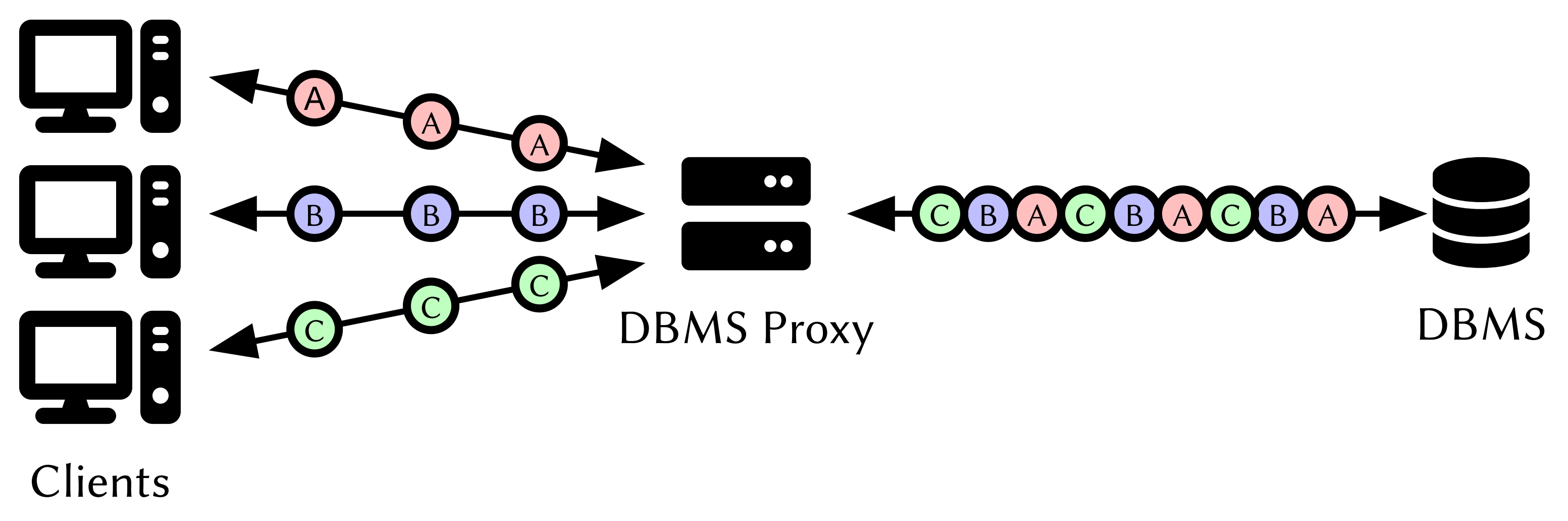
Tigger: A Database Proxy That Bounces With User-Bypass

Matthew Butrovich, Karthik Ramanathan, John Rollinson*, Wan Shen Lim, William Zhang,
Justine Sherry, Andrew Pavlo
Carnegie Mellon University, *Army Cyber Institute

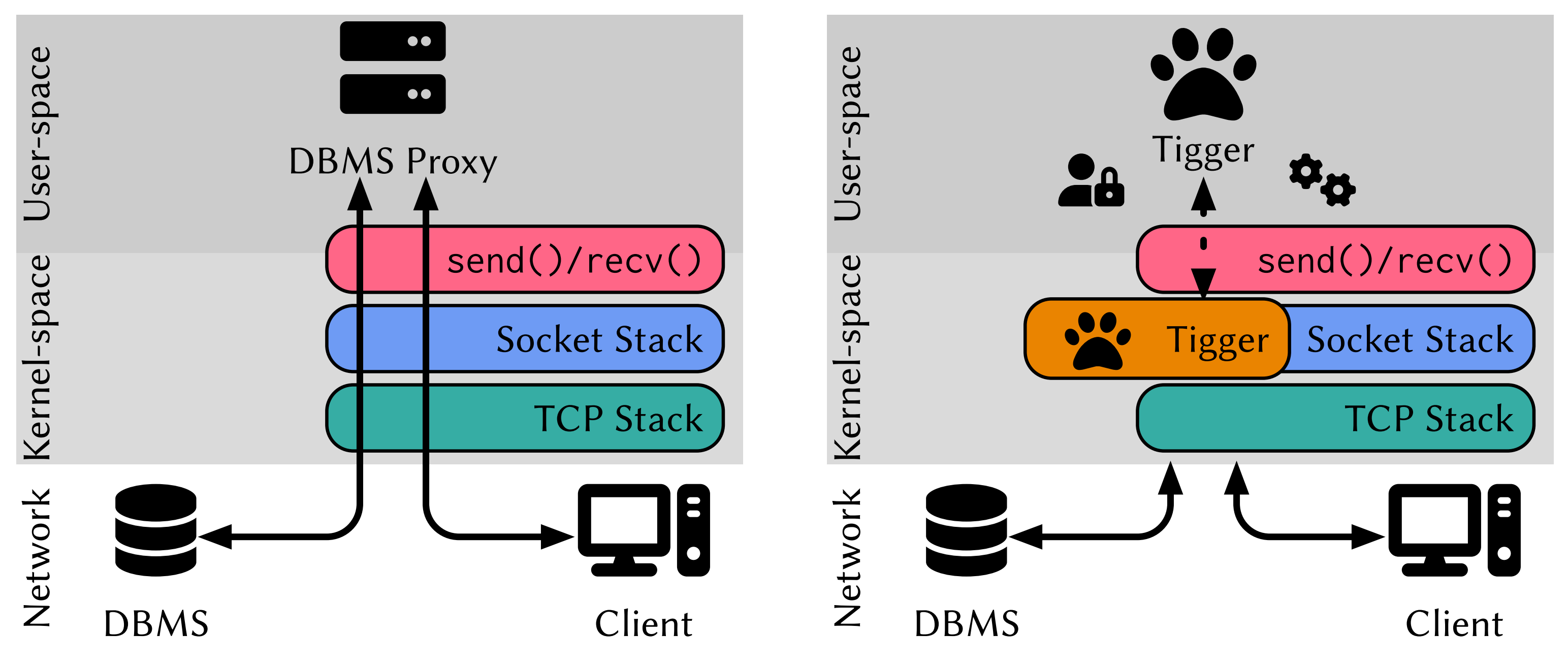
Background

Developers often deploy database-specific network proxies whereby applications connect transparently to the proxy instead of directly connecting to the database management system (DBMS).



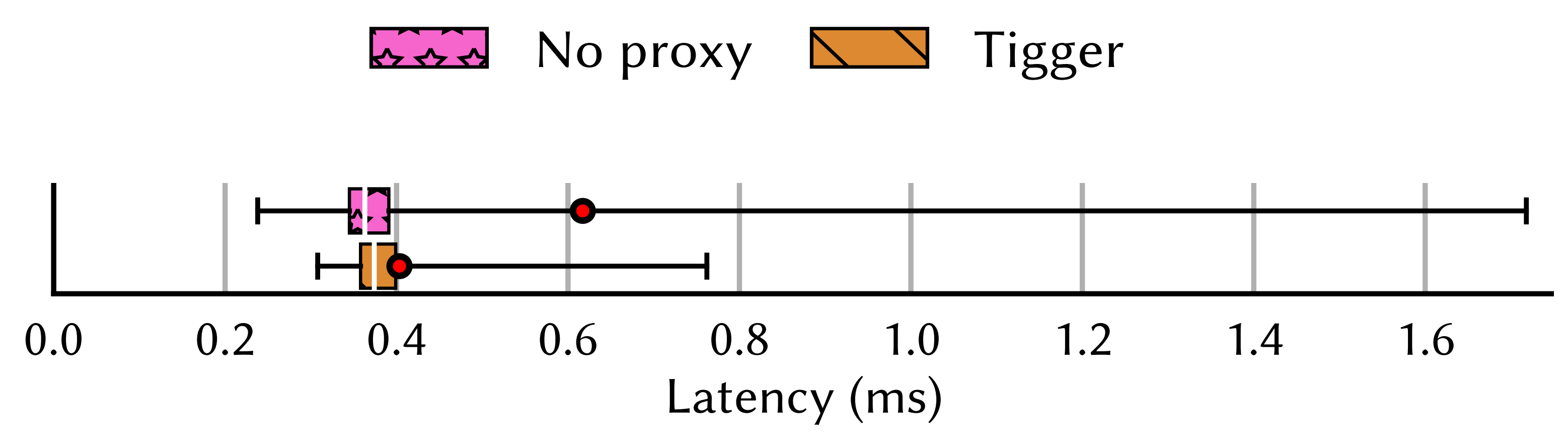
This indirection improves system performance because of connection pooling, load balancing, and other DBMS-specific optimizations. However, these Application Layer (L7) proxies are inefficient due to their reliance on OS system calls.

User-Bypass and Tigger

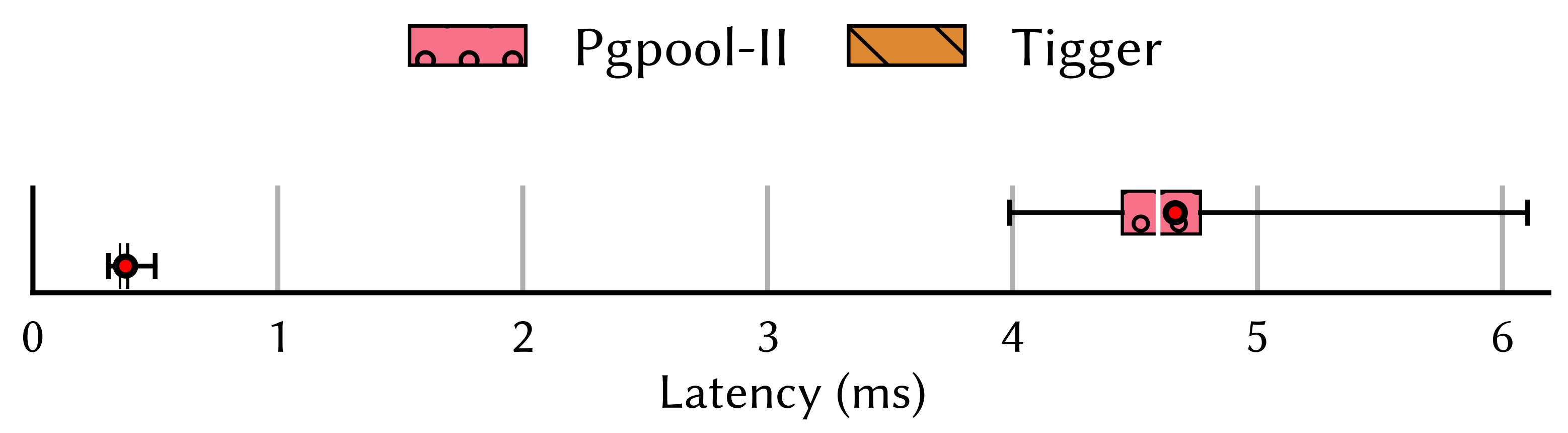


The **Tigger** DBMS proxy applies **user-bypass** to push application logic into kernel-space via Linux's eBPF infrastructure. This design eliminates the overheads associated with user-space proxies (e.g., copying buffers, scheduling user threads, and privilege escalation).

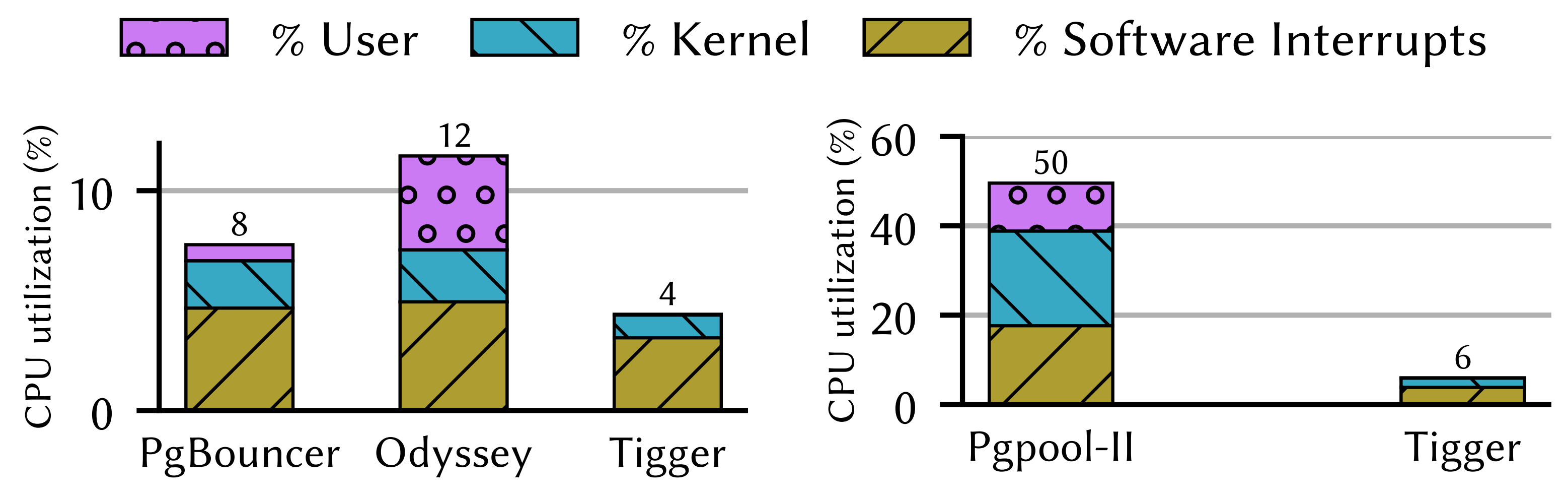
Evaluation



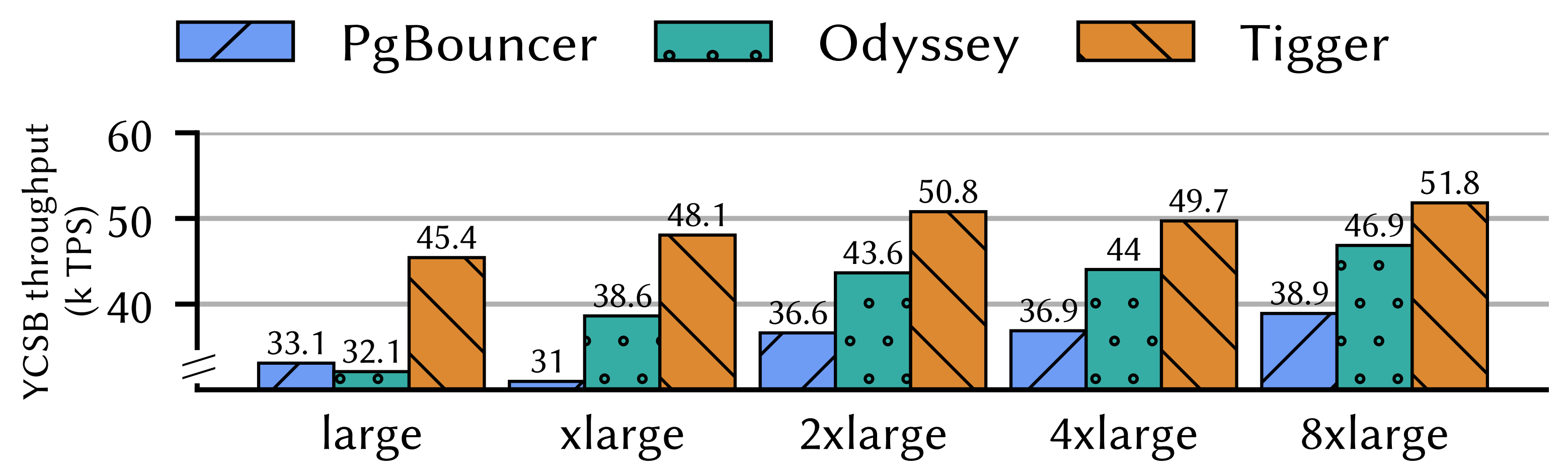
For transaction pooling with YCSB with 10,000 clients, Tigger reduces p99 latency by 56% and reduces mean latency by 35% in exchange for a small minimum latency increase for the extra network hop.



For workload replication with YCSB, Tigger's reduction in system calls, and thereby buffer copies, lowers transaction latencies by 92%.



The CPU utilizations for the previous experiments show how Tigger's user-bypass design reduces User and Kernel CPU time. Software Interrupt time is lower for Tigger as well due to handling a query in a single interrupt, instead of both read and write paths as with PgBouncer and Odyssey.



With 2 vCPUs (large), Tigger offers 40% more throughput than other proxies. Odyssey requires 8× the vCPUs (at 8× the cost) to match that baseline performance from Tigger. Administrators could deploy Tigger at a much lower cost than Odyssey or PgBouncer.

Future Work

Ephemeral I/Os: Identify I/O reads that the DBMS discards shortly after use. Emphasize reads where minimal logic is applied to the data (e.g., inspecting page headers, following pointers in index structures).

Shuffle Nodes: Explore if nodes in distributed query processing (e.g., Apache Uniffle) benefit from user-bypass.

Conclusion

We introduce a user-bypass technique to safely avoid system calls, which copy buffers to and from kernel-space. User-bypass pushes application logic into the Linux kernel using eBPF. We show how to apply user-bypass with our PostgreSQL-compatible proxy Tigger. When compared against other proxies, Tigger offers the best performance and lowest operating cost. Our evaluation shows the value of user-bypass to reduce data movement between kernel-space and user-space and avoid costly system calls. In the future, DBMS components that perform ephemeral I/Os should be designed with user-bypass in mind.