Designing MPI Library with On-Demand Paging (ODP) of InfiniBand: Challenges and Benefits

Mingzhe Li  Khaled Hamidouche  Xiaoyi Lu
Hari Subramani  Jie Zhang  Dhabaleswar K. Panda

Network-Based Computing Laboratory
Department of Computer Science and Engineering
The Ohio State University
Outline

• Introduction

• Problem Statement

• Analyze ODP Performance at Verbs-level

• Design an ODP-Aware MPI Runtime

• Performance Evaluation

• Conclusion and Future Work
Drivers of Modern HPC Cluster Architectures

- Remote Direct Memory Access (RDMA)-enabled networking (InfiniBand and RoCE)
- < 1usec latency, 100Gbps bandwidth
Existing RDMA Communication on InfiniBand

• Pin-down pages of communication buffer for RDMA operations
  – Pin-down/Unpin pages is costly

• MPI stacks on InfiniBand
  – Pinned bounce buffers: small messages
  – Pin-down cache: large messages

• Bottlenecks with existing scheme
  – Limit physical memory resources for computation
  – Contentions between multiple jobs on the same node
  – Pinned buffers have to fit in physical memory
On-Demand Paging (ODP)

• A new feature of IB introduced by Mellanox for RDMA communications

• Communication buffers are no longer pinned
  – Paged in when the host channel adapter (HCA) needs them
  – Swapped out when reclaimed by the OS
Overview of ODP

- How page fault is handled:
  - 1: Get pages, map to DMA
  - 2: Update mapping, resume communication

- Pre-fetch: a new verbs-level operation introduced by Mellanox to pre-fetch pages
  - Warm up communication buffers
  - No guarantee that pages remain resident
Outline

• Introduction
• Problem Statement
• Analyze ODP Performance at Verbs-level
• Design an ODP-Aware MPI Runtime
• Performance Evaluation
• Conclusion and Future Work
Problem Statement

- What are the performance characteristics of ODP performance at IB verbs-level?
- Can we design an high performance ODP-Aware MPI runtime for point-to-point and one-sided routines?
- What kind of benefits can be achieved at applications using ODP-Aware MPI runtime?
Outline

• Introduction
• Problem Statement
• Analyze ODP Performance at Verbs-level
• Design an ODP-Aware MPI Runtime
• Performance Evaluation
• Conclusion and Future Work
Experimental Setup

- ODP has not yet been deployed at any serious scale
- Nowlab Cluster
  - 4 Sandy Bridge, 2 Ivy Bridge, 2 Haswell Compute Nodes
  - MT4113 Connect-IB HCAs (56 Gbps data rate)
  - Mellanox OpenFabrics version 3.1-1.0.3
  - RHEL 7.1.1503 with kernel version 4.2.3
- A set of proposed verbs-level benchmarks to analyze ODP performance
Verbs-level Latency & Bandwidth

- A 4MB send/receive buffer on each process, one process per node
- ODP achieves the same performance as Pin-down if there is no page fault
- A memory region is touched does not imply that the HCA has a mapping for it
Page Fault / Page Pre-fetch Overhead

- Page fault only happens in the first iteration, but the overhead is high
- The pre-fetch operation needs around 5us to warm up 2 bytes buffer
Page Fault Overhead at Sender/Receiver

- Pre-fetch operation works very effectively to avoid page faults
- Overhead of resolving page fault at sender or receiver is different
- Page fault at receiver side needs to suspend Queue Pair (QP) to avoid congestion
Outline

• Introduction
• Problem Statement
• Analyze ODP Performance at Verbs-level
  • Design an ODP-Aware MPI Runtime
• Performance Evaluation
• Conclusion and Future Work
Challenges of High Performance ODP-Aware MPI Runtime

- A native design uses ODP for RDMA operations
  - Adds page fault overhead in critical paths

- An enhanced design with pre-fetch optimization
  - Incurs page pre-fetch overhead in critical paths

- The following challenges need to be resolved:
  - How to reduce the number of page faults?
  - When should pre-fetch operations be issued?
  - How to ensure that pre-fetched buffers are still valid (resident, mapping)?
  - How to apply these designs for different protocols?
MPI Point-to-Point with ODP (1)

• Most MPI stacks have two protocols for data transfers
  – Eager protocol: eagerly sends data to the other process
  – Rendezvous protocol: handshake before data exchange

• Eager Protocol:
  – Bounce buffers are around few MBs per process
  – Small messages are latency sensitive
  – Pre-fetch bounce buffer before send incurs overhead
 MPI Point-to-Point with ODP (2)

- Rendezvous protocol with blocking primitives
  - Pre-fetched buffer after RTS/CTS messages

- Rendezvous Protocol with non-blocking primitives
  - Issue pre-fetch for the same buffer again
  - Pre-fetch operations could be overlapped with the handshake phase
MPI Remote Memory Access (RMA) with ODP

• RMA window with ODP
  – Overhead of pre-fetch could not be overlapped
  – Pre-fetched window buffer could be invalidated

• RMA communication with ODP
  – Pre-fetch window buffers in synchronization operation
  – Get: pre-fetch origin buffers after RDMA operation has been issued
Overview of the MVAPICH2 Project

• High Performance open-source MPI Library for InfiniBand, Omni-Path, Ethernet/iWARP, and RDMA over Converged Ethernet (RoCE)
  – MVAPICH (MPI-1), MVAPICH2 (MPI-2.2 and MPI-3.0), Started in 2001, First version available in 2002
  – MVAPICH2-X (MPI + PGAS), Available since 2011
  – Support for GPGPUs (MVAPICH2-GDR) and MIC (MVAPICH2-MIC), Available since 2014
  – Support for Virtualization (MVAPICH2-Virt), Available since 2015
  – Support for Energy-Awareness (MVAPICH2-EA), Available since 2015
  – Support for InfiniBand Network Analysis and Monitoring (OSU INAM) since 2015
  – Used by more than 2,675 organizations in 83 countries
  – More than 401,000 (> 0.4 million) downloads from the OSU site directly
  – Empowering many TOP500 clusters (Nov ‘16 ranking)
    • 1st ranked 10,649,600-core cluster (Sunway TaihuLight) at National Supercomputing Center in Wuxi
    • 13th ranked 241,108-core cluster (Pleades) at NASA
    • 17th ranked 462,462-core cluster (Stampede) at TACC
    • 40th ranked 74,520-core cluster (Tsubame 2.5) at Tokyo Institute of Technology and many others
  – Available with software stacks of many vendors and Linux Distros (RedHat and SuSE)
  – http://mvapich.cse.ohio-state.edu

• Empowering Top500 systems for over a decade
  – System-X from Virginia Tech (3rd in Nov 2003, 2,200 processors, 12.25 TFlops)
  – Stampede at TACC (12th in Jun’16, 462,462 cores, 5.168 Plops)
Outline

• Introduction
• Problem Statement
• Analyze ODP Performance at Verbs-level
• Design an ODP-Aware MPI Runtime
  • Performance Evaluation
• Conclusion and Future Work
Both ODP-Enhanced and ODP-Optimal schemes achieve the same performance as the pin-down scheme.

The performance degradation with ODP-Native scheme is caused by page fault.
Application Level Experiments

- ODP-Optimal scheme achieves the same performance as the pin-down scheme.
- For AWP-ODC with input size 512X512X512, ODP-Optimal scheme could reduce the total execution time by 23X compared with other ODP schemes.
### Application Level Performance Analysis

<table>
<thead>
<tr>
<th>Apps.</th>
<th># Page Fault Events</th>
<th>#Pre-fetched Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>ODP-Native</td>
<td>ODP-Enhanced</td>
</tr>
<tr>
<td>NAS-CG</td>
<td>245</td>
<td>143</td>
</tr>
<tr>
<td>NAS-LU</td>
<td>280</td>
<td>180</td>
</tr>
<tr>
<td>NAS-MG</td>
<td>986</td>
<td>730</td>
</tr>
<tr>
<td>Lammmps-LJ</td>
<td>3023</td>
<td>2880</td>
</tr>
<tr>
<td>Lammmps-EAM</td>
<td>5701</td>
<td>4430</td>
</tr>
<tr>
<td>AWP-ODC (256^3)</td>
<td>2986</td>
<td>2014</td>
</tr>
<tr>
<td>AWP-ODC (512^3)</td>
<td>20638</td>
<td>17252</td>
</tr>
<tr>
<td>Boltzmann</td>
<td>1052</td>
<td>704</td>
</tr>
<tr>
<td>Lennard</td>
<td>1745</td>
<td>1623</td>
</tr>
</tbody>
</table>

- Profile the number of page fault events and pre-fetched pages to analyze the performance difference.
- For AWP-ODC with input size 512X512X512, ODP-Optimal scheme could significantly reduce the number of page fault events.
• Pin-down buffer size is significantly reduced with ODP schemes
• For LJ application, the size of pin-down buffers could be reduced by 11X
Conclusion & Future Work

- Presented the ODP feature and analyzed its performance characteristics with IB verbs-level micro-benchmarks
- Designed a high performance ODP-Aware MPI runtime to support MPI point-to-point and RMA routines
- Proposed ODP-Aware MPI runtime shows good performance at micro-benchmarks and applications
- Plan to evaluate proposed designs at scale
Thank You!

{limin, hamidouc, luxi, subramon, zhanjie, panda}@cse.ohio-state.edu

Network-Based Computing Laboratory
http://nowlab.cse.ohio-state.edu/

MVAPICH Web Page
http://mvapich.cse.ohio-state.edu/