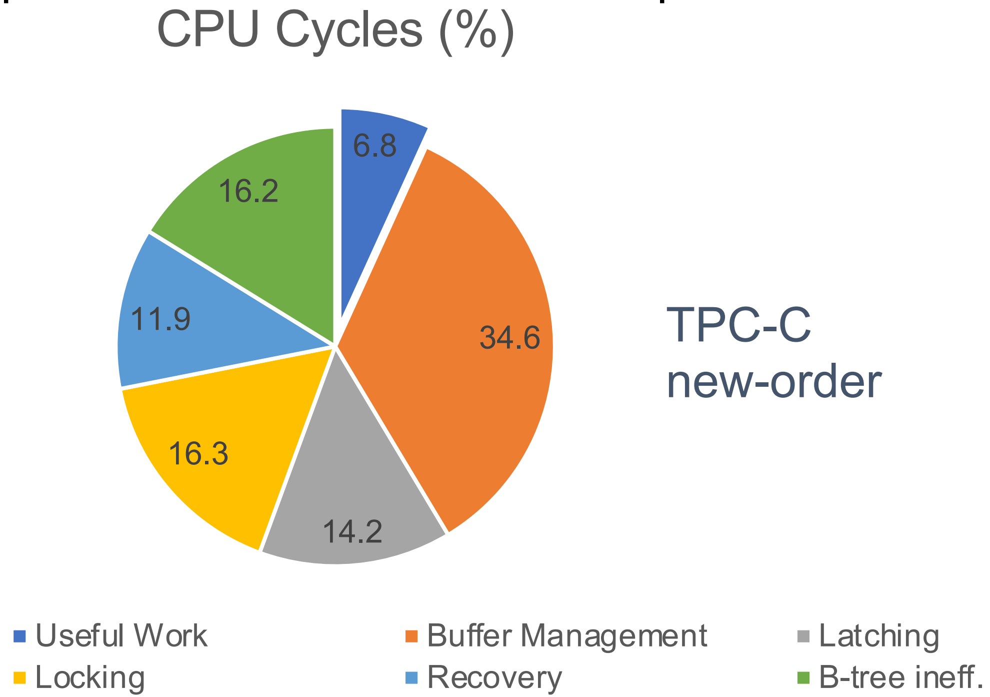
OLTP Through the Looking Glass 16 Years Later

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 core components when data fits in memory



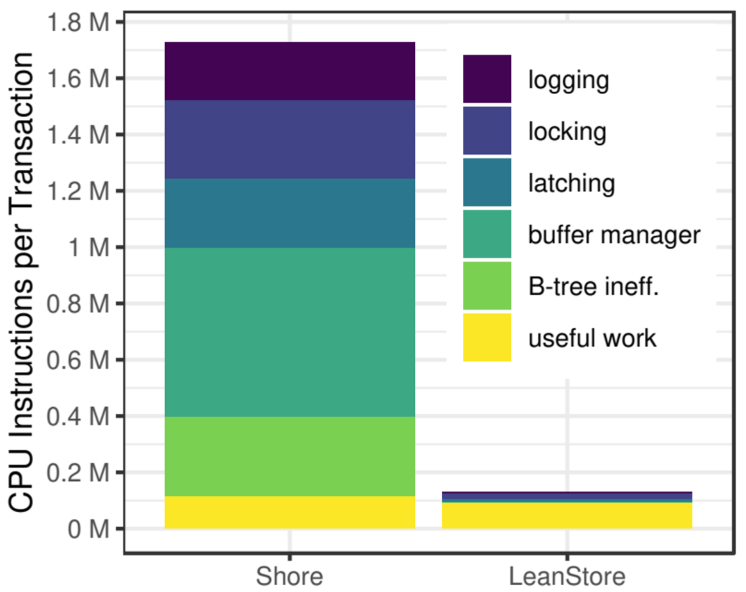
TPC-C
new-order

Many New OLTP Engines since then

H-Store VOLTDB leanstore

Silo HyPer UMBRA

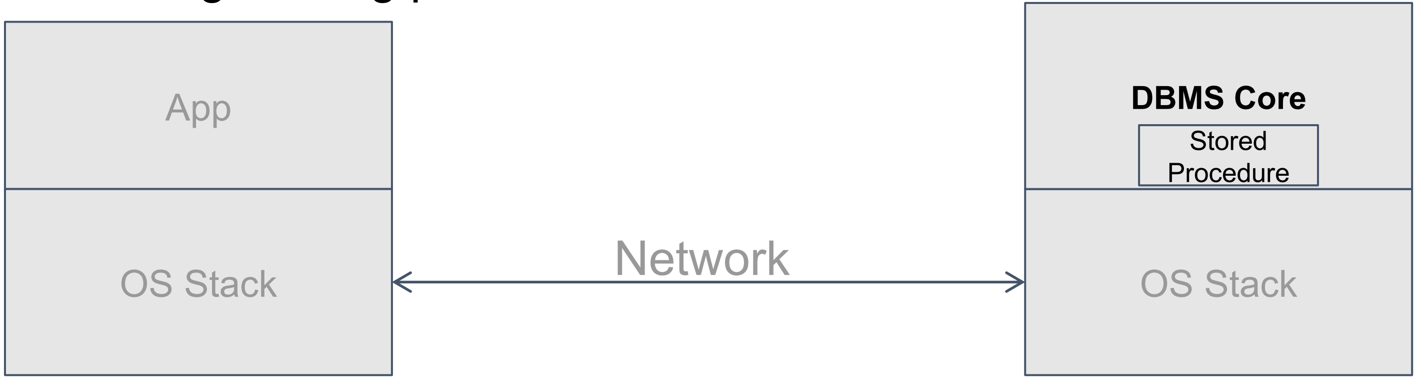
Hekaton ....



TPC-C
new-order

Problems of Previous Research

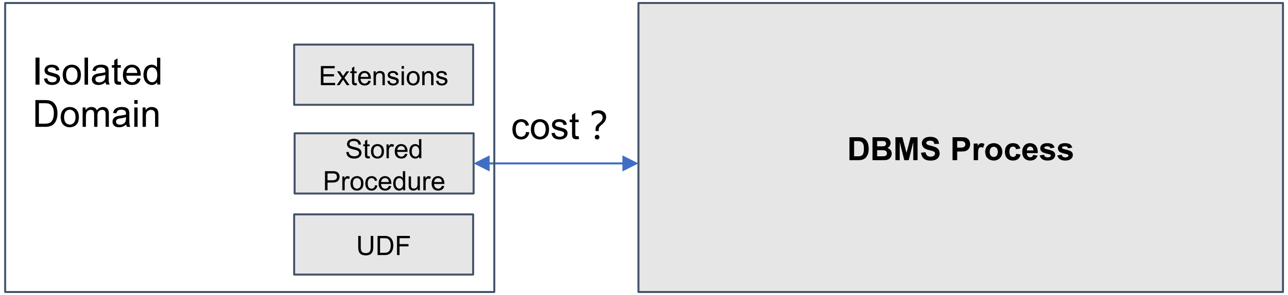
- Benchmarks ignore OS stack and communication
- Most assume stored-procedure as the core technique to reduce network overhead.
- The reality [1-3]: many apps prefer interactive transactions due to better software engineering practices



[1] Pavlo, Andrew. "What are we doing with our lives? Nobody cares about our concurrency control research." *SIGMOD 2017*.
[2] Gupta, Surabhi, and Karthik Ramachandra. "Procedural Extensions of SQL: Understanding their usage in the wild." *VLDB 2021*
[3] Hu, Gansen, et al. "WeBridge: Synthesizing Stored Procedures for Large-Scale Real-World Web Applications." *SIGMOD 2024*.

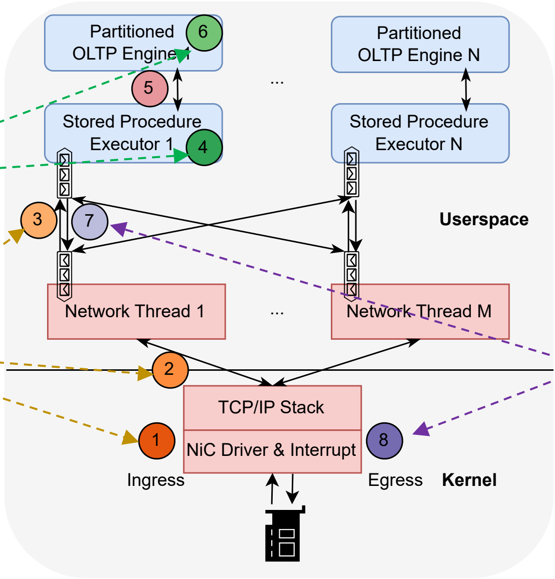
Security of Stored-procedure

- Procedure 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 stacks and communication
- Consider procedure isolation
- Assume previous bottlenecks were solved after more than a decade of research - We use VoltDB as the testbed.
- Assume single-partition transactions and single-node setup

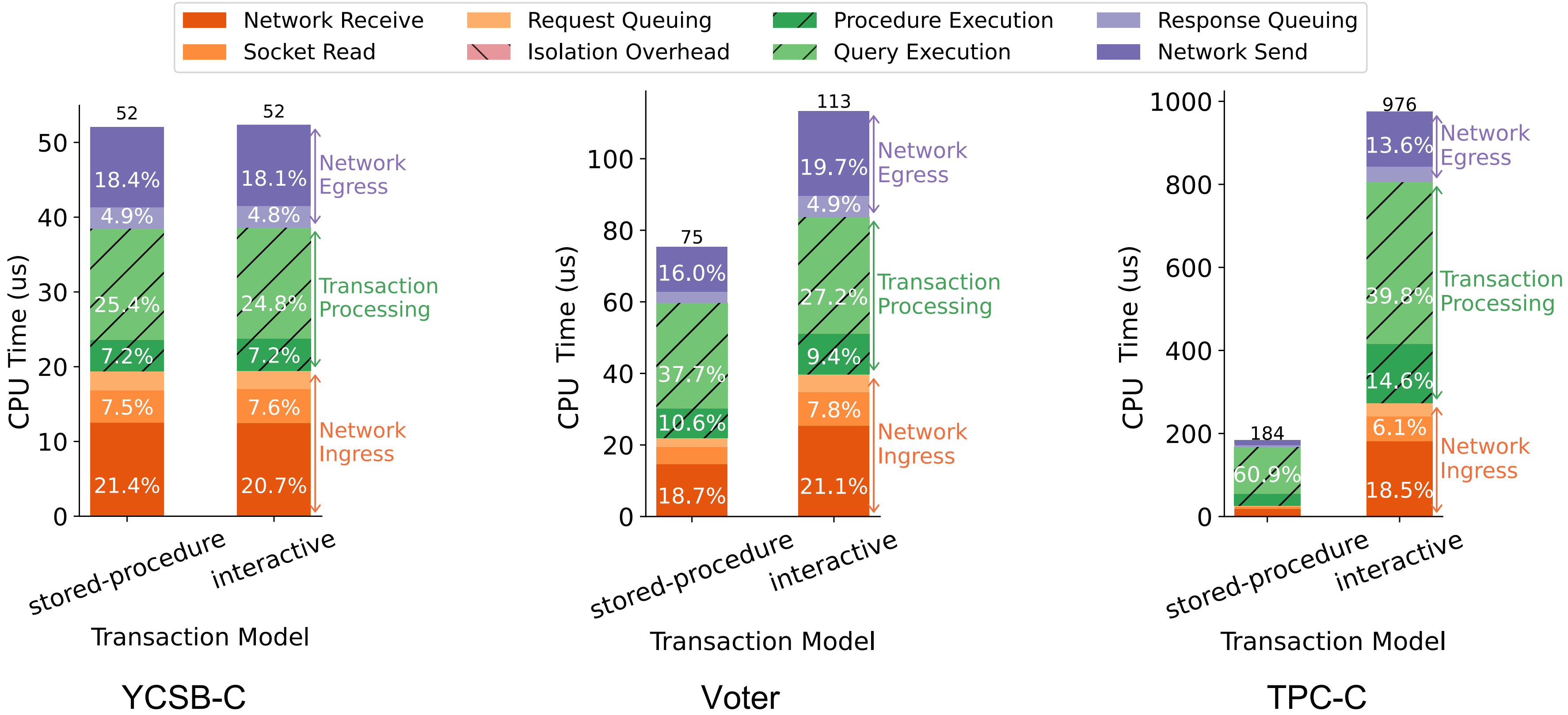


Transaction Processing

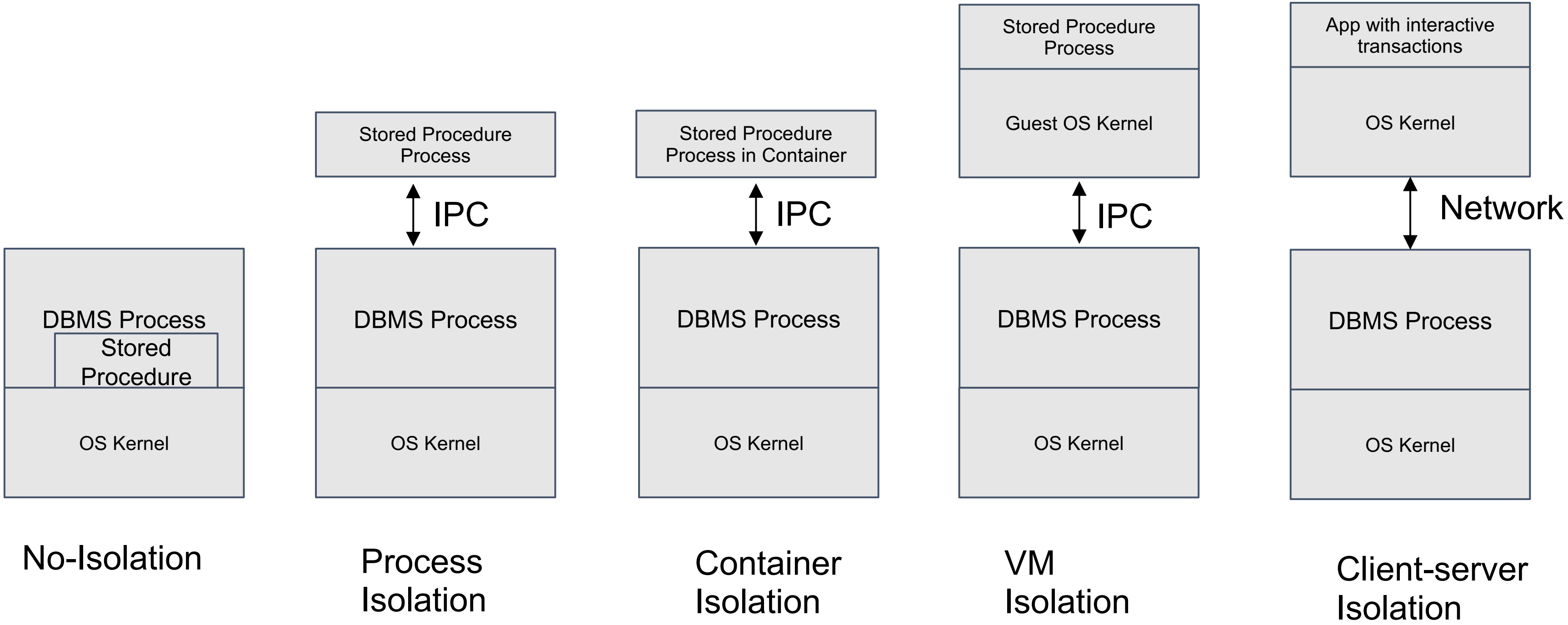
Network Ingress

Network Egress

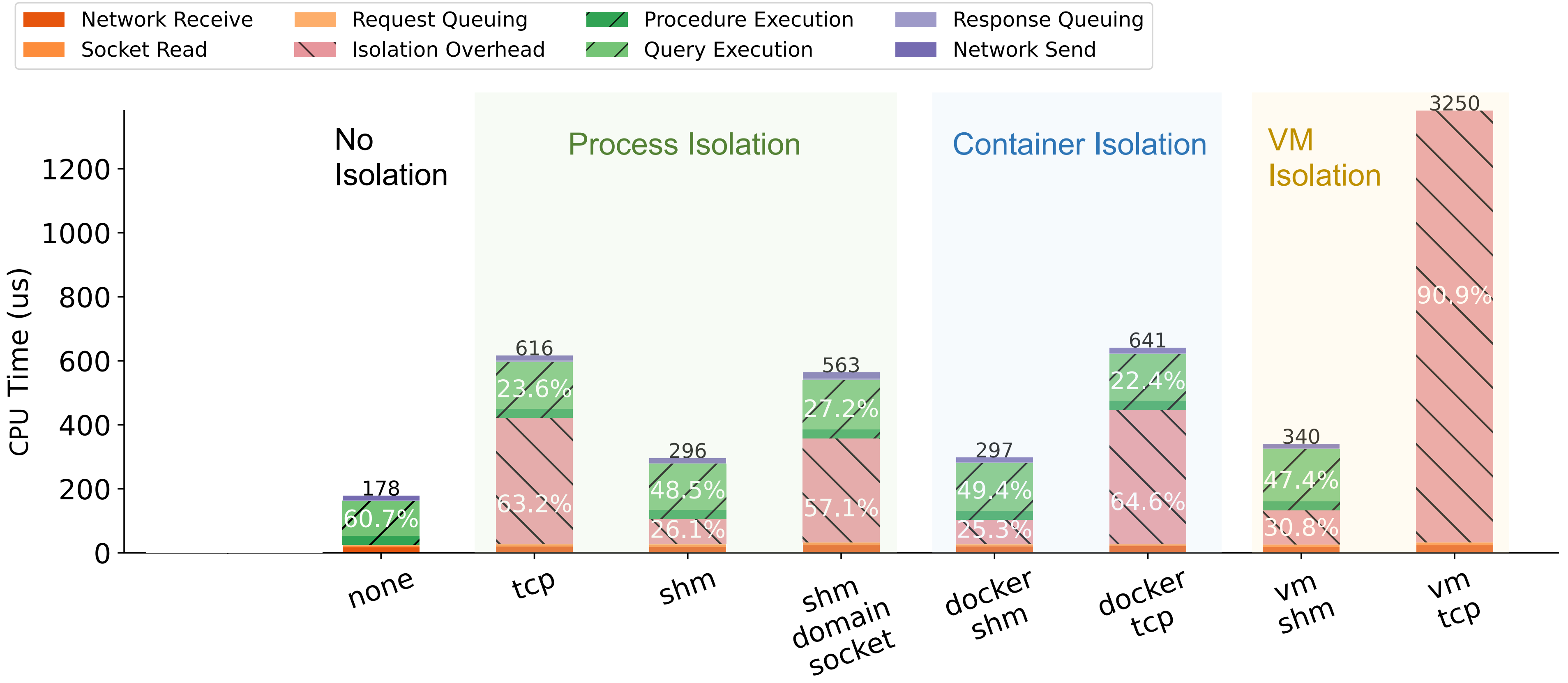
No-Isolation – Server-side CPU-time Breakdown, Communication is the bottleneck



Isolating Procedure



Isolated Stored Procedure Execution, Communication for Isolation is the bottleneck



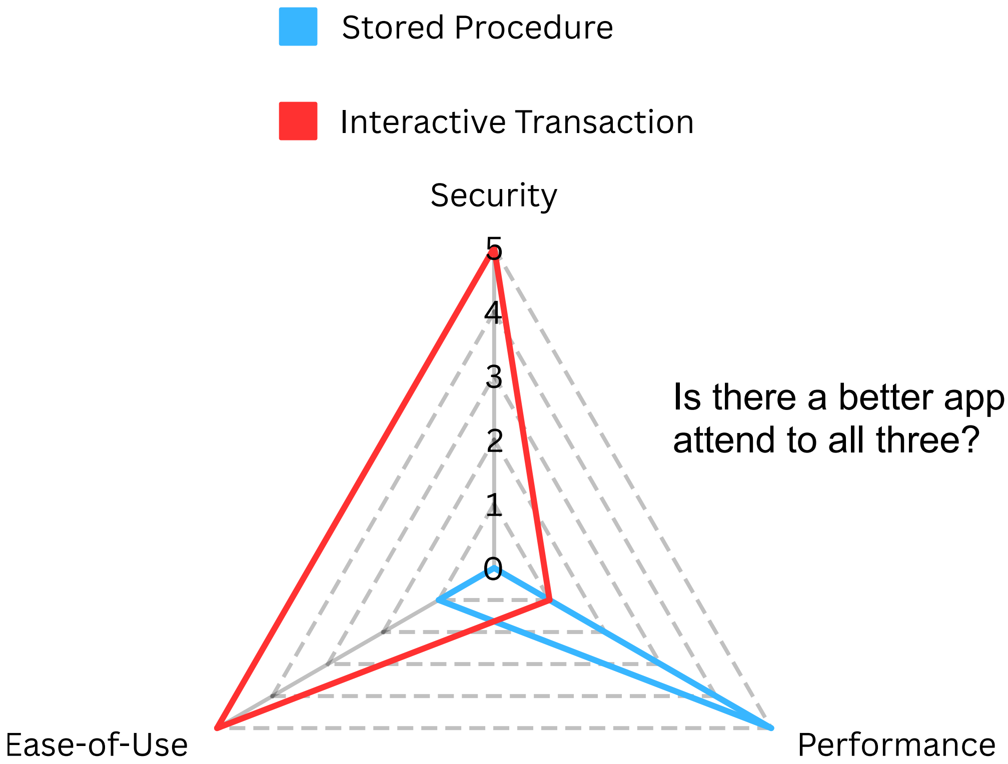
Isolation Mechanisms

TPC-C

Wish #1: 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

Conclusion

- Communication and OS stack should be more focused by the community for future DB systems research.
- 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. The trade-off space is under-explored.

CIDR 2025 Preprint