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

DBMS

- Data serialization format
- Network & transport protocol
- Physical interconnect

Client Application

- Data serialization format
- Network & transport protocol
- Physical interconnect
Communication Performance

Figure: TPC-C throughput using Silo
Communication Performance

Figure: TPC-C throughput using Silo

(a) 1 Thread

(b) 8 Threads

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

![Kernel based communication](image)

**Figure**: Kernel based communication
Understanding the Bottleneck

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

![Diagram of message buffer and memory access]

**Figure:** Direct memory access
Co-hosted on the same machine

Latency similar to embedded DBs, e.g. SQLite

Ideal interconnect for container / Docker environment
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
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:

```
<table>
<thead>
<tr>
<th>Size of message [Byte]</th>
<th>Sync. Throughput [msgs/s]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Write + Polling</td>
</tr>
<tr>
<td>4</td>
<td>Two Writes</td>
</tr>
<tr>
<td>16</td>
<td>Write + Immediate</td>
</tr>
<tr>
<td>64</td>
<td>Send + Receive</td>
</tr>
</tbody>
</table>
```

![Graph showing Sync. Throughput vs Size of message for different operations](image-url)
Low-Latency Communication Using RDMA

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

![Graph showing performance metrics for different data sizes and operations.](attachment:image)

- Sync. Throughput [msgs/s]
- Size of message [Byte]
Low-Latency Communication Using RDMA

- Asymmetric connections
- Many message buffers → 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

![Diagram of mailbox and message buffer]

```
[28] SELECT a FROM r WHERE x = 28
written after message
in-flight
```

```
[30] SELECT e FROM r WHERE x = 81
```
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

![Diagram showing mailbox and message buffer with SQL queries]

SELECT a FROM r WHERE x = 28
SELECT e FROM r WHERE x = 81
## Results

### Local YCSB-C

<table>
<thead>
<tr>
<th></th>
<th>[sync. tx/s]</th>
<th>TCP</th>
<th>SHM</th>
<th>NP</th>
<th>DS</th>
<th>RDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Silo + L5</td>
<td>50.5 K</td>
<td>50.5 K</td>
<td><strong>685 K</strong></td>
<td>—</td>
<td>72.1 K</td>
<td>364 K</td>
</tr>
<tr>
<td>DBMS X</td>
<td>7.56 K</td>
<td>7.56 K</td>
<td>11.5 K</td>
<td>11.5 K</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MySQL</td>
<td>10.0 K</td>
<td>10.0 K</td>
<td>45.9 K</td>
<td>27.6 K</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SQLite</td>
<td>—</td>
<td>—</td>
<td>378 K</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>
### Results

#### Local YCSB-C

<table>
<thead>
<tr>
<th>[sync. tx/s]</th>
<th>TCP</th>
<th>SHM</th>
<th>NP</th>
<th>DS</th>
<th>RDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Silo + L5</td>
<td>50.5 K</td>
<td><strong>685 K</strong></td>
<td>—</td>
<td>72.1 K</td>
<td>364 K</td>
</tr>
<tr>
<td>DBMS X</td>
<td>7.56 K</td>
<td>11.5 K</td>
<td>11.5 K</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>MySQL</td>
<td>10.0 K</td>
<td>45.9 K</td>
<td>27.6 K</td>
<td>—</td>
<td>—</td>
</tr>
<tr>
<td>SQLite</td>
<td>—</td>
<td>378 K</td>
<td>—</td>
<td>—</td>
<td>—</td>
</tr>
</tbody>
</table>

#### Remote YCSB-C

<table>
<thead>
<tr>
<th>[sync. tx/s]</th>
<th>1 G Eth</th>
<th>56 G IB</th>
<th>RDMA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Silo + L5</td>
<td>15 K</td>
<td>27 K</td>
<td><strong>302 K</strong></td>
</tr>
<tr>
<td>DBMS X</td>
<td>3.1 K</td>
<td>3.7 K</td>
<td>—</td>
</tr>
<tr>
<td>MySQL</td>
<td>7.1 K</td>
<td>8.0 K</td>
<td>—</td>
</tr>
<tr>
<td>PostgreSQL</td>
<td>6.3 K</td>
<td>7.5 K</td>
<td>—</td>
</tr>
</tbody>
</table>
Results – Scale Out

The graph below shows the sync. throughput [tx/s] for different numbers of clients, with varying configurations of RDMA (4 servers, 2 servers, 1 server) and TCP. The y-axis represents the sync. throughput in transactions per second (tx/s), while the x-axis illustrates the number of clients. The graph demonstrates the performance differences across these configurations, indicating that RDMA 4 servers generally achieve higher throughput compared to RDMA 2 servers, RDMA 1 server, and TCP.
Conclusion

- **L5 – Low-Level, Low-Latency Library**
  https://github.com/pfent/L5RDMA
- Shared Memory and RDMA bring OLTP performance to clients
- fent@in.tum.de