Embedding Database System Logic in the Operating System Is Finally a Good Idea

Matthew Butrovich

PARALLEL DATA LABORATORY

Carnegie Mellon University

Carnegie Mellon
Parallel Data Laboratory

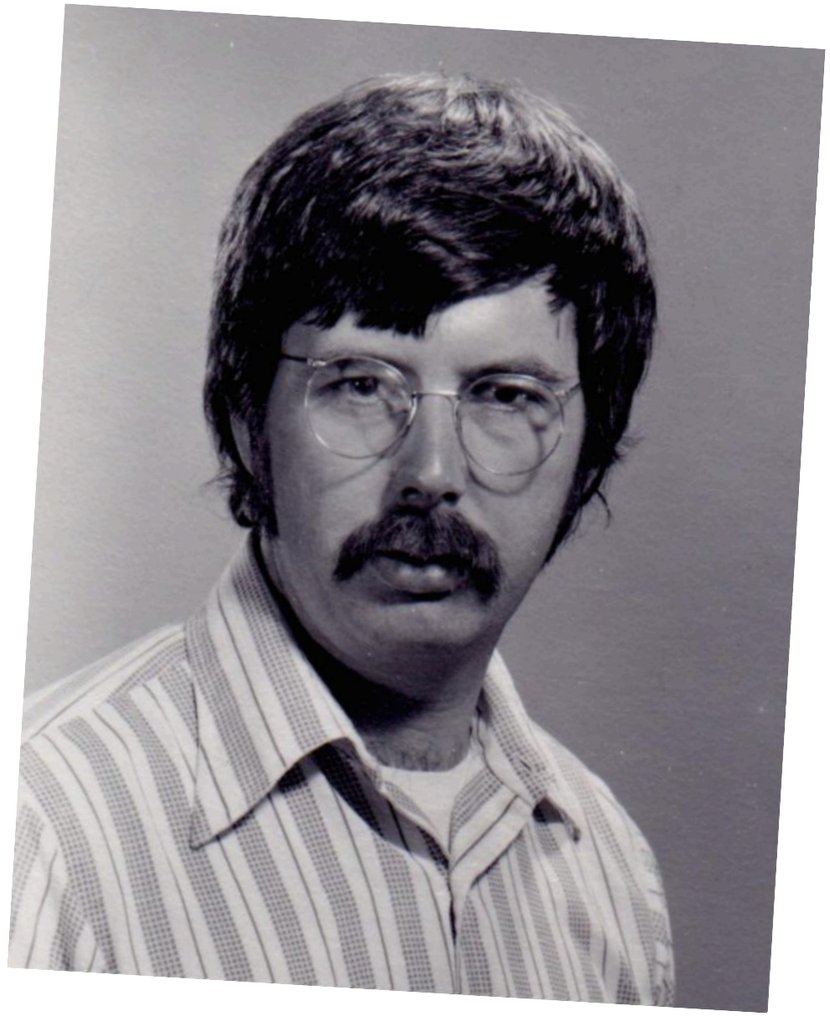
Outline

- User-bypass
- Prior work: User-bypass for DBMS proxies
- Future work: User-bypass DBMS

The OS Is Not Our Friend

The OS Is Not Our Friend

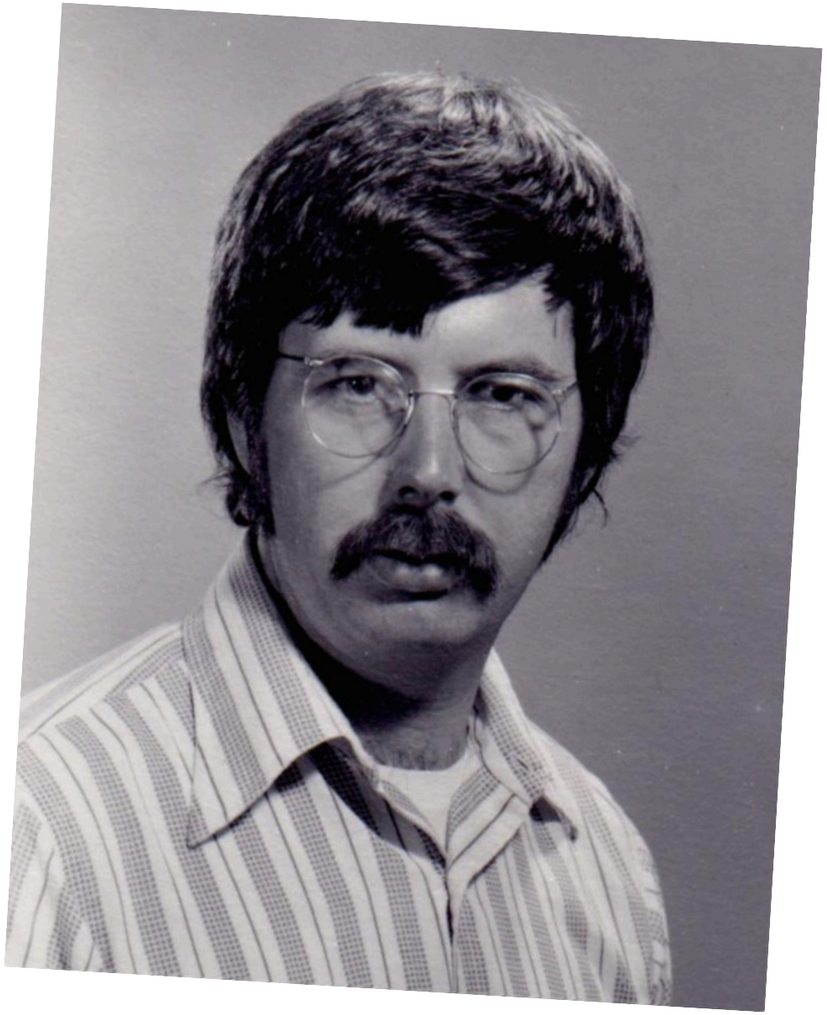
"The bottom line is that operating system services in many existing systems are either too slow or inappropriate."



Michael Stonebraker. Operating System Support for Database Management. *Commun. ACM*. 1981.

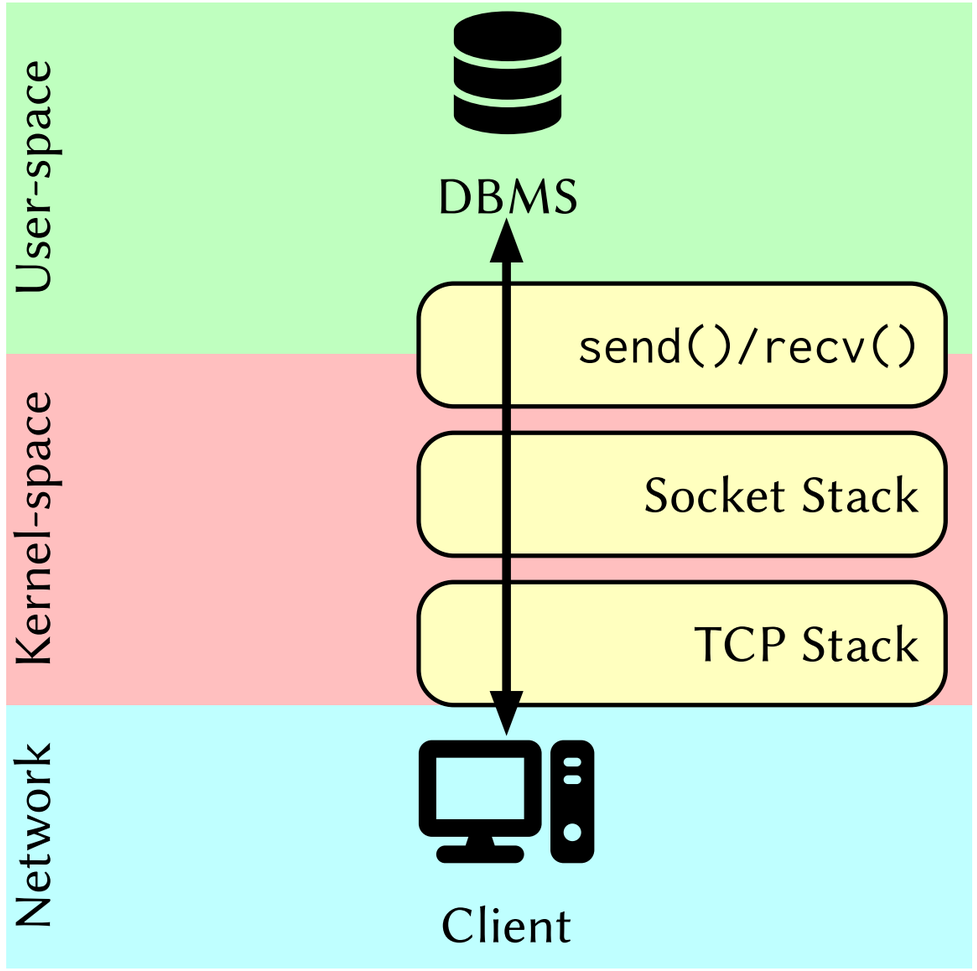
The OS Is Not Our Friend

"The bottom line is that operating system services in many existing systems are either too slow or inappropriate."



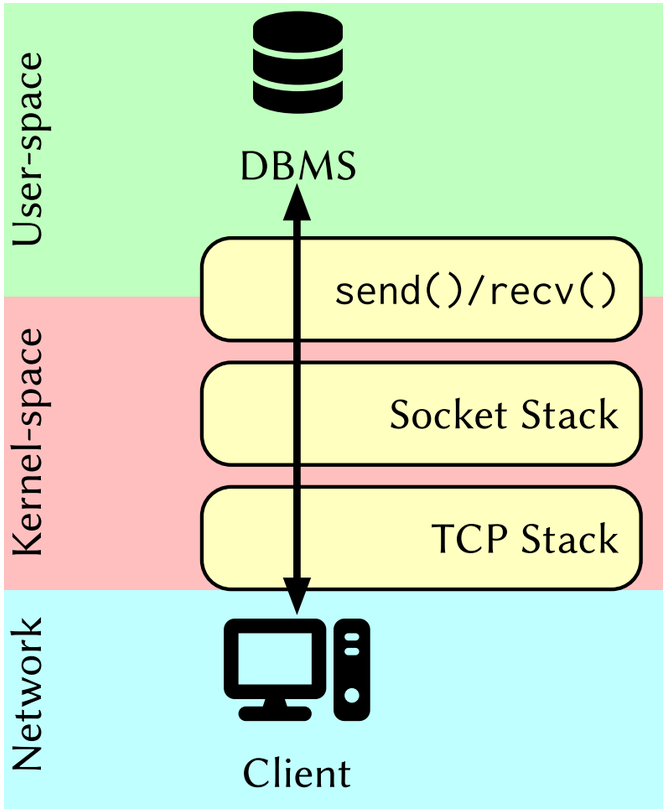
Michael Stonebraker. Operating System Support for Database Management. *Commun. ACM*. 1981.

Where Are the I/O Bottlenecks?



User-space DBMS

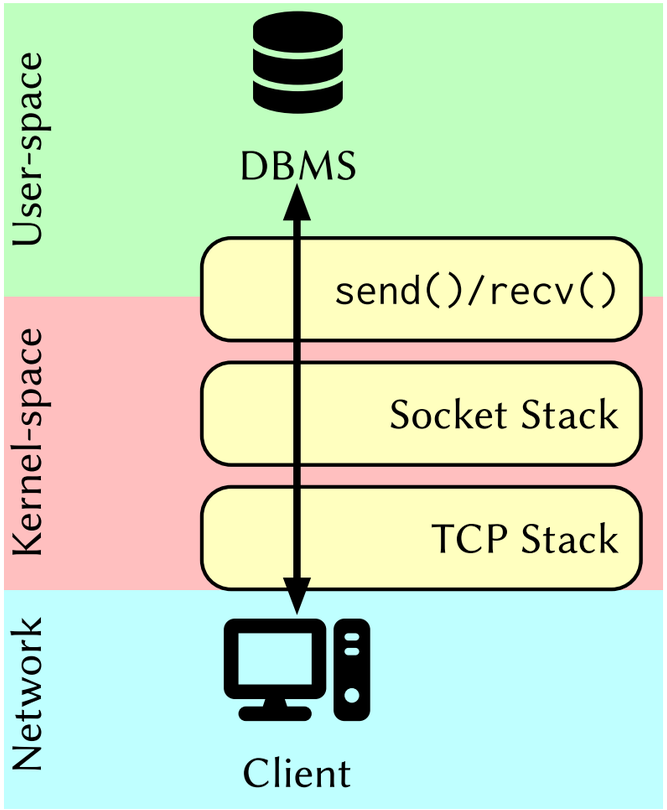
Where Are the I/O Bottlenecks?



User-space DBMS

Where Are the I/O Bottlenecks?

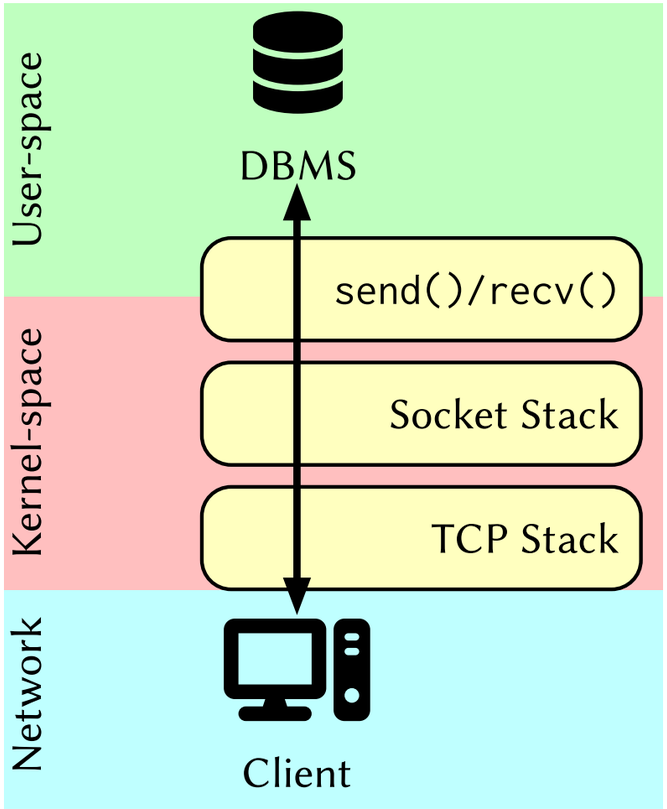
- I/O devices (network, disk) are faster



User-space DBMS

Where Are the I/O Bottlenecks?

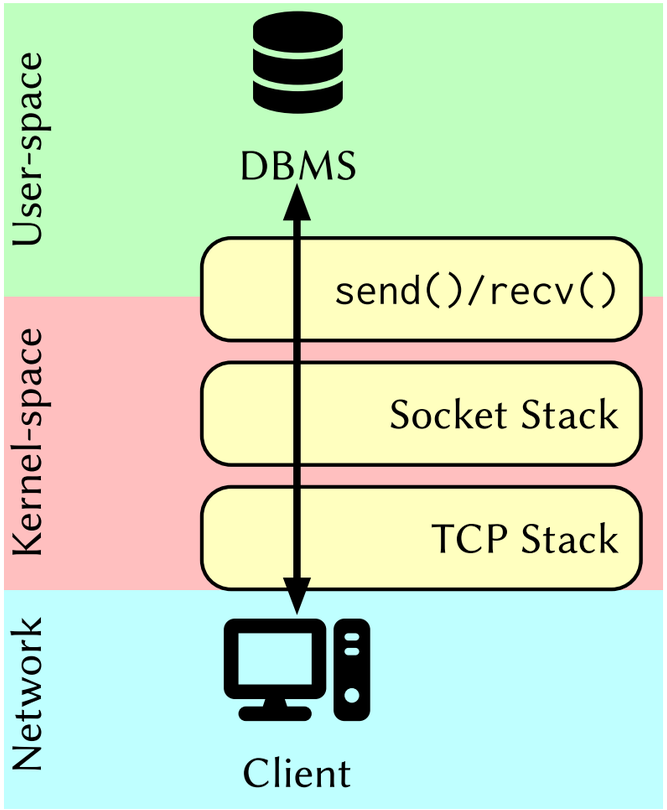
- I/O devices (network, disk) are faster
- Operating system logic is also faster



User-space DBMS

Where Are the I/O Bottlenecks?

- I/O devices (network, disk) are faster
- Operating system logic is also faster
- Max throughput: 42Gbps per CPU core



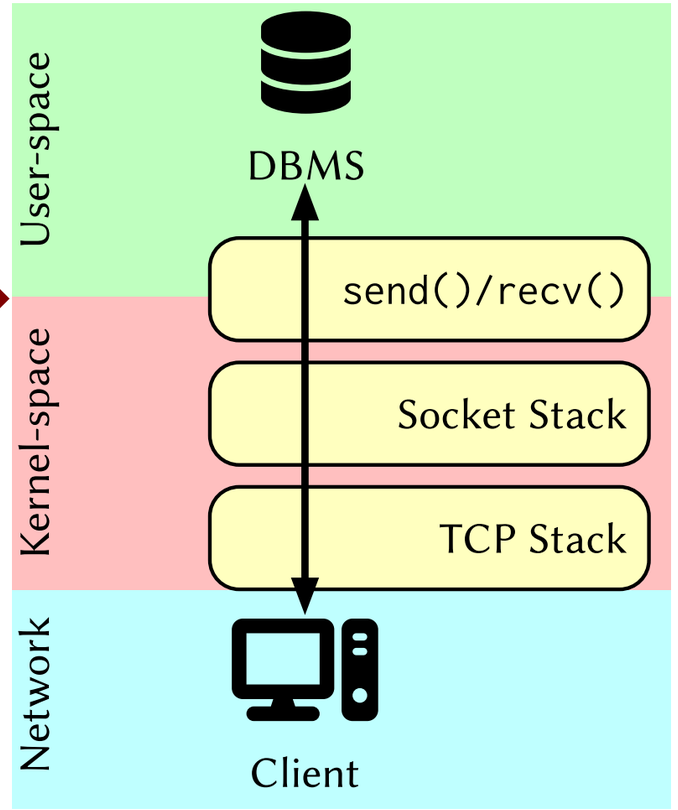
User-space DBMS

Qizhe Cai et al. Understanding host network stack overheads. *SIGCOMM*. 2021.

Where Are the I/O Bottlenecks?

- I/O devices (network, disk) are faster
- Operating sys
- Max throughput: 42Gbps per CPU core

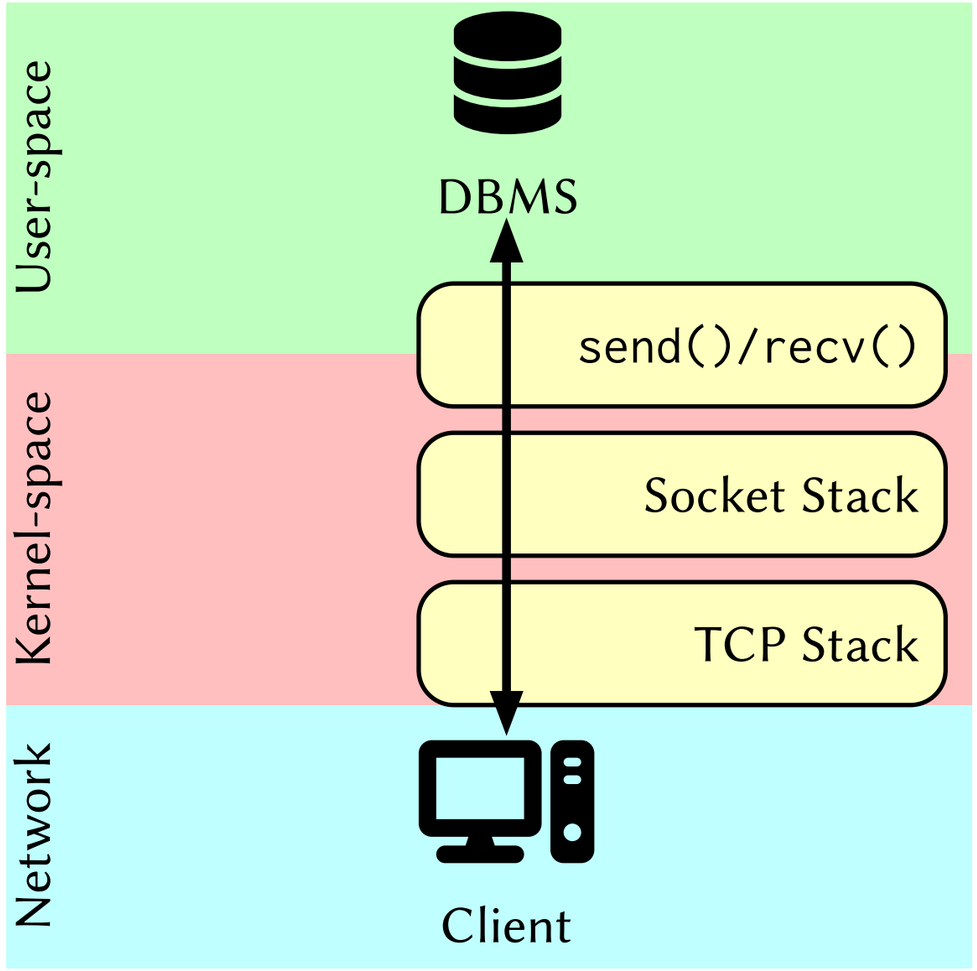
>50% of CPU cycles on memcpy()



User-space DBMS

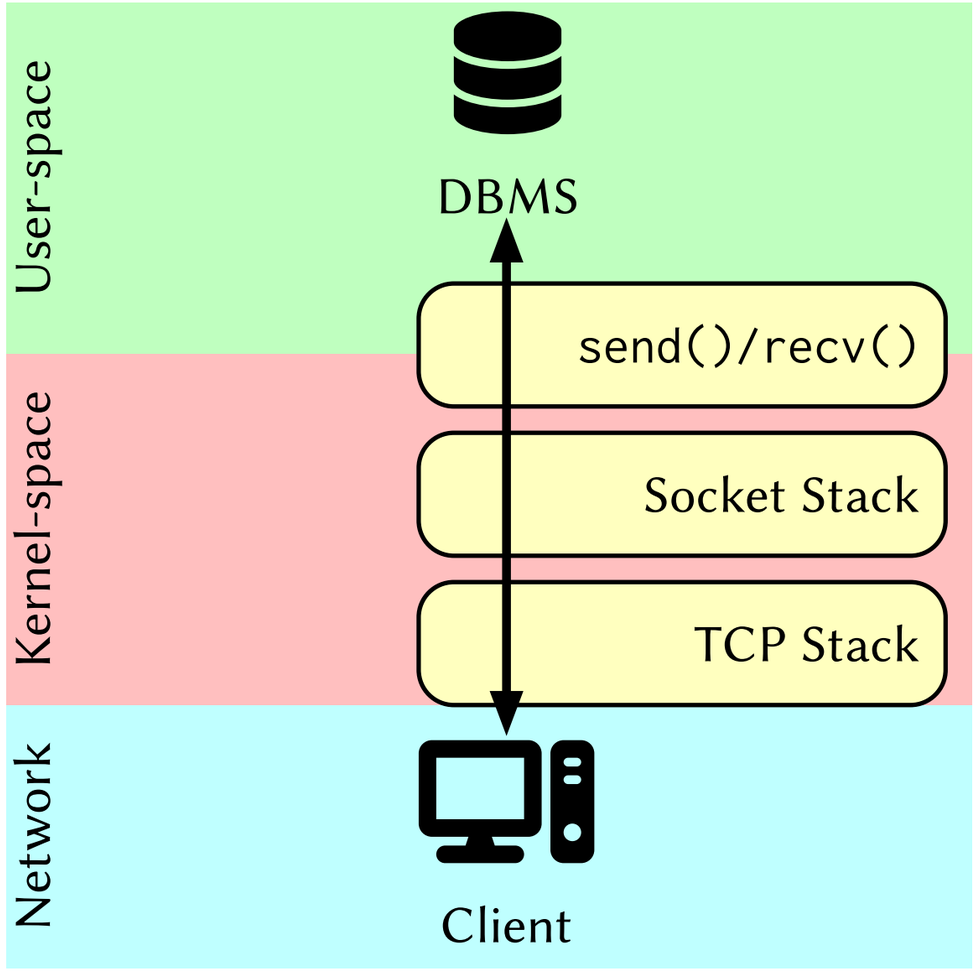
Qizhe Cai et al. Understanding host network stack overheads. *SIGCOMM*. 2021.

Kernel-Bypass



User-space DBMS

Kernel-Bypass

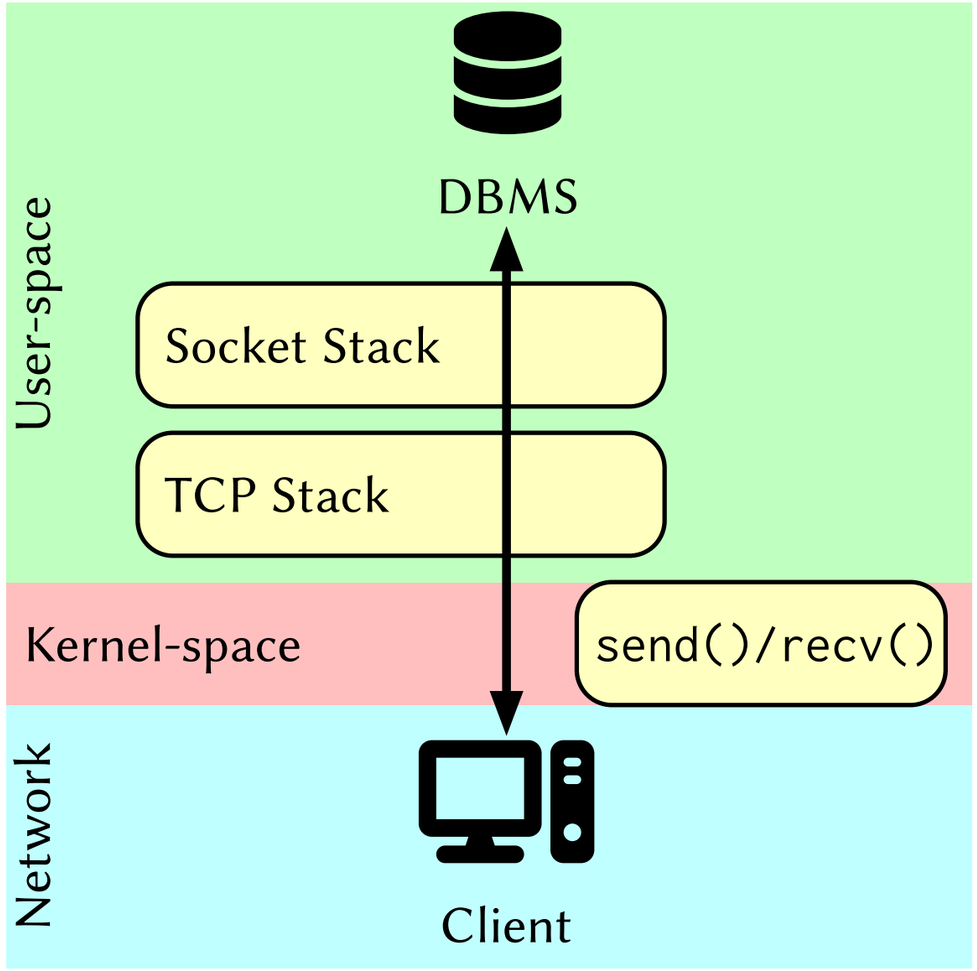


DPDK

SPDK

User-space DBMS

Kernel-Bypass

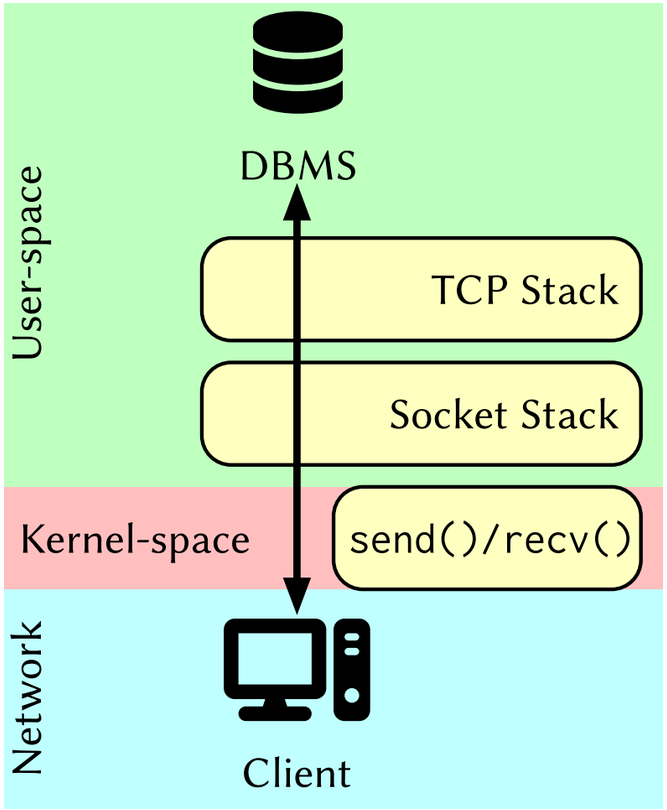


DPDK

SPDK

Kernel-bypass DBMS

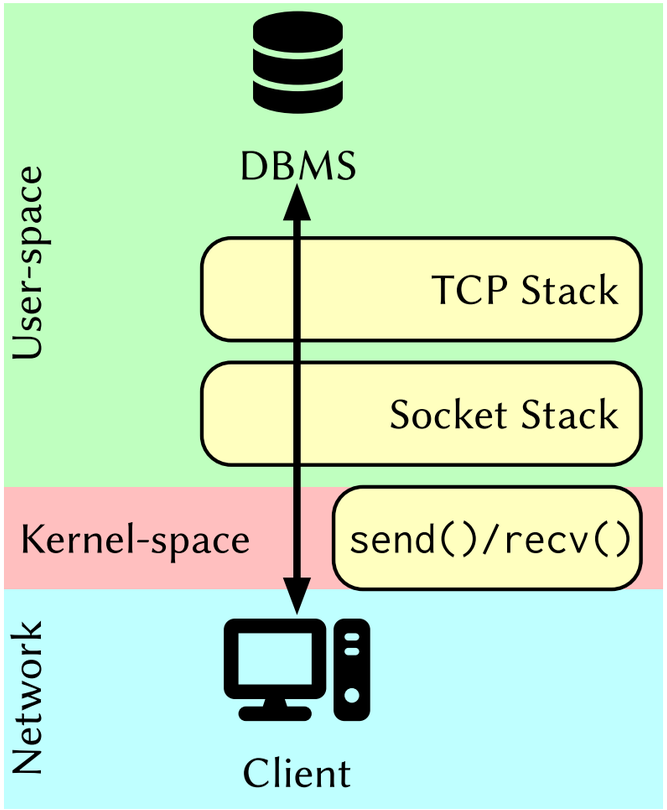
Kernel-Bypass



Kernel-bypass DBMS

Kernel-Bypass

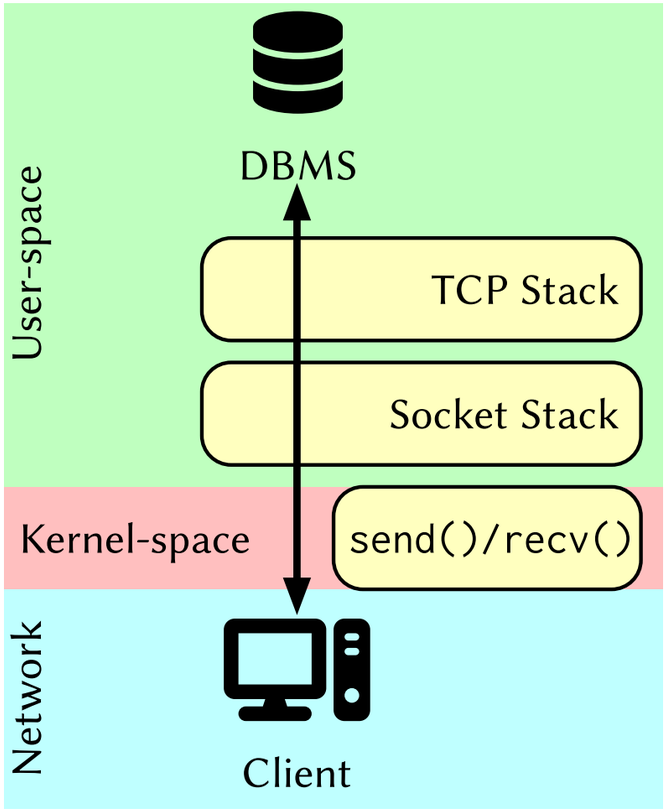
- Reimplement protocols in user-space



Kernel-bypass DBMS

Kernel-Bypass

- Reimplement protocols in user-space
- Difficult to debug, deploy, and maintain

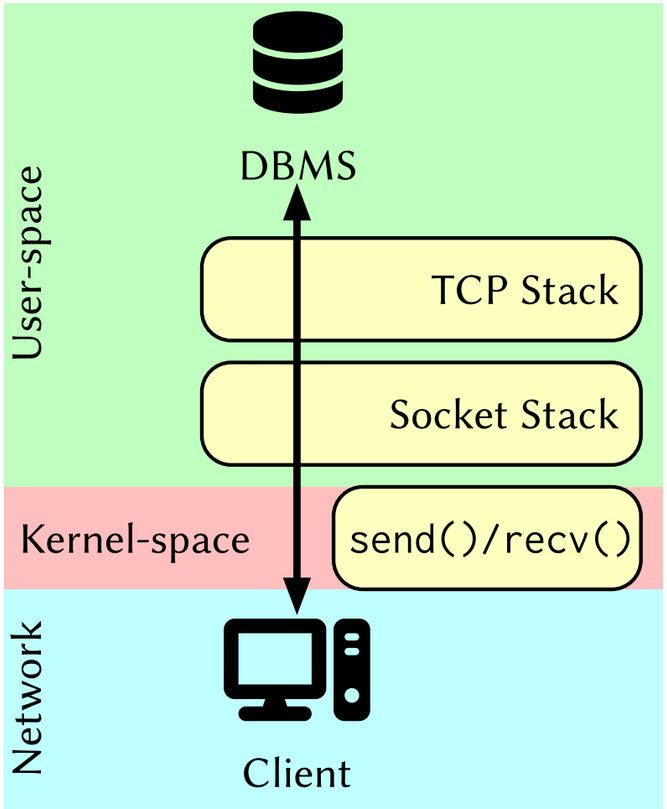


Kernel-bypass DBMS

William Tu et al. Revisiting the openvSwitch Dataplane Ten Years Later. *SIGCOMM*. 2021.

Kernel-Bypass

- Reimplement protocols in user-space
- Difficult to debug, deploy, and maintain
- Difficult to optimize



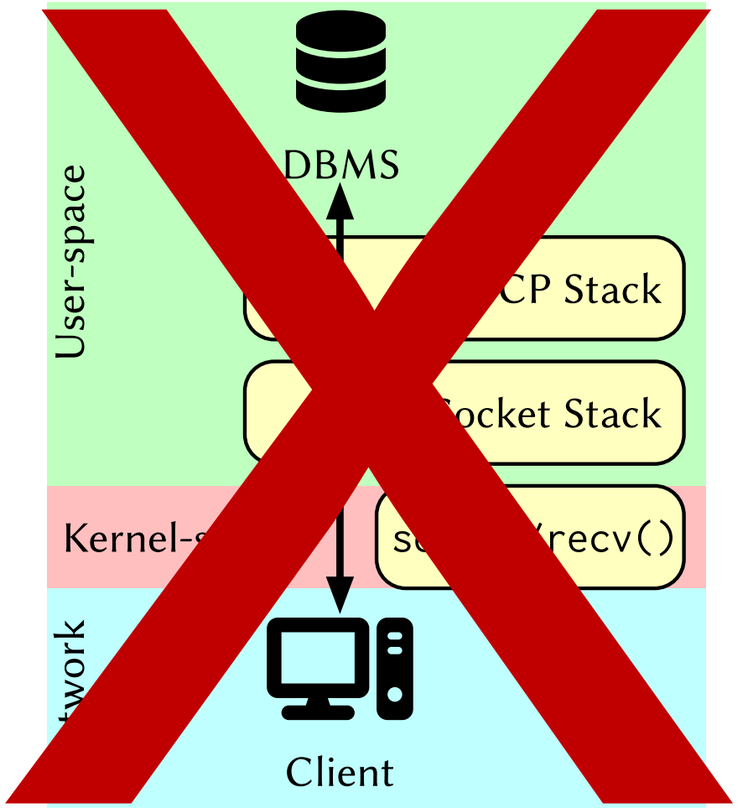
Kernel-bypass DBMS

William Tu et al. Revisiting the openvSwitch Dataplane Ten Years Later. *SIGCOMM*. 2021.

https://github.com/xrp-project/BPF-KV/issues/3

Kernel-Bypass

- Reimplement protocols in user-space
- Difficult to debug, deploy, and maintain
- Difficult to optimize

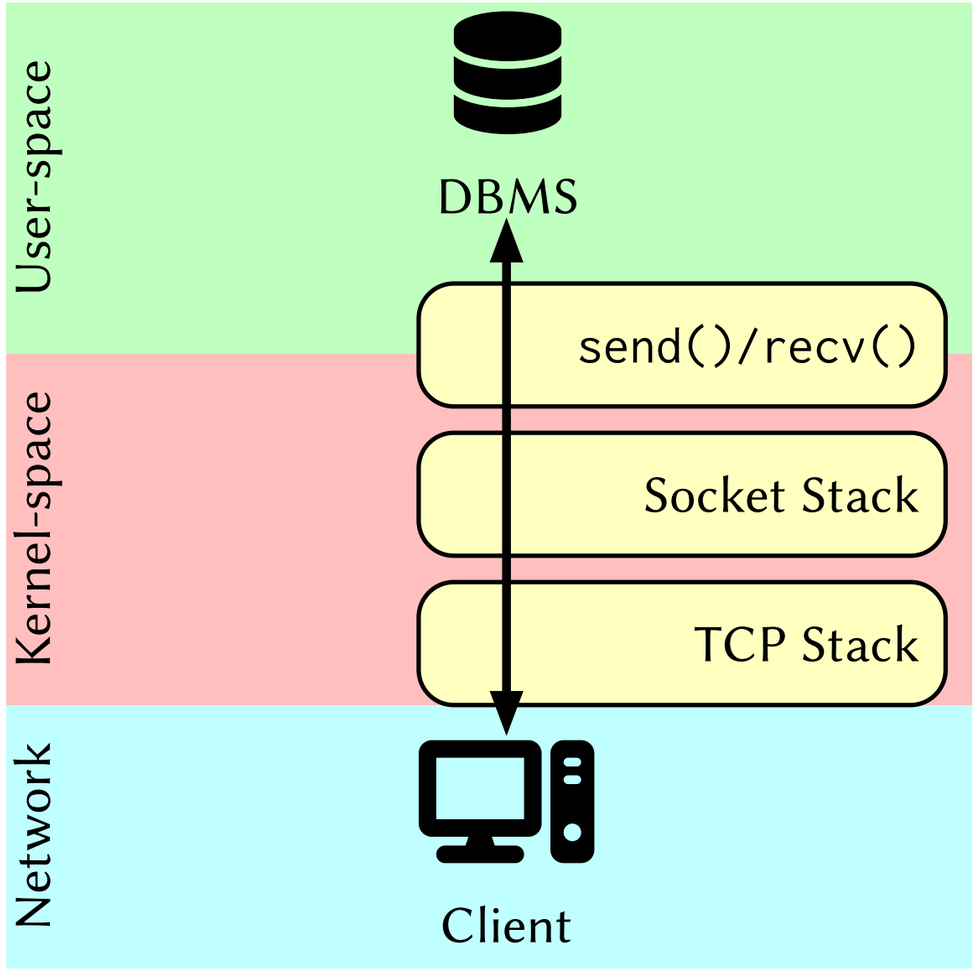


Kernel-bypass DBMS

William Tu et al. Revisiting the openvSwitch Dataplane Ten Years Later. *SIGCOMM*. 2021.

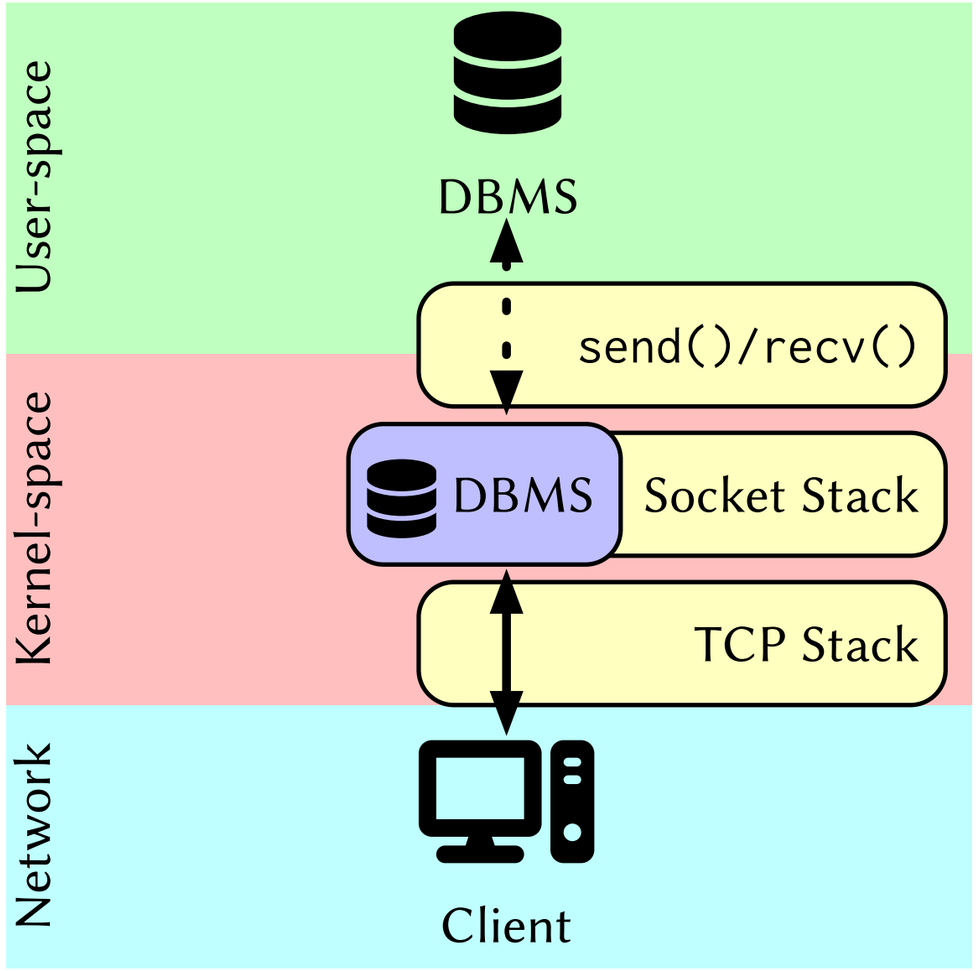
https://github.com/xrp-project/BPF-KV/issues/3

User-Bypass



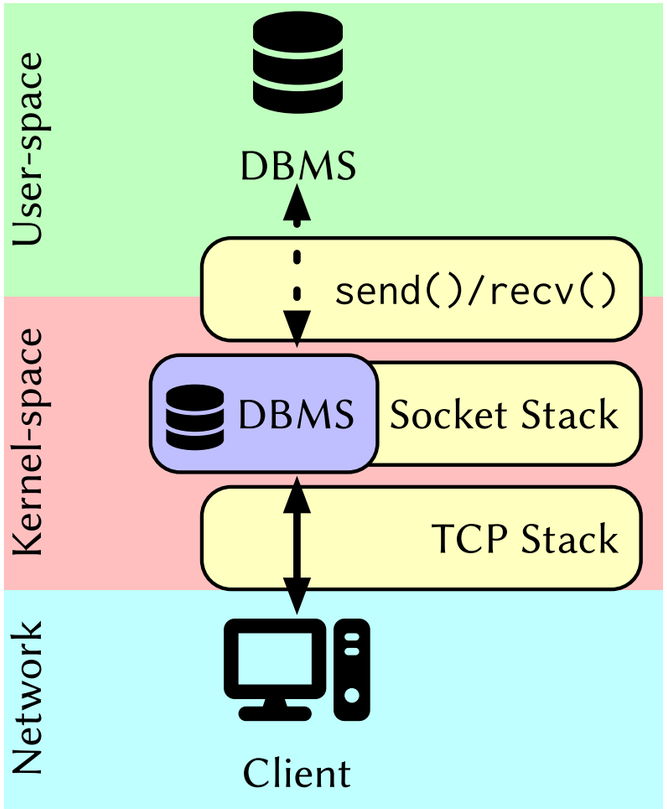
User-space DBMS

User-Bypass



User-bypass DBMS

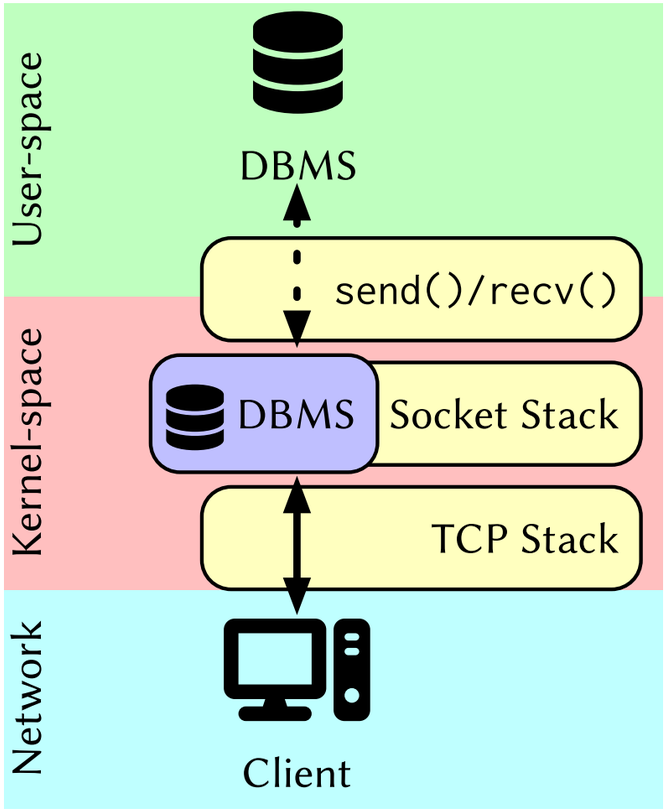
User-Bypass



User-bypass DBMS

User-Bypass

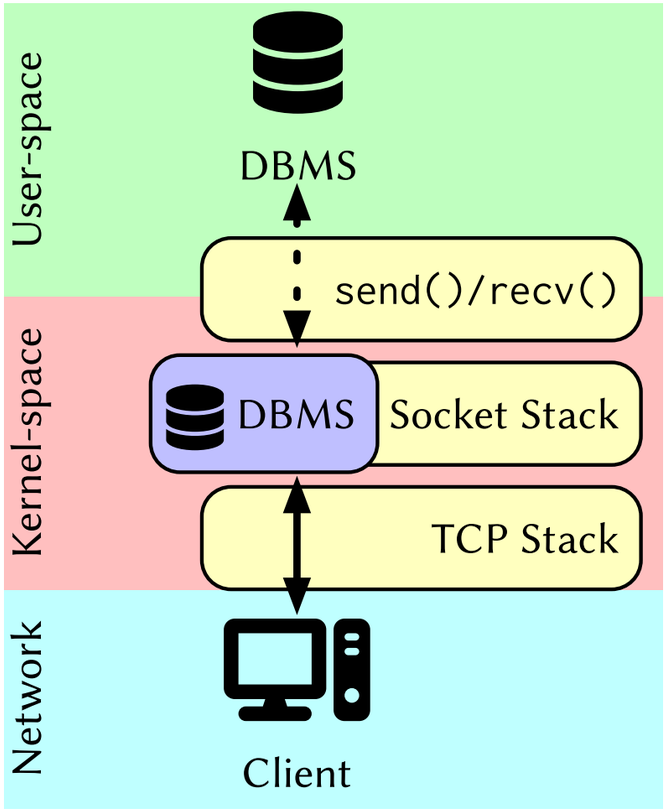
- Don’t pull DBMS data to user-space, push DBMS logic to kernel-space



User-bypass DBMS

User-Bypass

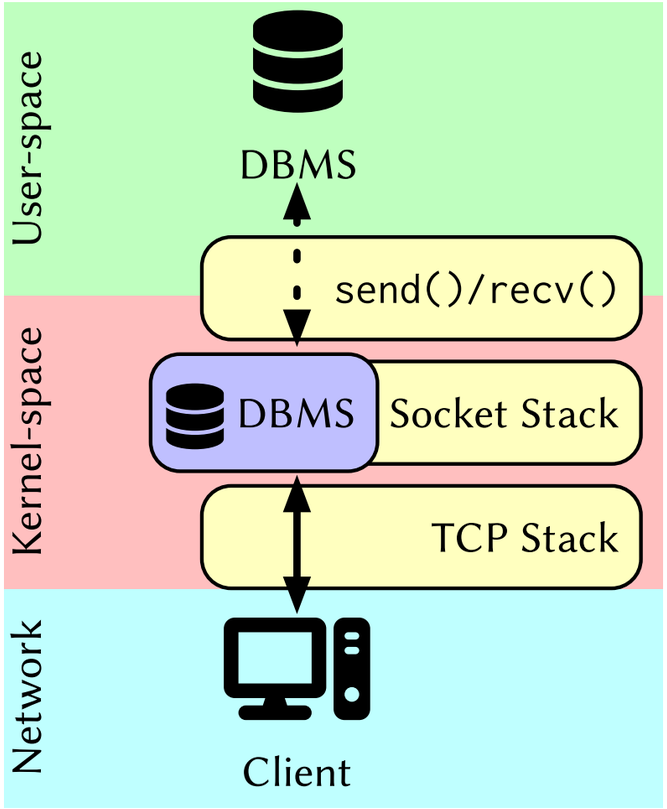
- Don’t pull DBMS data to user-space, push DBMS logic to kernel-space
- Avoid copying buffers, scheduling user threads, and system call overhead



User-bypass DBMS

User-Bypass

- Don’t pull DBMS data to user-space, push DBMS logic to kernel-space
- Avoid copying buffers, scheduling user threads, and system call overhead



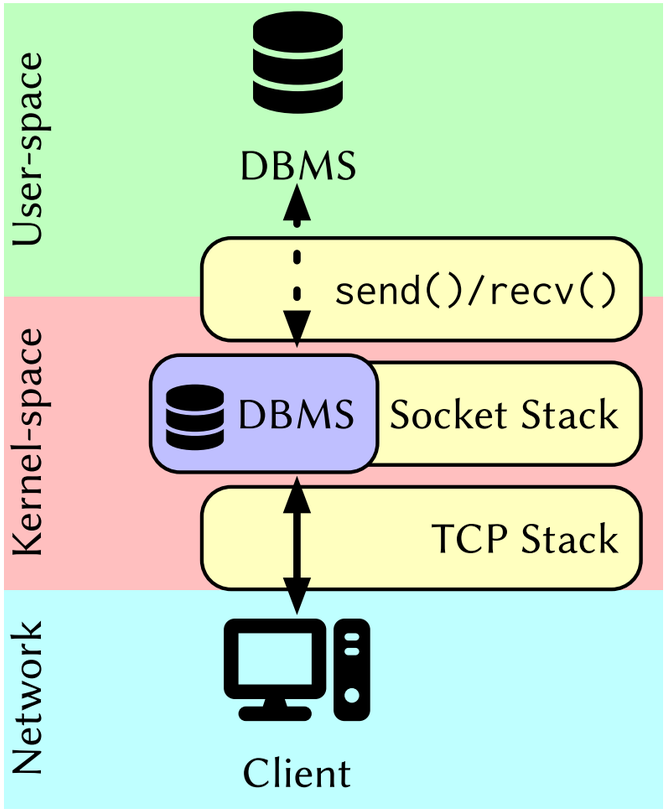
User-bypass DBMS

Brian N. Bershad et al. Extensibility, Safety and Performance in the SPIN Operating System. *SOSP*. 1995.

Greg Ganger et al. Fast and flexible application-level networking on exokernel systems. *ACM Trans. Comput. Syst*. 2002.

Margo I. Seltzer et al. Dealing with Disaster: Surviving Misbehaved Kernel Extensions. *OSDI*. 1996.

Applying User-Bypass



User-bypass DBMS

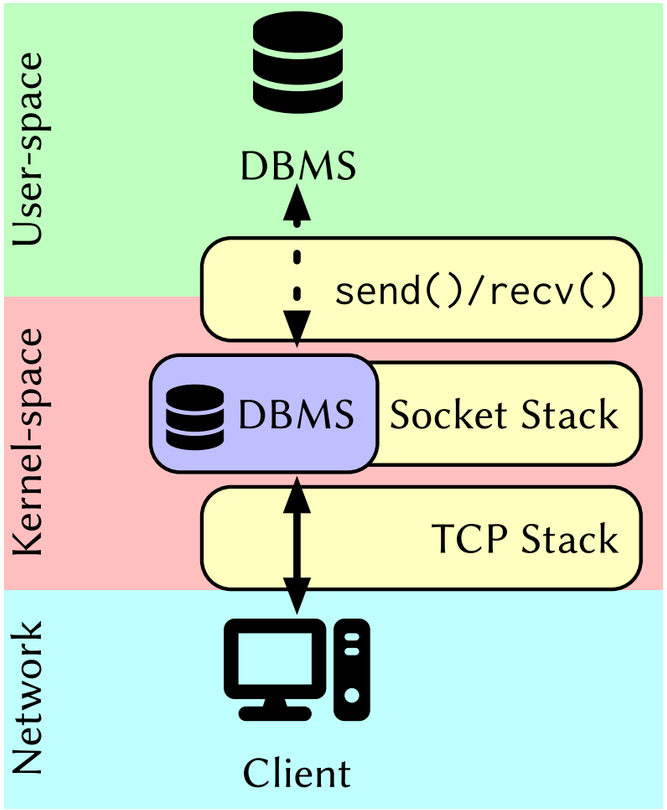
Brian N. Bershad et al. Extensibility, Safety and Performance in the SPIN Operating System. *SOSP*. 1995.

Greg Ganger et al. Fast and flexible application-level networking on exokernel systems. *ACM Trans. Comput. Syst*. 2002.

Margo I. Seltzer et al. Dealing with Disaster: Surviving Misbehaved Kernel Extensions. *OSDI*. 1996.

Applying User-Bypass

- Lack of standard API



User-bypass DBMS

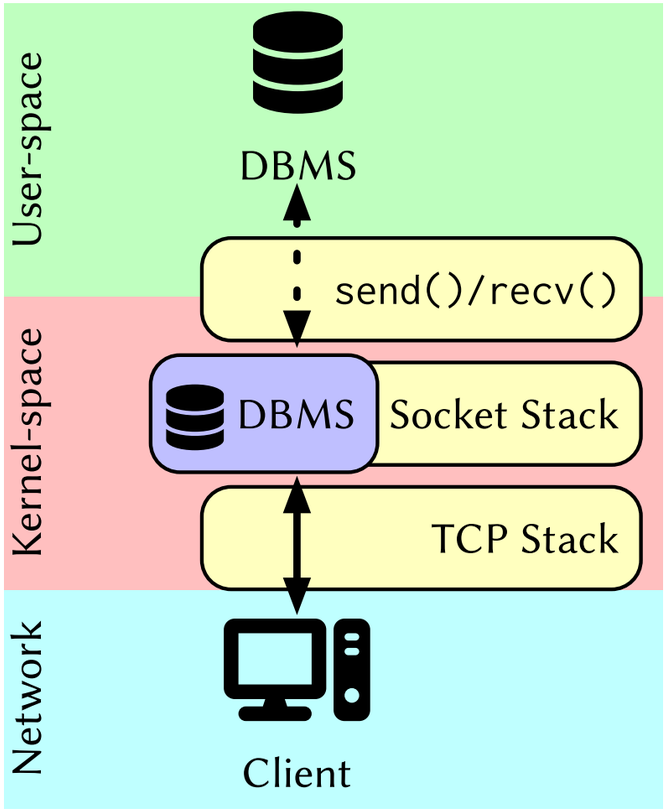
Brian N. Bershad et al. Extensibility, Safety and Performance in the SPIN Operating System. *SOSP*. 1995.

Greg Ganger et al. Fast and flexible application-level networking on exokernel systems. *ACM Trans. Comput. Syst*. 2002.

Margo I. Seltzer et al. Dealing with Disaster: Surviving Misbehaved Kernel Extensions. *OSDI*. 1996.

Applying User-Bypass

- Lack of standard API
- Stability and security issues



User-bypass DBMS

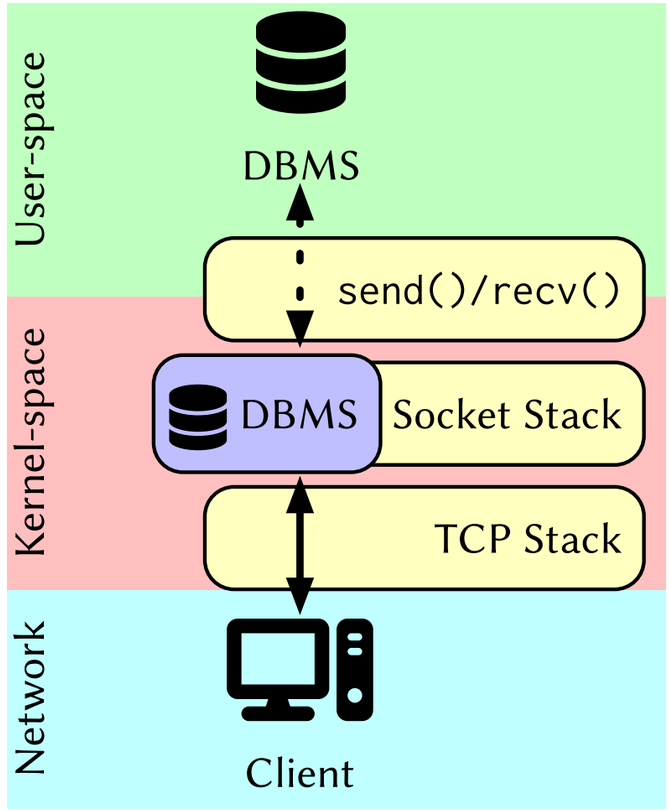
Brian N. Bershad et al. Extensibility, Safety and Performance in the SPIN Operating System. *SOSP*. 1995.

Greg Ganger et al. Fast and flexible application-level networking on exokernel systems. *ACM Trans. Comput. Syst*. 2002.

Margo I. Seltzer et al. Dealing with Disaster: Surviving Misbehaved Kernel Extensions. *OSDI*. 1996.

Applying User-Bypass

- Lack of standard API
- Stability and security issues
- Lack of OS adoption



User-bypass DBMS

Brian N. Bershad et al. Extensibility, Safety and Performance in the SPIN Operating System. *SOSP*. 1995.

Greg Ganger et al. Fast and flexible application-level networking on exokernel systems. *ACM Trans. Comput. Syst*. 2002.

Margo I. Seltzer et al. Dealing with Disaster: Surviving Misbehaved Kernel Extensions. *OSDI*. 1996.

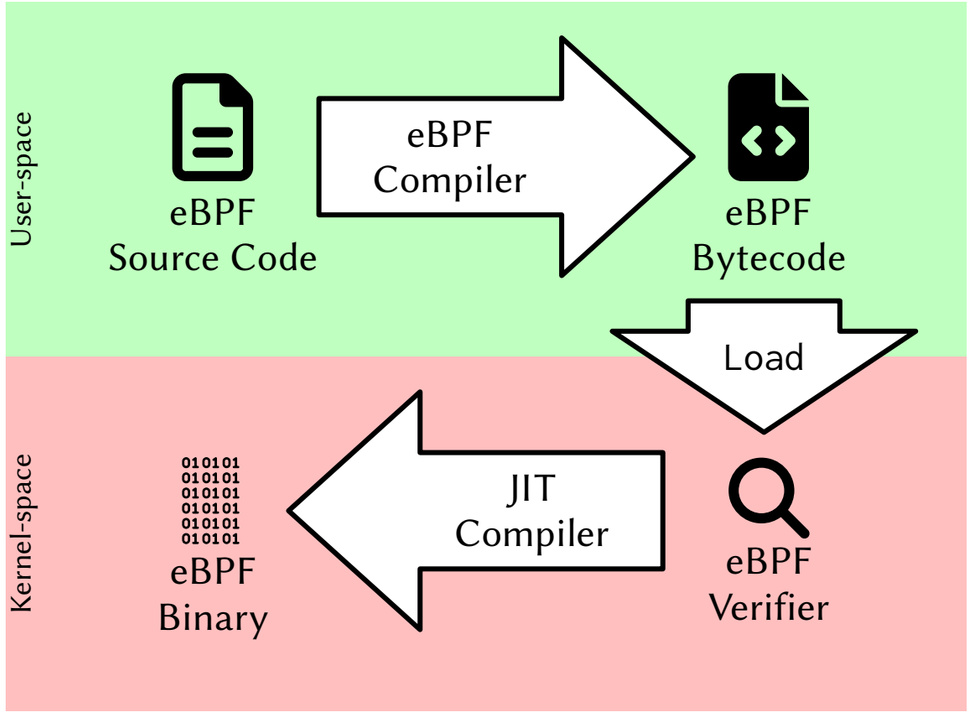
extended Berkeley Packet Filter

extended Berkeley Packet Filter

- Safe, event-driven programs in kernel-space

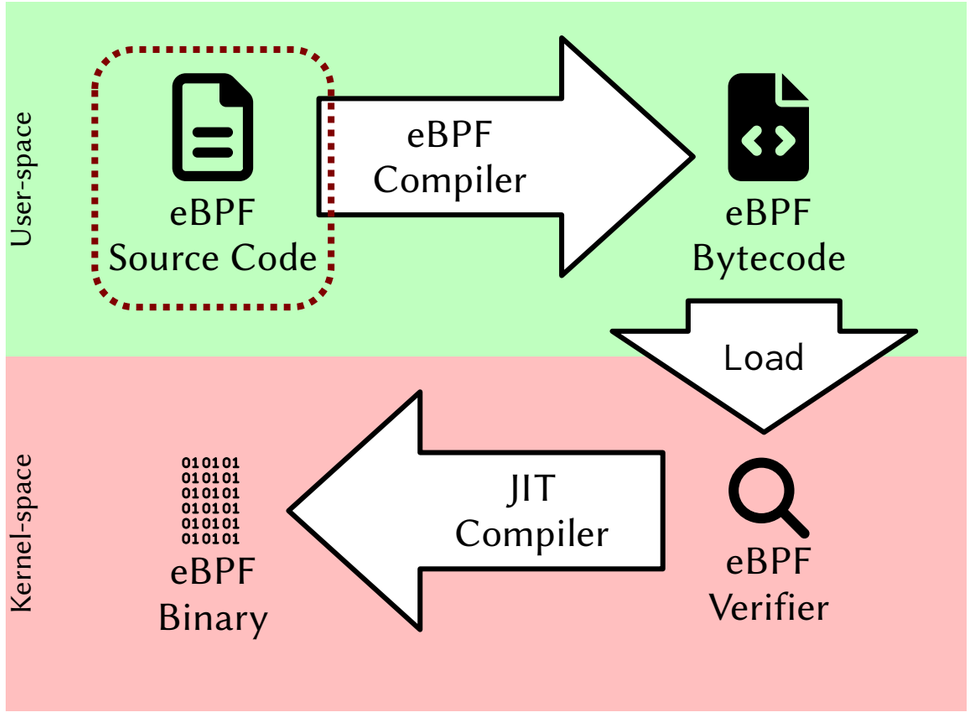
extended Berkeley Packet Filter

- Safe, event-driven programs in kernel-space



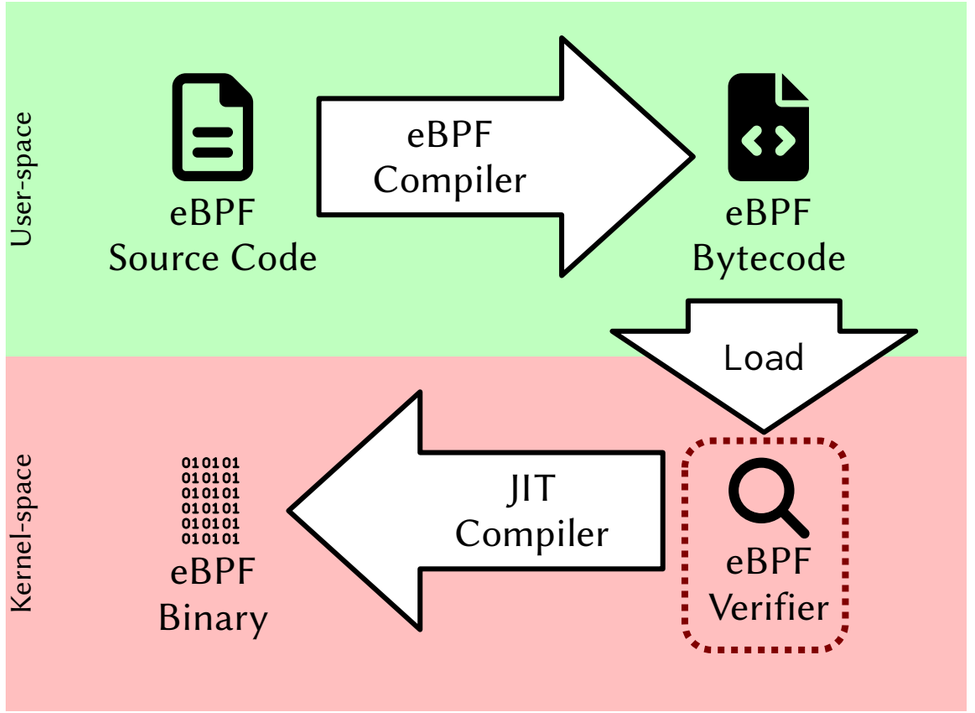
extended Berkeley Packet Filter

- Safe, event-driven programs in kernel-space
- Write in C and compile to eBPF



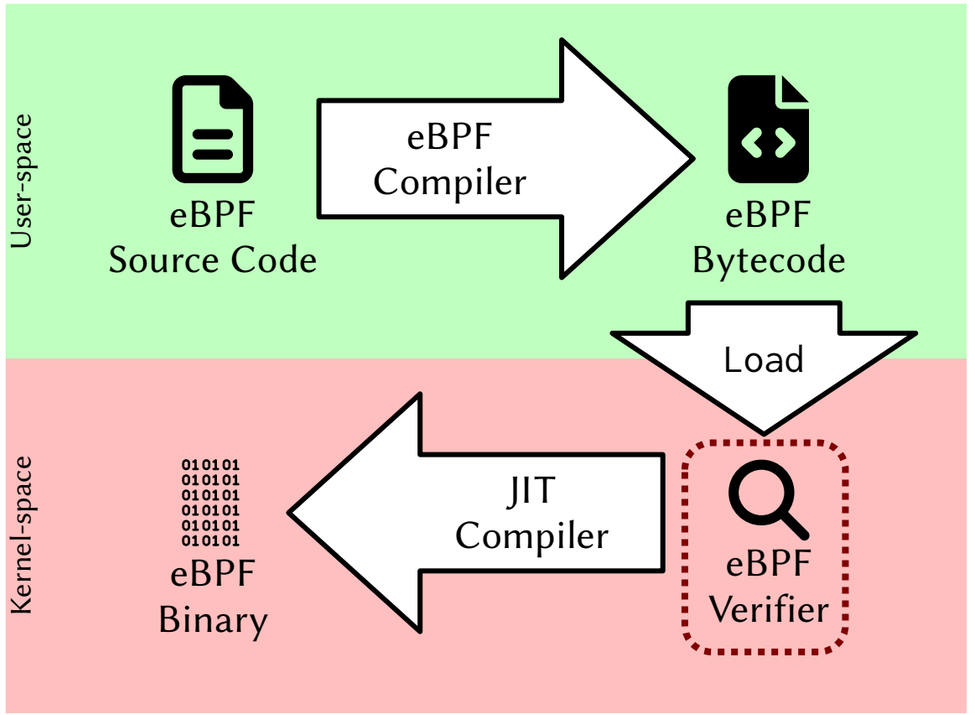
extended Berkeley Packet Filter

- Safe, event-driven programs in kernel-space
- Write in C and compile to eBPF



extended Berkeley Packet Filter

- Safe, event-driven programs in kernel-space
- Write in C and compile to eBPF
- Verifier constraints:
  - # instructions, boundedness, memory safety, limited API



eBPF Environment

eBPF Environment

- Attach to user-space or kernel-space hooks

eBPF Environment

- Attach to user-space or kernel-space hooks
  - User-space ⇒ “new system call”

eBPF Environment

- Attach to user-space or kernel-space hooks
  - User-space ⇒ “new system call”
  - Kernel-space ⇒ observe/modify OS logic

eBPF Environment

- Attach to user-space or kernel-space hooks
  - User-space ⇒ “new system call”
  - Kernel-space ⇒ observe/modify OS logic

- Ephemeral program execution

eBPF Environment

- Attach to user-space or kernel-space hooks
  - User-space ⇒ “new system call”
  - Kernel-space ⇒ observe/modify OS logic
- Ephemeral program execution
- eBPF maps: kernel-resident data structures

eBPF Environment

- Attach to user-space or kernel-space hooks
  - User-space ⇒ “new system call”
  - Kernel-space ⇒ observe/modify OS logic
- Ephemeral program execution
- eBPF maps: kernel-resident data structures
  - Key-value interface

eBPF Environment

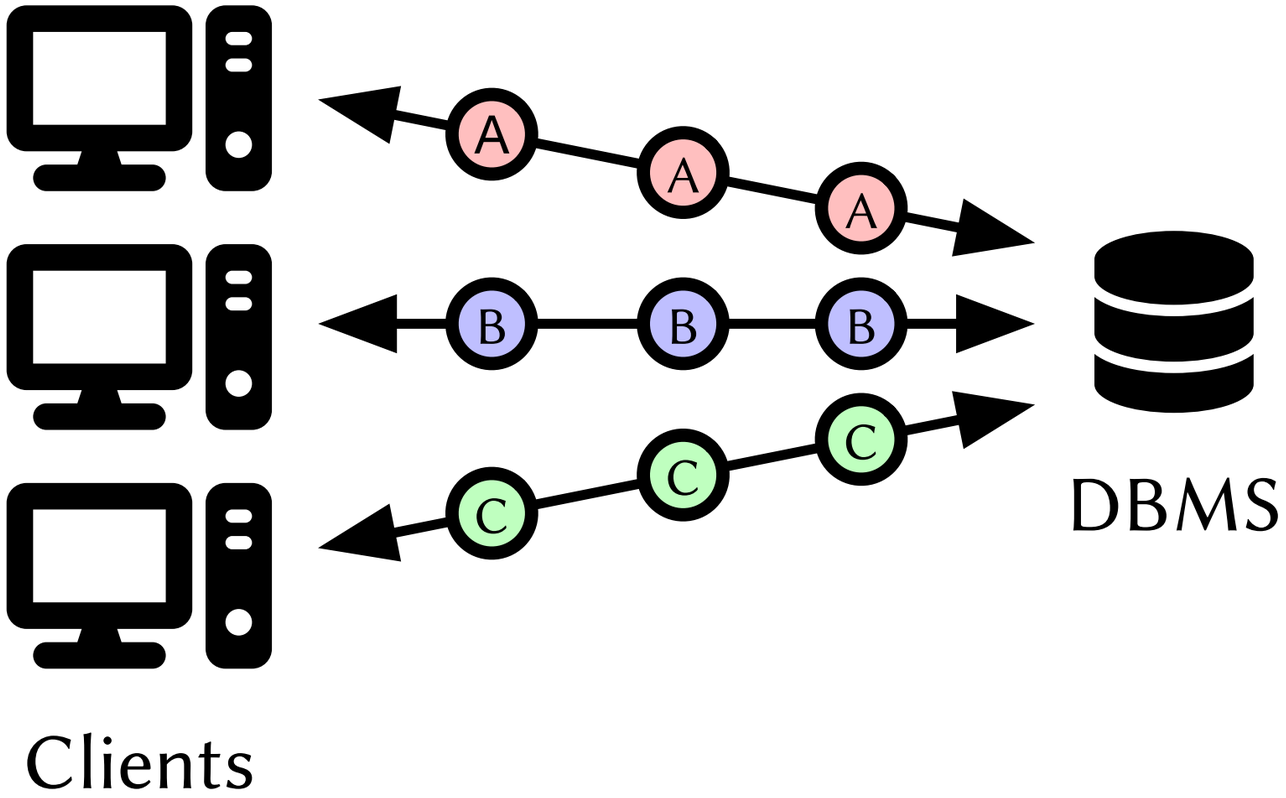
- Attach to user-space or kernel-space hooks
  - User-space ⇒ “new system call”
  - Kernel-space ⇒ observe/modify OS logic
- Ephemeral program execution
- eBPF maps: kernel-resident data structures
  - Key-value interface
  - Hash tables, stacks/queues, arrays, etc.

Outline

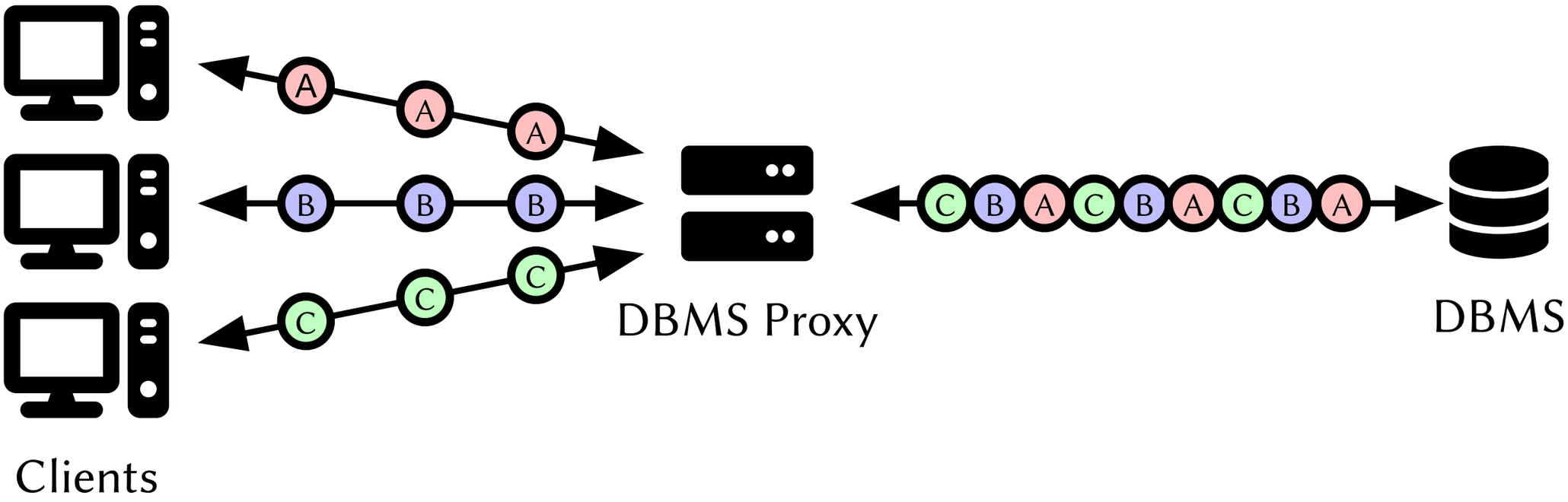
- User-bypass
- Prior work: User-bypass for DBMS proxies
- Future work: User-bypass DBMS

Butrovich et al. Tigger: A Database Proxy That Bounces With User-Bypass. *VLDB*. 2023.

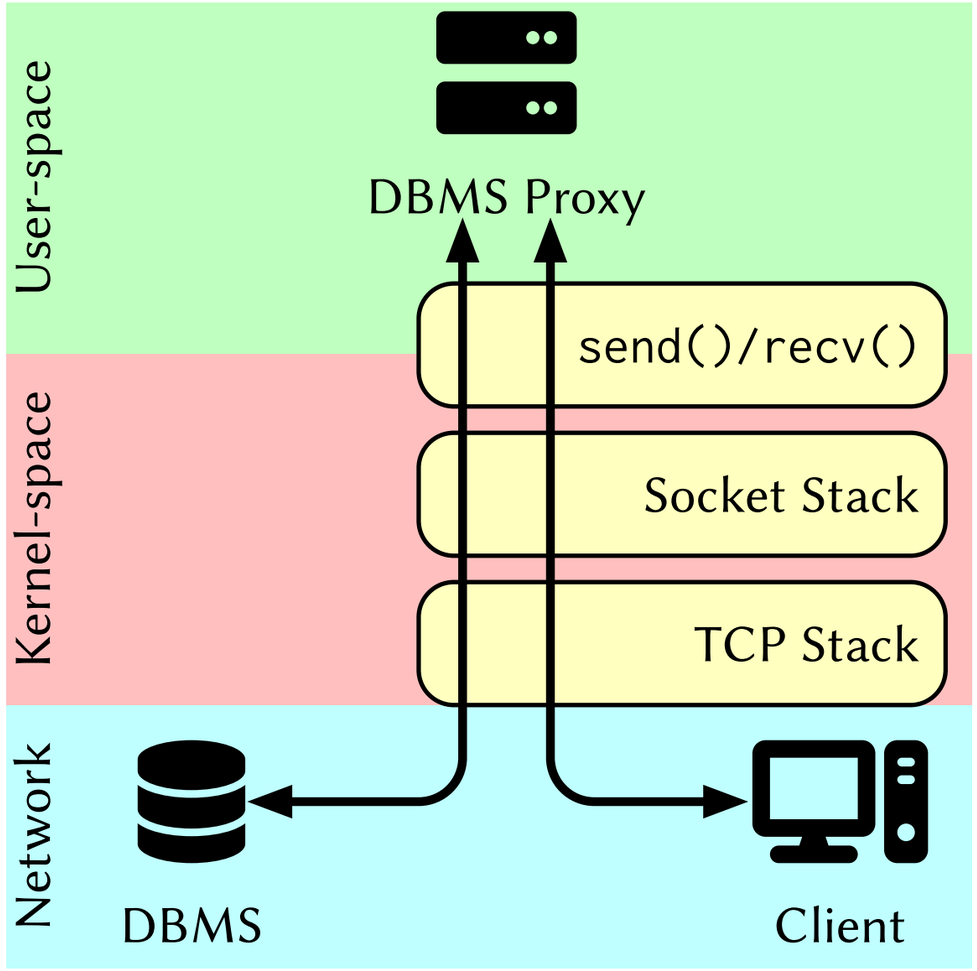
Connection Pooling with DBMS Proxies



Connection Pooling with DBMS Proxies

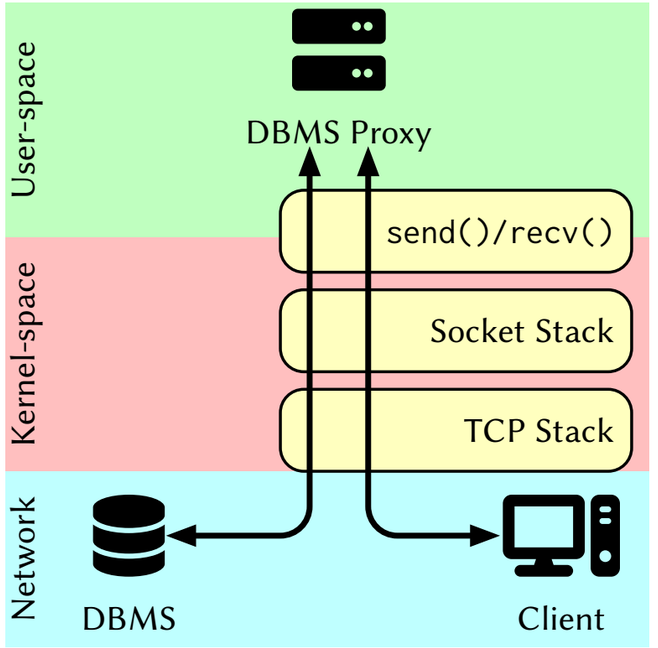


User-Space DBMS Proxy



User-space DBMS Proxy

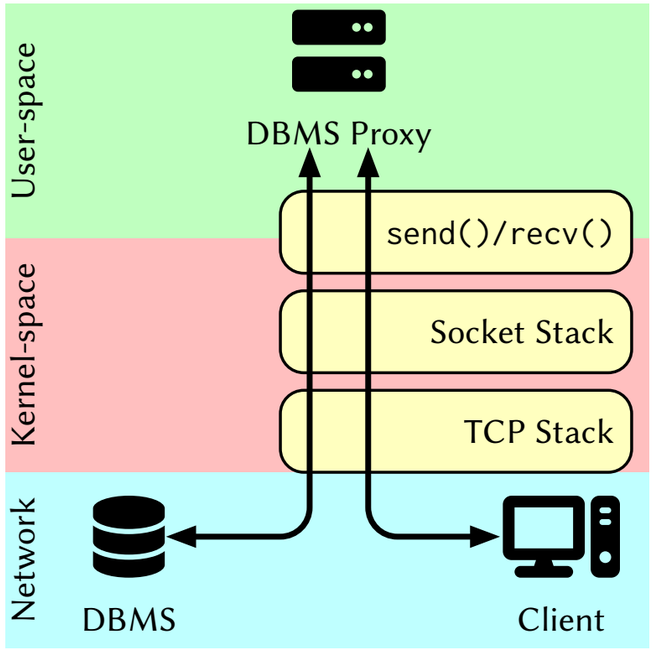
User-Space DBMS Proxy



User-space DBMS proxy

User-Space DBMS Proxy

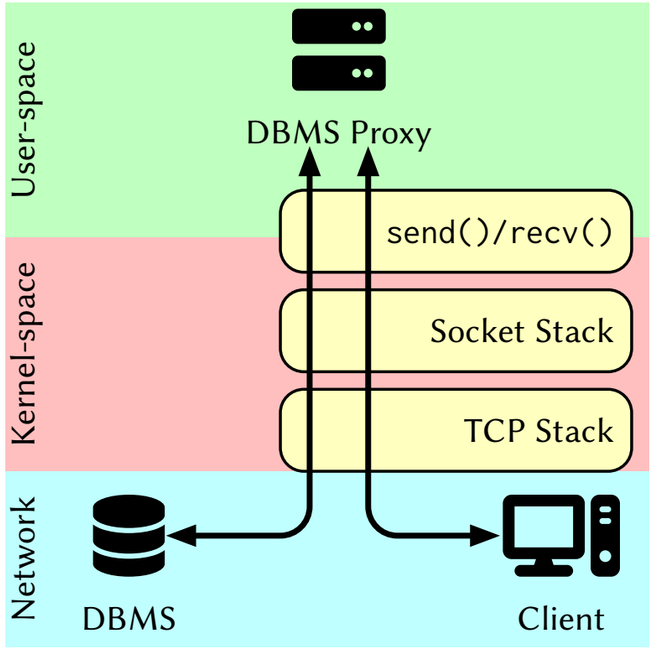
- Traffic goes through OS network stack to apply DBMS protocol logic



User-space DBMS proxy

User-Space DBMS Proxy

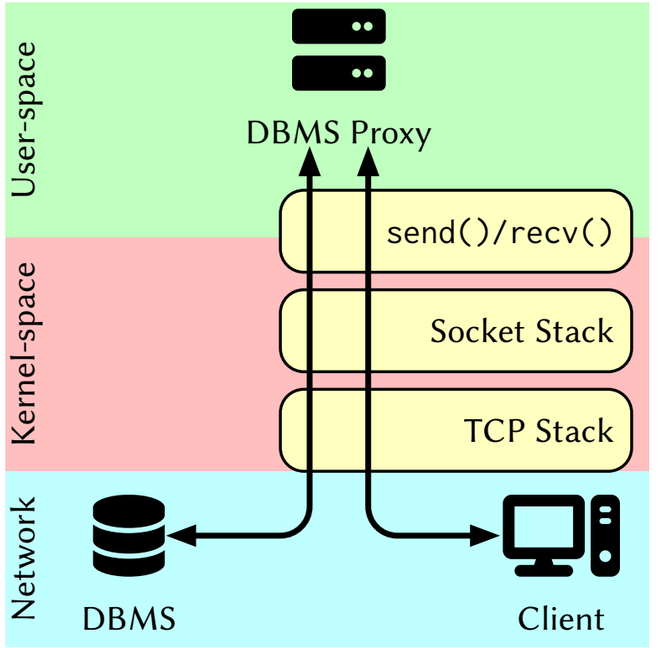
- Traffic goes through OS network stack to apply DBMS protocol logic
- User-space applications of varying complexity to express parallelism



User-space DBMS proxy

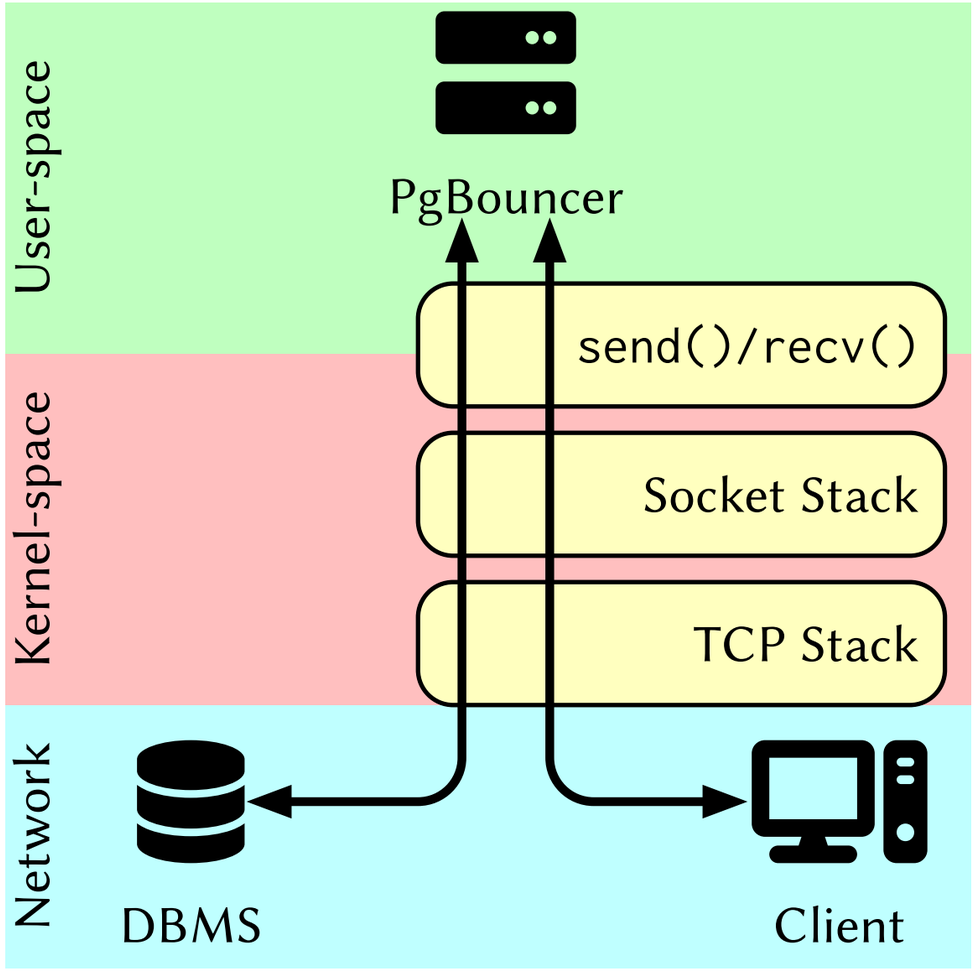
User-Space DBMS Proxy

- 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



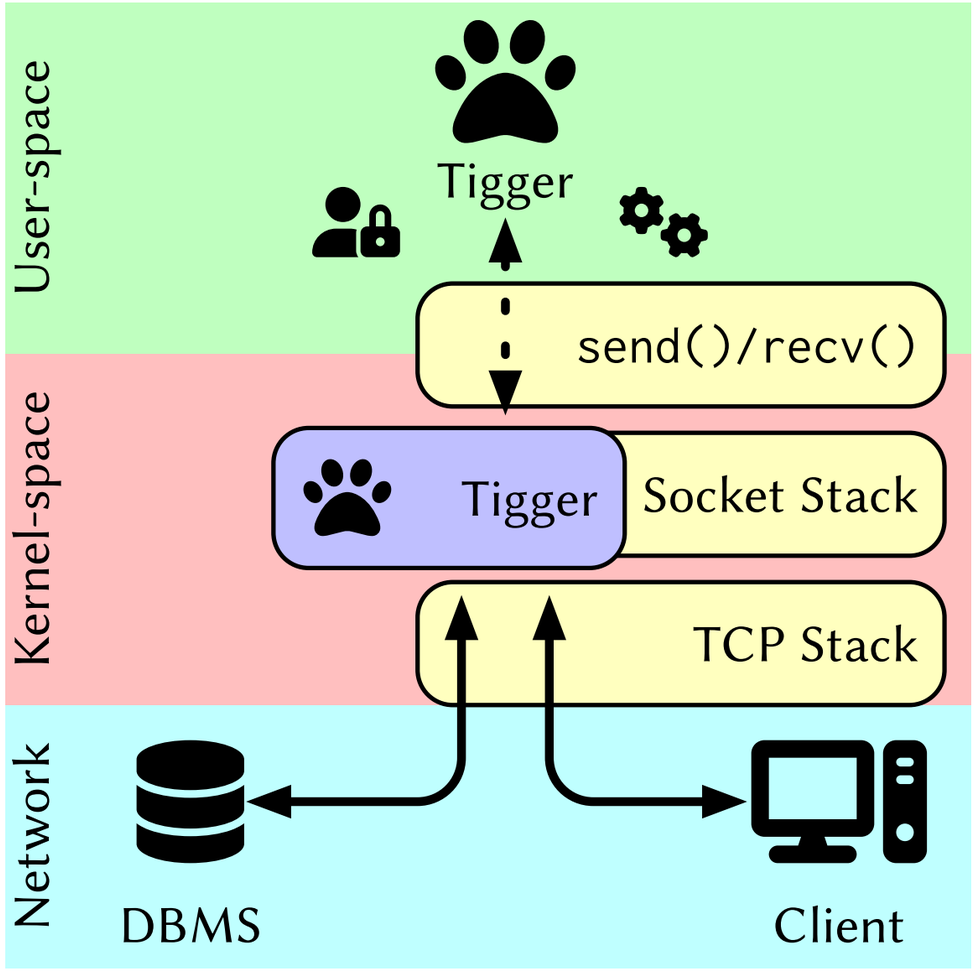
User-space DBMS proxy

Tigger DBMS Proxy



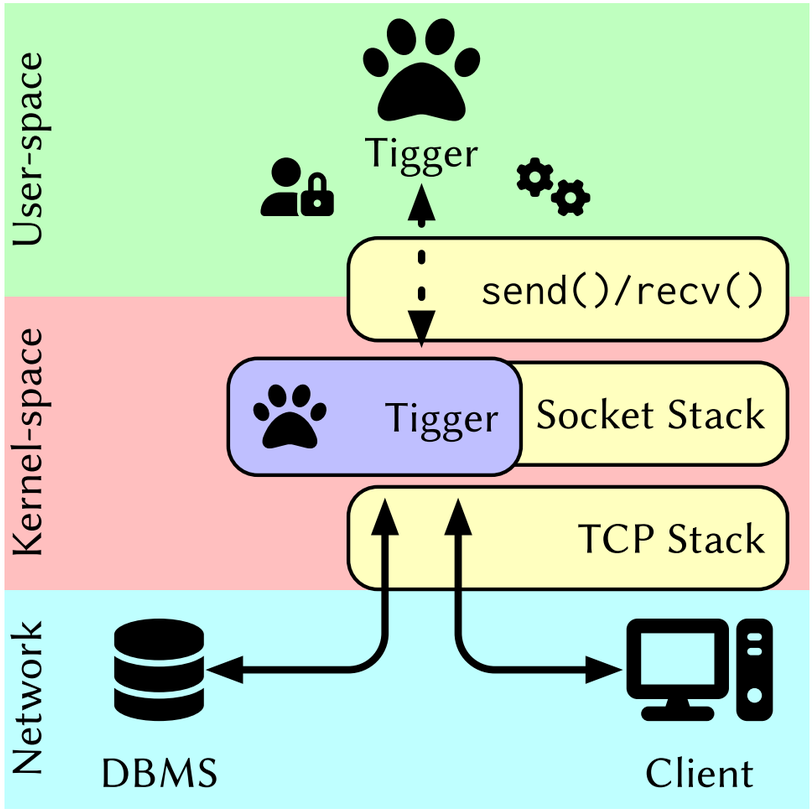
User-space DBMS Proxy

Tigger DBMS Proxy



User-Bypass DBMS Proxy

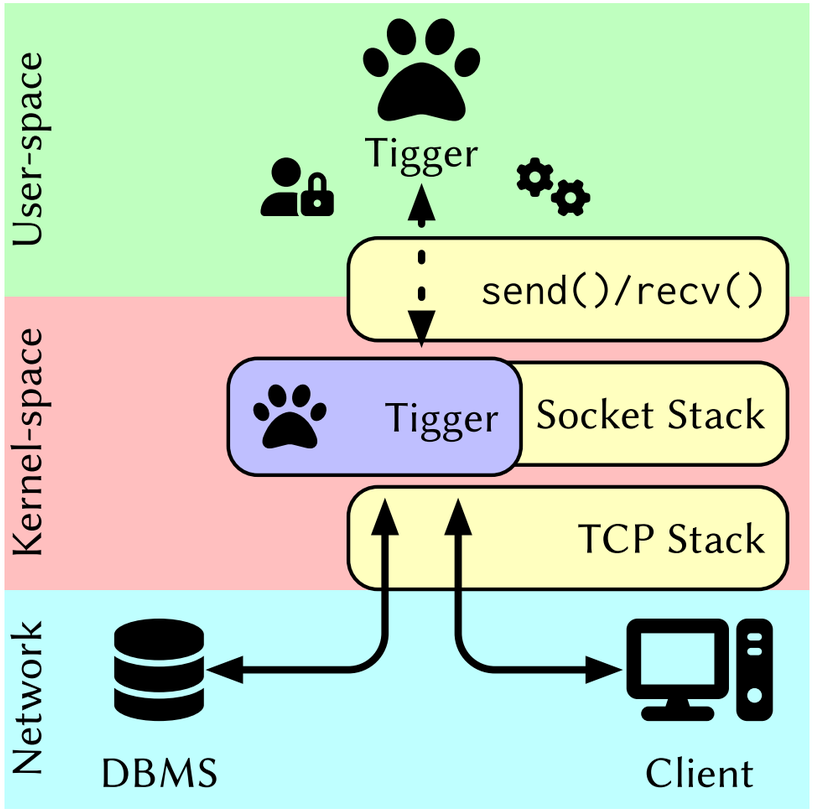
Tigger DBMS Proxy



User-Bypass DBMS Proxy

Tigger DBMS Proxy

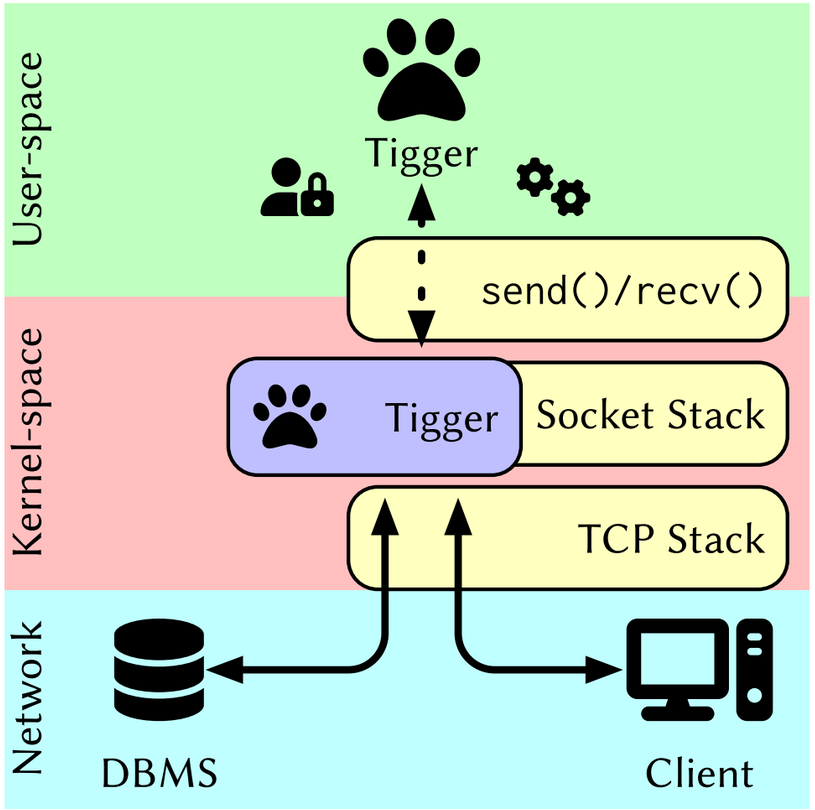
- Frequent operations use User-bypass:
  - Transaction-aware pooling
  - Workload replication



User-Bypass DBMS Proxy

Tigger DBMS Proxy

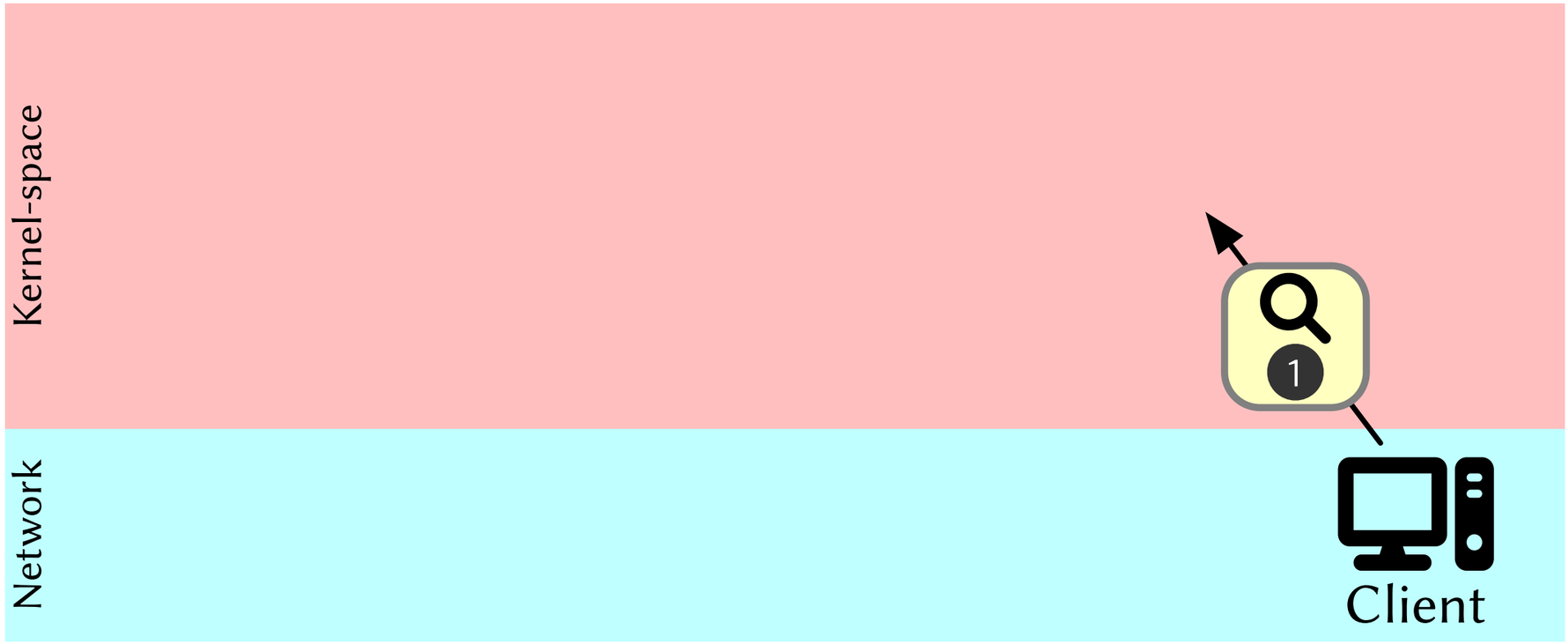
- Frequent operations use User-bypass:
  - Transaction-aware pooling
  - Workload replication
- User-space operations:
  - Authentication
  - Settings



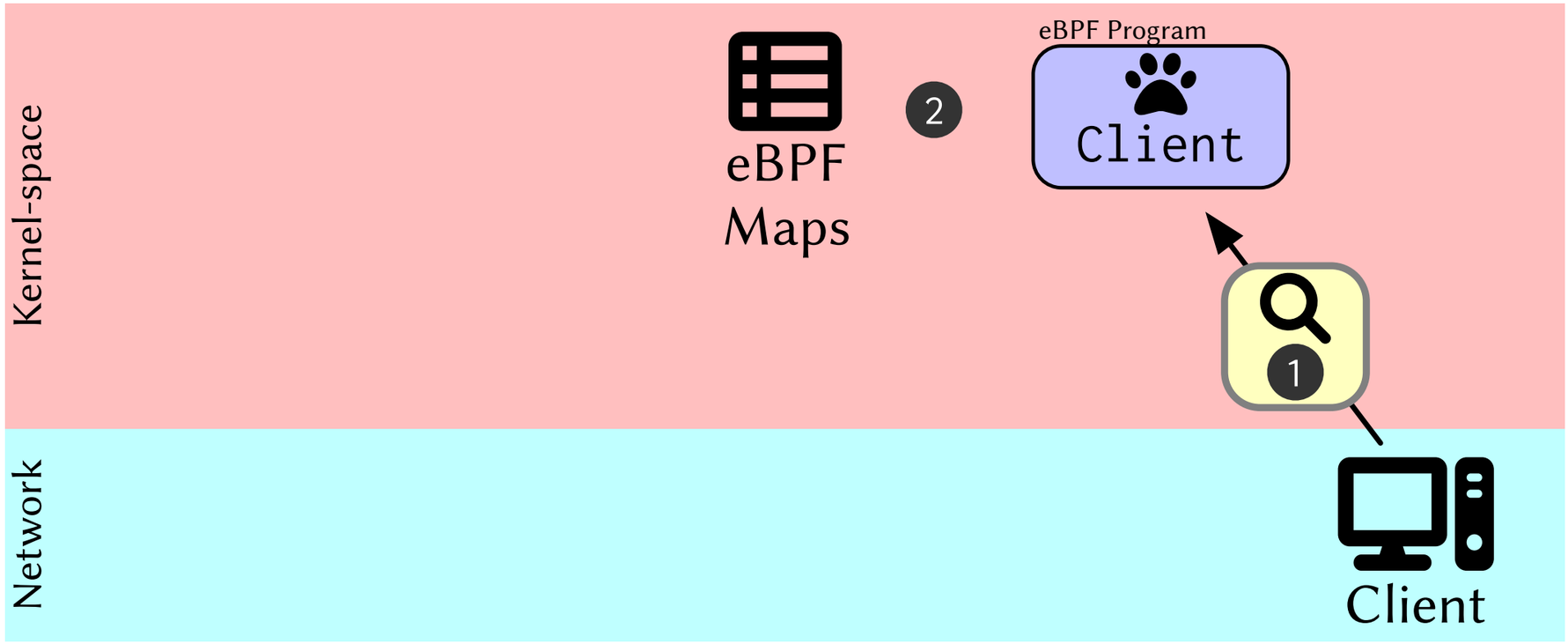
User-Bypass DBMS Proxy

Tigger Connection Pooling

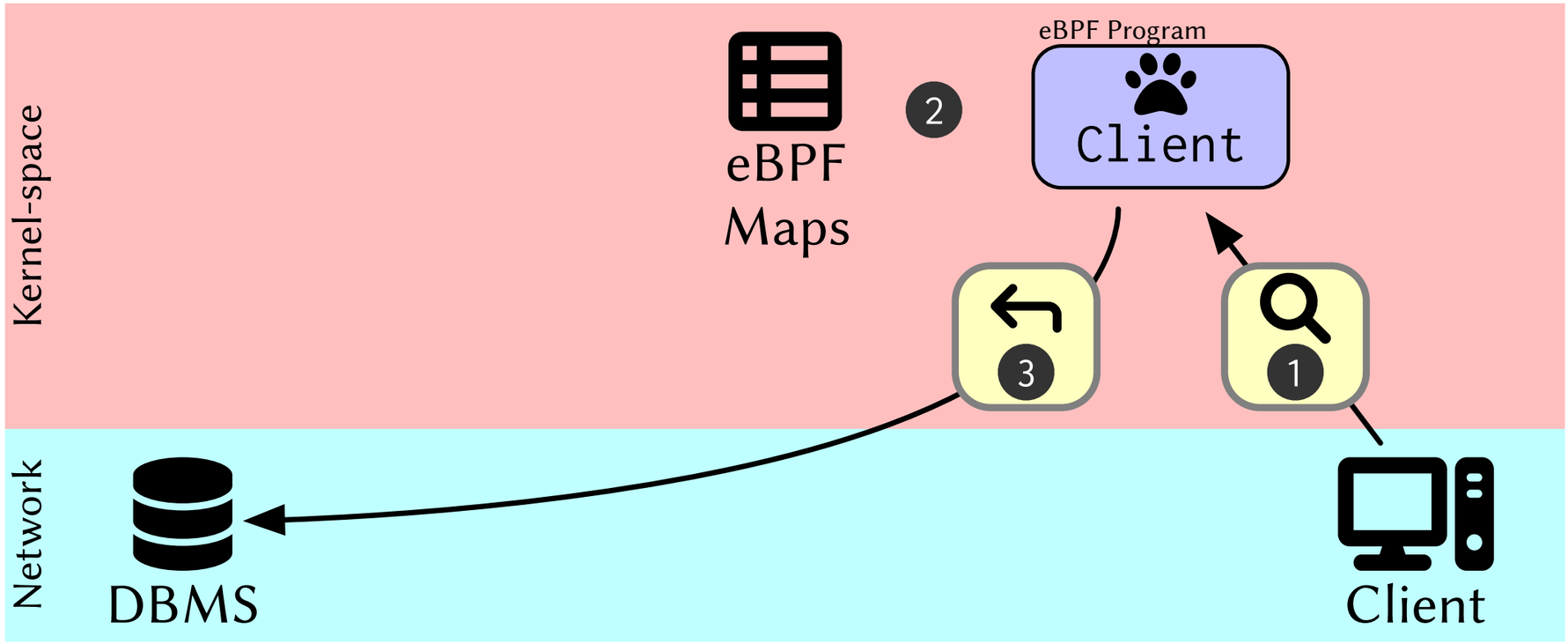
Tigger Connection Pooling



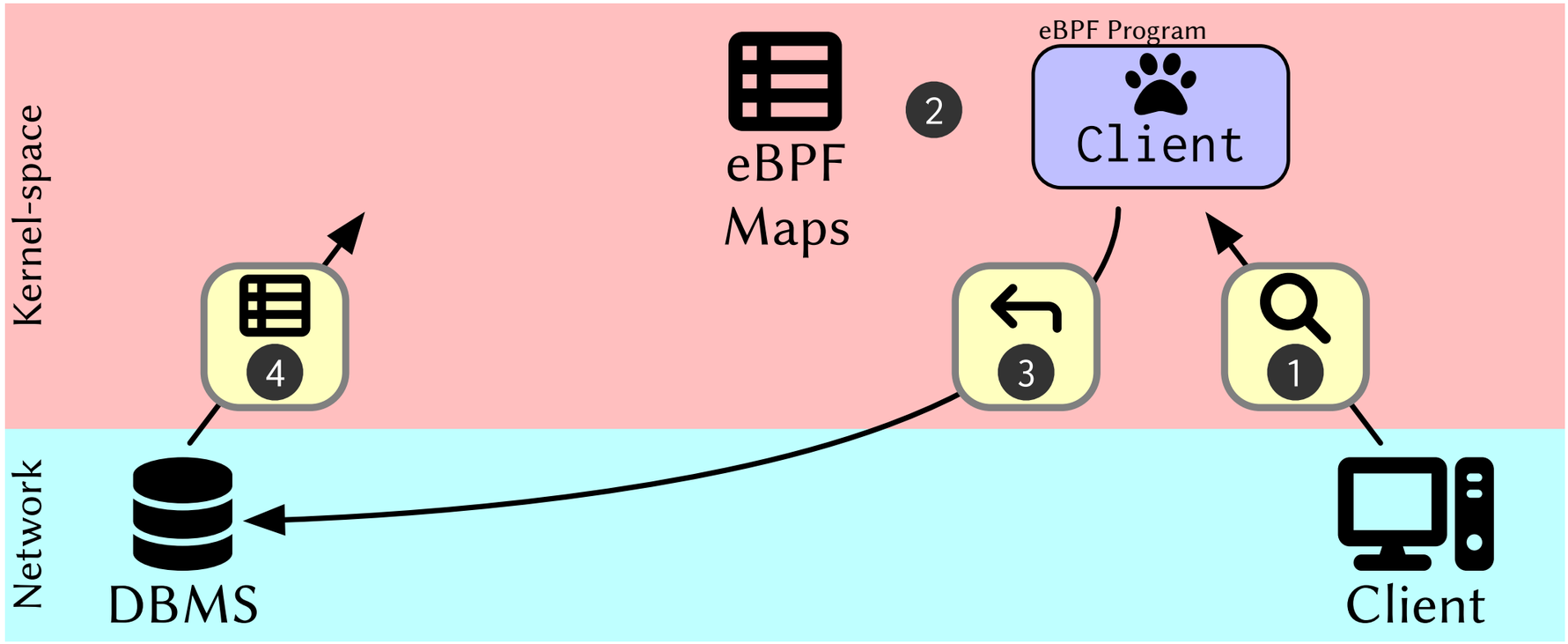
Tigger Connection Pooling



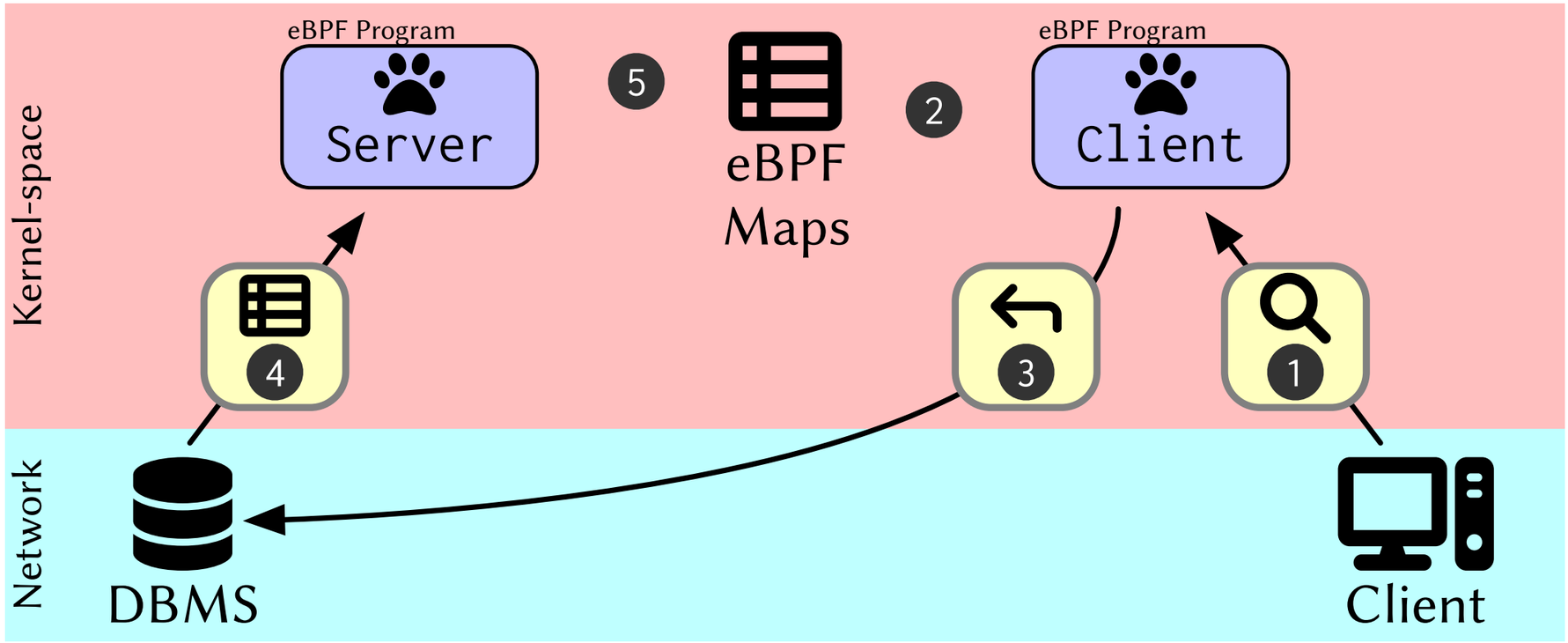
Tigger Connection Pooling



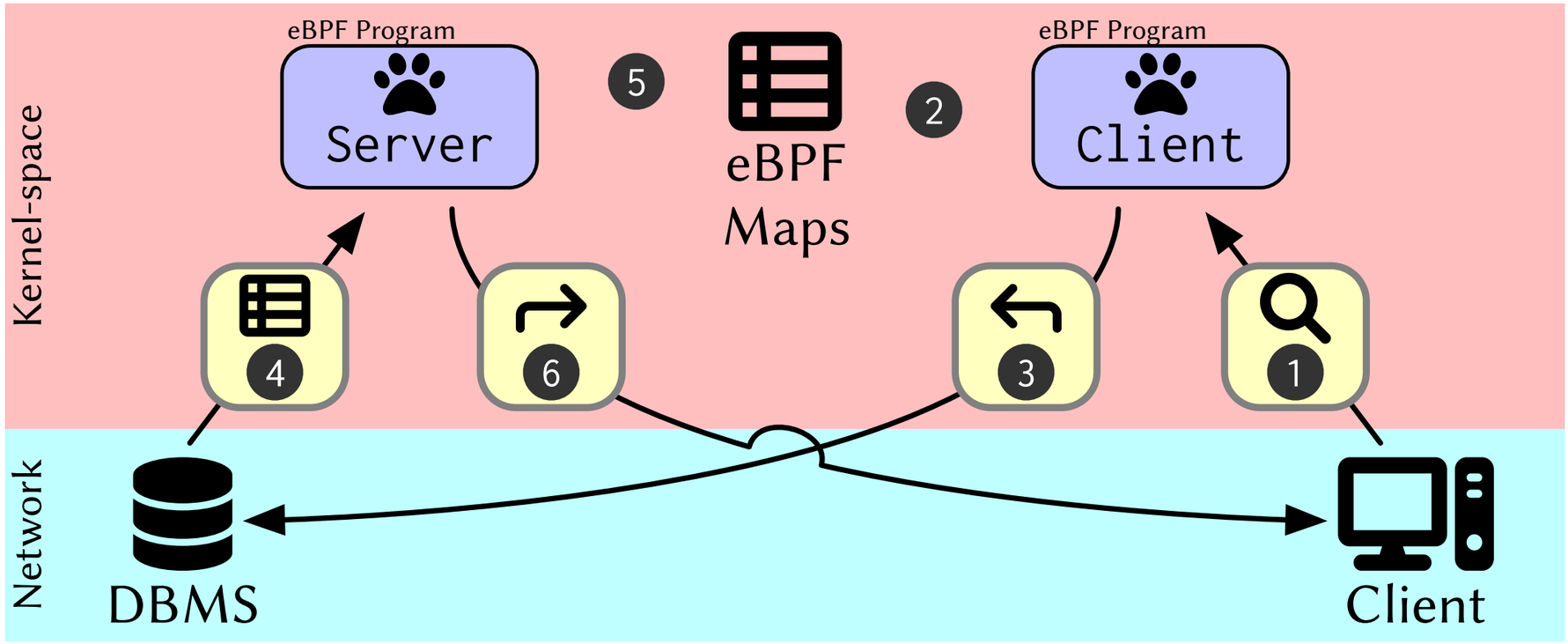
Tigger Connection Pooling



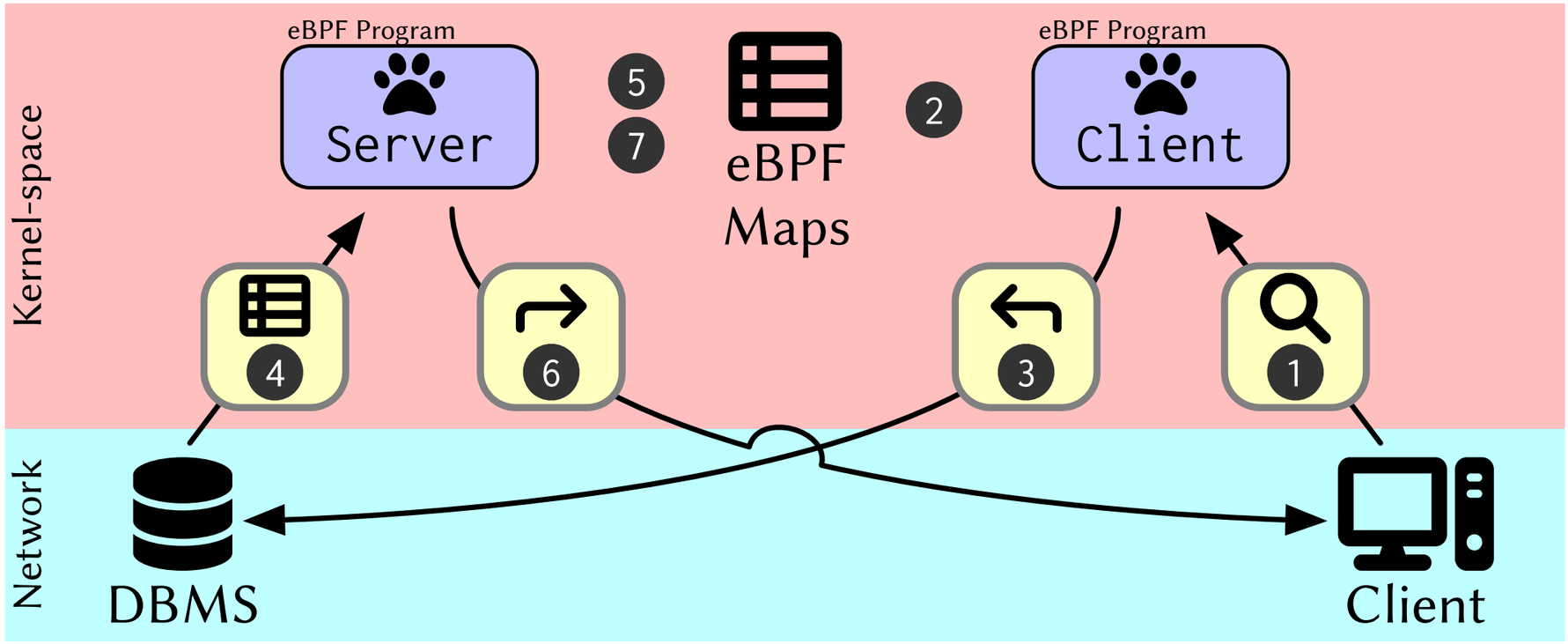
Tigger Connection Pooling



Tigger Connection Pooling



Tigger Connection Pooling



Experimental Setup

Experimental Setup

- Proxies:
  - PgBouncer
  - Yandex Odyssey
  - Tigger

Experimental Setup

- Proxies:
  - PgBouncer
  - Yandex Odyssey
  - Tigger

- Dedicated AWS EC2 c6i instances, PostgreSQL 14.5, Ubuntu 22.04 LTS

Experimental Setup

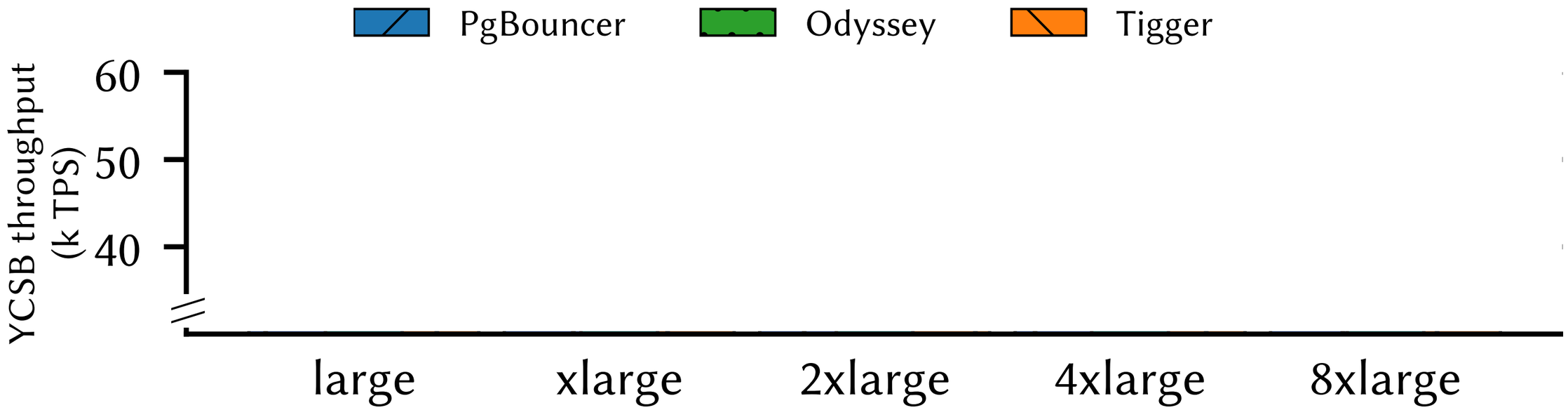
- Proxies:
  - PgBouncer
  - Yandex Odyssey
  - Tigger

- Dedicated AWS EC2 c6i instances, PostgreSQL 14.5, Ubuntu 22.04 LTS

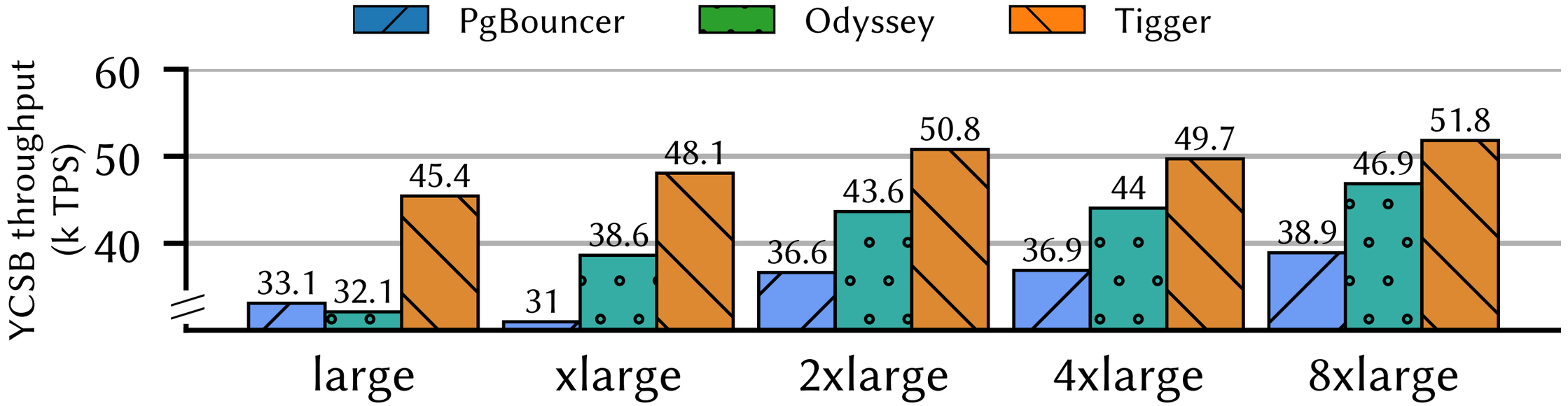
- BenchBase: YCSB

Connection Pooling Throughput

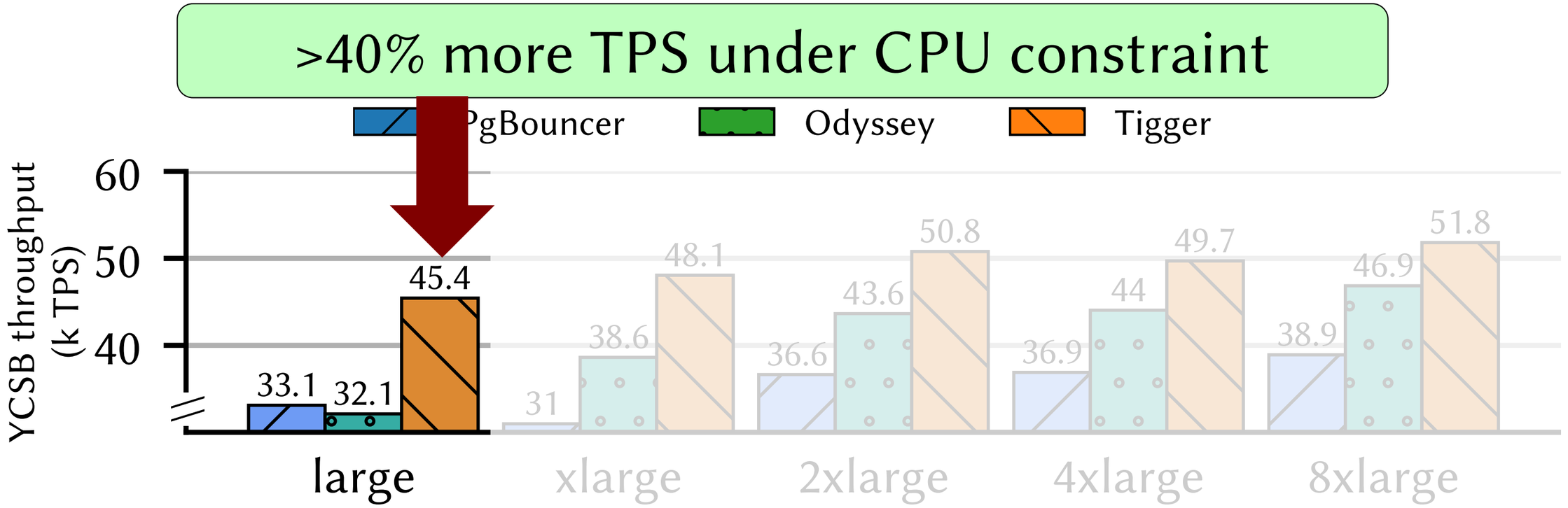
Connection Pooling Throughput



Connection Pooling Throughput

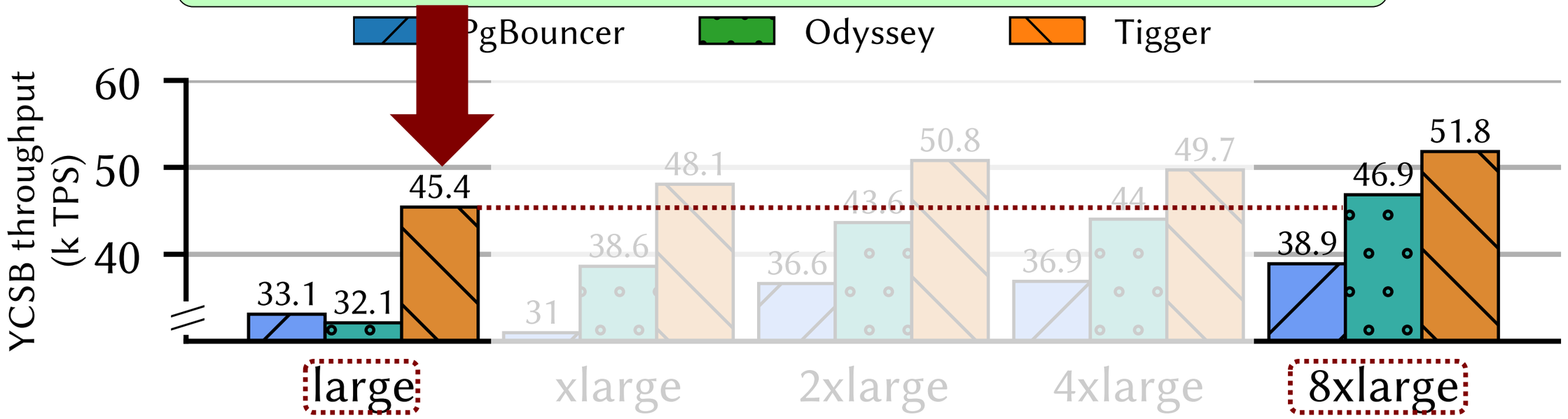


Connection Pooling Throughput



Connection Pooling Throughput

>40% more TPS under CPU constraint

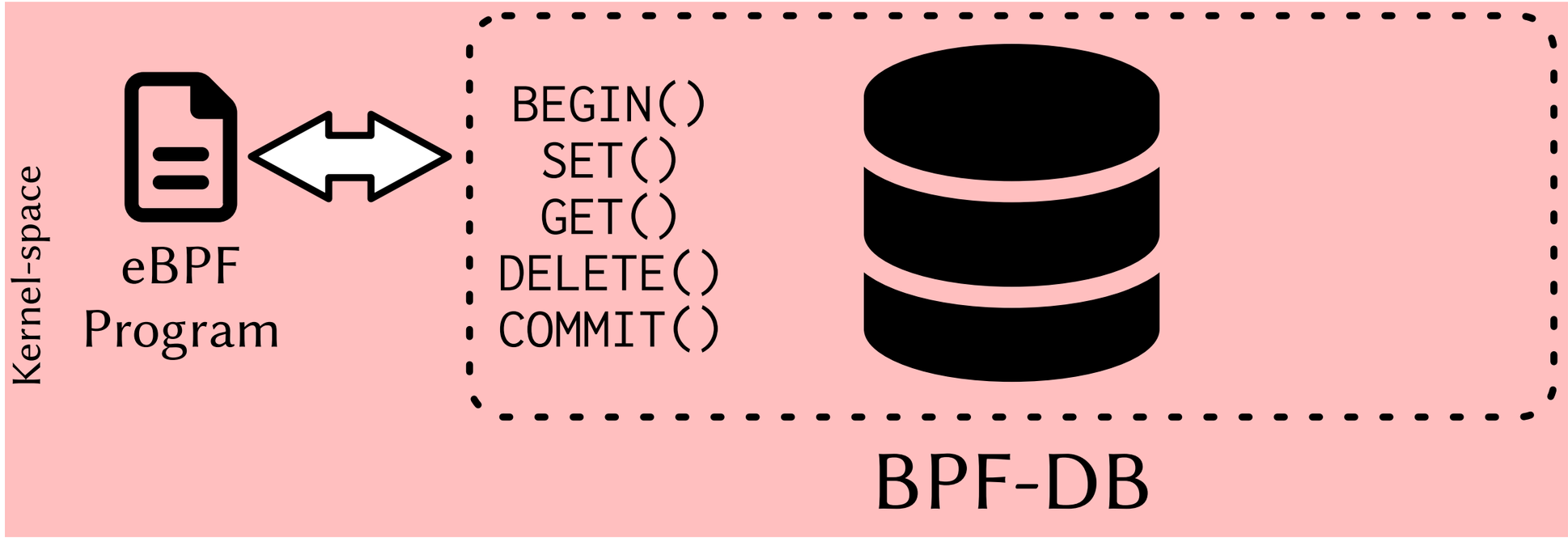


8x cost for Odyssey to match Tigger's performance

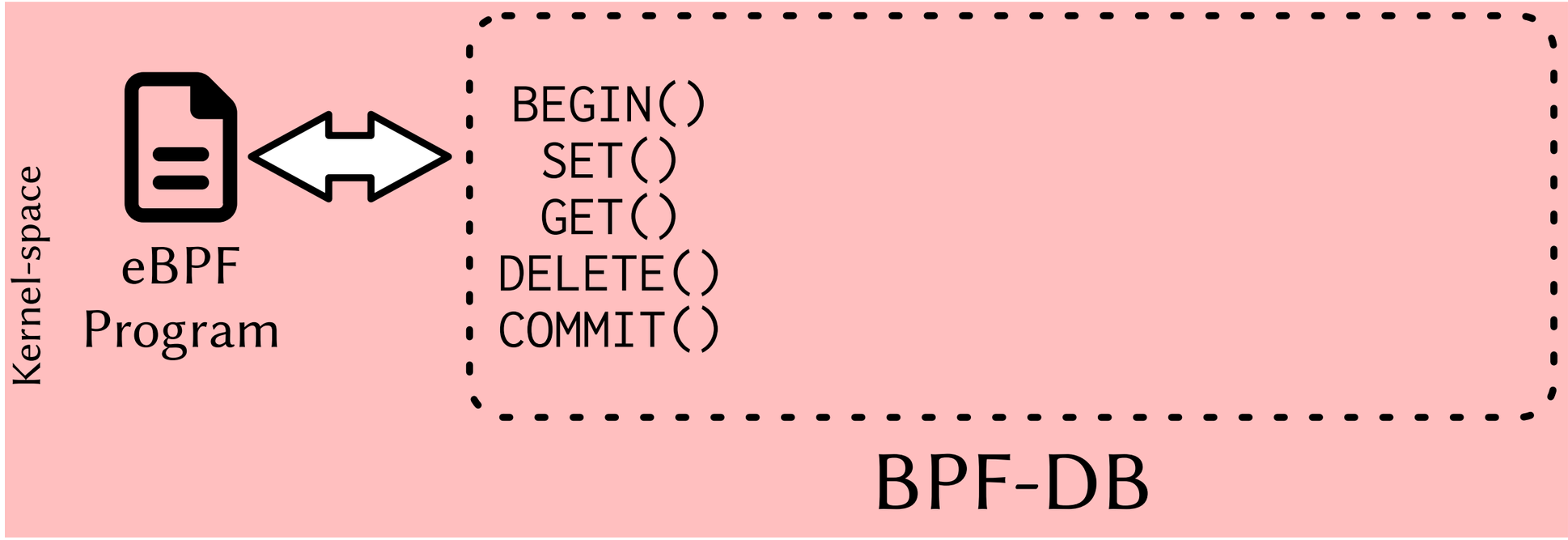
Outline

- User-bypass
- Prior work: User-bypass for DBMS proxies
- Future work: User-bypass DBMS

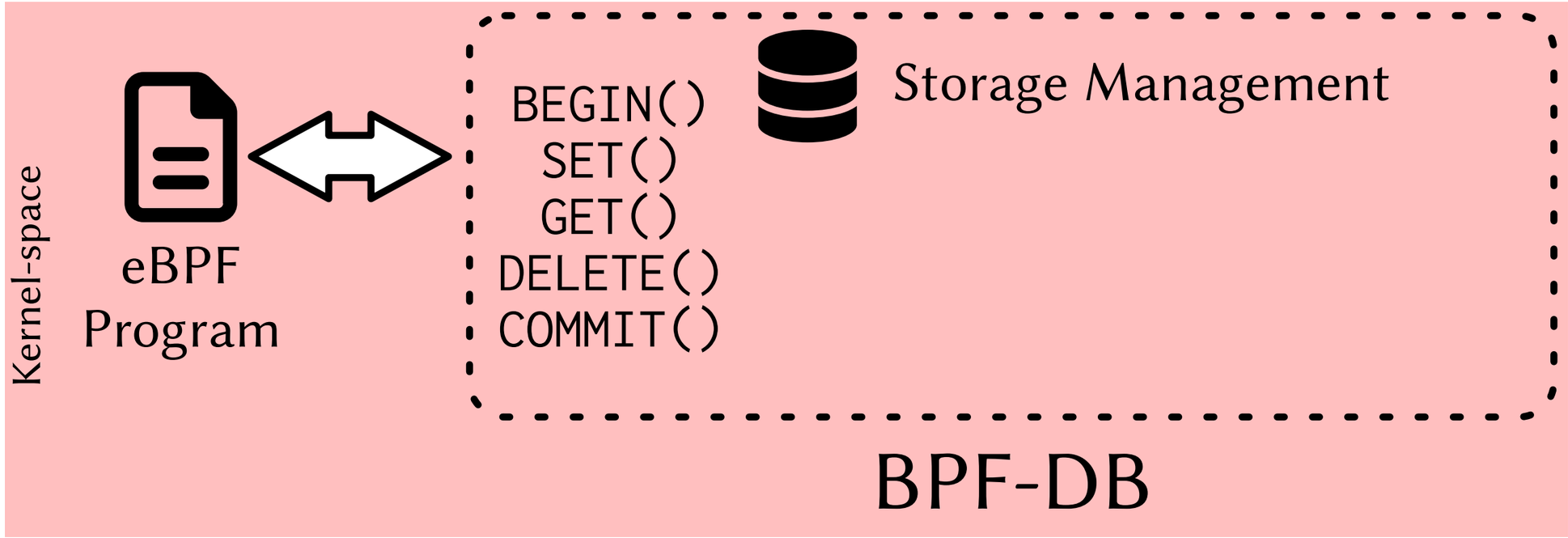
User-Bypass DBMS



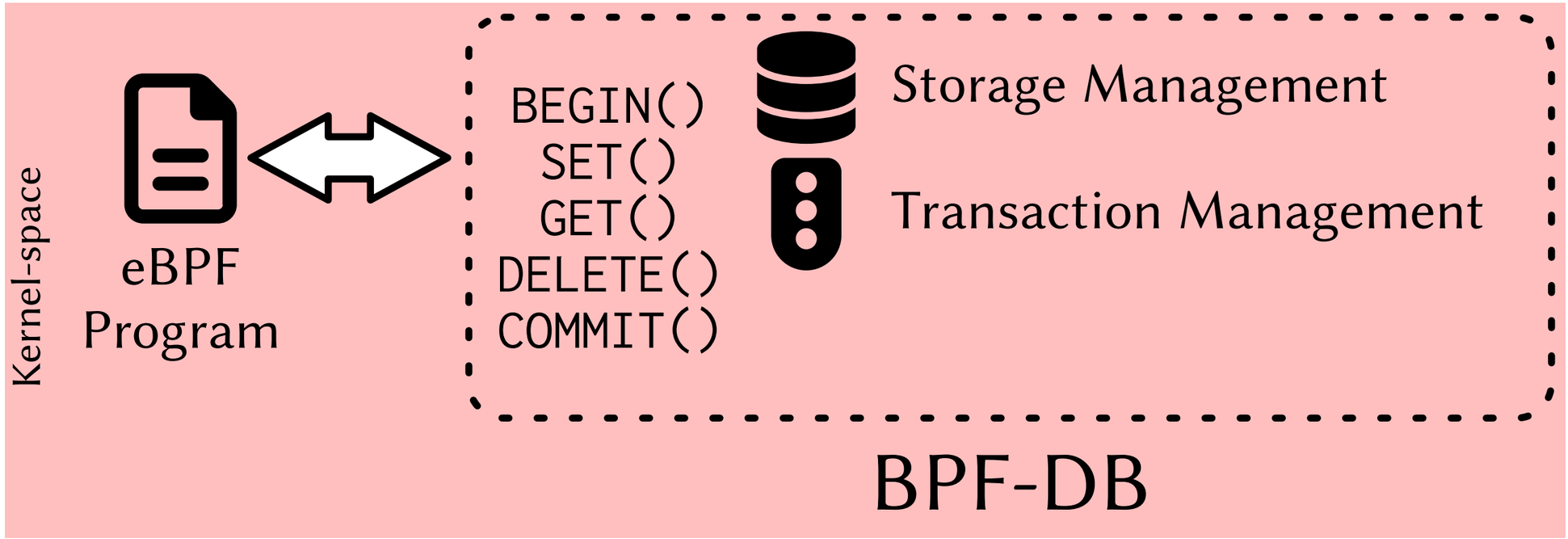
User-Bypass DBMS



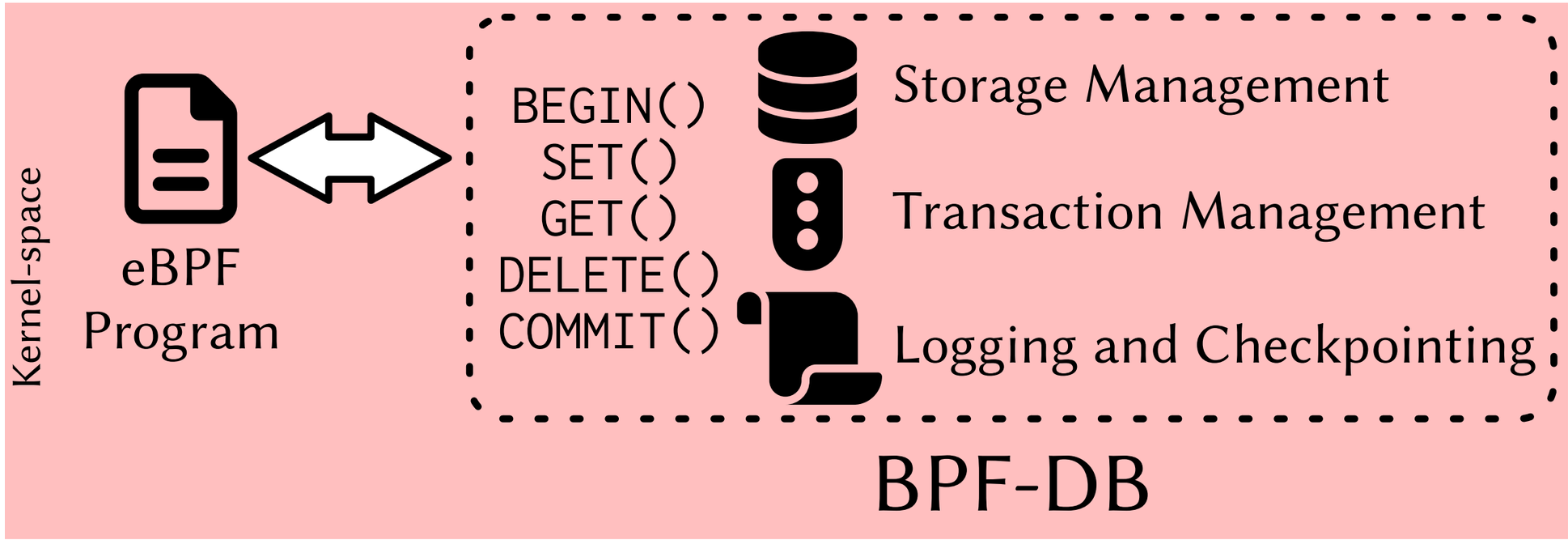
User-Bypass DBMS



User-Bypass DBMS



User-Bypass DBMS



Storage Management

Storage Management

- Goal: Store database contents in kernel-resident data structures

Storage Management

- Goal: Store database contents in kernel-resident data structures

- Challenges:

Storage Management

- Goal: Store database contents in kernel-resident data structures

- Challenges:
  - No `malloc`

Storage Management

- Goal: Store database contents in kernel-resident data structures

- Challenges:
  - No `malloc`
  - eBPF maps use fixed size keys and values

Storage Management

- Goal: Store database contents in kernel-resident data structures

- Challenges:
  - No `malloc`
  - eBPF maps use fixed size keys and values
  - Version chains cannot be unbounded

Transaction Management

Transaction Management

- Goal: Multi-statement transactions that ensure ACID properties

Transaction Management

- Goal: Multi-statement transactions that ensure ACID properties

- Challenges:

Transaction Management

- Goal: Multi-statement transactions that ensure ACID properties

- Challenges:
  - Restrictive atomic primitives

Transaction Management

- Goal: Multi-statement transactions that ensure ACID properties

- Challenges:
  - Restrictive atomic primitives
  - Boundedness limits spinning

Transaction Management

- Goal: Multi-statement transactions that ensure ACID properties

- Challenges:
  - Restrictive atomic primitives
  - Boundedness limits spinning
  - eBPF execution cannot yield

Logging and Checkpointing

Logging and Checkpointing

- Goal: Persist database contents to disk both through write-ahead logging and checkpointing

Logging and Checkpointing

- Goal: Persist database contents to disk both through write-ahead logging and checkpointing

- Challenges:

Logging and Checkpointing

- Goal: Persist database contents to disk both through write-ahead logging and checkpointing

- Challenges:
  - eBPF programs cannot initiate disk access

Logging and Checkpointing

- Goal: Persist database contents to disk both through write-ahead logging and checkpointing

- Challenges:
  - eBPF programs cannot initiate disk access
  - eBPF program execution cannot yield

Logging and Checkpointing

- Goal: Persist database contents to disk both through write-ahead logging and checkpointing

- Challenges:
  - eBPF programs cannot initiate disk access
  - eBPF program execution cannot yield
  - Database contents are stored in kernel-resident data

Experimental Setup

Experimental Setup

- v0.1 user-bypass DBMS (BPF-DB):
  - In-memory storage manager, variable value sizes
  - No multi-statement transactions
  - Atomic GET/SET/DELETE via spinlocks, ordering, and RCU “MVCC”

Experimental Setup

- v0.1 user-bypass DBMS (BPF-DB):
  - In-memory storage manager, variable value sizes
  - No multi-statement transactions
  - Atomic GET/SET/DELETE via spinlocks, ordering, and RCU “MVCC”

- In-memory databases:
  - Redis (C, single-threaded)
  - KeyDB (C, multi-threaded)
  - Dragonfly (C++, multi-threaded)
  - BPF-DB (eBPF, multi-threaded)

Experimental Setup

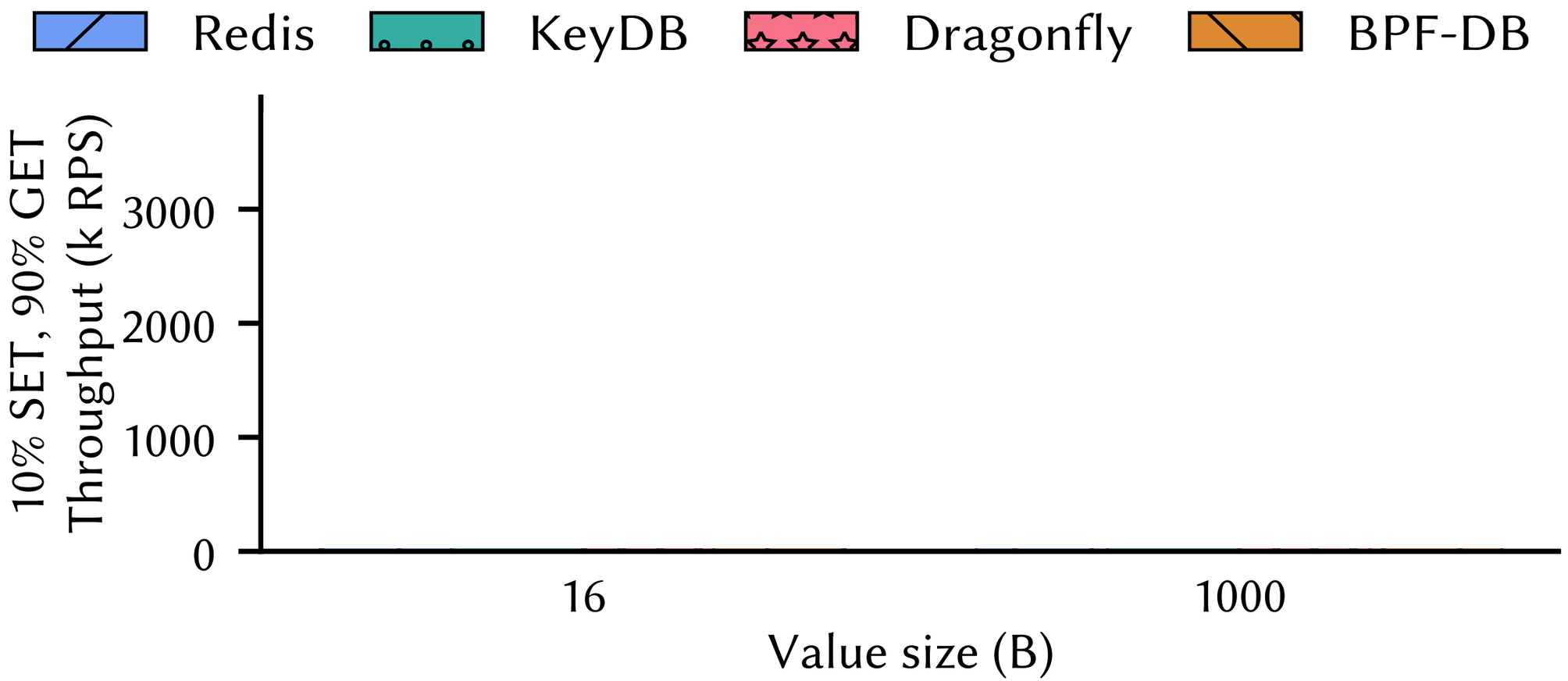
- v0.1 user-bypass DBMS (BPF-DB):
  - In-memory storage manager, variable value sizes
  - No multi-statement transactions
  - Atomic GET/SET/DELETE via spinlocks, ordering, and RCU “MVCC”

- In-memory databases:
  - Redis (C, single-threaded)
  - KeyDB (C, multi-threaded)
  - Dragonfly (C++, multi-threaded)
  - BPF-DB (eBPF, multi-threaded)

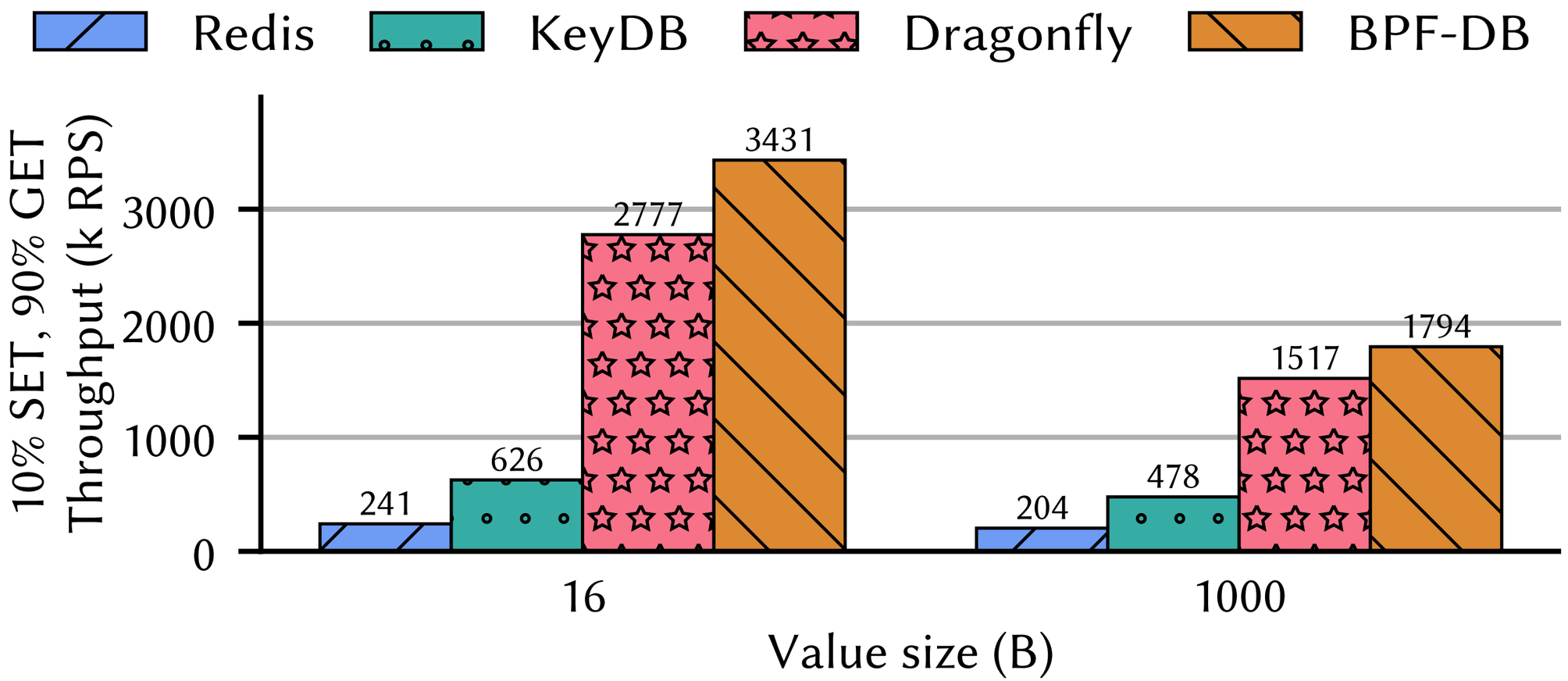
- 2×20-core Intel Xeon Gold 5218R CPUs, 196 GB DRAM, memtier_benchmark, Ubuntu 22.04 LTS

Preliminary In-Memory Results

Preliminary In-Memory Results



Preliminary In-Memory Results



Conclusion

Conclusion

- User-bypass is an approach for designing systems that reduces execution overhead and data movement associated with OS system calls

Conclusion

- User-bypass is an approach for designing systems that reduces execution overhead and data movement associated with OS system calls

- User-bypass is already feasible for multiple applications including DBMS proxies

Conclusion

- User-bypass is an approach for designing systems that reduces execution overhead and data movement associated with OS system calls
- User-bypass is already feasible for multiple applications including DBMS proxies
- Our proposed user-bypass DBMS will benefit from storing database contents in kernel-resident data structures and enable new classes of eBPF applications