Tigger: A Database Proxy That Bounces With User-Bypass

Matt Butrovich

Karthik Ramanathan, John Rollinson*,

Wan Shen Lim, William Zhang, Justine Sherry, Andrew Pavlo

PARALLEL DATA LABORATORY

Carnegie Mellon University, *Army Cyber Institute

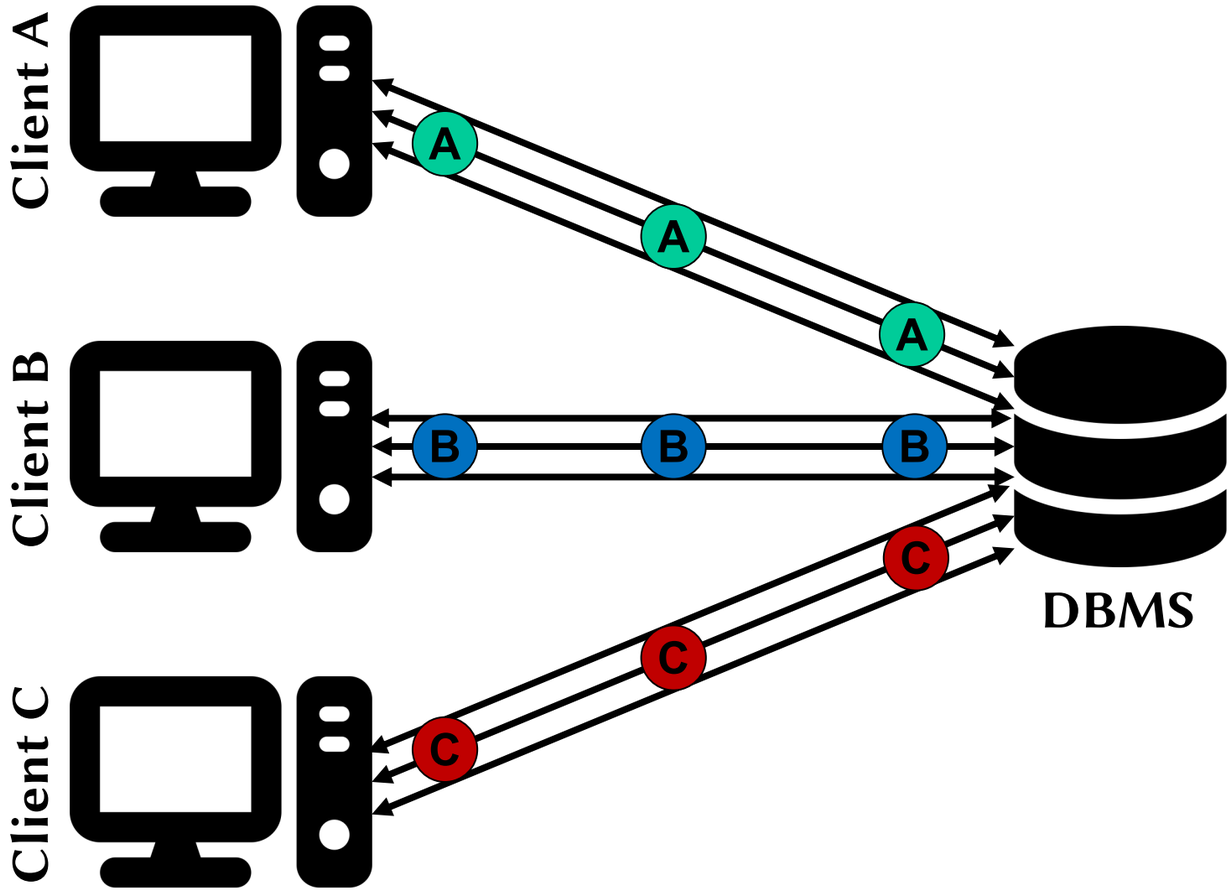
Carnegie Mellon
Parallel Data Laboratory

Where We're Going

- Database management system (DBMS) proxies are useful in large cloud deployments
- Current proxy designs are inefficient due to buffer copies with system calls
- We propose a technique called *user-bypass* to push DBMS proxy logic into kernel-space

Why do we need DBMS proxies?

DBMS Connections



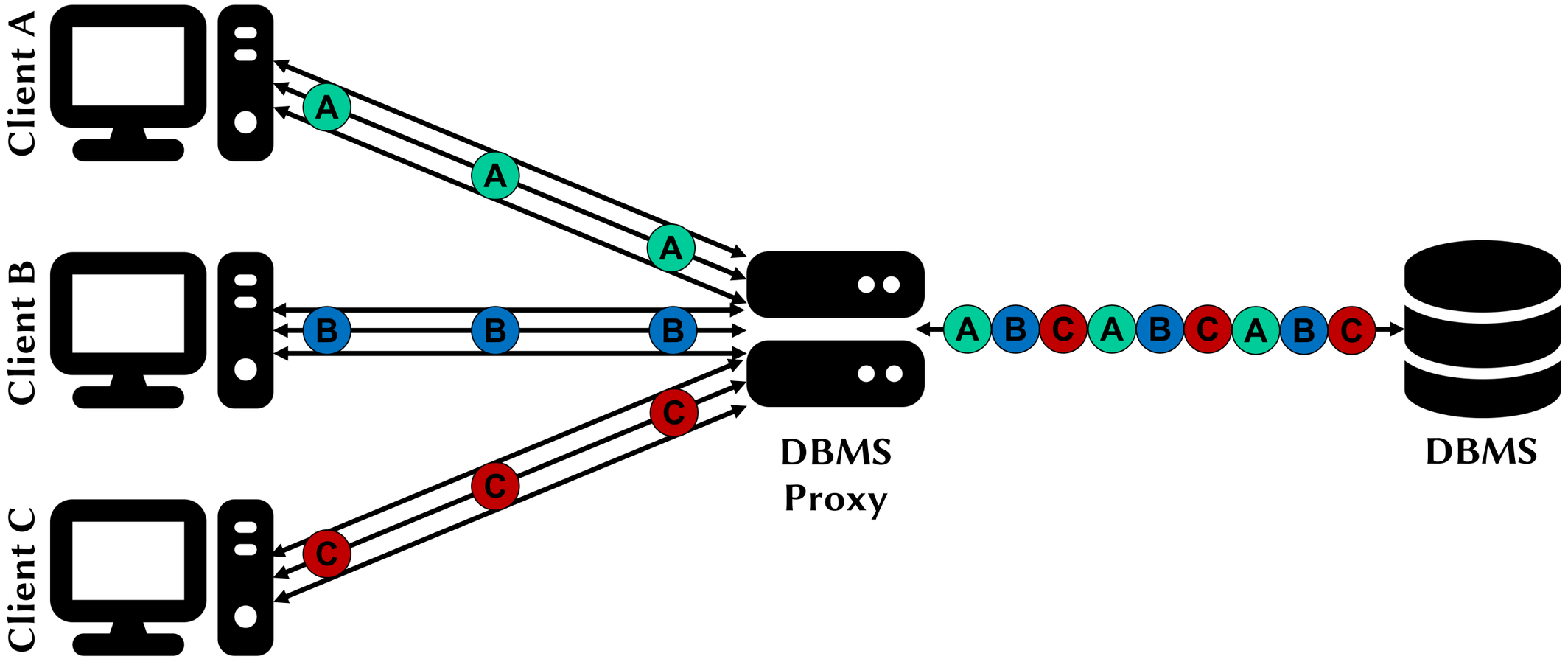
Connection Scaling

- Autoscaling applications open a lot of connections
- More connections = slower transaction latencies
- Each connection = worker (e.g., thread, process)
- Each PostgreSQL connection = MBs of RAM

[0] **Measuring the Memory Overhead of a Postgres Connection**
Postgres From Below

[1] **Resources consumed by idle PostgreSQL connections**
AWS Database Blog

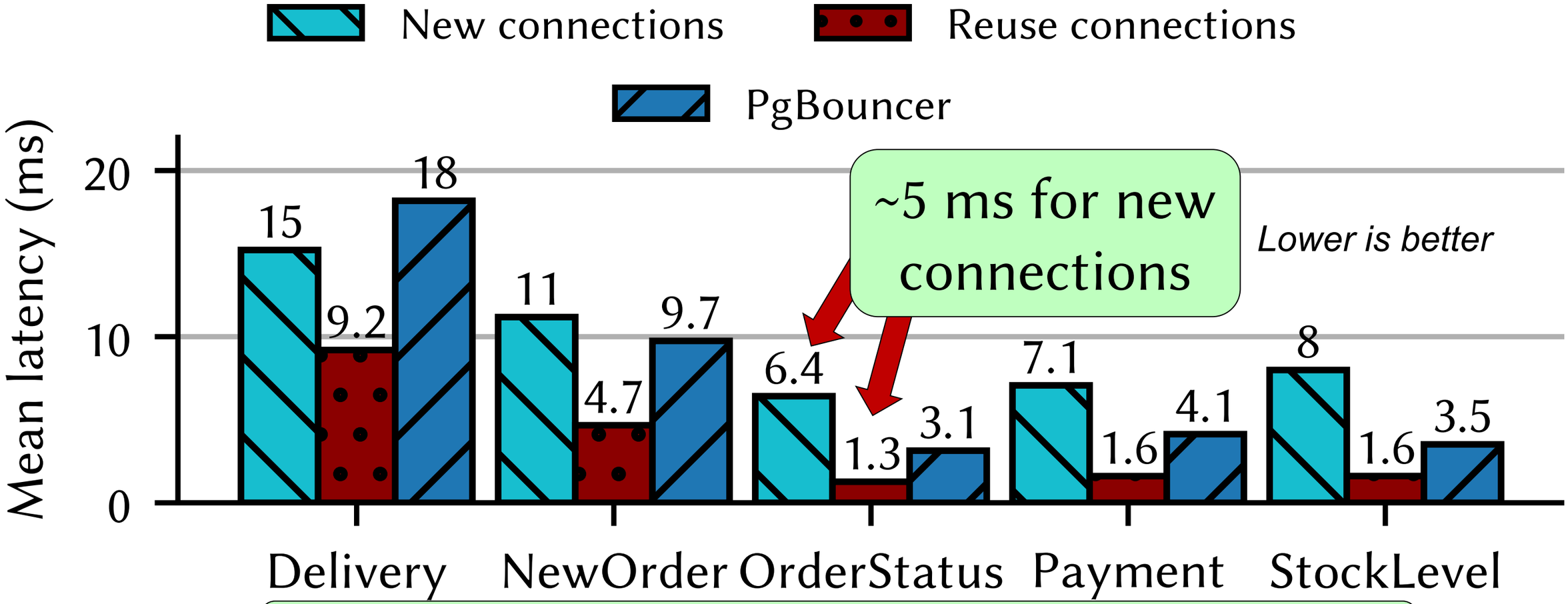
Connection Pooling



Connection Establishment

- Serverless systems open short-lived connections
- Popular web frameworks discard connections
- Establishment requires multiple operations:
  - Task allocation, socket allocation, TCP handshake, TLS handshake, client authentication, querying DBMS knobs

New Connection Overhead

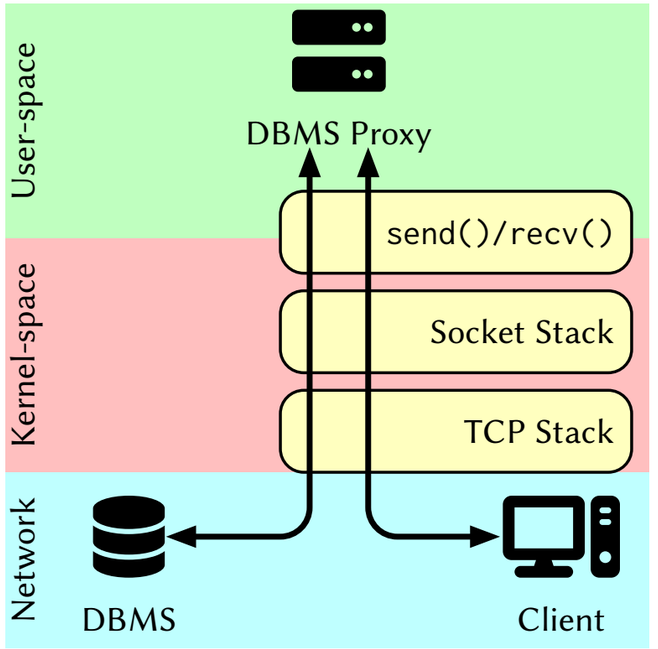


4 out of 5 transactions are faster with a proxy

Why do we need a new approach to DBMS proxies?

User-Space DBMS Proxy

- Traffic goes through OS network stack to apply DBMS protocol logic
- User-space applications of varying complexity to express parallelism
- Coordination mechanisms around `send()` and `recv()` system calls

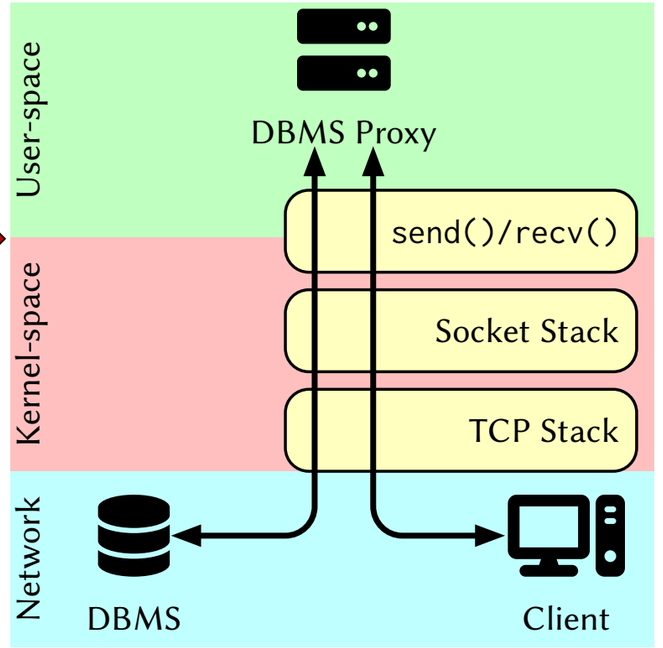


User-space proxy

OS Network Stack Overhead

- How fast can the Linux kernel process data?

>50% of CPU cycles copying buffers

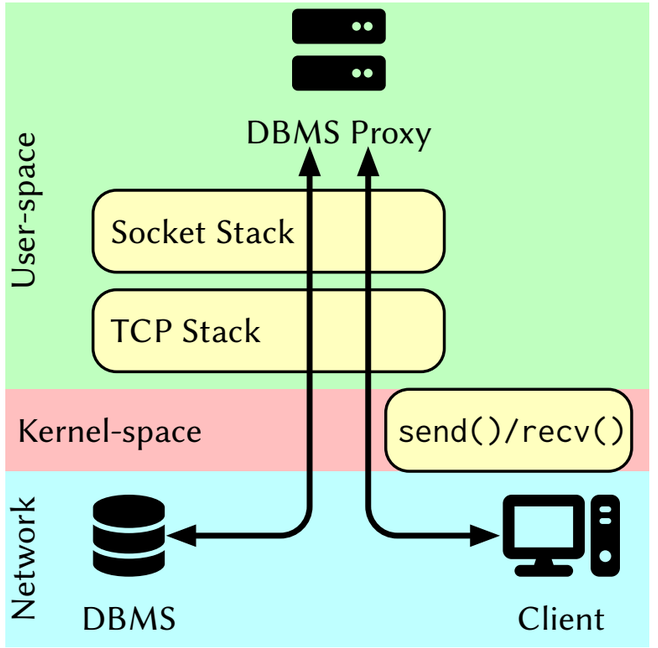


User-space proxy

- 42Gbps per CPU core

Kernel-Bypass DBMS Proxy

- Software tied to DPDK versions
- Dedicated drivers and device mode with no kernel management
- Standard networking tools don't work for debugging

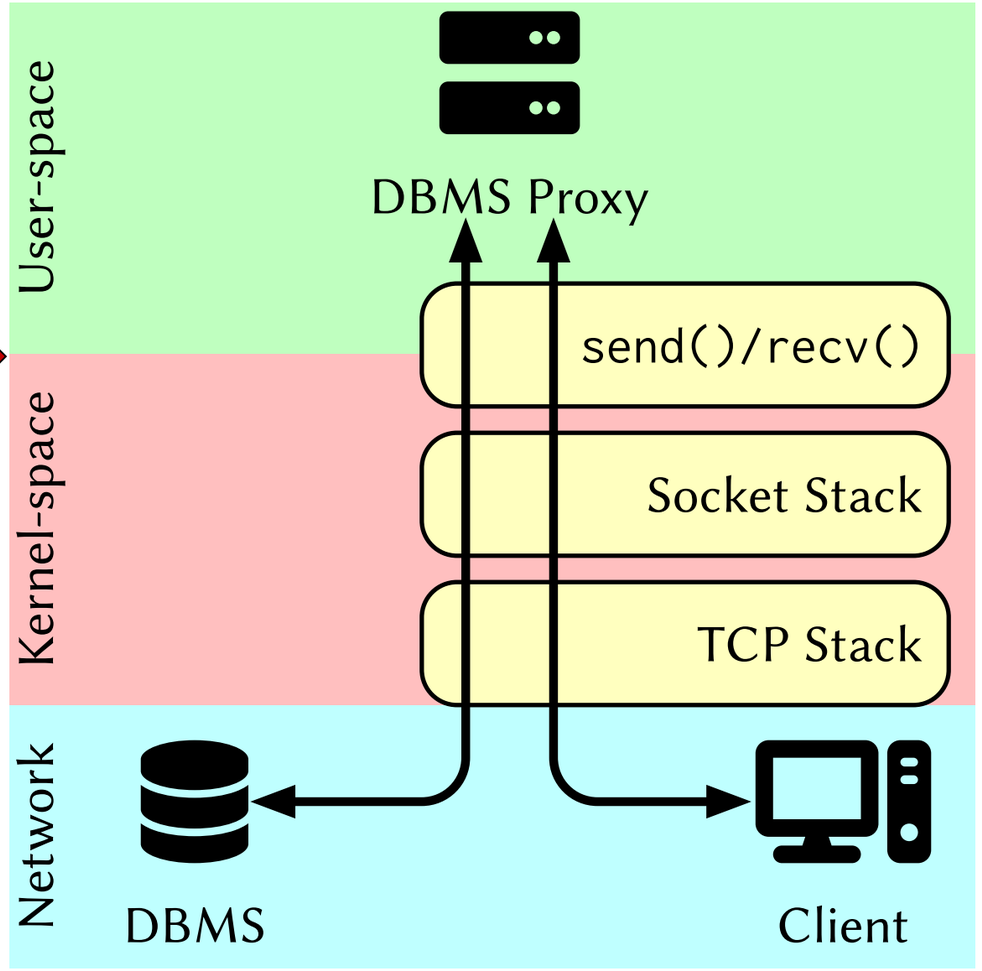


Kernel-bypass proxy

Kernel-Bypass DBMS Proxy

The Big Problem

>50% of CPU cycles copying buffers

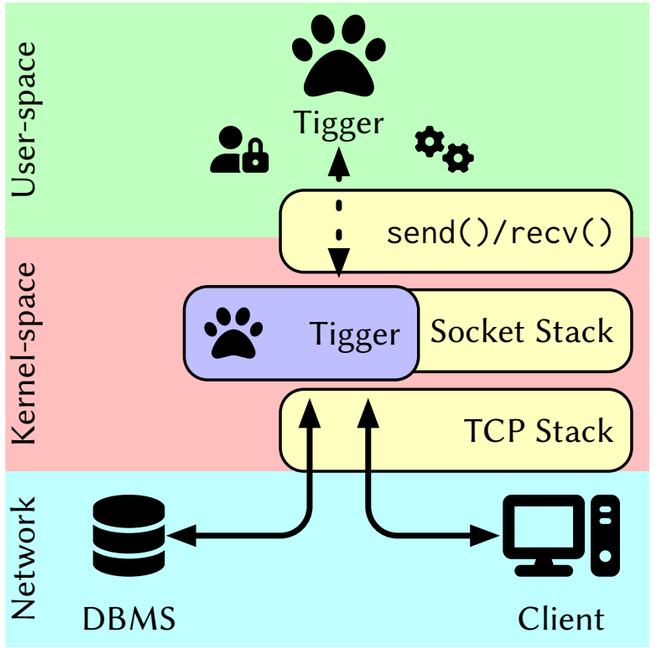


User-space proxy

What if we could bypass user-space entirely?

User-Bypass DBMS Proxy

- Don’t pull application data up to user-space logic
- Push application logic down to kernel-space data
- Zero-copy kernel APIs, avoid system calls



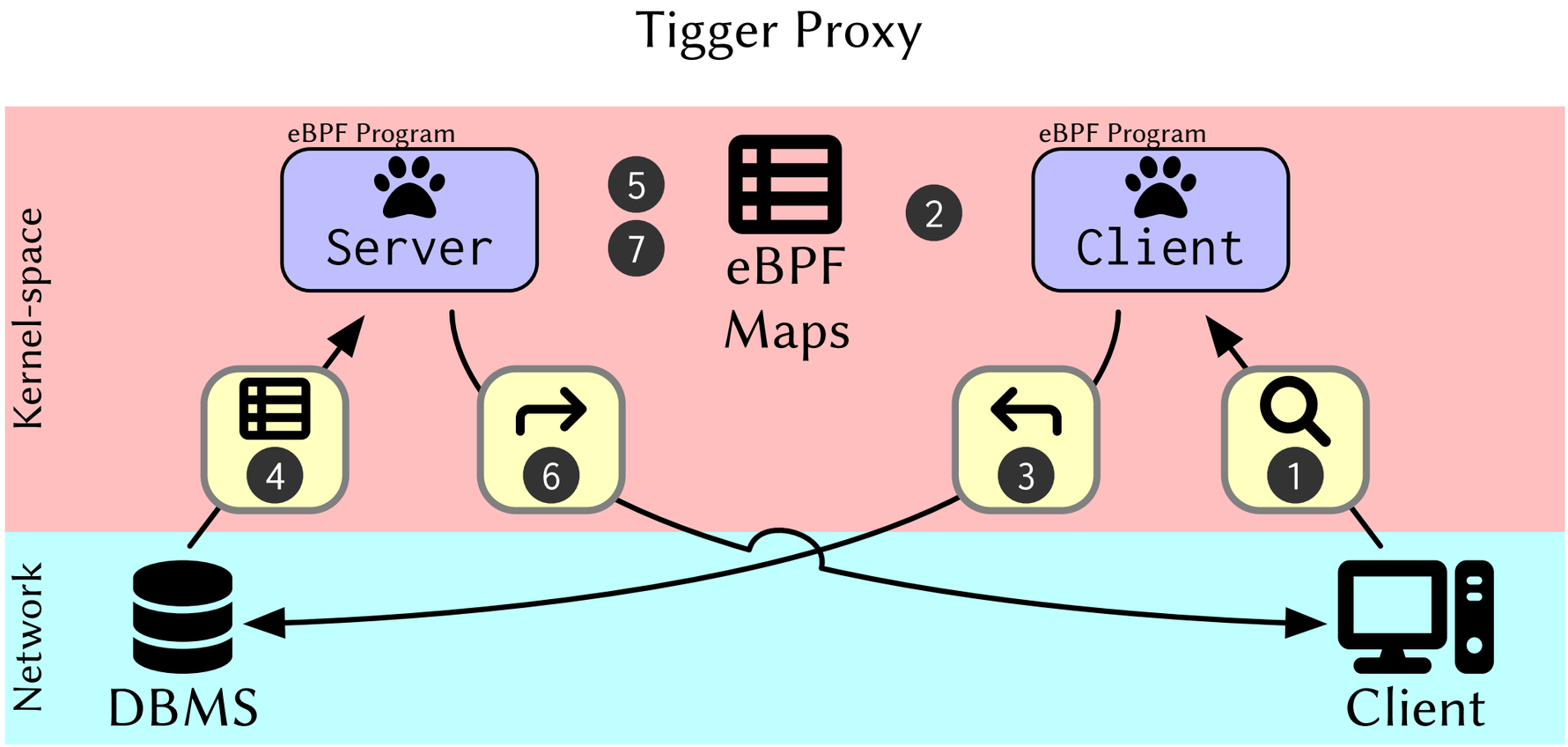
User-bypass proxy

How do we achieve user-bypass?

eBPF

- Safe, event-driven programs in Linux kernel
  - Carry state across events with *eBPF Maps*
- Verifier constraints:
  - # instructions, boundedness, memory safety, limited API
- Uses:
  - Networking, observability, security

Tigger Connection Pooling

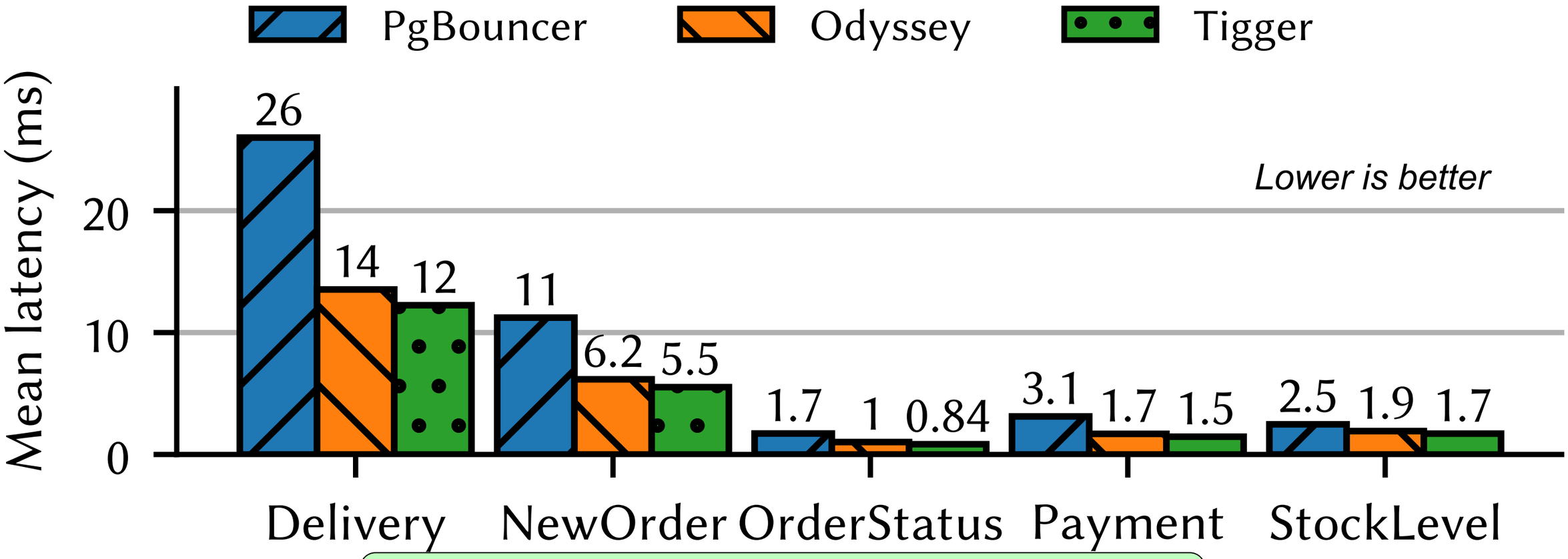


Evaluation

Experimental Setup

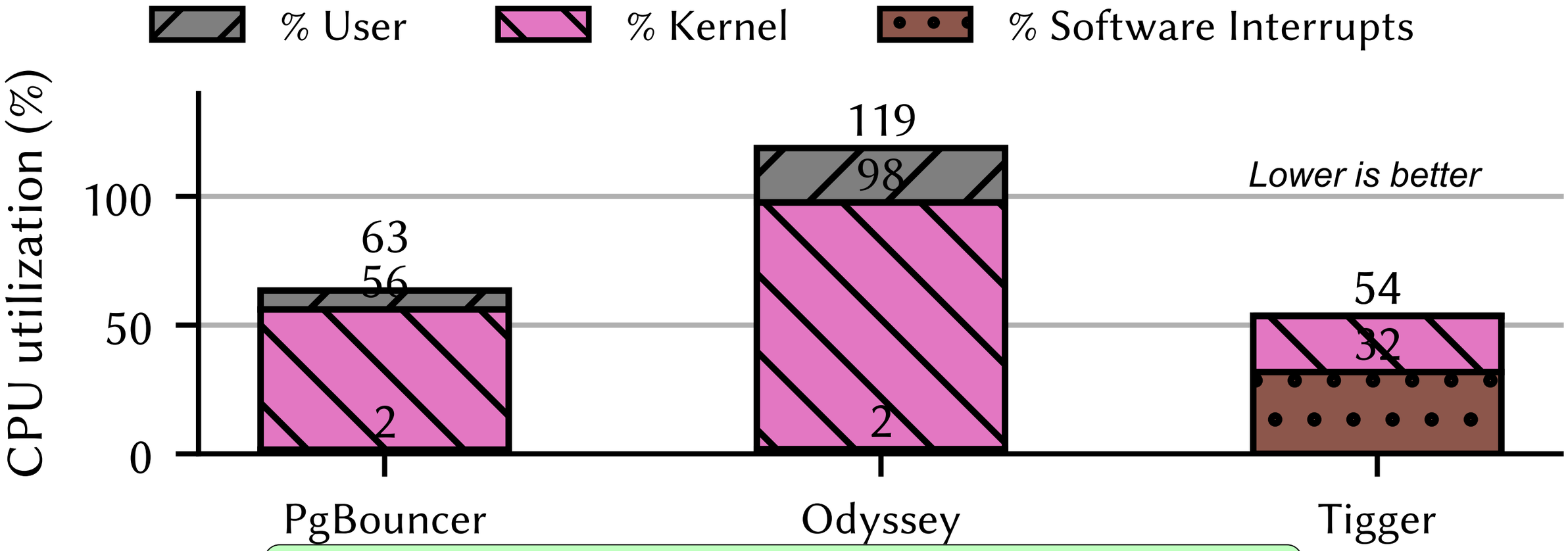
- Dedicated servers:
  - 2×20-core Intel Xeon Gold 5218R CPUs
  - 192 GB DRAM
  - Samsung PM983 SSD
  - Dual-port 10GbE network adapter
  - Ubuntu 22.04 LTS with Linux (v5.15)
  - Cisco Nexus 3064 10GbE network switch
- Software:
  - PostgreSQL (v14.5), BenchBase

Connection Pooling



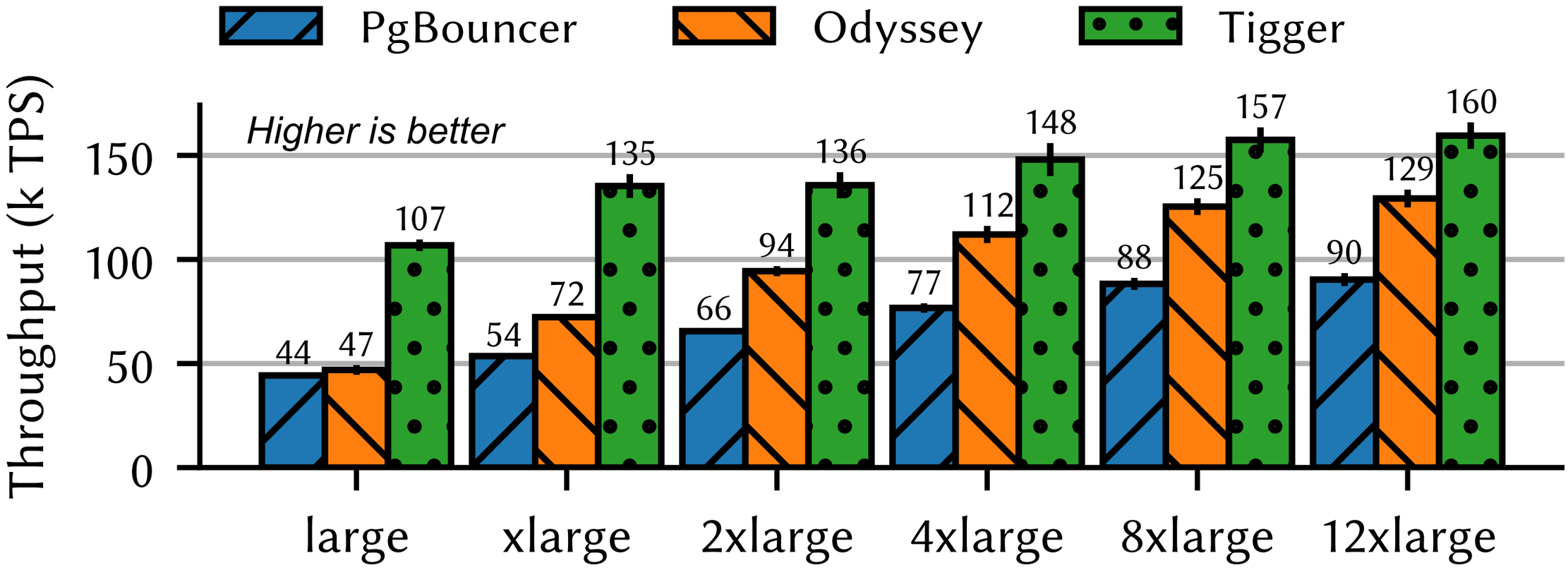
Tigger offers the lowest latencies

Connection Pooling CPU



Tigger consumes the fewest CPU cycles

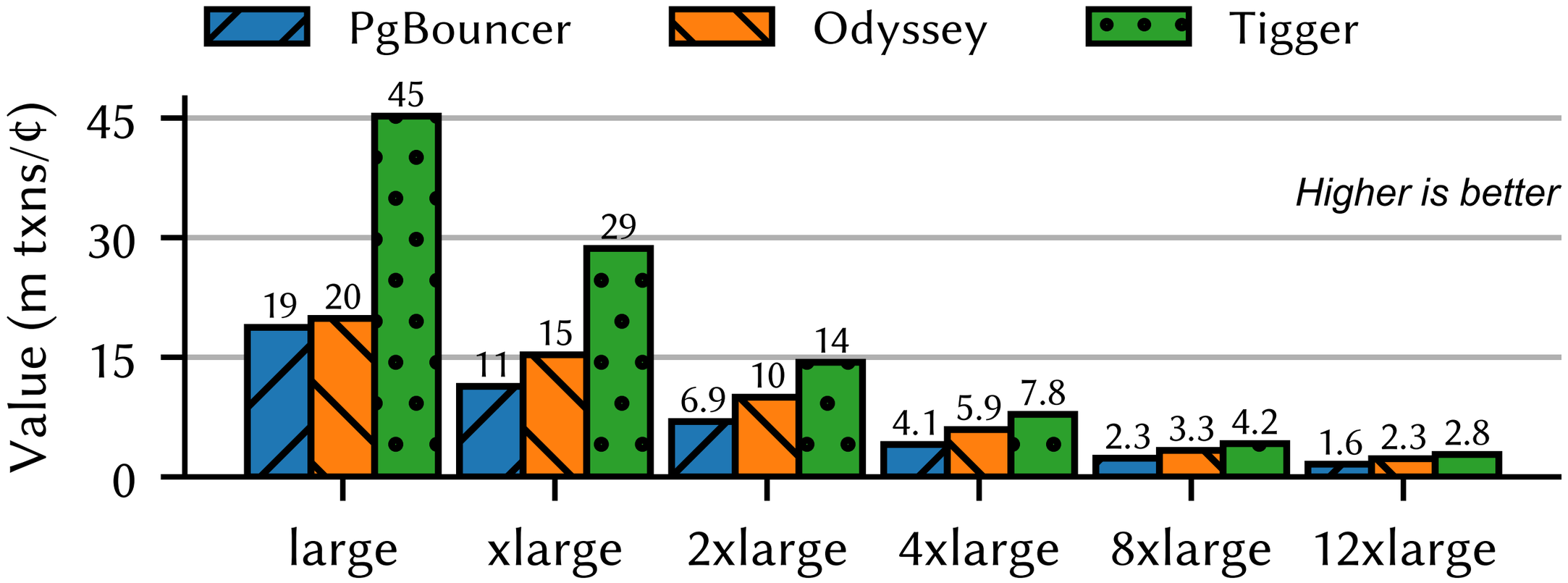
Cloud Instance Throughput



Tigger maximizes throughput with fewer resources

Errors bars are SEM (n=20).

Cloud Instance Value



Tigger does more while costing less

Where to next?

DBMS Proxy Comparisons

- Kernel-bypass:
  - WIP version of PgBouncer using DPDK

- `io_uring`:
  - Linux asynchronous IO with shared ring buffers
  - Doesn't eliminate all copies, system calls, or scheduling

Tigger Future Enhancements

- More proxy features:
  - Sharding, caching, query rewriting
- Automatic proxy tuning:
  - Pool sizes, instance sizes
- Hardware offload:
  - SmartNICs, FPGAs

Takeaways

- *User-bypass* is a new approach to safely push application logic into the kernel, avoiding system call overhead

- *Tigger*'s user-bypass offers the lowest latency and lowest CPU utilization for DBMS proxies

https://mattbutrovi.ch