OLTP Through the Looking Glass 16 Years Later:

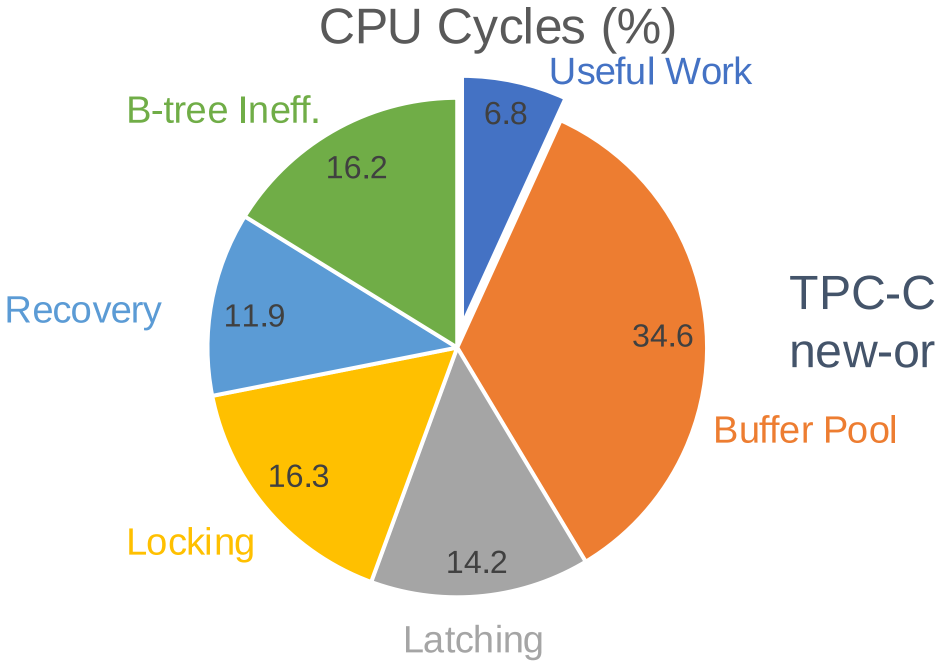
Communication is the New Bottleneck

Xinjing Zhou, **Viktor Leis**, Xiangyao Yu, Michael Stonebraker

MIT CSAIL, **TUM**, UW-Madison, MIT CSAIL

OLTP Looking Glass Back in 2008

- A performance study of a disk-based OLTP system - Shore
- Bottlenecks were spread across various components when data fits in memory



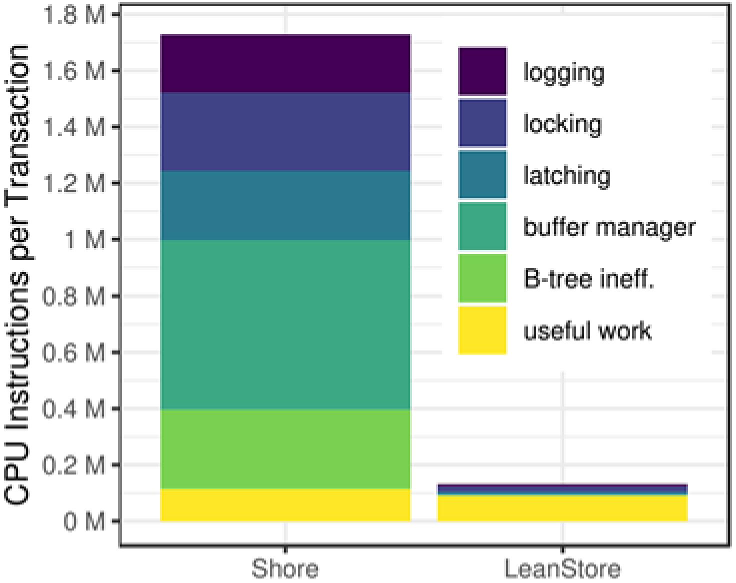
TPC-C
new-order

Many New OLTP Engines since then

H-Store VOLTDB leanstore

Silo HyPer UMBRA

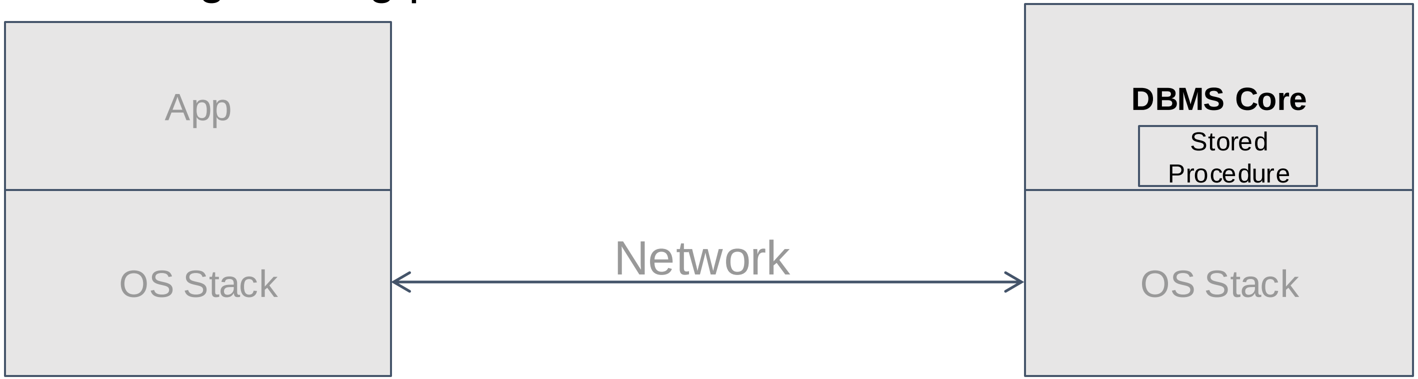
Hekaton ....



TPC-C
new-order

Problems of Previous Research

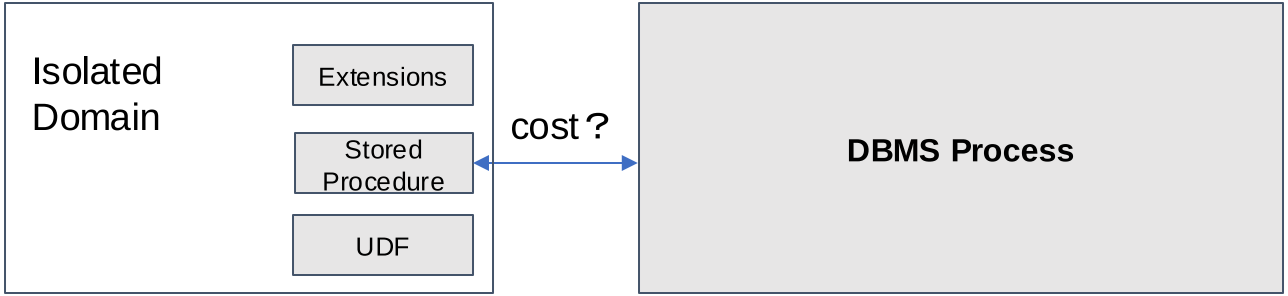
- Benchmarks ignore OS stacks and communication
- Most assume stored procedure as the core technique to reduce network overhead.
- The reality [1-2]: many apps prefer interactive transactions due to better software engineering practices



[1] Pavlo, Andrew. "What are we doing with our lives? Nobody cares about our concurrency control research." *SIGMOD 2017*.
[2] Hu, Gansen, et al. "WeBridge: Synthesizing Stored Procedures for Large-Scale Real-World Web Applications." *SIGMOD 2024*.

Security of Stored Procedures

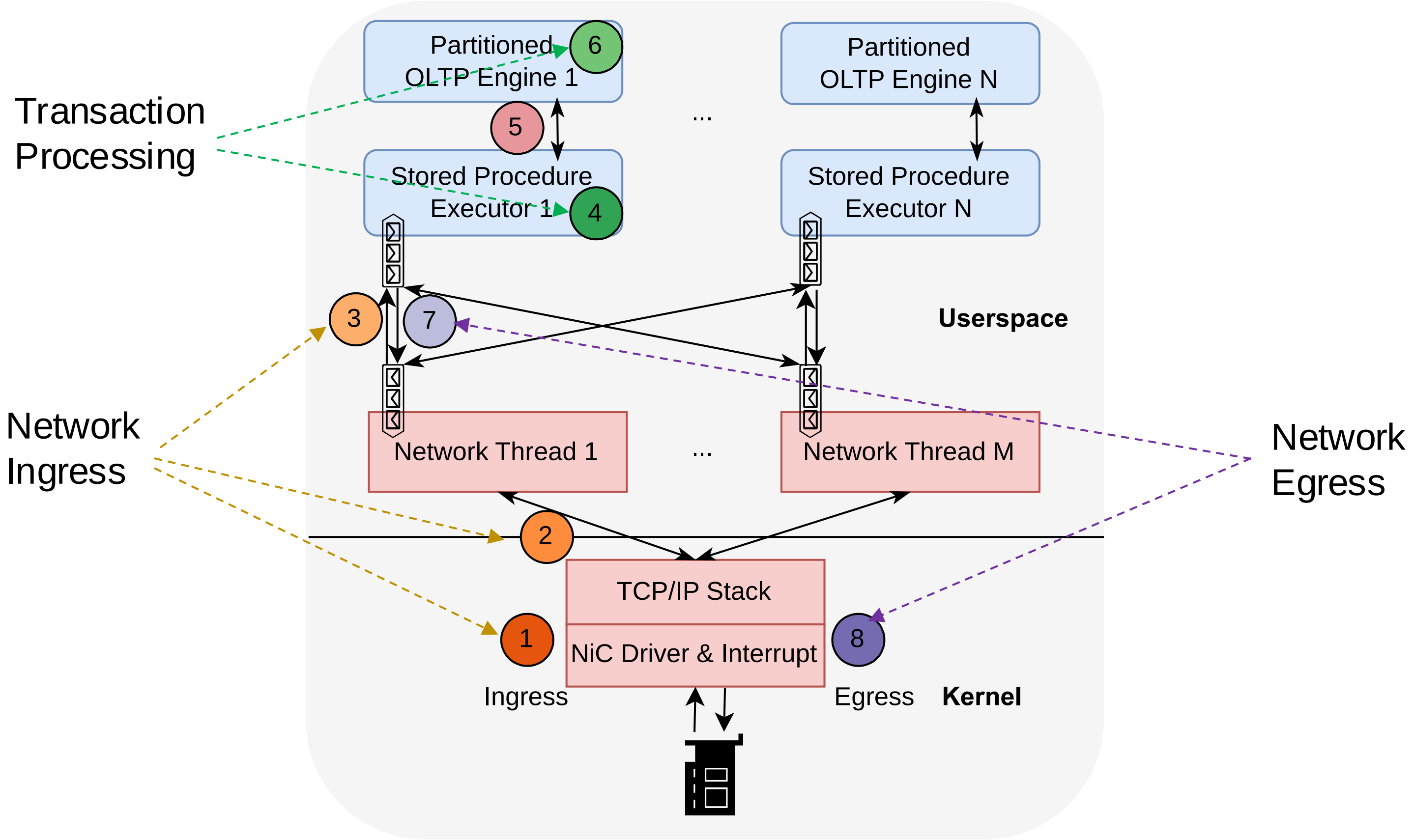
- Procedures run in the same address space of DBMS process for performance
  - written in various languages: PL/SQL, C/C++, Java, Python
- Malicious/errant procedures could read unauthorized data or crash DBMS
- DBMSs are becoming more multi-tenant as people move to the cloud
- This applies to other extensibility mechanisms: UDF and extensions



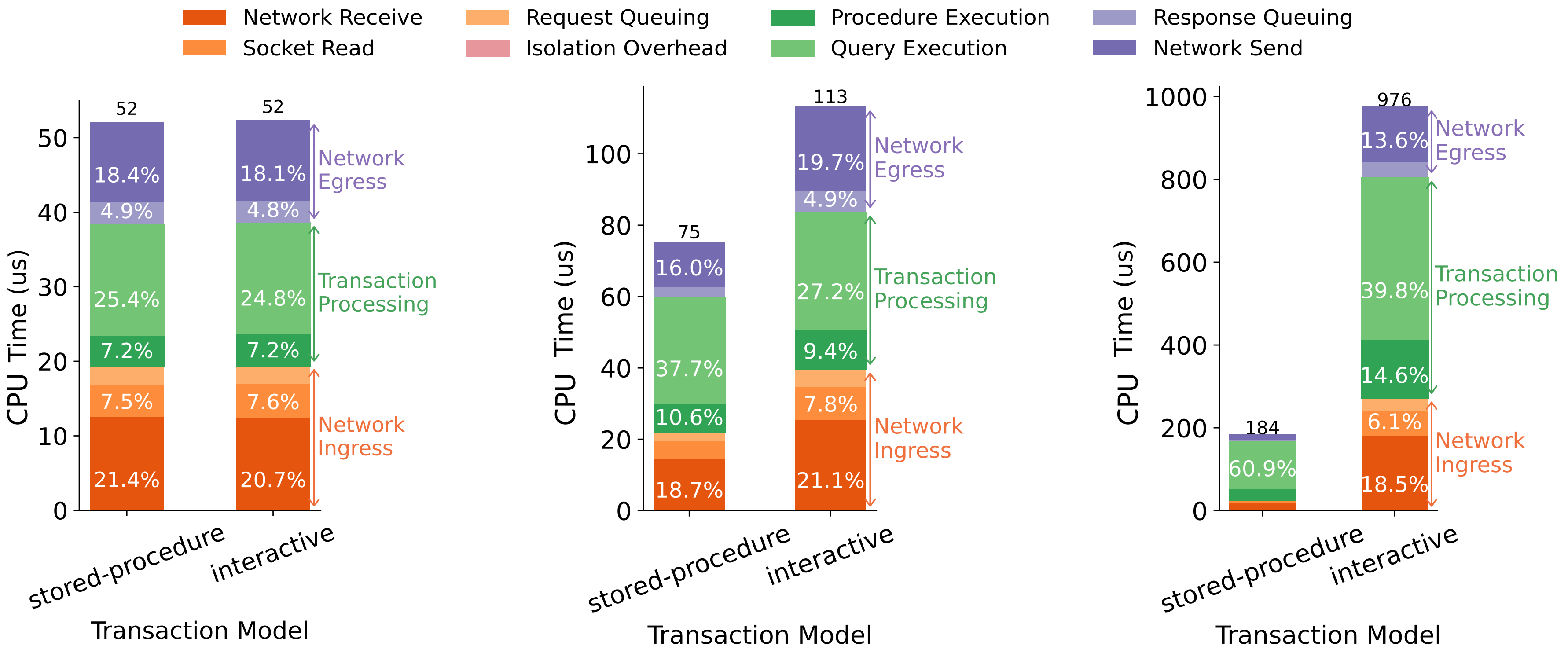
OLTP Looking Glass 2.0

- Consider OS network stacks
- Consider both stored procedures and interactive transactions
- Consider procedure isolation
- Assume previous bottlenecks were solved after more than a decade of research - We use VoltDB as the testbed.
- 2 Google cloud instances with 10Gbps NIC and 16-core 2.3Ghz CPU
- Increase the load until the server is CPU-bound

VoltDB Architecture



No-Isolation – Server-side CPU-time Breakdown, Communication is the bottleneck

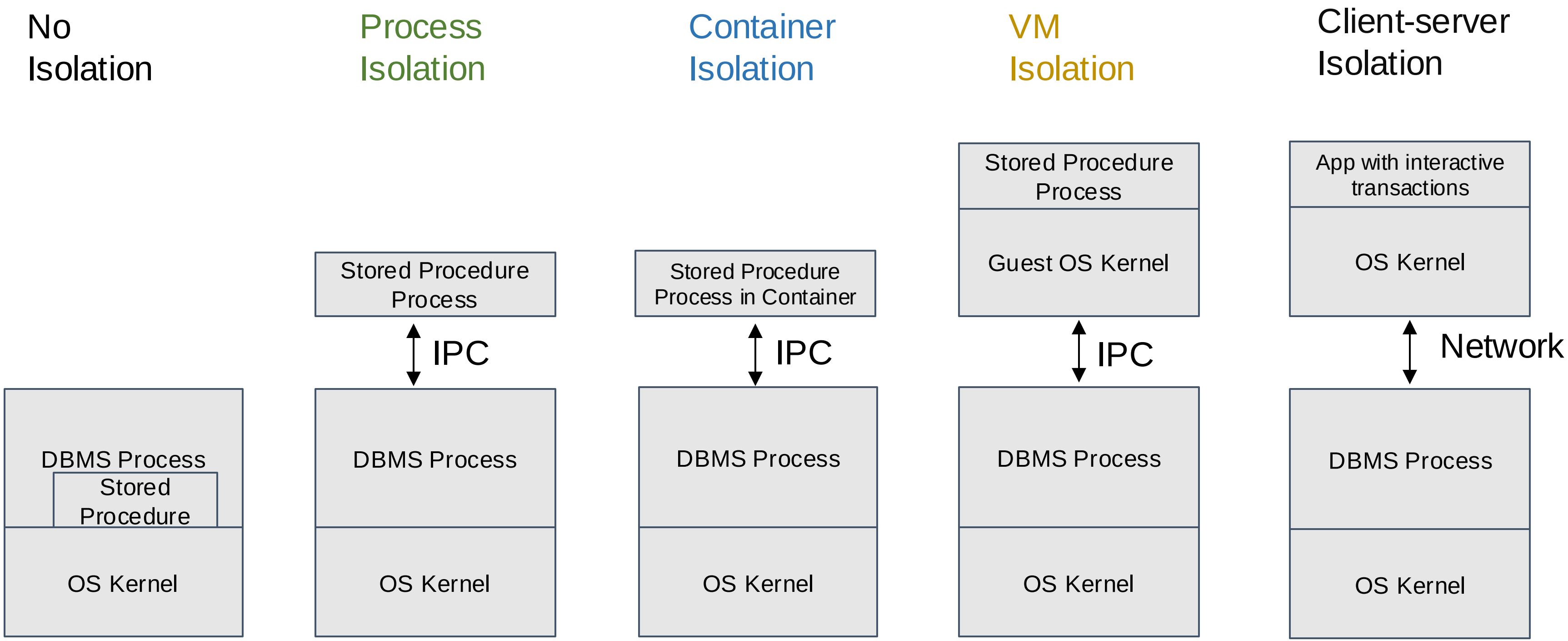


YCSB-C

Voter

TPC-C

Isolating Procedure



No Isolation

DBMS Process

Stored Procedure

OS Kernel

Process Isolation

Stored Procedure Process

IPC

DBMS Process

OS Kernel

Container Isolation

Stored Procedure Process in Container

IPC

DBMS Process

OS Kernel

VM Isolation

Stored Procedure Process

Guest OS Kernel

IPC

DBMS Process

OS Kernel

Client-server Isolation

App with interactive transactions

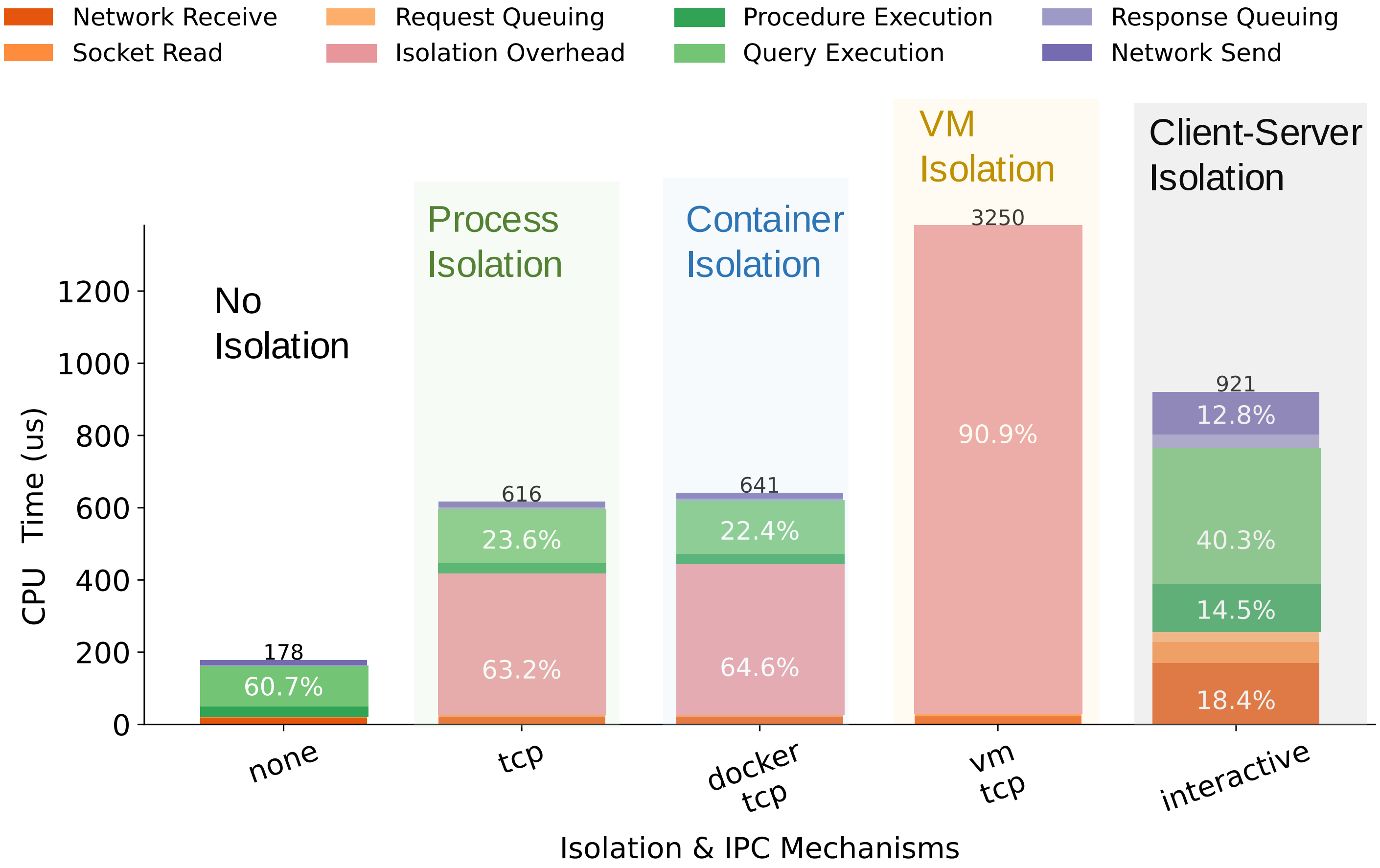
OS Kernel

Network

DBMS Process

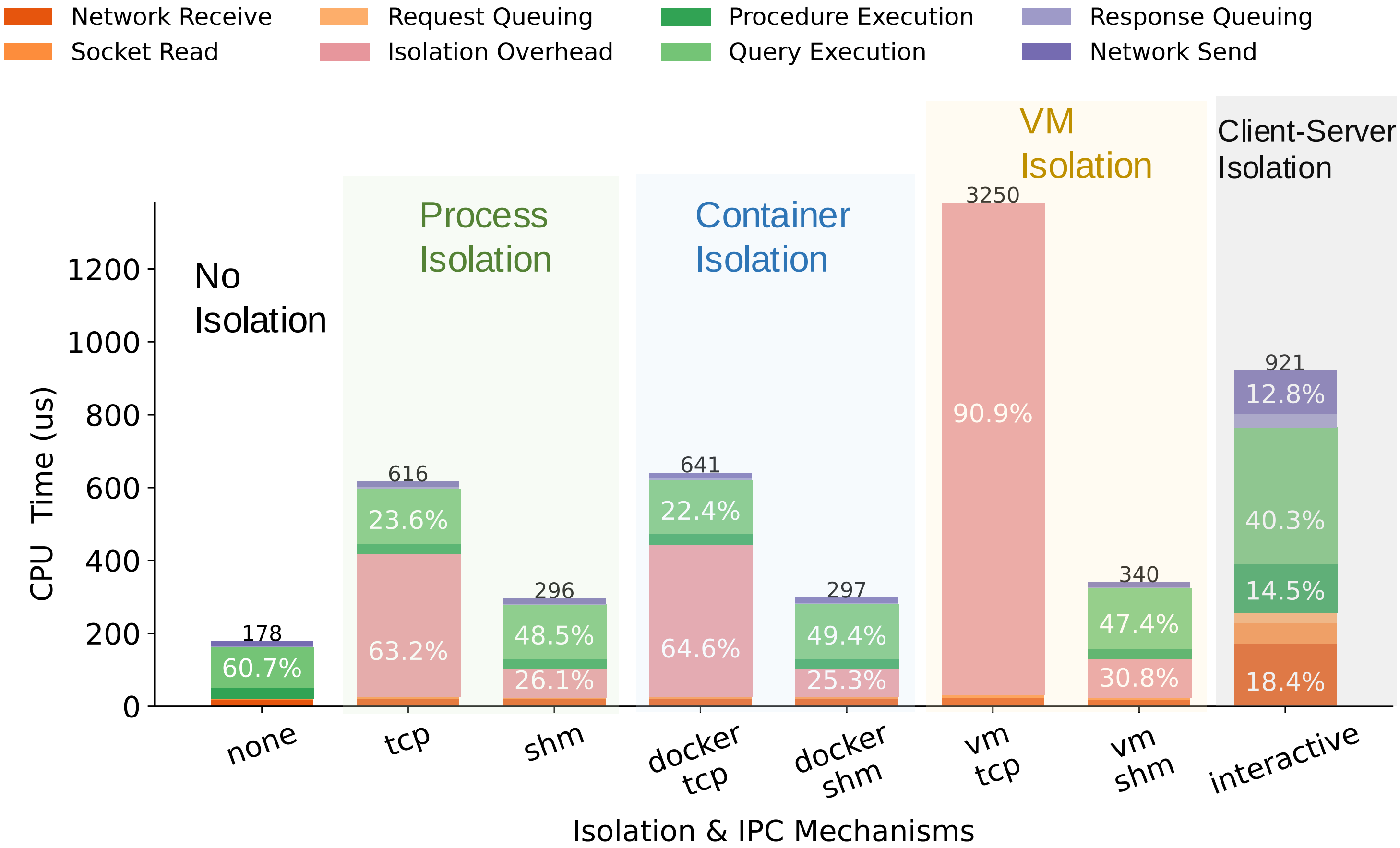
OS Kernel

Isolated Stored Procedure Execution, Communication for Isolation is the bottleneck



TPC-C

Isolated Stored Procedure Execution, Communication for Isolation is the bottleneck

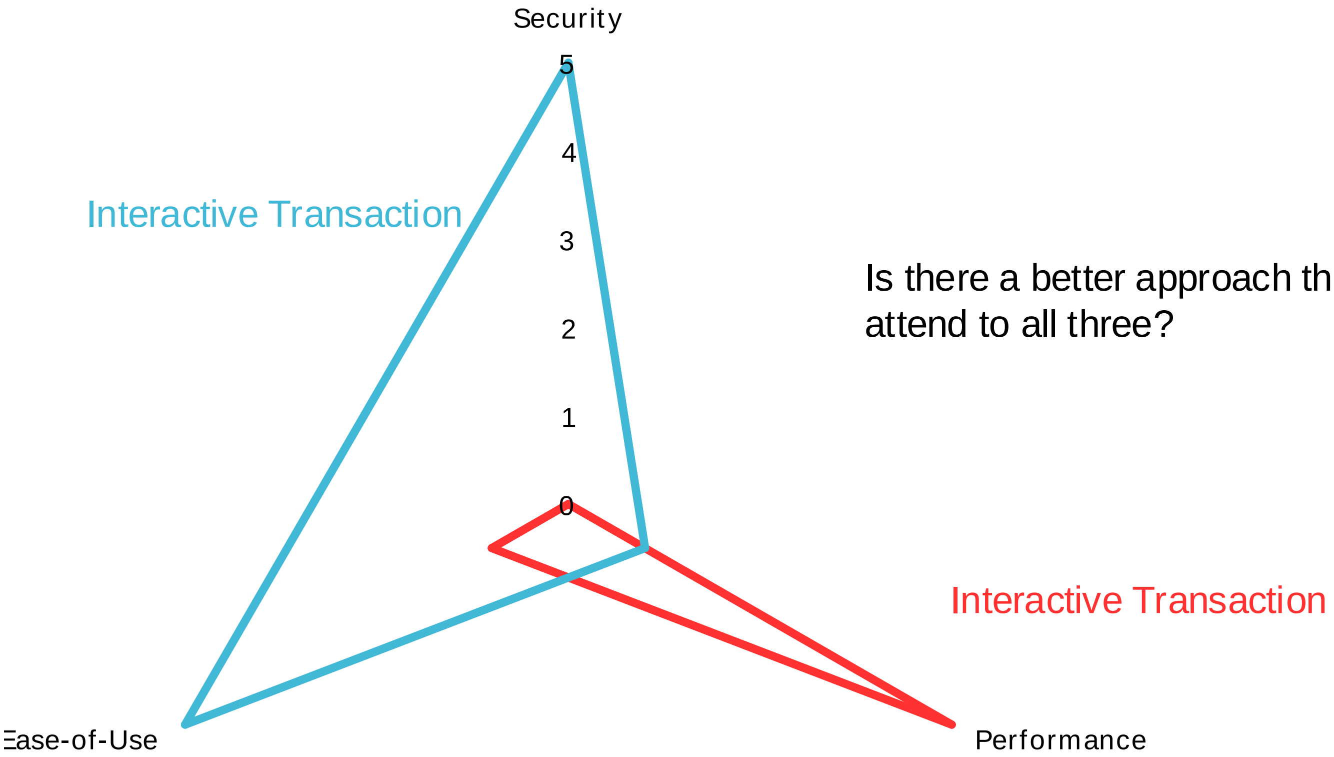


TPC-C

Wish #1: Towards Usable Kernel Bypass

- DPDK + User space TPC/IP stack (F-Stack)
  - Reduces kernel network stack overhead of VoltDB by 85%
- Only two DBMS vendors support kernel-bypass: Yellowbrick and ScyllaDB
- Three Problems
  - **Interface-Mismatch**: DPDK is a layer-2 stack – no transport/routing layer support
  - **Design Limitation**: A DPDK app requires complete control of a NIC
    - Linux tooling are not available on DPDK-managed NIC, making debugging and deployment hard.
  - **Engineering and Maintenance:** User-space TCP/IP stacks often require DBMS to rewrite their network layer code due to API differences.

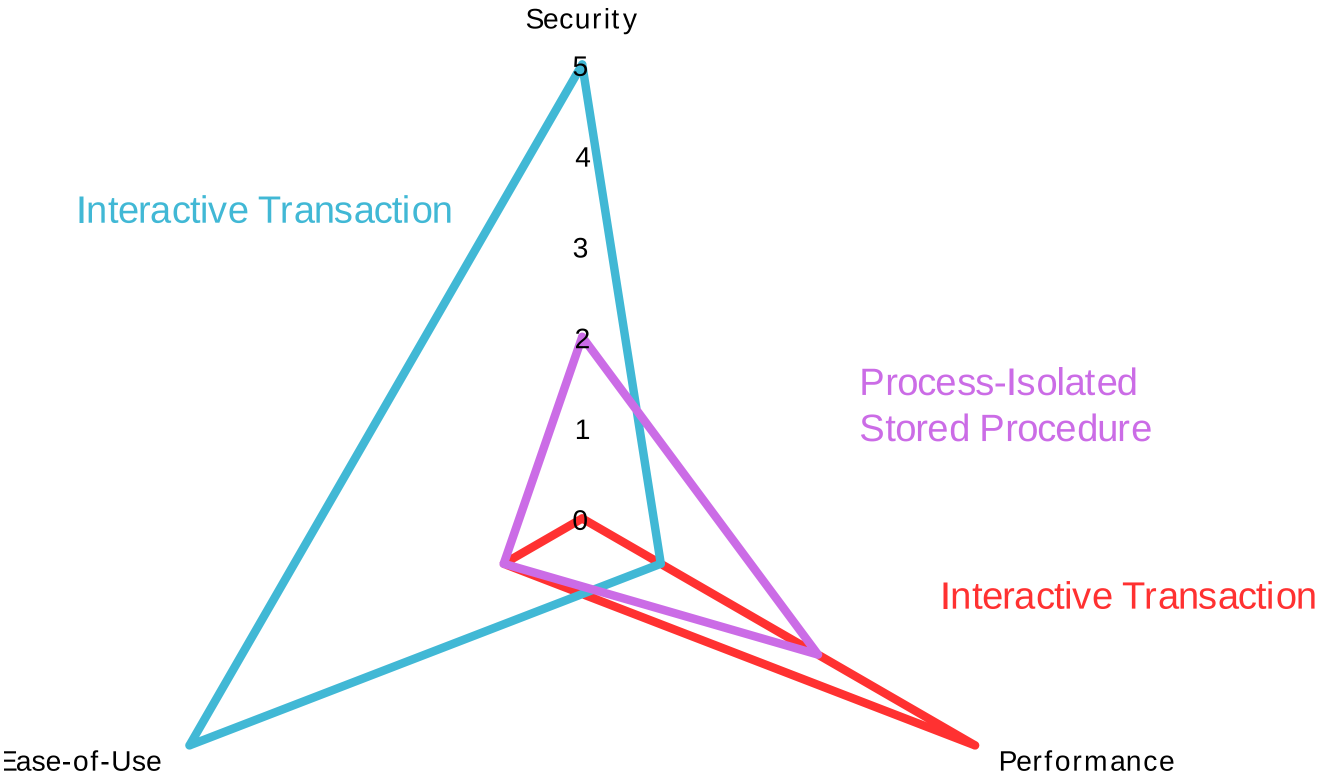
Wish #2: More Exploration in the Trade-off Space



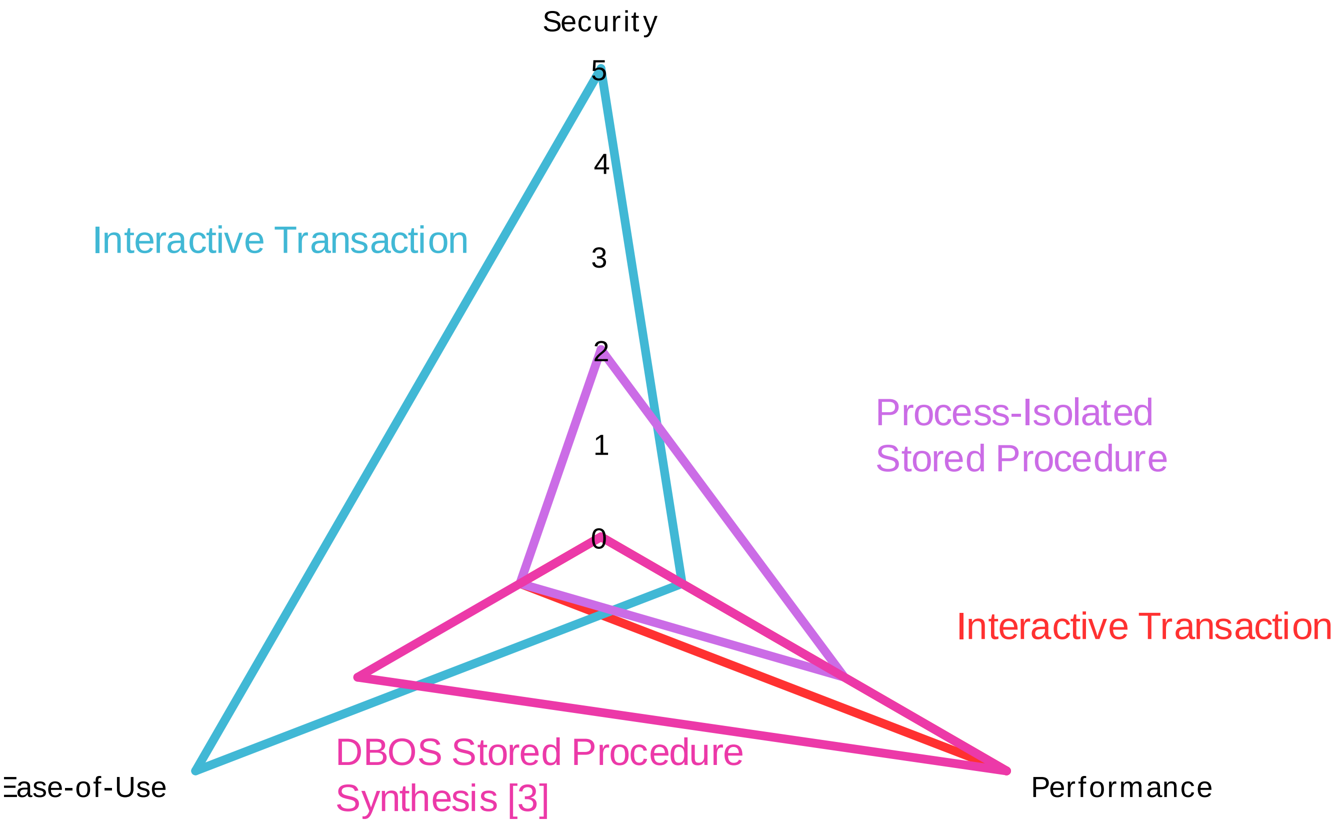
Is there a better approach that attend to all three?

Debuggability
Testing
Language Flexibility
DBMS-agnostic
Version-Control

Wish #2: More Exploration in the Trade-off Space



Wish #2: More Exploration in the Trade-off Space



[3] https://www.dbos.dev/blog/stored-procedures-good-bad-elegant

Conclusion

- We should focus more on intra-DBMS communication and OS network stack.
- We need more usable and efficient kernel bypass abstractions to make larger impact on DBMS.
- We should revisit the debate about stored-procedure and interactive transaction, factoring in security and usability.

CIDR 2025 Preprint