



arm

# HPC Network Stack Update

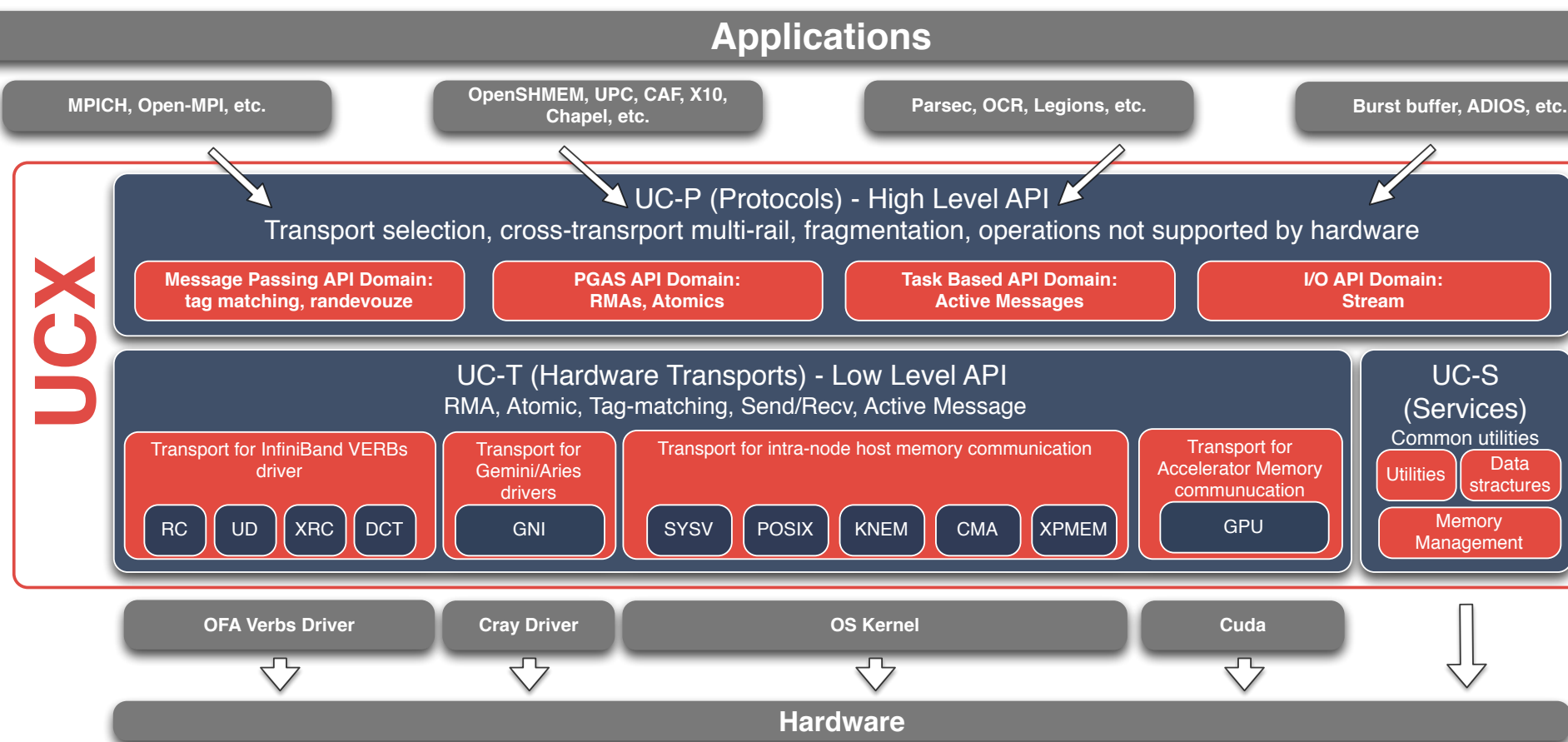
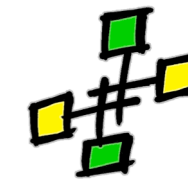
Pavel Shamis/Pasha, Principal Research  
Engineer

# RDMA Update

# VERBs API on Arm

- Besides bug fixes not much work was required
- Mellanox OFED 2.4 and above supports Arm
- Linux Kernel 4.5.0 and above (maybe even earlier)
- Linux Distribution Support – on going process
- OFED – no official ARMv8 support





[WWW.OPENUCLX.ORG](http://WWW.OPENUCLX.ORG)

<https://github.com/openucx/ucx>



# UCX 1.3: <https://github.com/openucx/ucx/releases/tag/v1.3.0>

- **Multi-rail** support for eager and rendezvous protocols
- Added **stream-based communication** API
- Added support for **GPU** platforms: Nvidia CUDA and AMD ROCM software stacks
- Added API for **Client-Server** based connection establishment
- Added support for **TCP** transport (Send/Receive semantics)
- Support for InfiniBand **hardware tag-matching** for DC and accelerated transports
- Added support for **tag-matching communications with CUDA** buffers
- Initial support for **Java bindings**
- **Progress engine** optimizations
- Improved scalability of **software tag-matching** by using a hash table
- Added transparent **huge-pages** allocator
- Added **non-blocking flush and disconnect** semantics
- Added registration cache for **KNEM**



# UCX Roadmap

## v1.4 – end of July August

- Bitwise atomics support
- Improvements for message injection (medium message size)
- Client/server connection establishment to support large address
- Support multiple connections between same pair of endpoints
- CUDA-IPC support

## v1.5 – November 2018

- Bugfixes and optimizations
- Active Message API
- New Client-Server API
- Full functionality over TCP
- Full functionality over legacy RDMA devices
- Full functionality over uGNI API

## V2.0 – 2019

- Updated API – not backward compatible with 1.x
- Binary distribution will provide v1.x version of the library (in addition for 2.x) for backward compatibility
  - All codes should work as it is

# MPI

# Scaling



• **HPE Apollo 70:** This dense, scalable platform – and HPE's first ARM-based HPC system – brings more choice and flexibility to HPC customers. It provides easy access to HPC technology with support for standard HPE provisioning, cluster management and performance software. The Apollo 70, using Cavium's 64-bit ARMv8-A ThunderX2™ Server Processor, is purpose-built for memory intensive HPC workloads and delivers up to 33 percent more memory bandwidth than today's industry standard servers. The Apollo 70 also provides access to HPE's partnership ecosystem delivering key HPC components including Red Hat® Enterprise Linux®, SUSE® Linux Enterprise Server for ARM, and Mellanox® high-speed InfiniBand & Ethernet fabric solutions.

University of BRISTOL GW4 January 17th 2017

Announcing the **GW4 Tier 2 HPC service, 'Isambard'**: named after Isambard Kingdom Brunel

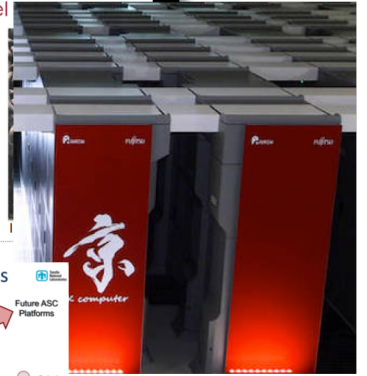
**System specs:**

- Cray CS-400 system
- **10,000+ ARMv8 cores**
- HPC optimised software stack
- Technology comparison:
  - x86, KNL, Pascal
- To be installed March-Dec 2017
- £4.7m total project cost over 3 years

Simon McIntosh Smith, simonm@cs.bris.ac.uk, @simonm

Sandia's NNSA/ASC ARM Platforms

Timeline diagram showing the evolution of Sandia's NNSA/ASC ARM Platforms from Sept 2011 to TODAY. The timeline includes milestones such as Applied Micro S-Cores 47 nodes, Cray ThunderX2 47 nodes, and Future ASC Platforms.



OpenMP™

MPI  
OpenMP™



MPI  
OpenMP™



MPI



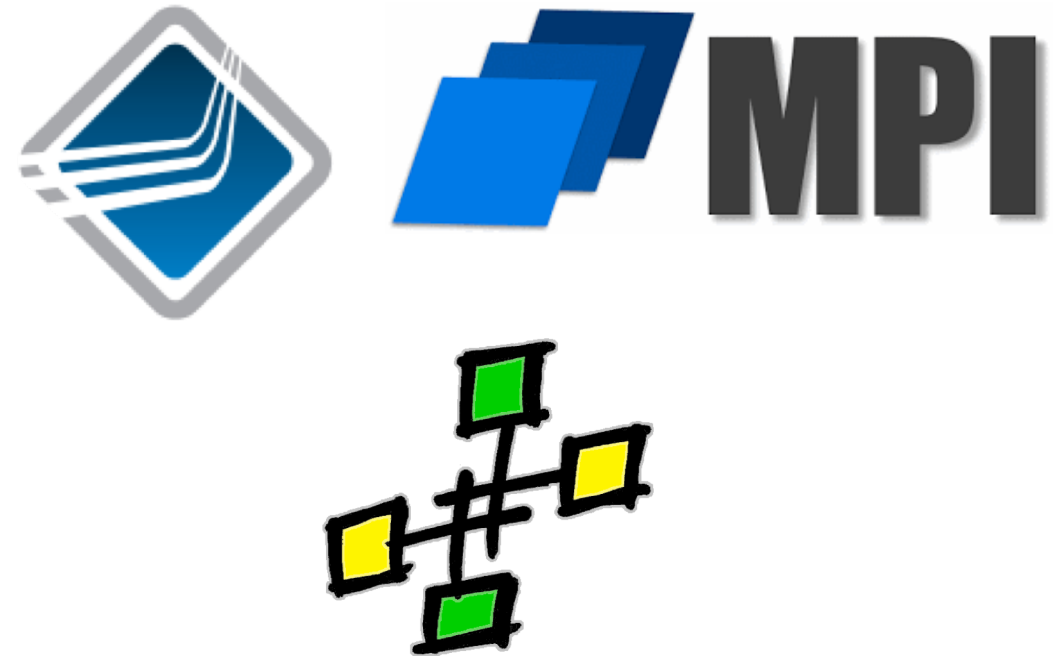
# Programing models

**MVAPICH 2.3 – works on ARMv8**

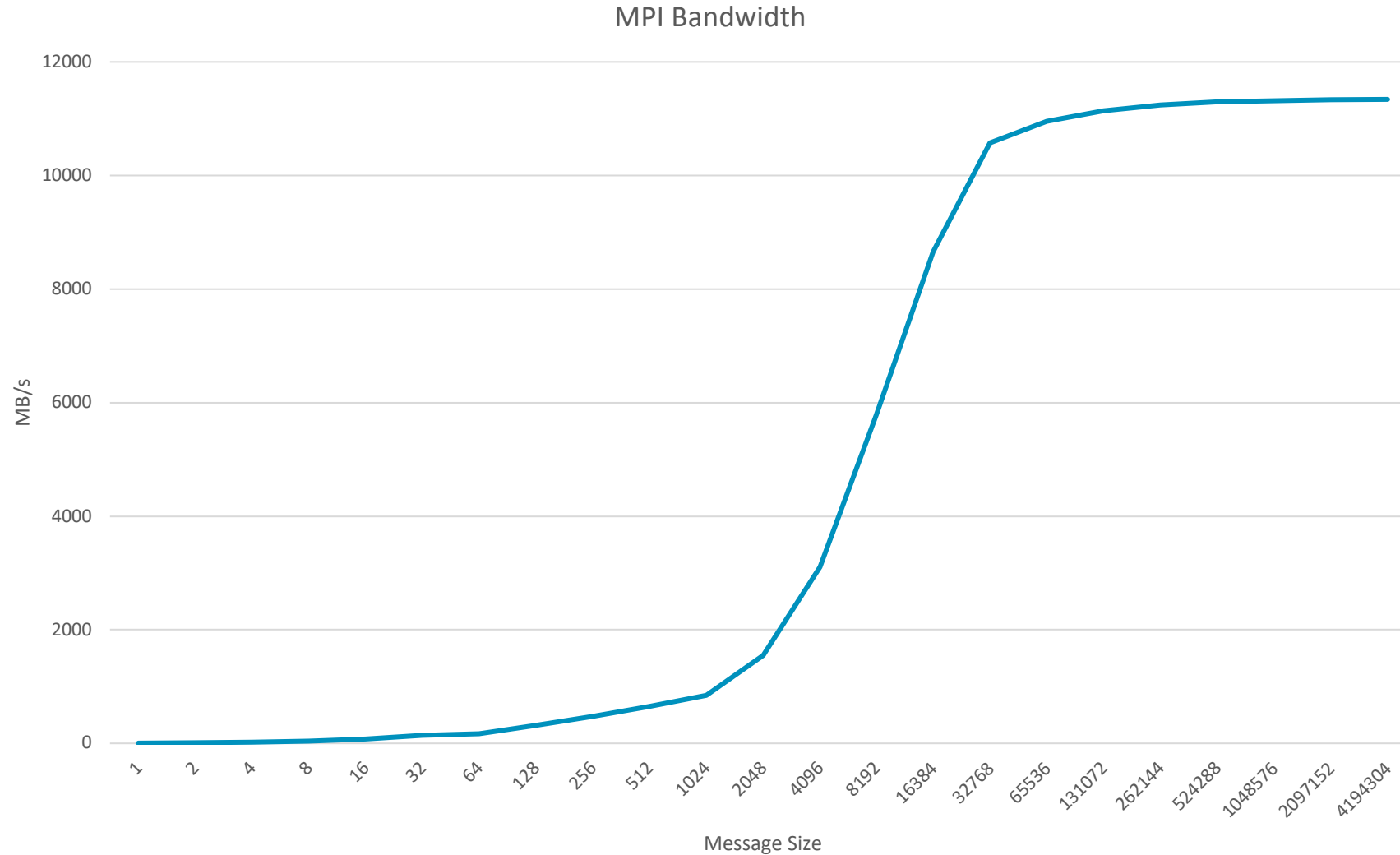
MPICH 3.3b – works on ARMv8

Open MPI 3.x – works on ARMv8

OSHMEMP – work on ARMv8

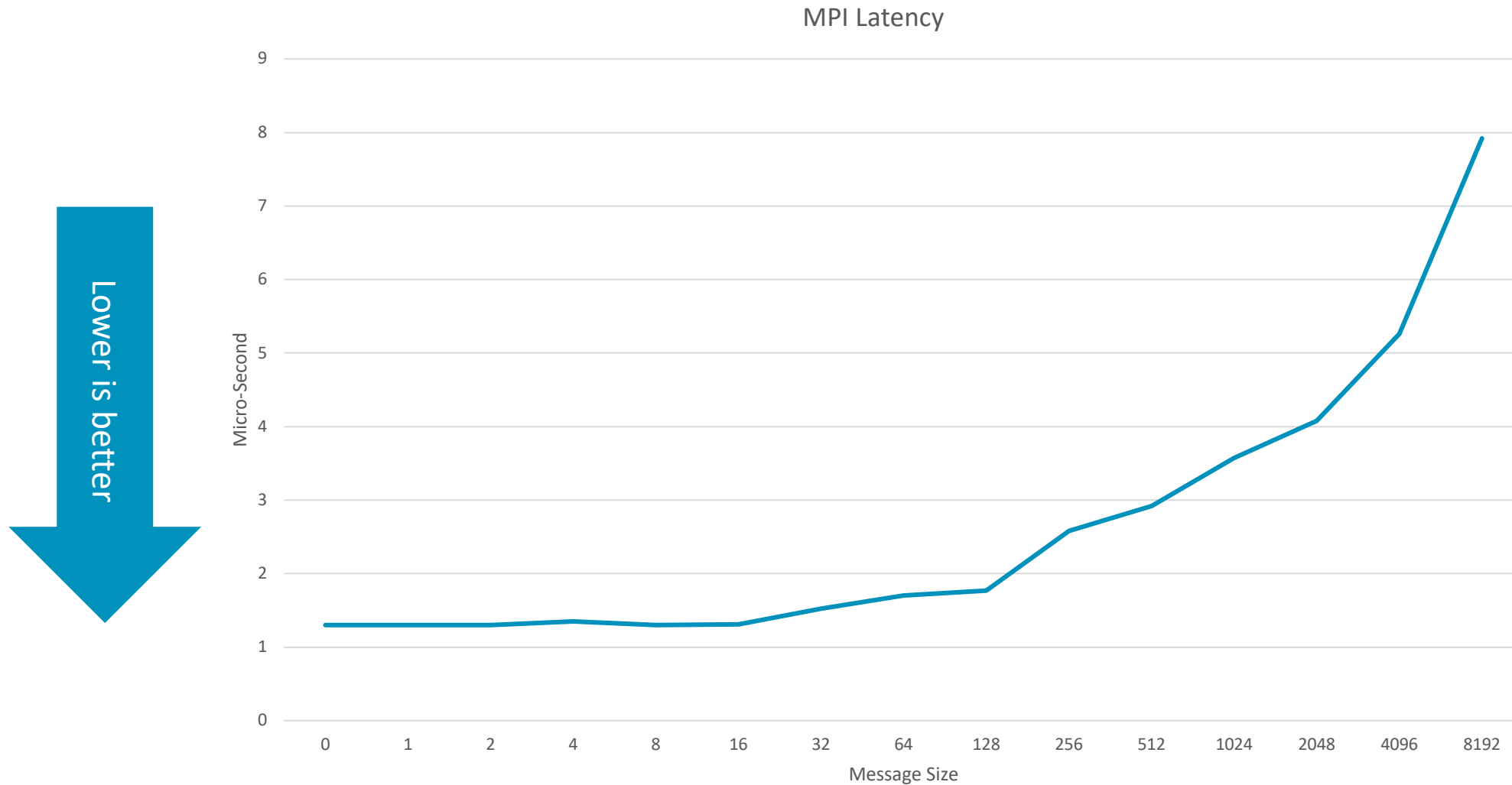


# HPE Comanche (Apollo 70) with Cavium Thunder X2 **SINGLE** core, Mellanox ConnexX-4 100Gb/s (EDR) - Bandwidth

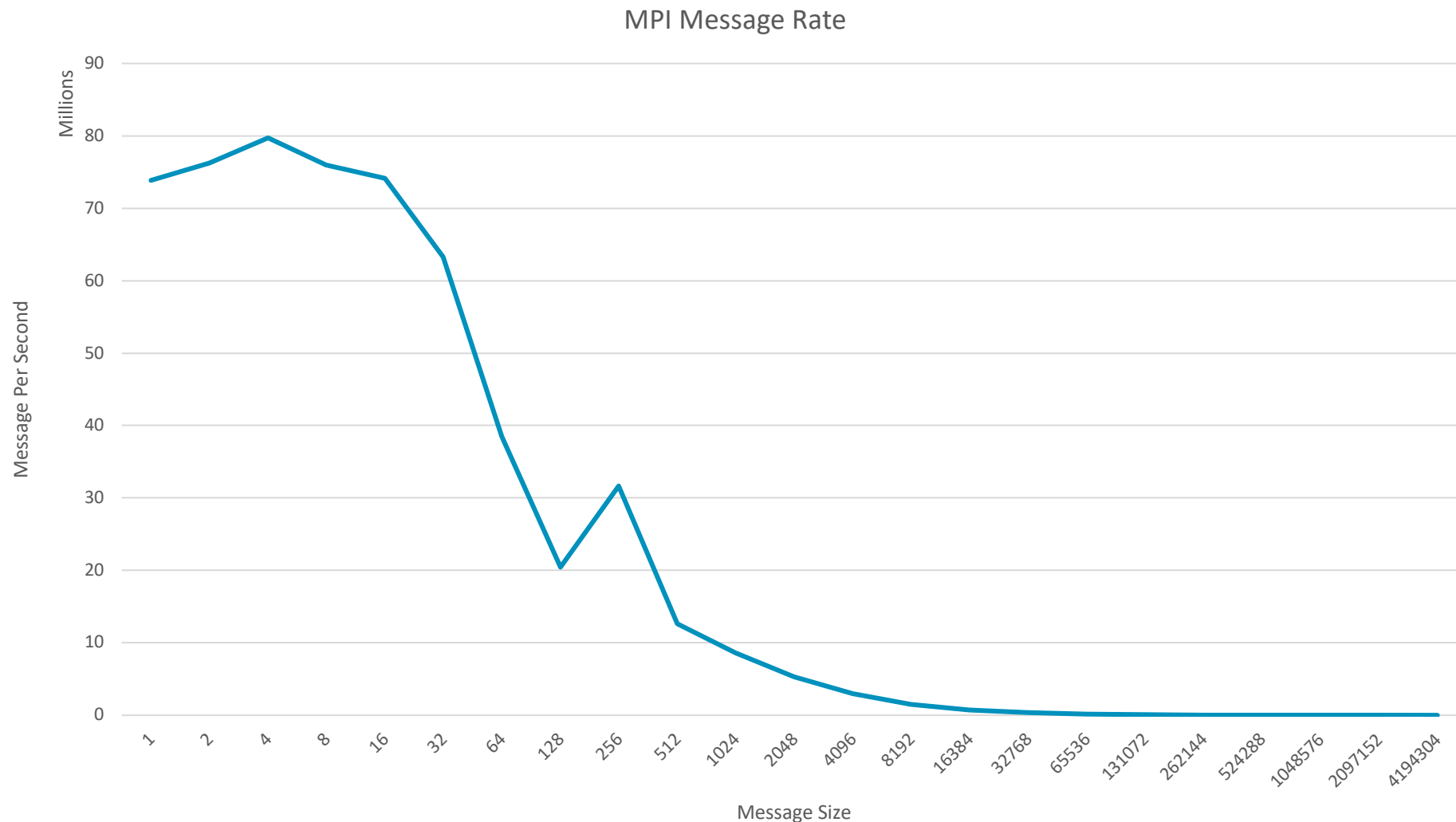




# HPE Comanche (Apollo 70) with Cavium Thunder X2, Mellanox ConnectX-4 100Gb/s (EDR) – Latency/Ping Pong



# HPE Comanche (Apollo 70) with Cavium Thunder X2, Mellanox ConnectX-4 100Gb/s (EDR) – MPI Message Rate (28 cores)



Higher is better

# MVAPICH Update

# Building with Arm compiler

## Example:

```
my_cc=armclang
```

```
my_cxx=armclang++
```

```
my_fc=armflang
```

```
../configure CC=${my_cc} CXX=${my_cxx} F77=${my_fc} FC=${my_fc} --prefix=$INSTALL_DIR --with-device=ch3:mrail --with-rdma=gen2 --enable-cxx --enable-fc
```

## Post configure fix for libtool ( it does not get the right flags for armflang):

```
sed -i -e 's#wl=""#wl="-Wl,"#g' libtool
```

```
sed -i -e 's#pic_flag=""#pic_flag=" -fPIC -DPIC"#g' libtool
```

[Arm Allinea Studio](https://developer.arm.com) 18.4 release is now available on [developer.arm.com](https://developer.arm.com)

# Weakly Ordered Memory Model

Weakly ordered memory access means that changes to memory can be applied in any order as long as *single-core* execution sees the data needed for program correctness

## Benefits:

- The processor can make many optimizations to reduce memory access
  - This has power (pushing bits is expensive) and memory bandwidth benefits
- The optimizations are transparent to single-core execution

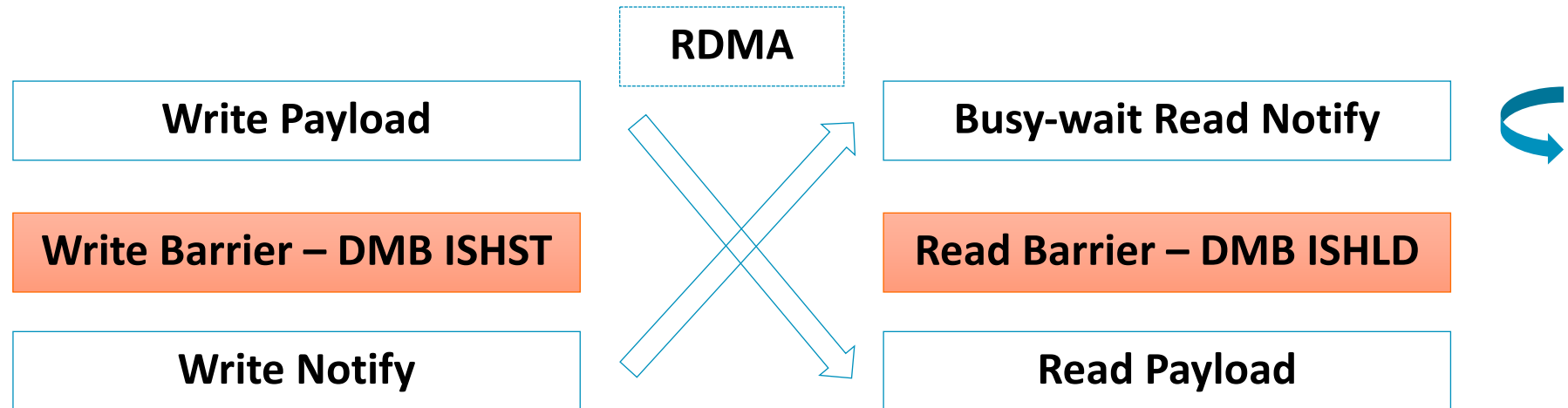
## Challenges:

- Synchronization of data between cores must be explicit
- Popular legacy architectures (EG: x86\_64, x86) provide “almost” strongly ordered memory access
  - This means that existing multi-core codes may be dependent on strongly ordered accesses

# Memory Barriers on Arm

## Memory Barriers

- Multithread environment
- Software-hardware interaction
- Examples <https://github.com/openucx/ucx/blob/master/src/ucs/arch/aarch64/cpu.h#L25>
- You can “fish” for these bugs in MPI implementations around Eager-RDMA and shared memory protocols



*Maranget, Luc, Susmit Sarkar, and Peter Sewell. "A tutorial introduction to the Arm and POWER relaxed memory models." Draft available from <http://www.cl.cam.ac.uk/~pes20/ppc-supplemental/test7.pdf> (2012).*



# Memory barrier example:

```
1754 {
1755     i
1756     i
1757     i
1758     M
1759     M
1760     M
1761     M
1762     #if d
1763     i
1764     M
1765     M
1766     #endi
1767     s
1768     w
1769     w

3772         if (shmem->local_rank == root)
3773         #if defined(_ENABLE_CUDA_)
3774             if (rdma_enable_cuda) {
3775                 MPIR_Localcopy(buf,
3776                               shmem->queue[root].shm_slots[windex]->buf,
3777                               &count, &datatype);
3778             } else
3779             {
3780                 MPIU_Memcpy(shmem->queue[root].shm_slots[windex]->buf,
3781                             shmem->queue[root].shm_slots[windex]->buf,
3782                             len);
3783             }
3784         while (shmem->queue[root].shm_slots[windex]->psn != shmem->queue[root].shm_slots[windex]->tail_psn)
3785             nspin++;
3786         if (nspin % mv2_shm_progress == 0)
3787             mv2_shm_progress(&nspin);
3788     }
3789 }
3790 #if defined(_ENABLE_CUDA_)
3791 if (rdma_enable_cuda) {
3792     MPIR_Localcopy(shmem->queue[root].shm_slots[windex]->buf,
3793                   shmem->queue[root].shm_slots[windex]->buf,
3794                   len);
3795 } else
3796 {
3797     MPIU_Memcpy(buf, shmem->queue[root].shm_slots[windex]->buf,
3798               len);
3799 }
```

DMB ISHST

DMB ISHLD

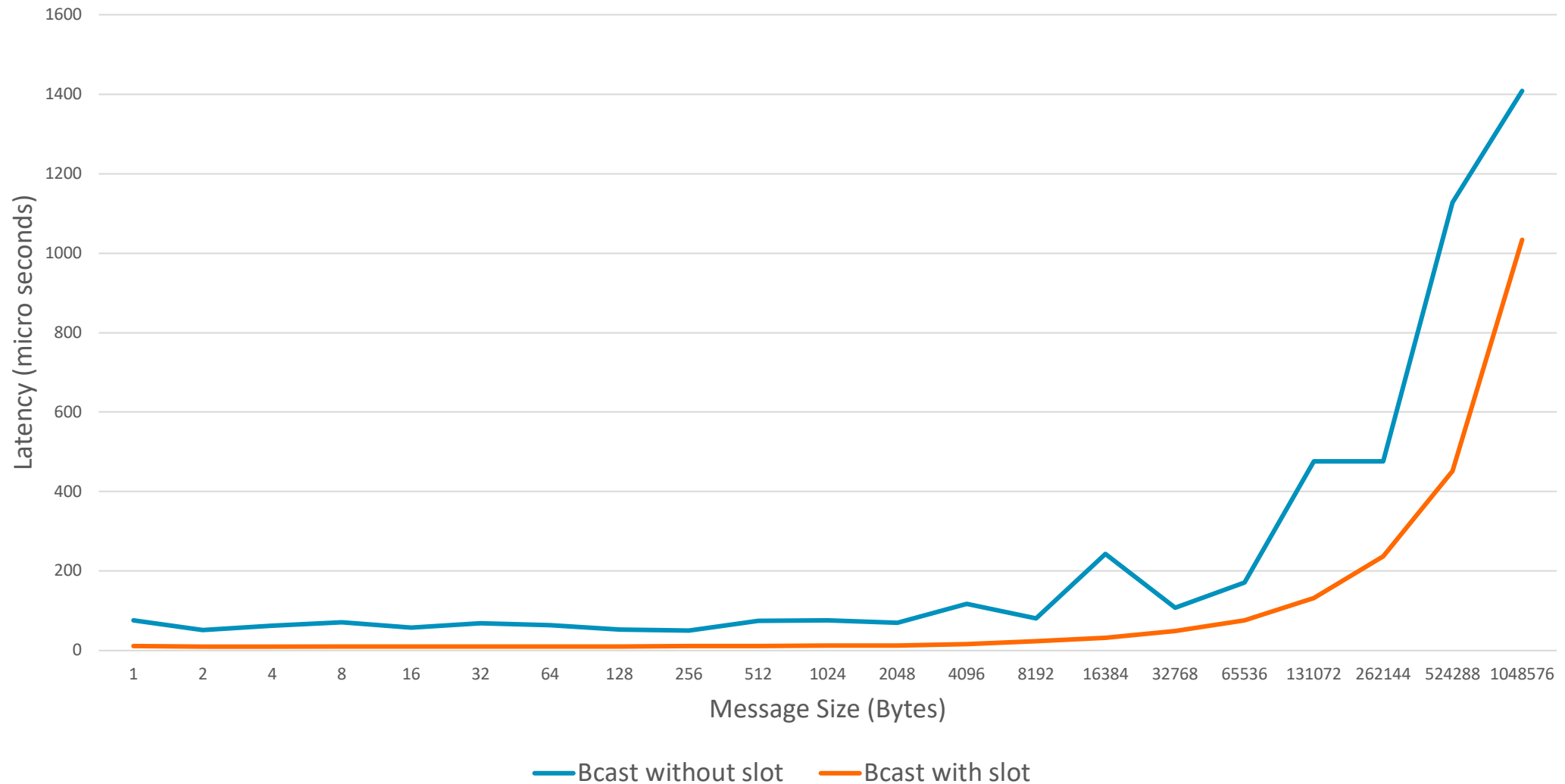
```
if (!is_cxx_uop) {
    (*MPIR_Process.cxx_call_op_fn) (
        void mv2_shm_tree_reduce(shmem_info_t * shmem, char *in_buf, int len,
        (*uop) (shmem->queue[i].shm_slots[rindex]->buf, buf, &count, &datatype);
    )
} else {
    WRITEBAR();
    shmem->queue[shmem->local_rank].shm_slots[windex]->psn = shmem->write;
}

@@ -3737,6 +3745,7 @@ void mv2_shm_tree_reduce(shmem_info_t * shmem, char *in_buf, int len,
MPIU_Memcpy(shmem->queue[shmem->local_rank].shm_slots[windex]->buf, in_buf, len);
}

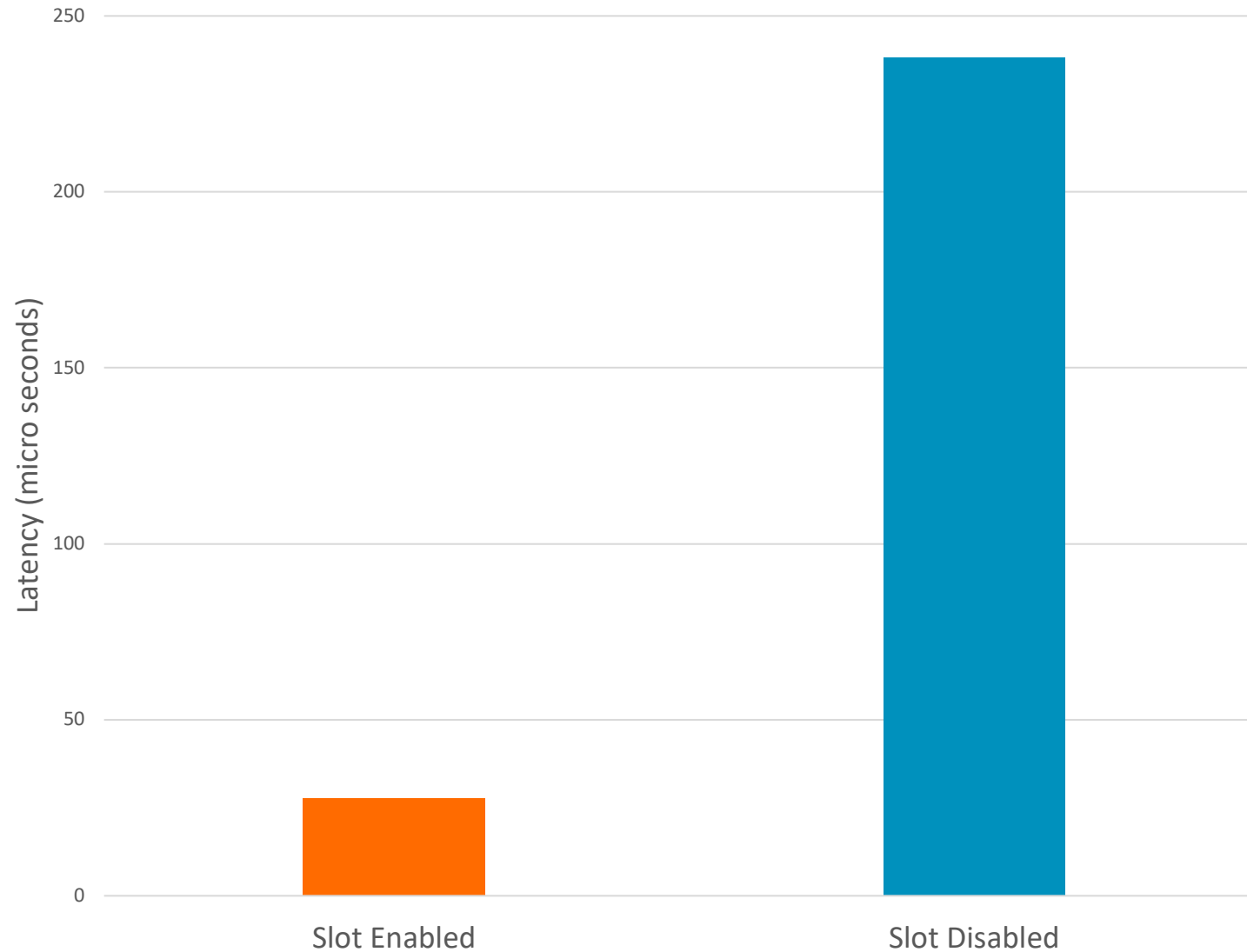
WRITEBAR();
shmem->queue[shmem->local_rank].shm_slots[windex]->psn = shmem->write;
}

@@ -3770,6 +3779,7 @@ int mv2_shm_bcast(shmem_info_t * shmem, char *buf, int len, int root)
{
    MPIU_Memcpy(shmem->queue[root].shm_slots[windex]->buf, buf, len);
}
WRITEBAR();
shmem->queue[root].shm_slots[windex]->psn = shmem->write;
} else {
    while (shmem->queue[root].shm_slots[rindex]->psn != shmem->read) {
        int mv2_shm_bcast(shmem_info_t * shmem, char *buf, int len, int root)
        mv2_shm_progress(&nspin);
    }
}
READBAR();
#if defined(_ENABLE_CUDA_)
if (rdma_enable_cuda) {
    MPIR_Localcopy(shmem->queue[root].shm_slots[rindex]->buf, len, MPI_BYTE,
    (volatile uint32_t *) (shmem->queue[intra_node_root].shm_slots[windex]->tail_psn)) {
        mv2_shm_progress(&nspin);
    }
    WRITEBAR();
    shmem->queue[intra_node_root].shm_slots[windex]->psn = shmem->write;
} else {
    /* node-level leader, and the root of the bcast */
    int mv2_shm_zcpy_bcast(shmem_info_t * shmem, char *buf, int len, int root,
    MPIU_Memcpy(shmem->queue[intra_node_root].shm_slots[windex]->buf, buf, len);
}
WRITEBAR();
shmem->queue[intra_node_root].shm_slots[windex]->psn = shmem->write;
WRITEBAR();
shmem->queue[intra_node_root].shm_slots[windex]->tail_psn = (volatile uint32_t *)
```

# OSU MVAPICH BCAST (448 processes)



# OSU MVAPICH Barrier (448 processes)



# HPCAC - <http://hpcadvisorycouncil.com/>



## Juno



- 1x GIGABYTE R270-T64 Chassis
    - 2 x Cavium ThunderX 48-core ARM processors
    - Memory: 64GB DDR4 2400 MHz
    - Mellanox ConnectX-4 EDR 100Gb/s InfiniBand/VPI adapter
    - SSD 480GB SATA 3
  - 2x GIGABYTE MT30-GS0 Chassis
    - 1x Cavium ThunderX 32-core ARM Processor
    - Memory: 128GB DDR4 2400 MHz
    - Mellanox ConnectX-5 EDR 100Gb/s InfiniBand/VPI adapter
    - SSD 1TB SATA 3
- Switch : Mellanox Switch-IB 2 SB7800 36-Port 100Gb/s EDR InfiniBand switches

[Apply for System Access](#)



The word "arm" in a white, lowercase, sans-serif font, positioned on the right side of a solid blue background.

The Arm trademarks featured in this presentation are registered trademarks or trademarks of Arm Limited (or its subsidiaries) in the US and/or elsewhere. All rights reserved. All other marks featured may be trademarks of their respective owners.

[www.arm.com/company/policies/trademarks](http://www.arm.com/company/policies/trademarks)

Thank You!

Danke!

Merci!

谢谢!

ありがとう!

Gracias!

Kiitos!

감사합니다

धन्यवाद

arm



# Backup