TeRM: Extending RDMA-Attached Memory with SSD

Zhe Yang¹, Qing Wang¹, Xiaojian Liao¹, Keji Huang², Jiwu Shu¹

¹Tsinghua University
²Huawei Technologies Co., Ltd
RDMA-based Storage System

• RDMA catalyzes in-memory storage systems
  • File systems, key-value stores, transactional databases, ...

- Assise [OSDI’20]
- Pilaf [ATC’13]
- FaRM [NSDI’14]
- Aurogon [FAST’22]
- Octopus [ATC’17]
- Cell [ATC’16]
- Sherman [SIGMOD’22]
- TH-DPMS [TOS’20]
- Orion [FAST’19]
- XStore [OSDI’20]
- Rowan [OSDI’23]
- FileMR [NSDI’20]
- RACE [ATC’21]
- FORD [FAST’22]
- DrTM+H [OSDI’18]
- ROLEX [FAST’23]
- FUSEE [FAST’23]
RDMA-attached Memory

• **Server**
  - Expose virtual memory via **RDMA MR (RDMA-attached Memory)**
  - RNIC accesses the virtual memory via DMA, bypassing the CPU
  - Pin pages in the physical memory; build the RNIC page table

• **Client**
  - Access the MR by one-sided RDMA READ/WRITE

**How to extend RDMA-attached memory with SSD?**
• **On-demand Paging MR**
  - Hardware solution by Mellanox [ASPLOS’17]
  - mmap an SSD and register as an ODP MR
  - The client submits normal RDMA READ/WRITE
• Not all pages are mapped
• Trigger an **RNIC page fault** when accessing an invalid virtual page
• Synchronizing between CPU and RNIC page tables
  • Three flows: faulting, invalidation, advising
ODP MR is not the silver bullet

• **Read 4KB performance**
  - 64GB virtual memory, 32GB physical memory
  - mmap() Intel Optane P5800X SSD
  - (a) 1 client thread
  - (b) 64 client threads

![Graph showing throughput comparison between PIN and ODP](image)

66.64x – 290.76x slowdown!
ODP MR is not the silver bullet

- **Two sources of overhead**
  - A normal read consumes 4μs
  - Hardware: stall & resume QP, trigger interrupt, update RNIC page table
  - Software: CPU page fault

<table>
<thead>
<tr>
<th>Hardware Events</th>
<th>Time (μs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stall QP &amp; Trigger Interrupt</td>
<td>127.37</td>
</tr>
<tr>
<td>CPU Page Fault</td>
<td>242.34</td>
</tr>
<tr>
<td>Update RNIC Page Table</td>
<td>74.17</td>
</tr>
<tr>
<td>Resume QP</td>
<td>128.86</td>
</tr>
</tbody>
</table>

1. Onload exception handling from HW to SW.
2. Eliminate CPU page faults from the critical path.
TeRM overview

• CPU VM
  • mmap; Serves local access (load/store) from the server-side application.

• TeRM MR
  • Serves remote access (memory read/write) from the client-side application.

• tLib-S/tLib-C
  • Server-side/client-side shared library; replaces libibverbs using LD_PRELOAD
• Magic physical page
  • Invalid virtual pages are mapped to this one.
  • Filled with magic pattern.

RDMA READ on invalid virtual pages returns with magic pattern.
Read workflow

• RDMA READ first
  ❶ submit an RDMA READ request
  ❷ receive the response
  ❸ check whether the data contains magic pattern
If no magic pattern is found, the read request completes.
Otherwise, ...

![Diagram showing RDMA READ workflow](image-url)
Read workflow

• RPC READ if necessary
  ① submit an RPC READ request
  ② tLib-S reads data
  ③ tLib-C receives data and completes the read

“principle 1: onload exception handling from HW to SW”
**Write workflow**

- **RPC WRITE for all**
  1. submit an RPC WRITE request
  2. tLib-S writes data
  3. tLib-C receives notification and completes the write

<table>
<thead>
<tr>
<th>Client</th>
<th>Server</th>
</tr>
</thead>
<tbody>
<tr>
<td>CPU</td>
<td>CPU</td>
</tr>
<tr>
<td>RDMA App</td>
<td>RDMA App</td>
</tr>
<tr>
<td>tLib-C</td>
<td>VM</td>
</tr>
<tr>
<td>QP</td>
<td>Physical Memory</td>
</tr>
<tr>
<td>QP</td>
<td>SSD</td>
</tr>
</tbody>
</table>

1. RDMA READ
2. response
3. RPC READ/WRITE
4. response
5. response
How can RPC access data efficiently?

- Load/store the CPU VM?
  - Still triggers CPU page faults!

- Convert memory load/store to file I/O
  - Read/write the SSD
  - “Principle 2: eliminate CPU page faults from the critical path”
How can RPC access data efficiently?

- Convert memory load/store to file I/O
  - SSD LBA range: \([\text{slba}, \text{slba} + \text{length}]\)
  - Virtual address range: \([\text{saddr}, \text{saddr} + \text{length}]\)
  - \(\text{lba} = \text{addr} - \text{saddr} + \text{slba}\)
Tiering IO

- Read/write data via two interfaces
  - Check the page cache
  - Buffer IO for cached data, using page cache
  - Direct IO for uncached data, bypassing page cache
Promoting Hotspots

- **Client-side**
  - Count accesses on each unit
- **Server-side**
  - Aggregate counters from all clients
  - Find most-accessed units as hotspots
  - Promote via `ibv_advise_mr()`
Evaluation

• **Testbed**
  - RDMA Cluster: server machine * 1, client machine * 2
  - SSD: Intel Optane P5800X 400GB
  - RNIC: ConnectX-5 100Gbps
  - Switch: IB 100Gbps

• **Settings**
  - Virtual memory: 64GB, physical memory: 32GB
  - 64 Client threads, 16 server threads
Evaluation

• **Comparing Targets**
  • **PIN**: ideal upper bound, *all pages in the physical memory*
  • **ODP**: *hardware* solution, **ODP MR**
  • **RPC**: *software* solution, all requests via **RPC**, access data via **memcpy**
  • **TeRM**: our solution.
Evaluation: Overall Performance

• Read
  • vs. ODP: 30.46x – 549.63x
  • vs. RPC: 9.05x – 45.19x
  • vs. PIN: 37.79% – 96.71%
Evaluation: Overall Performance

• **Write**
  - **vs. ODP:** ~1000x (ODP write is very unstable and jitters sharply)
  - **vs. RPC:** 7.73x – 12.60x
  - **vs. PIN:** 6.55% – 96.32%
Evaluation: Dynamic Workloads

• Change hotspots at the 60\textsuperscript{th} second
  • Performs stably: drops by only 6.82%
  • Promoting fast: returns to the peak in 1 second
Evaluation: RDMA-based storage system

- **Octopus: A File System [OSDI’20]**
  - **Workloads:** read/write the file
  - **Results:** up to $642.23x$ ODP, $7.68x$ RPC
Evaluation: RDMA-based storage system

• **XStore: A Key-Value System [ATC’17]**
  - **Workloads:** YCSB-C, read 8B keys and 128B values
  - **Results:** Up to 102.97x ODP, 2.69x RPC
Conclusion

• TeRM proposes an efficient approach to extending RDMA-attached memory with SSD.

• TeRM onloads exception handling from hardware to software and eliminates RNIC & CPU page faults on the critical path.

• TeRM implements a userspace shared library to replace libibverbs and run unmodified RDMA applications transparently.

• TeRM outperforms the hardware-only ODP MR by up to 642.23x, and the software-only RPC approach by up to 7.68x.
TeRM: Extending RDMA-Attached Memory with SSD

Zhe Yang, Qing Wang, Xiaojian Liao, Keji Huang, Jiwu Shu

yangz18@mails.tsinghua.edu.cn

https://github.com/thustorage/TeRM