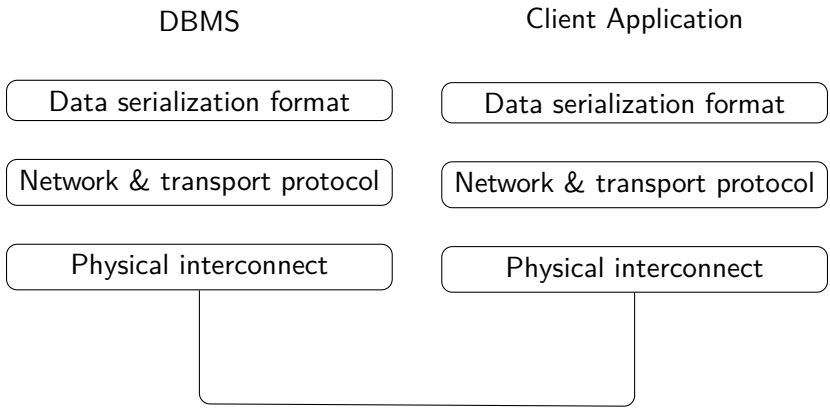
# Low-Latency Communication for Fast DBMS Using RDMA and Shared Memory

Philipp Fent, Alexander van Renen, Andreas Kipf,
Viktor Leis*, Thomas Neumann, Alfons Kemper

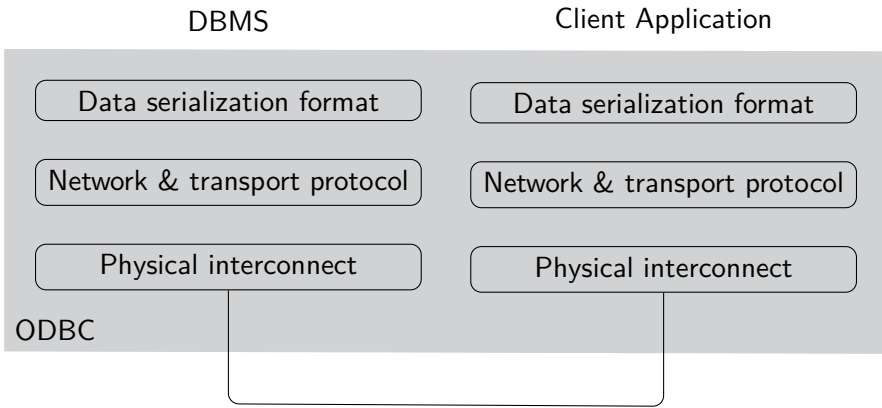
Technische Universität München,
Friedrich-Schiller-Universität Jena*

April 23, 2020

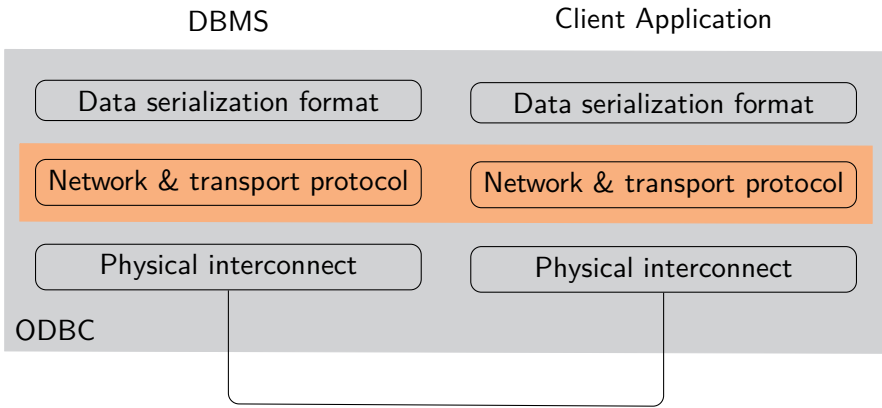
# Communication Performance



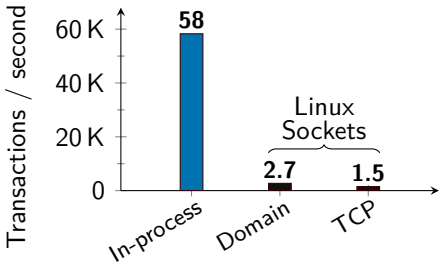
# Communication Performance



# Communication Performance



# Communication Performance




(a) 1 Thread

Figure: TPC-C throughput using Silo

# Communication Performance

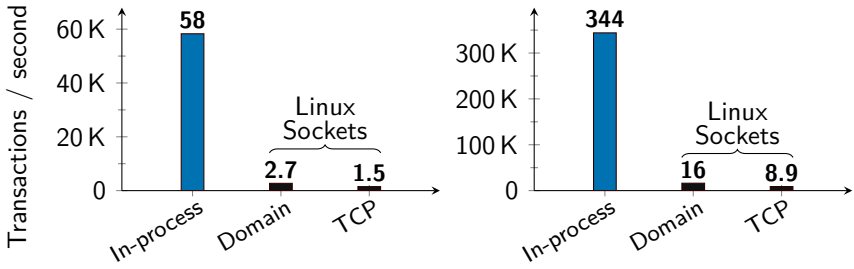



Figure: TPC-C throughput using Silo

# Understanding the Bottleneck

- Misconception: Network is slow

# Understanding the Bottleneck

- Misconception: Network is slow
- Twofold actual bottleneck:
  - TCP

# Understanding the Bottleneck

- Misconception: Network is slow
- Twofold actual bottleneck:
  - TCP
  - Through kernel communication

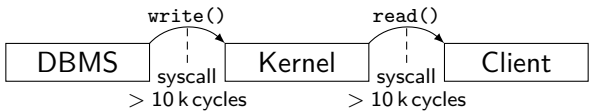



Figure: Kernel based communication

# Understanding the Bottleneck

- Misconception: Network is slow
- Twofold actual bottleneck:
  - TCP
  - Through kernel communication

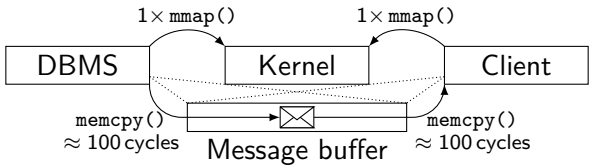



Figure: Direct memory access

# Low-Latency Communication Using Shared Memory

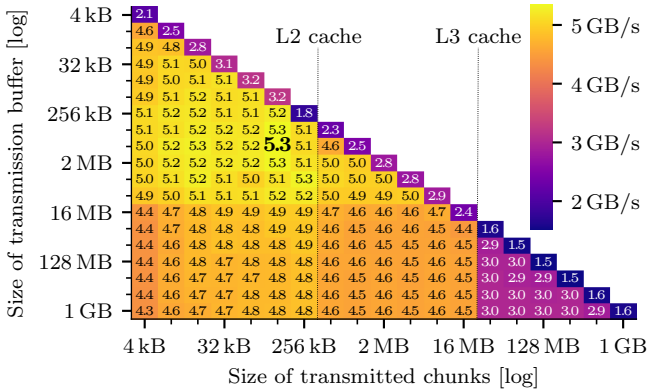
- Co-hosted on the same machine
- Latency similar to embedded DBs, e.g. SQLite
- Ideal interconnect for container / Docker environment

# Low-Latency Communication Using Shared Memory

- Co-hosted on the same machine
- Latency similar to embedded DBs, e.g. SQLite
- Ideal interconnect for container / Docker environment
- Bootstrapped via Domain Sockets
  - Pass message buffer via cmsg ancillary data
- Ringbuffer with polling to transfer serialized data

# Low-Latency Communication Using Shared Memory

Available bandwidth depends on transmission parameters

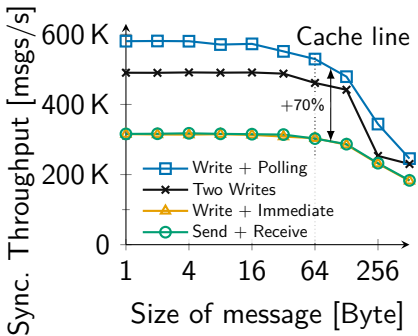


# Low-Latency Communication Using RDMA

- Co-located in the same datacenter
- Bootstrapped via regular TCP/IP
- Similar ringbuffer as with Shared Memory

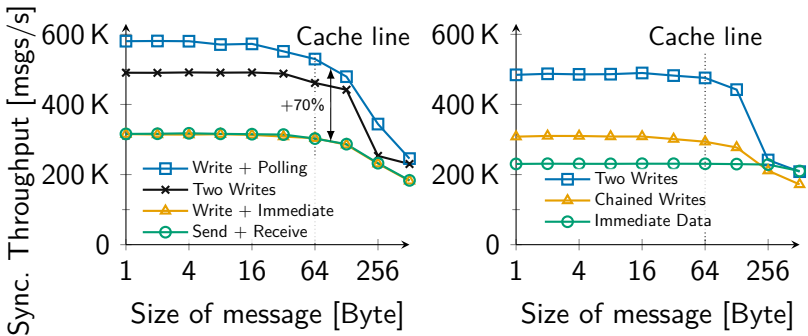
# Low-Latency Communication Using RDMA

- Co-located in the same datacenter
- Bootstrapped via regular TCP/IP
- Similar ringbuffer as with Shared Memory
- RDMA intricacies:



# Low-Latency Communication Using RDMA

- Co-located in the same datacenter
- Bootstrapped via regular TCP/IP
- Similar ringbuffer as with Shared Memory
- RDMA intricacies:

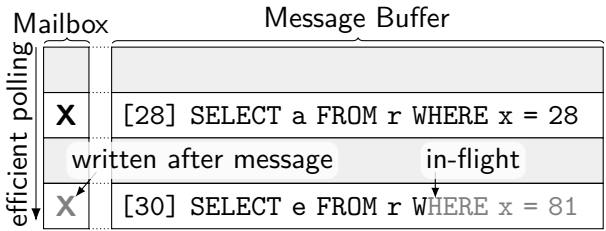


# Low-Latency Communication Using RDMA

- Asymmetric connections
- Many message buffers $\rightarrow$ random accesses for polling

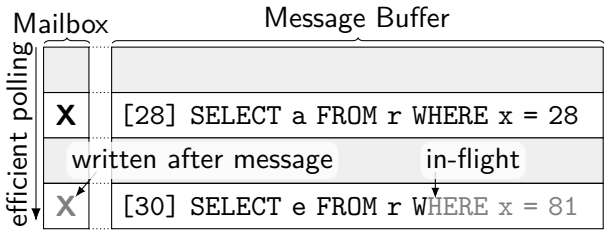
# Low-Latency Communication Using RDMA

- Asymmetric connections
- Many message buffers → random accesses for polling
- Cache efficient mailbox polling
  - Two writes to separate memory regions



# Low-Latency Communication Using RDMA

- Asymmetric connections
- Many message buffers → random accesses for polling
- Cache efficient mailbox polling
  - Two writes to separate memory regions
- Scales up to the limit of RDMA's reliable connections



# Results

Local YCSB-C

| [sync. tx/s] | TCP | SHM | NP | DS | RDMA |
|---|---|---|---|---|---|
| Silo + L5 | 50.5 K | **685 K** | — | 72.1 K | 364 K |
| DBMS X | 7.56 K | 11.5 K | 11.5 K | — | — |
| MySQL | 10.0 K | 45.9 K | 27.6 K | — | — |
| SQLite | — | 378 K | — | — | — |

# Results

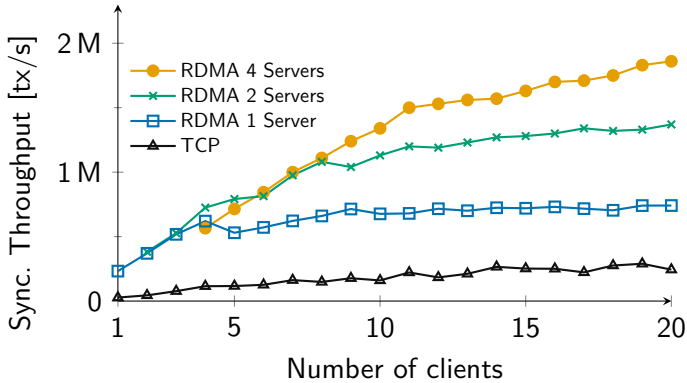
Local YCSB-C

| [sync. tx/s] | TCP | SHM | NP | DS | RDMA |
|---|---|---|---|---|---|
| Silo + L5 | 50.5 K | **685 K** | — | 72.1 K | 364 K |
| DBMS X | 7.56 K | 11.5 K | 11.5 K | — | — |
| MySQL | 10.0 K | 45.9 K | 27.6 K | — | — |
| SQLite | — | 378 K | — | — | — |

Remote YCSB-C

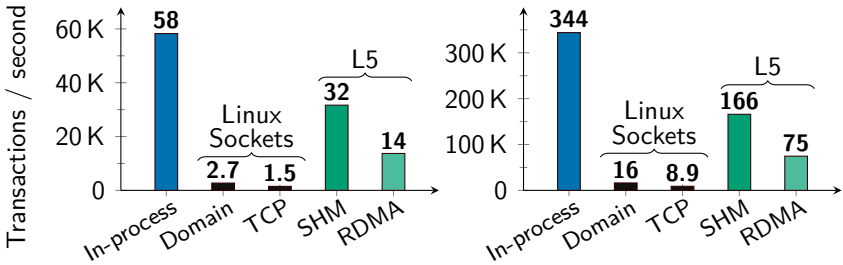
| [sync. tx/s] | 1 G Eth | 56 G IB | RDMA |
|---|---|---|---|
| Silo + L5 | 15 K | 27 K | **302 K** |
| DBMS X | 3.1 K | 3.7 K | — |
| MySQL | 7.1 K | 8.0 K | — |
| PostgreSQL | 6.3 K | 7.5 K | — |

# Results – Scale Out



# Conclusion

- **L5** – **L**ow-**L**evel, **L**ow-**L**atency **L**ibrary
  `https://github.com/pfent/L5RDMA`
- Shared Memory and RDMA bring OLTP performance to clients
- `fent@in.tum.de`




(a) 1 Thread (b) 8 Threads