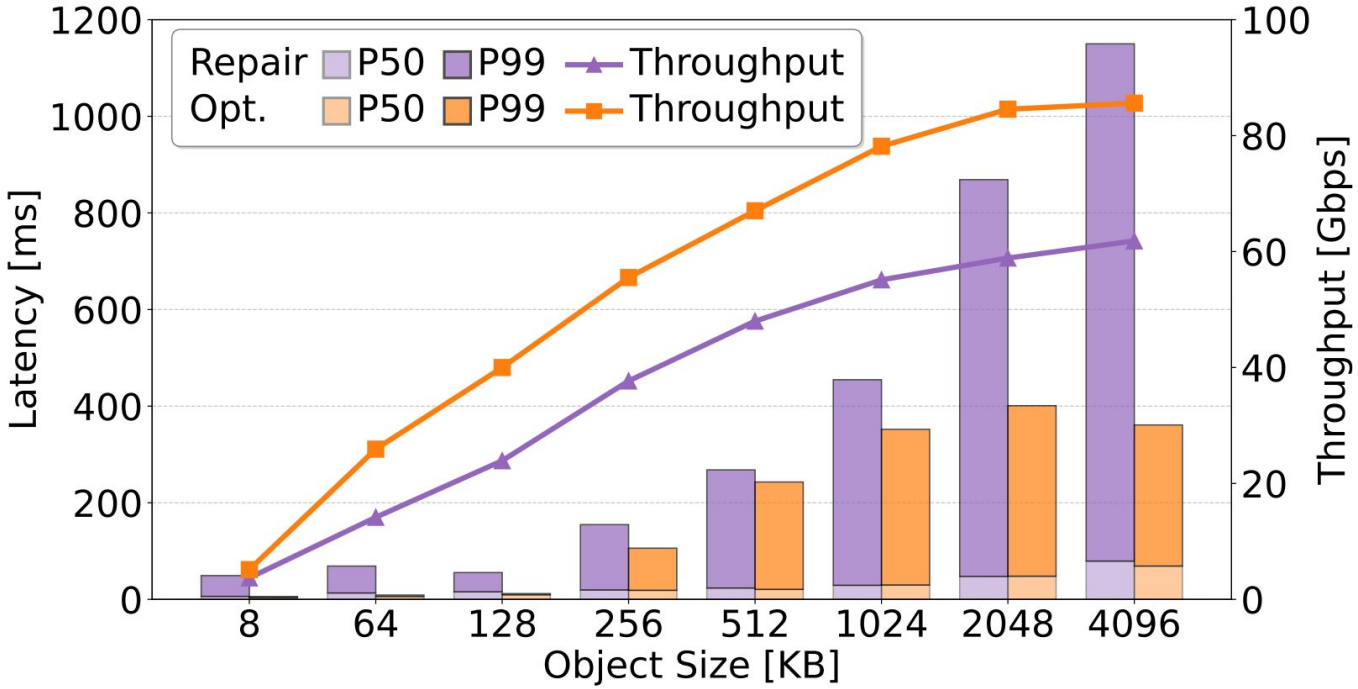
Operation	Latency [µs]	
(L) Block flow	4	
(L) Serialize TLS	2	
(L) Serialize TCP	11	
(L->S)State transferring to the server	148	
(L) Sending RPC: BEGIN_OFFLOAD		
(S) Restore TCP		39
(S) Restore TLS		15
(S) Install source IP rewrite rule		5
(L) Received RPC: SERVER_READY		
(L) Install redirection rule and unblock	4	
(L) Send RPC: L7LB_READY	56	
Total	**225**	

State transfer can be faster (if we modify the stack)

- Connection serialization takes 13 syscalls, each locking the socket

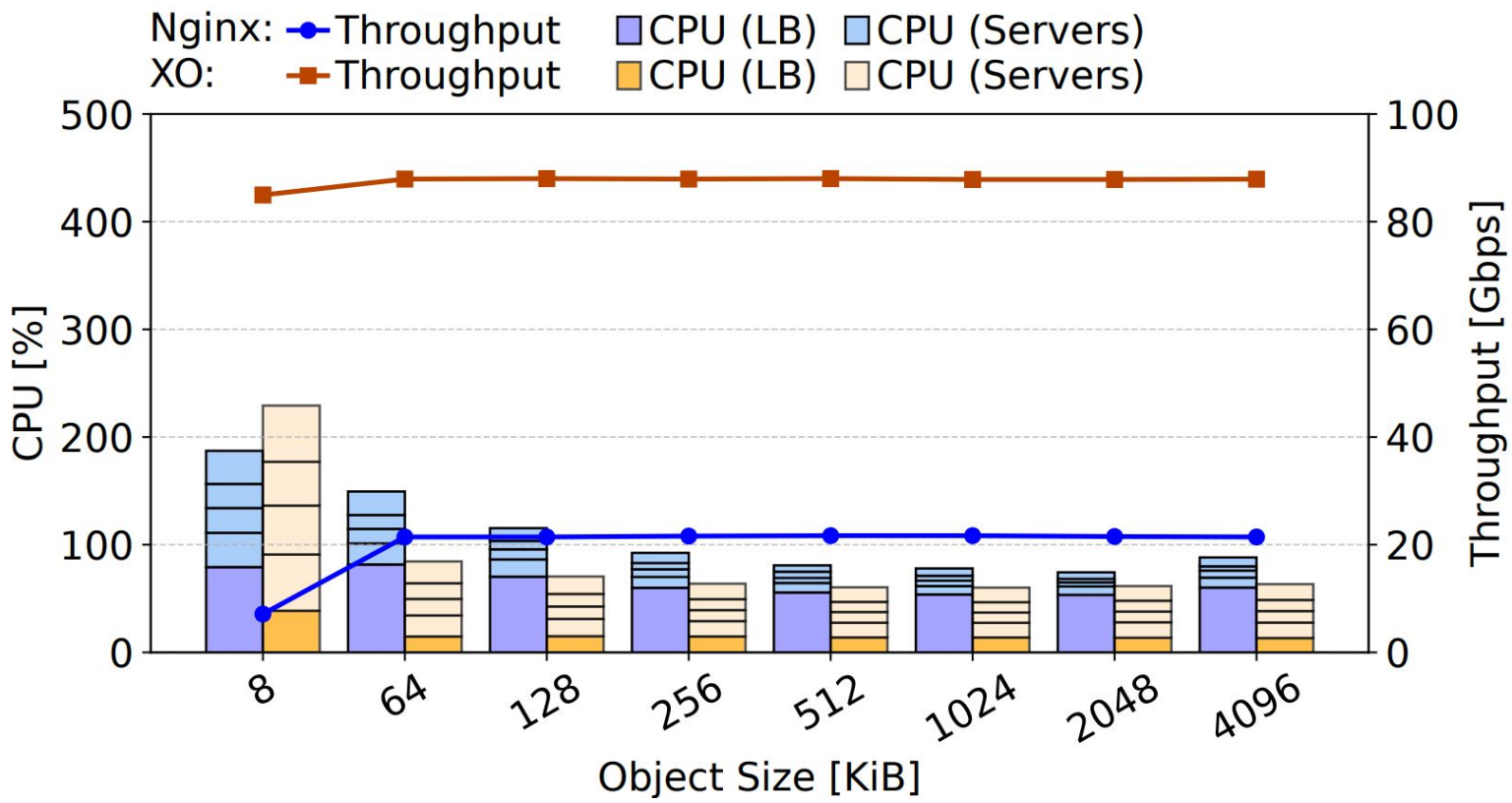
State transfer can be faster (if we modify the stack)

- Connection serialization takes 13 syscalls, each locking the socket
- We implement consolidated serialization or deserialization call to amortise syscall and socket locking costs



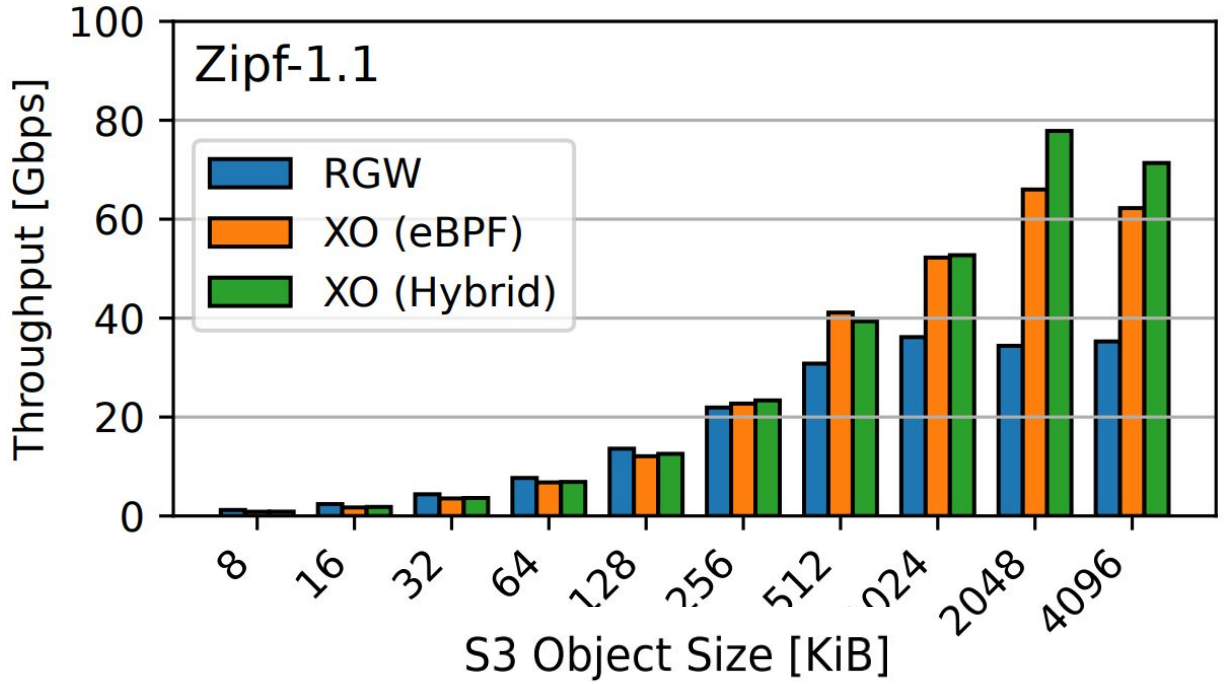
XO-nginx performance

- 25Gbps L7LB and server links
  - Better BW utilization
  - Reduced L7LB CPU usage



XO-Ceph performance

- 100Gbps L7LB and server links
  - Better throughput with 256KiB or larger objects
    - Typical Ceph depoloyments hosts many larger data*



*https://ceph.io/en/news/blog/2025/benchmarking-object-part2/

Discussion

- Virtualized infrastructure
  - We have eBPF-only option

Discussion

- Virtualized infrastructure
  - We have eBPF-only option
- Congestion window, RACK state etc
  - Currently not restored (i.e., slow-start)
  - Future work will (carefully) restore it: Important for long-RTT flows

Discussion

- Virtualized infrastructure
  - We have eBPF-only option
- Congestion window, RACK state etc
  - Currently not restored (i.e., slow-start)
  - Future work will (carefully) restore it: Important for long-RTT flows
- Concurrent requests
  - Packet level buffering/replication as with HA/TCP [NSDI'25]
    - At most awnd per connection
    - Trivial with eBPF

Summary

- XO: Remote TCP Connection Offload (from L7LB to the servers)
  - Ingress: Lightweight packet-level redirection at the host
  - Egress: Direct Server Return
  - Real app integration: Nginx and Ceph
  - Reduced L7LB CPU and BW usage and better server resource utilization
- Details and more results in the paper
- Code: https://github.com/uoenoplab/xo
  - AE confirmed it works with xl170 nodes on Cloudlab