



UCX ON AWS: ADDING SUPPORT FOR AMAZON EFA

HESSAM MIRSADEGHI, AKSHAY VENKATESH, JIM DINAN, SREERAM POTLURI

OUTLINE

- HPC in the cloud
- Elastic Fabric Adapter (EFA) from Amazon
- EFA support in UCX
- Performance results

HPC IN THE CLOUD



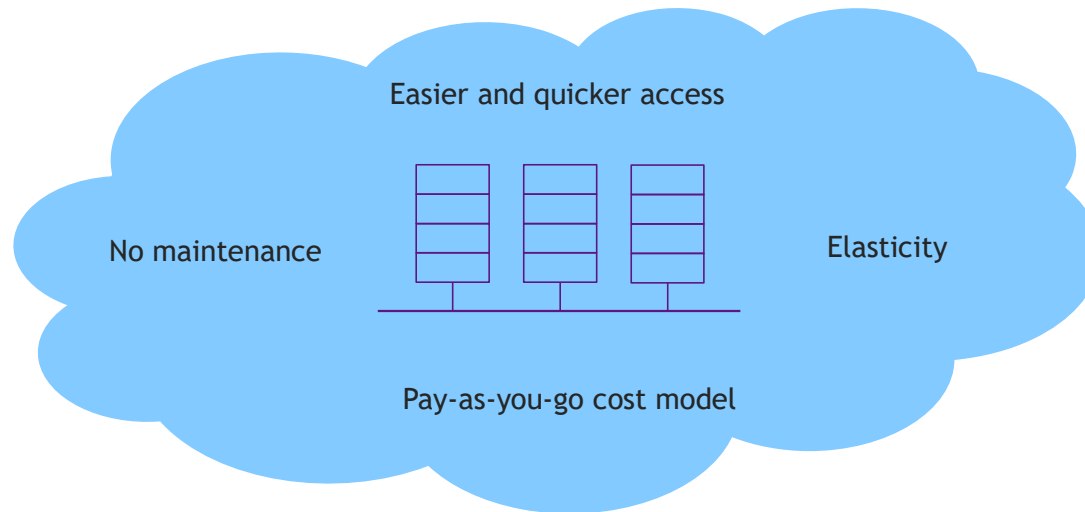
Traditional HPC



Deep learning



Data science



WHAT ABOUT PERFORMANCE?



Networking



Compute

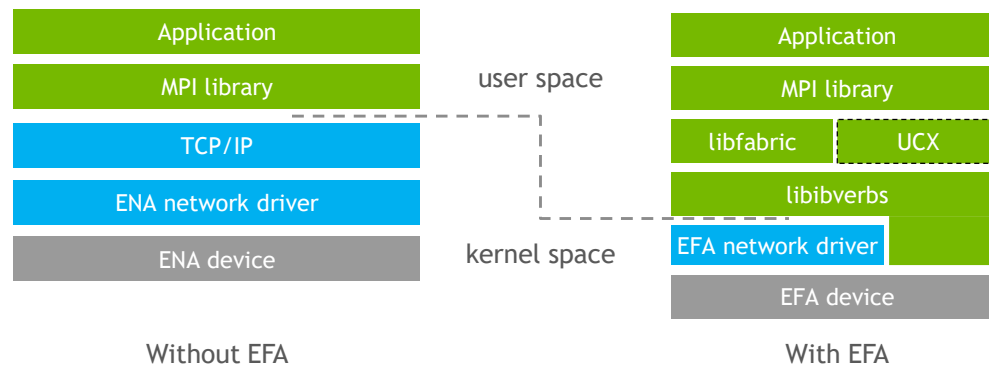


Storage

HPC ON AWS

- **Elastic Fabric Adapter (EFA)**

- HPC-optimized network interface for EC2 instances
- 100 Gbps link bandwidth
- OS bypass and RDMA
- GPUDirect RDMA



EFA NETWORK TRANSPORT PROTOCOLS

| | Unreliable Datagram (UD) | Scalable Reliable Datagram (SRD) |
|--------------|---|----------------------------------|
| Connected | No | No |
| Reliable | No | Yes |
| Ordered | No | No |
| SEND/RECEIVE | Yes, 4 KB MTU | Yes, 8 KB MTU |
| RDMA READ | No | Yes |
| RDMA WRITE | No | No |
| Flow limit | Yes, 10 Gbps in a cluster placement group | No |

WHY SRD?

AWS Datacenter Characteristics

- Avoid creating islands of specialized networks
 - Customers like instance choice
 - Must have the capacity right away
- Commodity Ethernet switches
- Equal Cost Multipath Routing (ECMP)
 - Static hash-based mapping of flows to paths (load balancing)
 - Preserves per-flow order for TCP
 - Hash collisions → hotspots → packet drop → decreased throughput

WHY SRD?

Relaxed ordering benefits

- Avoid ECMP limitations to maximize multipath routing
 - SRD sender controls ECMP path selection by manipulating packet encapsulation
- A single flow is sprayed over multiple paths
 - Minimize the chance of creating hotspots
 - Avoid existing hotspots and suboptimal paths
 - No head-of-line blocking
 - Faster recovery in case of link failures

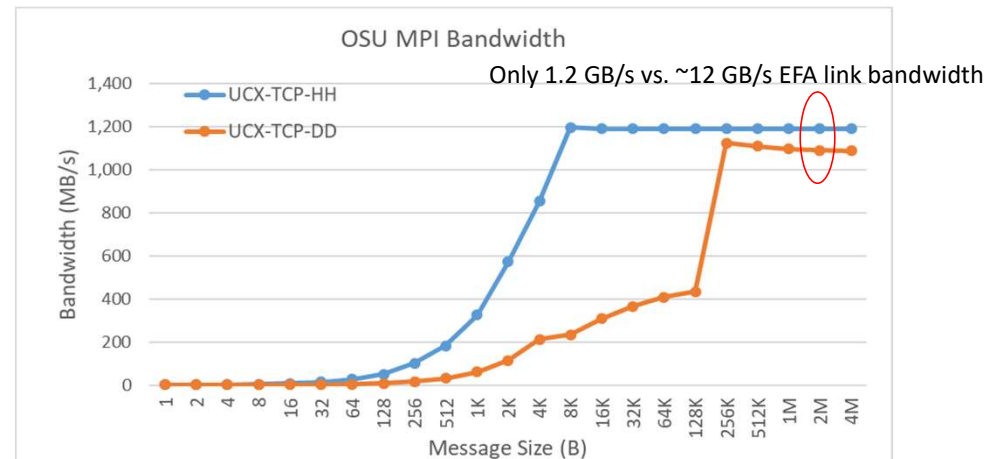
INSTANCE TYPES WITH EFA SUPPORT

| Instance Type | vCPU | Memory (GB) | NIC | GPU | GPUDirect RDMA | Cost (\$ / hour) |
|---------------|-------------------|-------------|--------------|----------------------|----------------|------------------|
| c5n.18xlarge | 72 (Skylake) | 192 | 1 EFA | None | No | \$3.88 |
| g4dn.metal | 96 (Cascade lake) | 384 | 1 EFA | 8 T4, 128 GB | No | \$7.82 |
| p3dn.24xlarge | 96 (Skylake) | 768 | 1 EFA | 8 V100, 256 GB | No | \$31.21 |
| p4d.24xlarge | 96 (Cascade lake) | 1152 | 4 EFA | 8 A100, 40 GB | Yes | \$32.77 |

EFA SUPPORT IN UCX

Motivation

- Deliver better-than-TCP performance on AWS
- Achieve the full potential of EFA
- Enable UCX software ecosystem on AWS
 - RAPIDS (spark, ucx-py), DASK
 - A lot of effort has been put into making UCX a good match
 - More isolated error handling
 - Better integration with python progress model



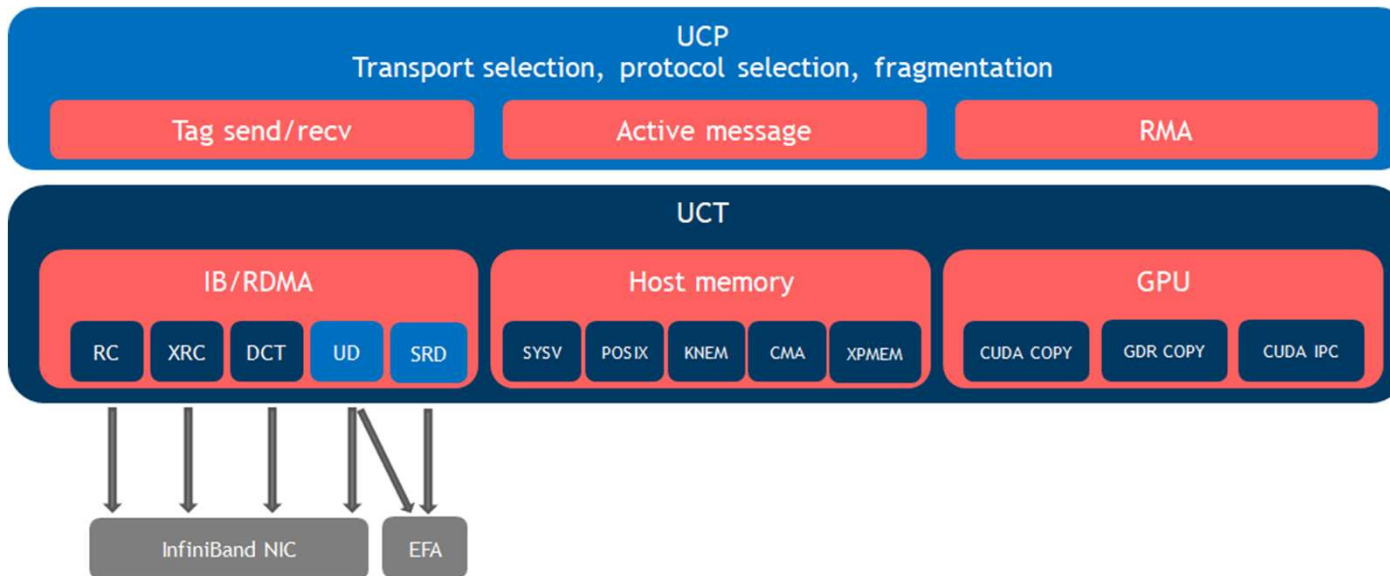
OSU MPI bandwidth with Open MPI+UCX+TCP (two p4d instances)

HH: host-to-host communication


DD: device-to-device communication

EFA SUPPORT IN UCX

UCT Layer Updates

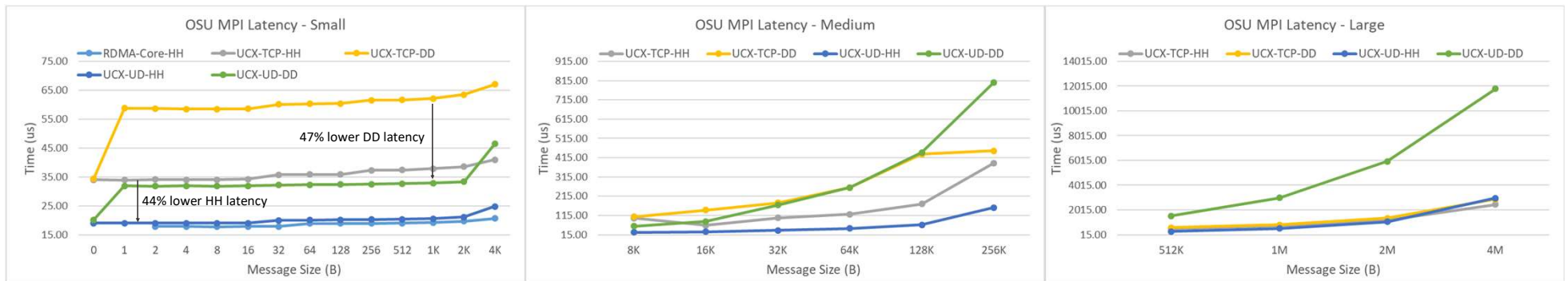


UCX+UD OVER EFA

- UCX already has support for IB UD verbs
 - EFA UD is exposed through the verbs API
 - EFA is not 100% IBTA compliant (EFA is not InfiniBand)
 - No guarantees on the completion order of work requests
 - Cannot use batch completion processing
 - Supports only a subset of memory registration access flags
 - `IBV_ACCESS_LOCAL_WRITE`
 - `IBV_ACCESS_REMOTE_READ` (p4d instances)
 - No interrupt-based completion notifications
- 
- Add a new EFA Memory Domain (MD) in UCT
 - Capture EFA-specific features/limitations
 - Max inline size, RDMA READ and its max size, etc.
 - `uct_ib_efa_md_open`, `uct_ib_efadv_check`, `uct_ib_efadv_query`
 - Add to UCT IB device
 - Supported registration access flags
 - Whether it provides CQ notifications
 - Whether it provides in-order WR completion
 - Update UD verbs accordingly
 - Don't use `IBV_SEND_SOLICITED` if CQ notification not supported
 - Don't use batch completion processing if in-order completion is not guaranteed

UCX+UD OVER EFA

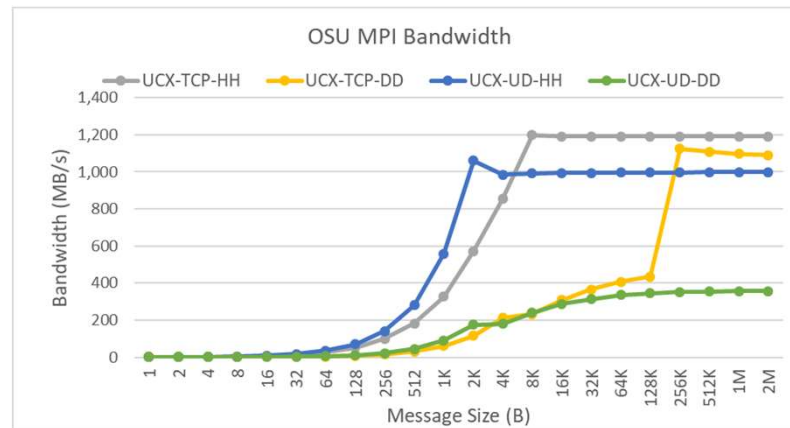
Latency Results



- Lower latency for small and medium messages with UCX+UD vs. UCX+TCP
 - Very close to RDMA-core
- Higher latency for device-to-device transfers compared to host-to-host
 - Extra host-to-device copy overhead at the receiver side
- Very high latency with UCX+UD for large-message device-to-device transfers
 - No GPUDirect RDMA over UD
 - Copy-based pipeline protocol with a small copy chunk size (~4 KB for UD vs. 256 KB for TCP)

UCX+UD OVER EFA

Bandwidth Results

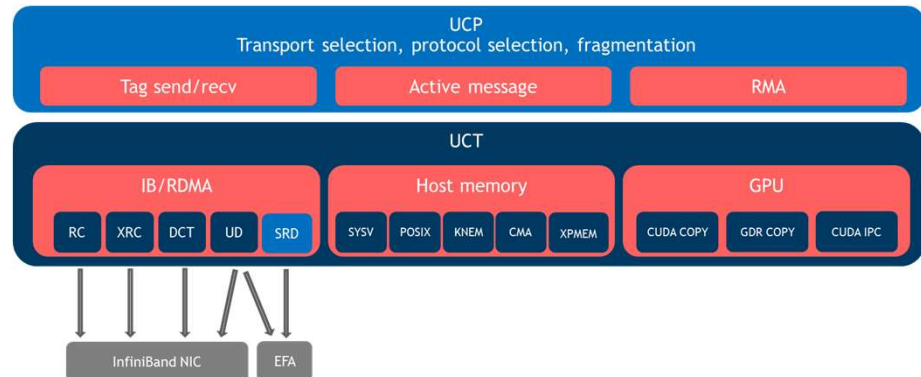


- Low saturation bandwidth with UCX+UD (only 1 GB/s)
 - 10 Gbps flow limit
- Very low bandwidth with UCX+UD for device-to-device transfers
 - No GPUDirect RDMA over UD
 - Small (~4 KB) pipeline copy chunk size with UD due to MTU size

UCX+SRD OVER EFA

SRD UCT Interface

- UCT Active Message (AM) API
 - am_short, am_bcopy, am_zcopy
- UCT RMA API
 - get_short, get_bcopy, get_zcopy



SRD UCT AM INTERFACE

Relaxed ordering impacts

- Packets must be delivered to UCP in order (MPI tag matching)
 - Add sequence number to each packet
 - Buffer out-of-order packets at the receiver end point
 - Use a window-based flow control to avoid sender overwhelming receiver
- UCP already handles fragmentation and reassembly
 - UCT interface advertises the maximum message size it can deliver without fragmentation

UCT FLUSH WITH SRD

- Flush operation: Notify user when all communication requests posted so far are completed

- With guarantee for in-order request completion

- Push each outstanding request into a queue
- Completion of request R_k implies completion of all requests before R_k too

- Without guarantee for in-order completion

- Enqueue each request (including flush)
- Remove only the completed request from the queue
- While queue head is flush, dequeue and call the callback



Outstanding queue



Outstanding queue

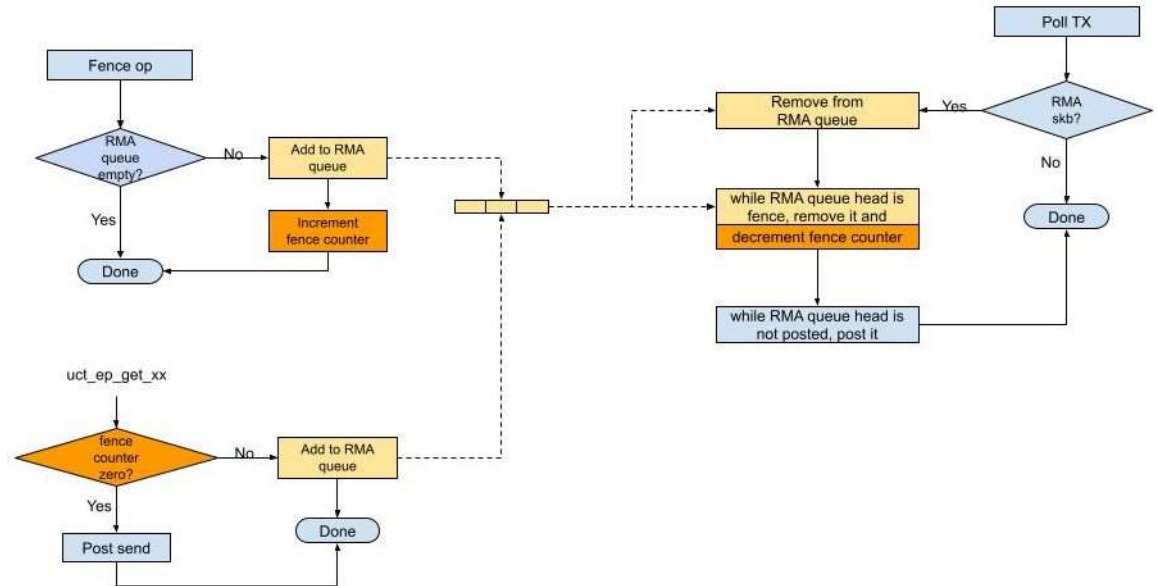


Outstanding queue

SRD UCT RMA INTERFACE

- Implement `get` operations using RDMA READ
- No `put` operations due to lack of RDMA WRITE
- EFA takes care of fragmentation/reassembly for RDMA READ (up to 1 GB)
- Two RDMA READ operations can complete out-of-order
 - The fence operation cannot be no-op anymore

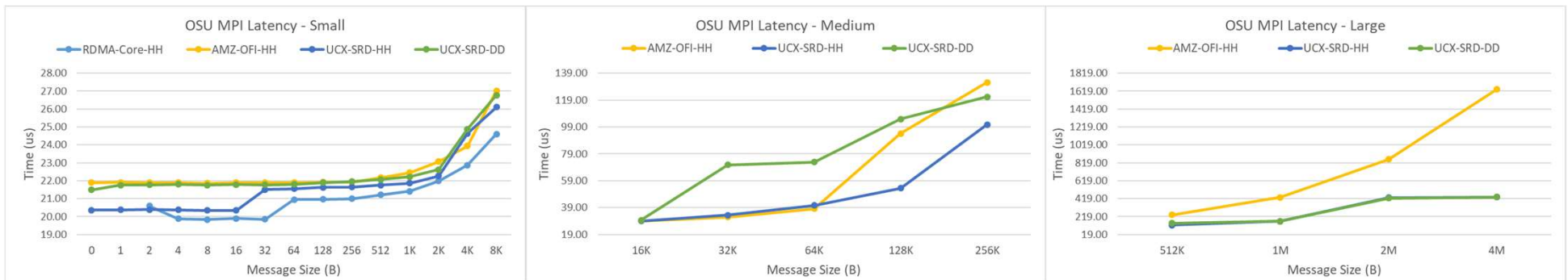
UCT FENCE WITH SRD



- Do not post the get request to EFA device if there is an outstanding fence
 - Add to a queue instead (RMA queue)
- Fence op inserts a request into the RMA queue if there is an outstanding get op
- Completion of a queue-head get request clears it's immediately following fence operation

UCX+SRD OVER EFA

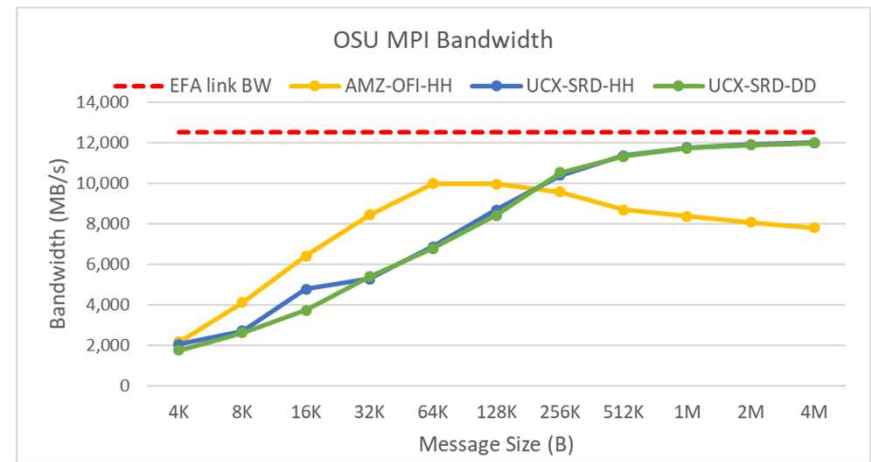
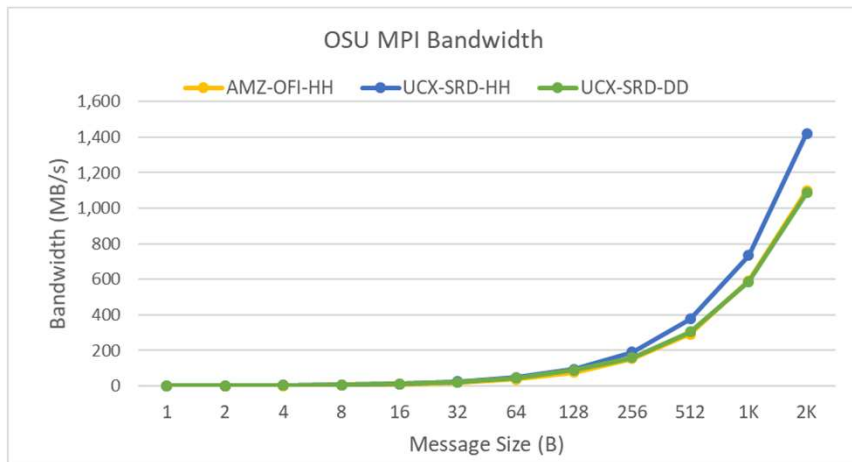
Latency Results



- AMZ-OFI: Open MPI + libfabric with the EFA/SRD provider from Amazon
 - Does not support device buffers yet
 - FI_EFA_USE_DEVICE_RDMA not set (setting it will give better results for libfabric)
- Small-message latency close to RDMA-core with UCX+SRD
- Lower/similar latency with UCX+SRD compared to libfabric for host-to-host communications
- Low latency for small-message device-to-device communications with UCX+SRD
- Large-message device-to-device latency same as host-to-host with UCX+SRD
 - Taking advantage of GPUDirect RDMA

UCX+SRD OVER EFA

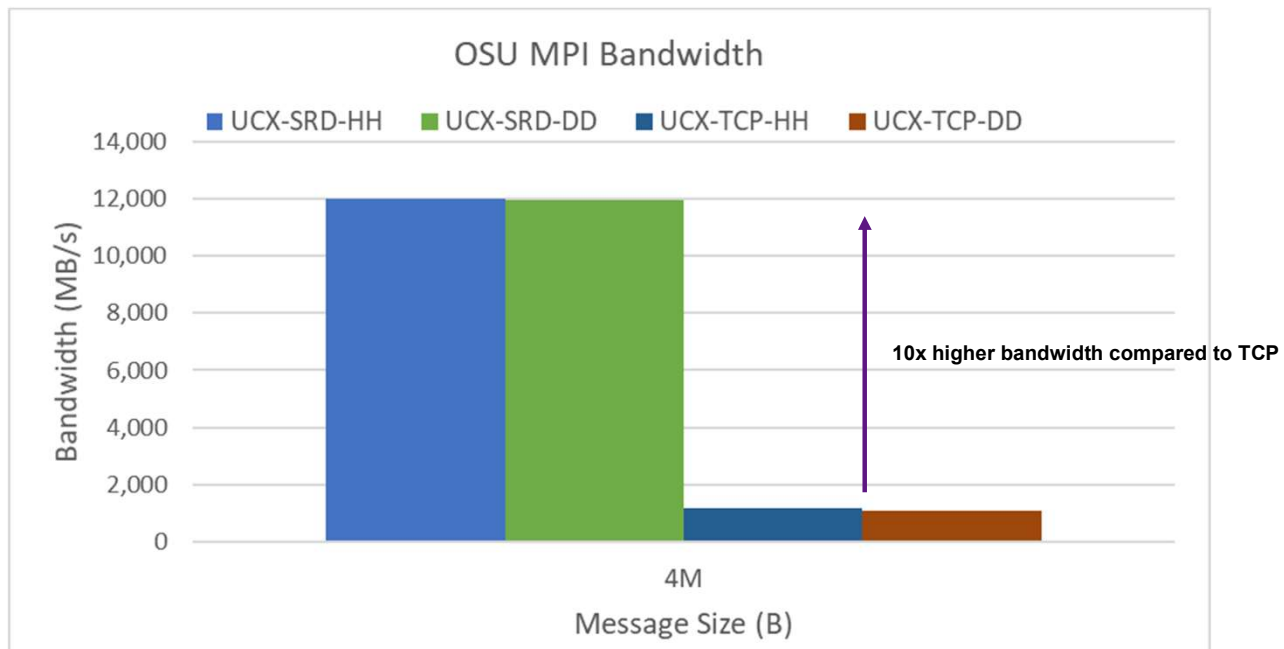
Bandwidth Results



- UCX+SRD successfully saturates EFA link bandwidth and achieves 12 GB/s
 - For both host and device buffers (GPUDirect RDMA)
- FI_EFA_USE_DEVICE_RDMA not set for libfabric
 - setting it will result in higher bandwidth with libfabric

UCX+SRD OVER EFA

Bandwidth versus TCP



CONCLUSION

- EFA+SRD provides AWS users with HPC-grade communication performance
- SRD uses relaxed ordering semantics to achieve high bandwidth on AWS network
- We added an SRD UCT transport in UCX to deliver EFA benefits to UCX users
 - 10x higher bandwidth compared to UCX+TCP on AWS
- Code status
 - Still WIP and requiring review and testing
 - UCX EFA memory domain: <https://github.com/openucx/ucx/pull/6653>
 - UCX UD over EFA: <https://github.com/openucx/ucx/pull/6353>
 - UCX SRD: <https://github.com/openucx/ucx/pull/6636>

Thank You!

Hessam Mirsadeghi hmirsadeghi@nvidia.com

