Agenda

• Introduction
• Overview of Amazon EFA
• Designing MPI Libraries for EFA
• Experimental Evaluations
• Conclusion
Cloud Computing widely adopted in industry computing environment

Cloud Computing provides high resource utilization and flexibility

Virtualization is the key technology to enable Cloud Computing

Intersect360 study shows cloud is the fastest growing class of HPC

HPC Meets Cloud: The convergence of Cloud Computing and HPC
Agenda

• Introduction

• **Overview of Amazon EFA**
  
• Designing MPI Libraries for EFA

• Experimental Evaluations

• Conclusion
Amazon Elastic Fabric Adapter (EFA)

- Enhanced version of Elastic Network Adapter (ENA)
- Allows OS bypass, up to 100 Gbps bandwidth
- Network aware multi-path routing
- Exposed through libibverbs and libfabric interfaces
- Introduces new Queue-Pair (QP) type
  - Scalable Reliable Datagram (SRD)
  - Also supports Unreliable Datagram (UD)
  - No support for Reliable Connected (RC)

<table>
<thead>
<tr>
<th>Model</th>
<th>Bandwidth</th>
<th>Latency</th>
</tr>
</thead>
<tbody>
<tr>
<td>C1</td>
<td>1 Gbps</td>
<td>~100us</td>
</tr>
<tr>
<td>CC1</td>
<td>10 Gbps</td>
<td>~100us</td>
</tr>
<tr>
<td>C3</td>
<td>~100us</td>
<td>optimiz ed</td>
</tr>
<tr>
<td>C4</td>
<td>EBS</td>
<td>~50 us</td>
</tr>
<tr>
<td>C5</td>
<td>ENA 25 Gbps</td>
<td>~50 us</td>
</tr>
<tr>
<td>C5n</td>
<td>EFA 100 Gbps</td>
<td>~15 us</td>
</tr>
</tbody>
</table>
## IB Transport Types and Associated Trade-offs

<table>
<thead>
<tr>
<th>Attribute</th>
<th>Reliable Connection</th>
<th>Reliable Datagram</th>
<th>Dynamic Connected</th>
<th><strong>Scalable Reliable Datagram</strong></th>
<th>Unreliable Connection</th>
<th>Unreliable Datagram</th>
<th>Raw Datagram</th>
</tr>
</thead>
<tbody>
<tr>
<td>Scalability (M processes, N nodes)</td>
<td>M(^2)N QPs per HCA</td>
<td>M QPs per HCA</td>
<td>M QPs per HCA</td>
<td>M QPs per HCA</td>
<td>M(^2)N QPs per HCA</td>
<td>M QPs per HCA</td>
<td>1 QP per HCA</td>
</tr>
<tr>
<td>Corrupt data detected</td>
<td>Yes</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Delivery Guarantee</td>
<td>Data delivered exactly once</td>
<td></td>
<td></td>
<td>No guarantees</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>Data Order Guarantees</td>
<td>Per connection</td>
<td>One source to multiple destinations</td>
<td>Per connection</td>
<td>No Unordered, duplicate data detected</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Data Loss Detected</td>
<td>Yes</td>
<td></td>
<td></td>
<td>No</td>
<td>No</td>
<td>No</td>
<td></td>
</tr>
<tr>
<td>Error Recovery</td>
<td>Errors (retransmissions, alternate path, etc.) handled by transport layer. Client only involved in handling fatal errors (links broken, protection violation, etc.)</td>
<td></td>
<td></td>
<td>Errors are reported to responder</td>
<td>None</td>
<td>None</td>
<td></td>
</tr>
</tbody>
</table>
## Scalable Reliable Datagrams (SRD): Features & Limitations

<table>
<thead>
<tr>
<th>Feature</th>
<th>UD</th>
<th>SRD</th>
</tr>
</thead>
<tbody>
<tr>
<td>Send/Recv</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Send w/ Immediate</td>
<td>✖</td>
<td>✖</td>
</tr>
<tr>
<td>RDMA</td>
<td>✖</td>
<td>✖</td>
</tr>
<tr>
<td>Read/Write/Atomic</td>
<td>✖</td>
<td>✖</td>
</tr>
<tr>
<td>Scatter Gather Lists</td>
<td>✔</td>
<td>✔</td>
</tr>
<tr>
<td>Shared Receive Queue</td>
<td>✖</td>
<td>✖</td>
</tr>
<tr>
<td>Reliable Delivery</td>
<td>✖</td>
<td>✔</td>
</tr>
<tr>
<td>Ordering</td>
<td>✖</td>
<td>✖</td>
</tr>
<tr>
<td>Inline Sends</td>
<td>✖</td>
<td>✖</td>
</tr>
<tr>
<td>Global Routing Header</td>
<td>✔</td>
<td>✖</td>
</tr>
<tr>
<td>Max Message Size</td>
<td>4KB</td>
<td>8KB</td>
</tr>
</tbody>
</table>

- **Similar to IB Reliable Datagram**
  - No limit on number of outstanding messages per context

- **Out of order delivery**
  - No head-of-line blocking
  - Bad fit for MPI, can suit other workloads

- **Packet spraying over multiple ECMP paths**
  - No hotspots
  - Fast and transparent recovery from network failures

- **Congestion control designed for large scale**
  - Minimize jitter and tail latency

---

*Amazon Elastic Fabric Adapter: Anatomy, Capabilities, and the Road Ahead, Raghu Raja, OpenFabrics Workshop 2019*
Verbs level evaluation of EFA performance

- SRD adds 8-10% overhead compared to UD
- Due to hardware based acks used for reliability

- Instance type: c5n.18xlarge
- CPU: Intel Xeon Platinum 8124M @ 3.00GHz
Agenda

• Introduction
• Overview of Amazon EFA
• Designing MPI Libraries for EFA
• Experimental Evaluations
• Conclusion
Designing MPI libraries for EFA

- MPI offer various communication primitives
  - Point-to-point, Collective, Remote Memory Access
  - Provides strict guarantees about reliability and ordering
  - Allows message sizes much larger than allowed by the network

- How to address these semantic mismatches between the network and programming model in a scalable and high-performance manner?

Shared Memory Model
SHMEM, DSM

Distributed Memory Model
MPI (Message Passing Interface)

Partitioned Global Address Space (PGAS)
OpenSHMEM, UPC, UPC++, CAF …
Challenge 1: Reliable and in-order delivery

- MPI guarantees reliable and in-order message matching to applications
- UD does not provide reliability or ordering
- SRD provides reliability but not in-order delivery

- Solution: use acknowledgements and retransmissions for reliability
- Piggy back acks on application messages for reducing overhead
- Use sequence number and sliding window for re-ordering packets at the receiver process

Challenge 2: Zero-copy transmission of large messages

- MPI allows sending and receiving very large messages
- Network message size bound by MTU size (4KB for UD, 8KB for SRD)
- Need to handle segmentation and reassembly
- Existing zero-copy designs* can not be used
  - Utilizes send-with-immediate for sequence numbers (not supported by EFA)
  - Retransmits entire message if out-of-order arrival is detected
- Solution: propose new design for zero-copy rendezvous transfers
  - Maintain a pool of dedicated QPs for zero-copy transfers
  - Use scatter gather lists for sequence numbers
  - Reorder out-of-order packets at the receiver

Issues with out-of-order packets for zero-copy transfers

Send Buffer

Send\textsubscript{1} \hspace{1cm} Send\textsubscript{2} \hspace{1cm} Send\textsubscript{3}

Recv\textsubscript{1} \hspace{1cm}Recv\textsubscript{2} \hspace{1cm}Recv\textsubscript{3}

In-order Message \hspace{1cm} Out-of-order Message

Not enough Space In Receive buffer!

Receive Buffer

Need to use temporary buffers to ensure enough space for incoming messages!
Handling out-of-order packets for zero-copy transfers

Two MTU sized buffers are required (one for receiving, one for re-ordering)
Agenda

• Introduction
• Overview of Amazon EFA
• Designing MPI Libraries for EFA
• **Experimental Evaluations**
• Conclusion
Experimental Setup

- Instance type: c5n.18xlarge
- CPU: Intel Xeon Platinum 8124M @ 3.00GHz
- Cores: 2 Sockets, 18 cores / socket
- KVM Hypervisor, 192 GB RAM, One EFA adapter / node
- MVAPICH2 version: MVAPICH2-X 2.3rc2 + SRD support
- OpenMPI version: Open MPI v3.1.3 with libfabric 1.7
Point-to-Point Performance

- Both UD and SRD shows similar latency for small messages.
- SRD shows higher message rate due to lack of software reliability overhead.
- SRD is faster for large messages due to larger MTU size.
Point-to-Point Performance

Pt2pt Latency – Small Messages

- MV2X
- OpenMPI
- IntelMPI

Pt2pt Latency Large Messages

- MV2X
- OpenMPI
- IntelMPI

Pt2pt BW – Small Messages

- MV2X
- OpenMPI
- IntelMPI

Instance type: c5n.18xlarge
CPU: Intel Xeon Platinum 8124M @ 3.00GHz
MVAPICH2 version: Latest MVAPICH2-X + SRD support
OpenMPI version: Open MPI v4.0.2 with libfabric 1.8
Collective Performance: MPI Allreduce

- Up to 18% improvement with SRD compared to UD
- Bidirectional communication pattern allows piggybacking of acks
- Modest improvement compared to asymmetric communication patterns
Collective Performance: Large Scale

- Up to 25x improvement in Allreduce with large messages
- Up to 6x better performance in Bcast with large messages
- Up to 12x better performance in Gather with small messages
• Up to 10% performance improvement for MiniGhost on 8 nodes

• Up to 27% better performance with CloverLeaf on 8 nodes
Agenda

• Introduction
• Overview of Amazon EFA
• Designing MPI Libraries for EFA
• Experimental Evaluations
• Conclusion
Conclusion

• HPC workloads are being run on cloud environments
  – Networks differ significantly from traditional on-premise HPC clusters
  – MPI libraries need to leverage the features and address the limitations

• Amazon Elastic Fabric Adapter provides lower latency
  – Introduces Scalable Reliable Datagram transport
  – Take advantage of hardware level reliable delivery in MPI
  – Proposed designs for zero-copy transmission of large messages
  – Show significant improvement in microbenchmarks and applications

• MVAPICH2-X for AWS 2.3 released on 04/12/2019
  – Includes support for SRD and XPMEM based transports
  – Available for download from http://mvapich.cse.ohio-state.edu/downloads/
Thank You!

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

The High-Performance MPI/PGAS Project
http://mvapich.cse.ohio-state.edu/

The High-Performance Big Data Project
http://hibd.cse.ohio-state.edu/

The High-Performance Deep Learning Project
http://hidl.cse.ohio-state.edu/