Experiences Using MVAPICH in a Production HPC Environment at TACC

Karl W. Schulz

Director, Scientific Applications

Texas Advanced Computing Center (TACC)

MVAPICH User's Group ◆ August 2013, Columbus, OH

Acknowledgements

- *Sponsor: National Science Foundation*
  - NSF Grant #OCI-1134872 Stampede Award, “Enabling, Enhancing, and Extending Petascale Computing for Science and Engineering”
  - NSF Grant #OCI-0926574 - “Topology-Aware MPI Collectives and Scheduling”

- *Professor D.K. Panda and his team at OSU*

Outline

- Brief clustering history at TACC
  - InfiniBand evaluation
  - MVAPICH usage
  - Optimizations
- Stampede
  - System overview
  - MPI for heterogeneous computing
  - Other new goodies

Brief Clustering History at TACC

- Like many sites, TACC was deploying small clusters in early 2000 timeframe
- First “large” cluster was Lonestar2 in 2003
    - 300 compute nodes originally
    - Myrinet interconnect
    - debuted at #26 on Top500
- In 2005, we built another small research cluster: ***Wrangler*** (128 compute hosts)
    - 24 hosts had both Myrinet and early IB
    - single 24-port Topspin switch
    - used to evaluate price/performance of commodity Linux Cluster hardware

Early InfiniBand Evaluation

- Try to think back to the 2004/2005 timeframe……
  - only 296 systems on the Top500 list were clusters
  - multiple IB vendors and stacks
  - "multi-core" meant dual-socket
  - we evaluated a variety of stacks across the two interconnects
  - *our first exposure to MVAPICH* (0.9.2 via Topspin and 0.9.5 via Mellanox)

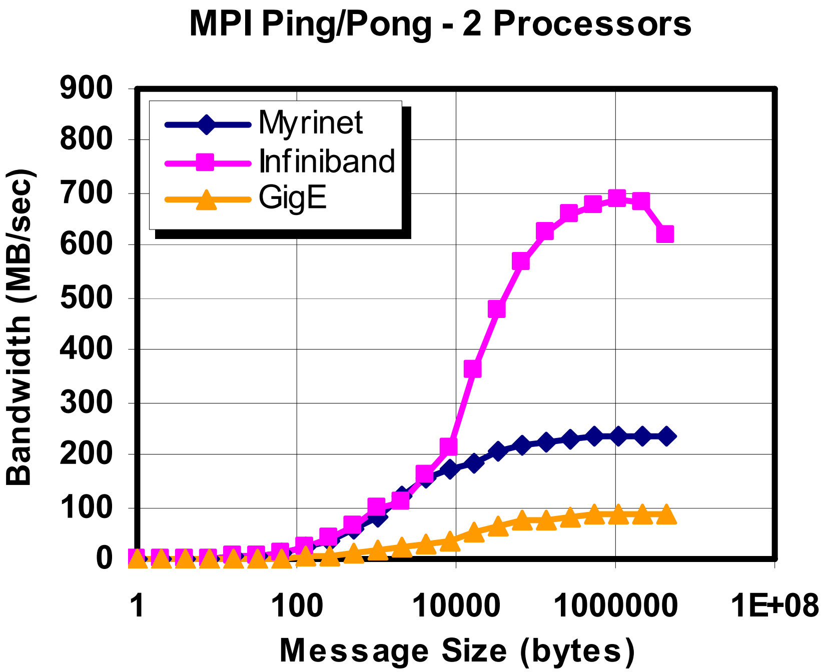
Example MPI Latency Measurements, circa 2005

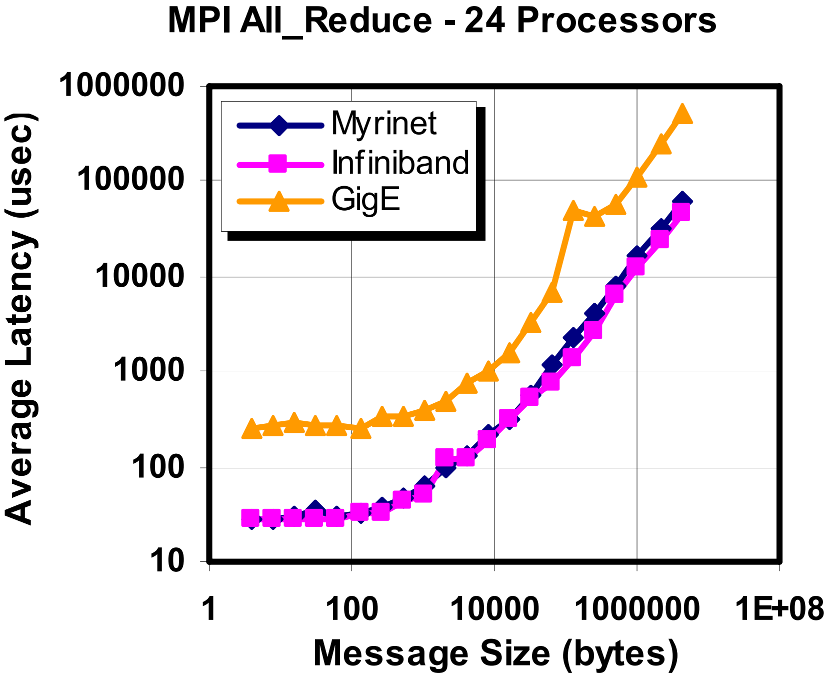
Myrinet	MPICH-GM	7.05 µs
	MPICH-MX	3.25 us
	LAM-GM	7.77 µs
	VMI-GM	8.42 µs
Infiniband	IB-Gold	4.68 µs
	IB-Topspin	5.24 µs
	LAM-IB	14.96 µs
GigE	MPICH	35.06 µs
	LAM-TCP	32.63 µs
	VMI-TCP	34.29 µs

Early InfiniBand Evaluation

- In addition to latency considerations, we were also attracted to BW performance and influence on applications

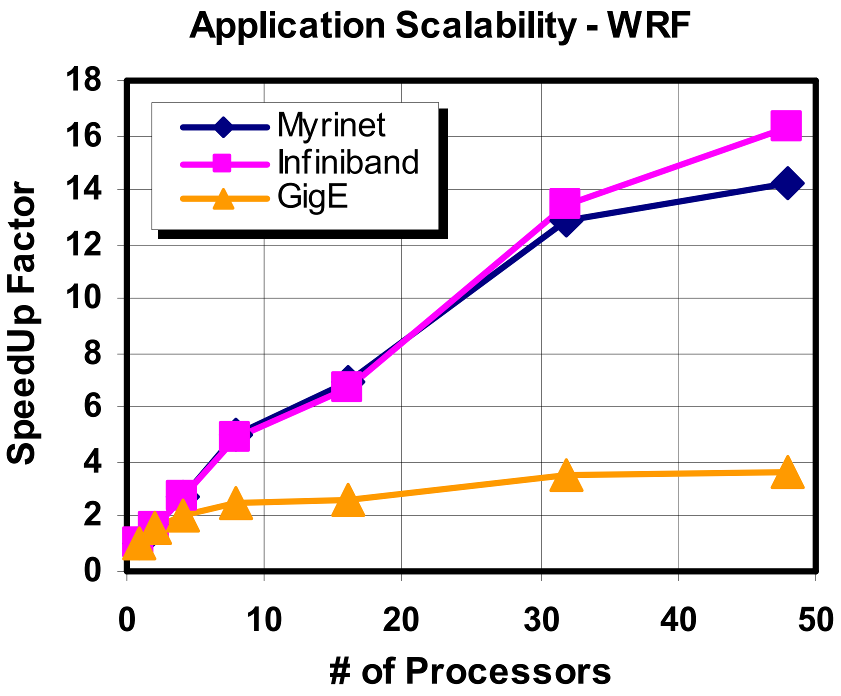
TACC Internal Benchmarking, circa 2005

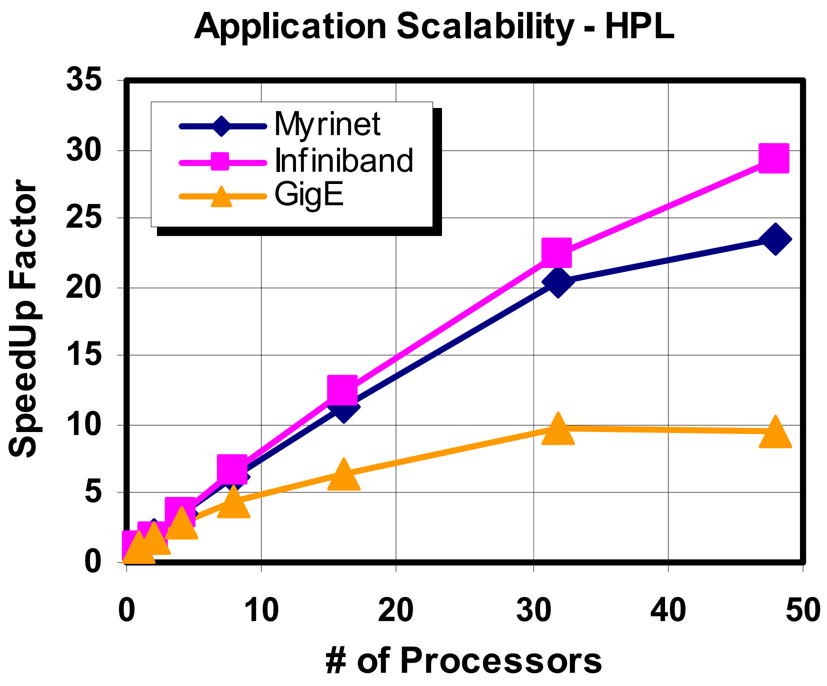




Early InfiniBand Evaluation

TACC Internal Benchmarking, circa 2005

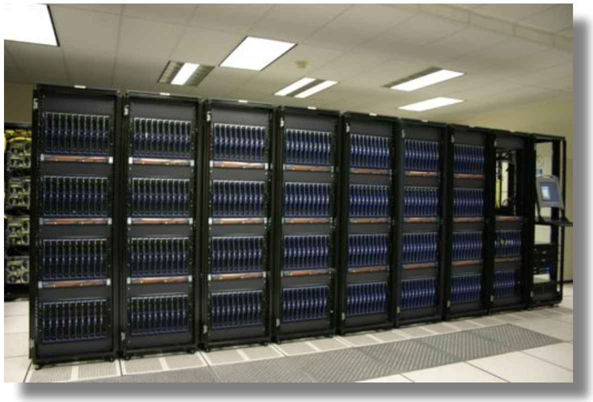




Brief Clustering History at TACC

- Based on these evaluations and others within the community, our next big cluster was IB based

- Lonestar3 entered production in 2006:
  - OFED 1.0 was released in June 2006 (and we ran it!)
  - First production Lustre file system (also using IB)
  - MVAPICH was the primary MPI stack
  - workhorse system for local and national researchers, expanded in 2007



☛Debuted at #12 on Top500

Brief Clustering History at TACC

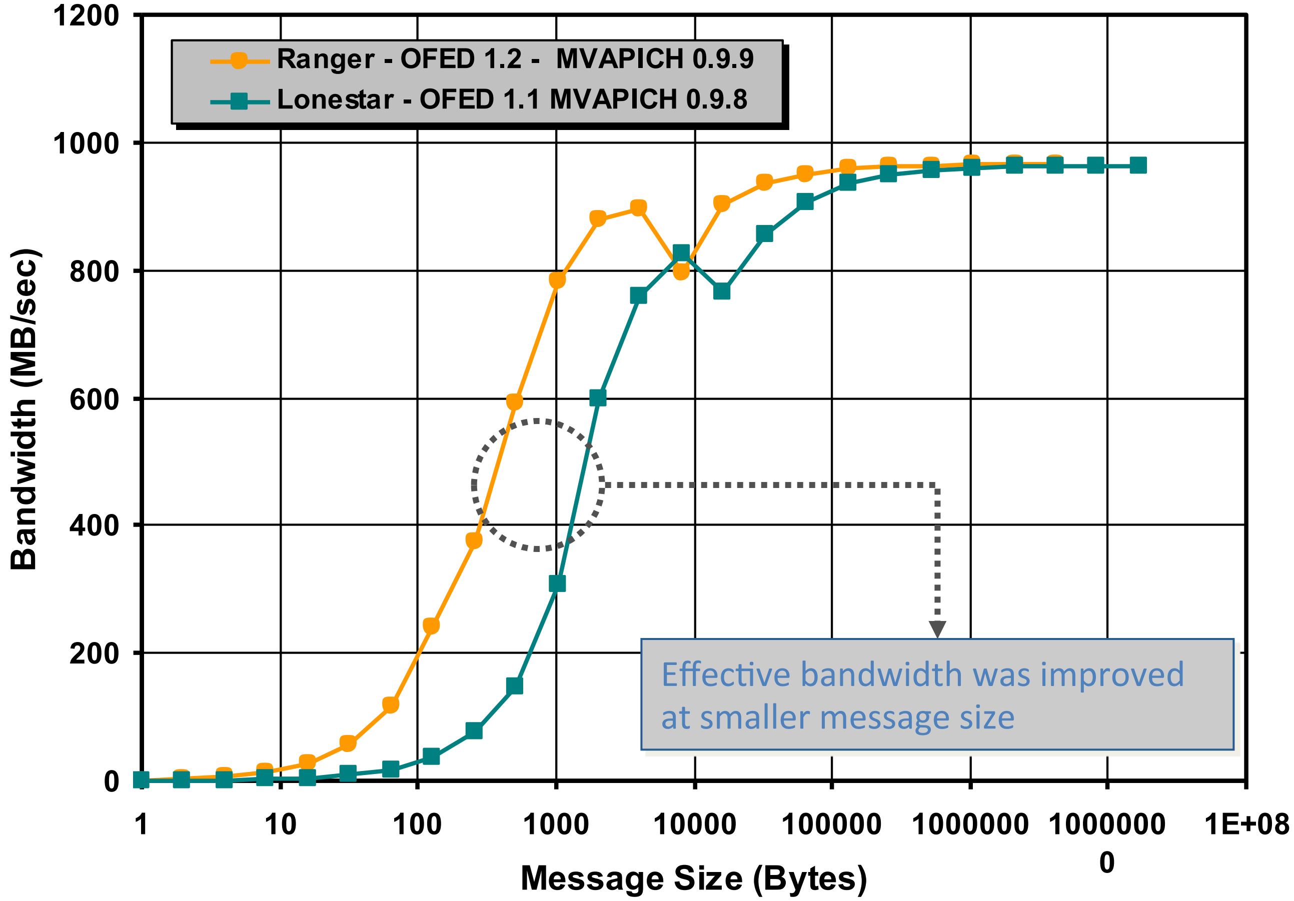
- These clustering successes ultimately led to our next big deployment in 2008, the first NSF "Track 2" system, ***Ranger:***
  - $30M system acquisition
  - 3,936 Sun four-socket blades
  - 15,744 AMD "Barcelona" processors
  - All IB all the time (SDR) - no ethernet
    - Full non-blocking 7-stage Clos fabric
    - ~4100 endpoint hosts
    - >1350 MT47396 switches
    - *challenges encountered at this scale led to more interactions and collaborations with OSU team*

☛ Debuted at #4 on Top500

Ranger: MVAPICH Enhancements

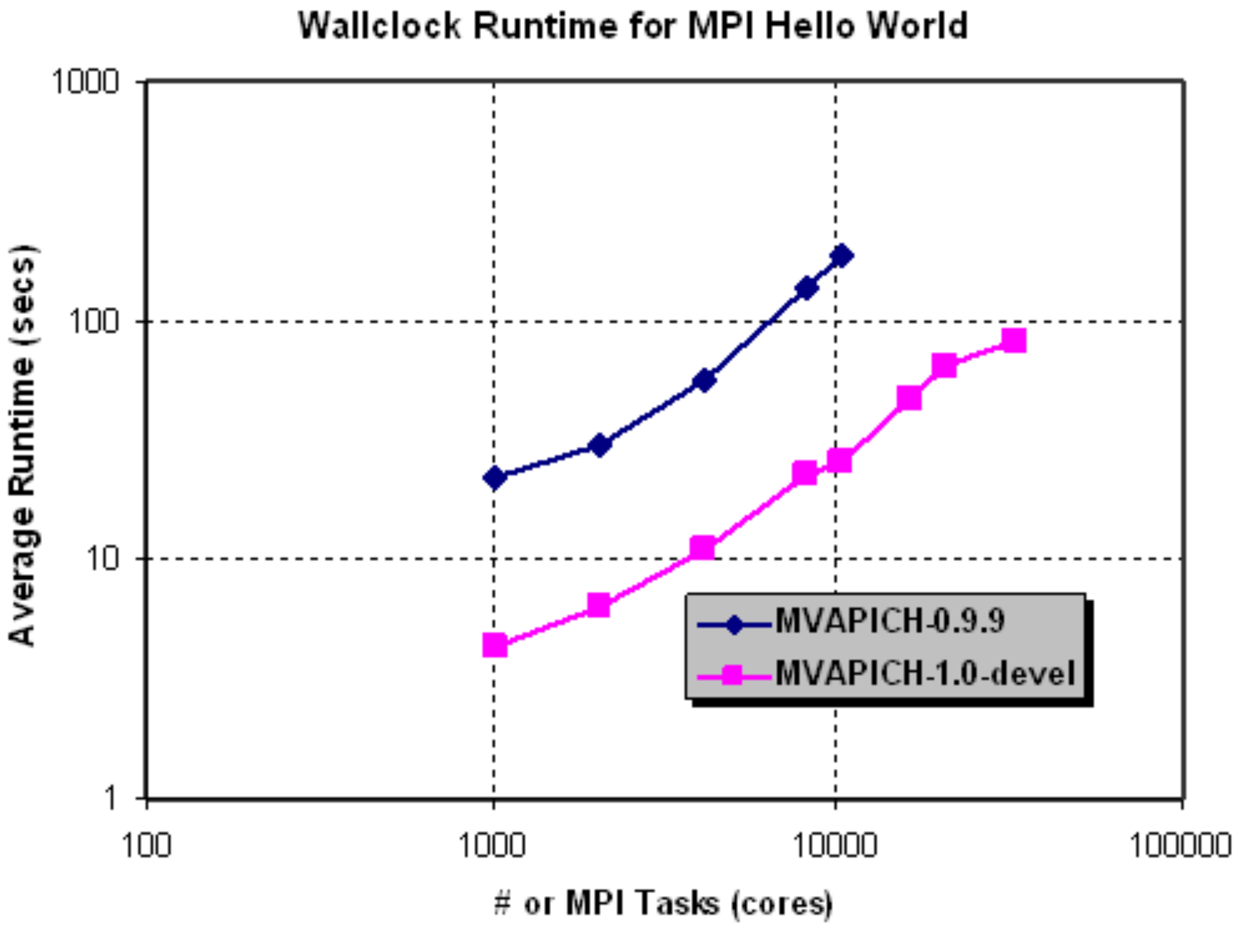
- The challenges encountered at this scale led to more direct interactions with the OSU team
- Fortunately, I originally met Professor Panda at IEEE Cluster 2007
  - original discussion focused on "`mpirun_rsh`" for which enhancements were released in MVAPICH 1.0
  - subsequent interactions focused on ConnectX collective performance, job startup scalability, SGE integration, shared-memory optimizations, etc.
  - DK and his team relentlessly worked to improve MPI performance and resolve issues at scale; helped to make Ranger a very productive resource with MVAPICH as the default stack for thousands of system users

Ranger: MVAPICH Enhancements

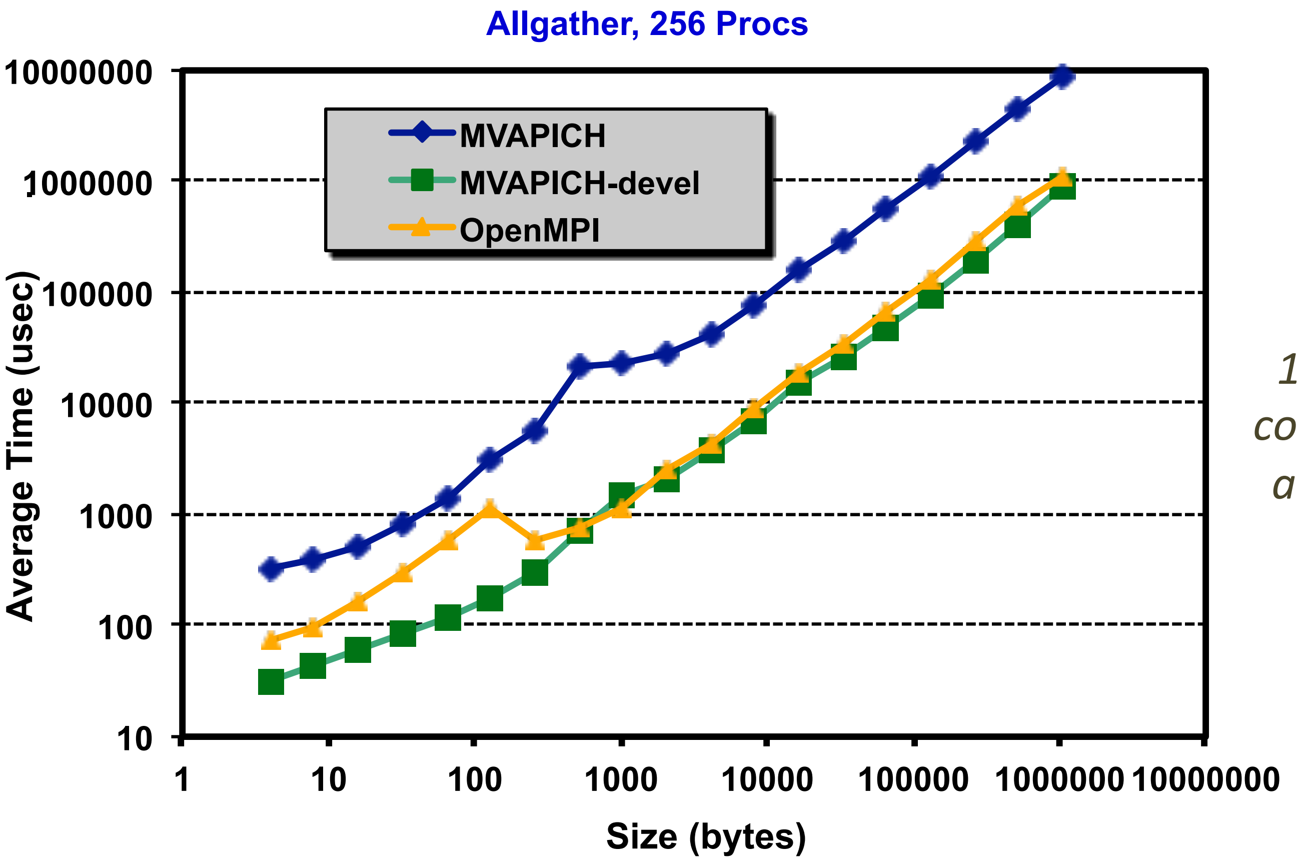


Effective bandwidth was improved at smaller message size

Ranger: MVAPICH Enhancements

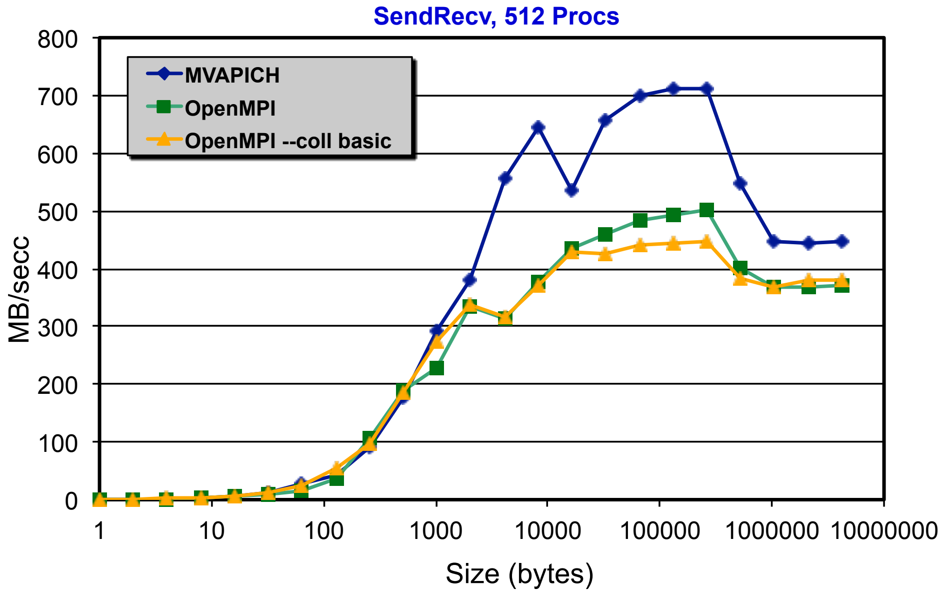


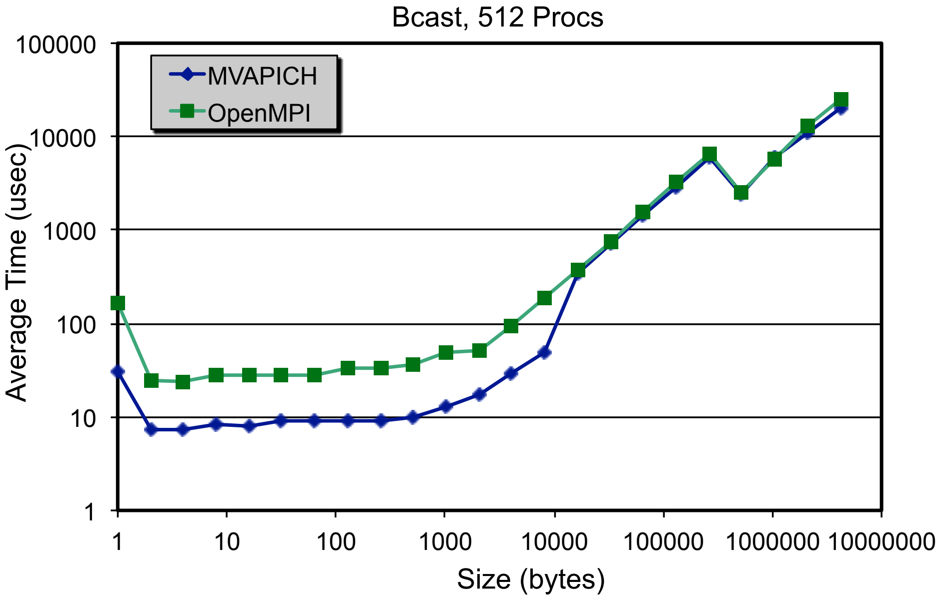
MVAPICH Improvements



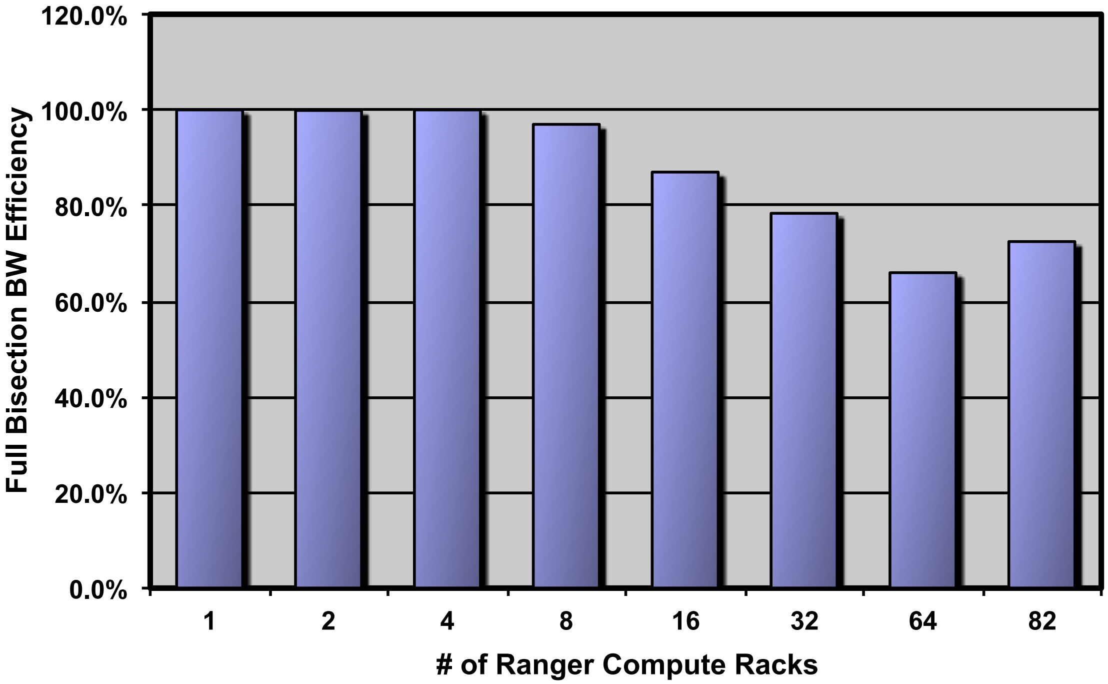
1st large 16-core IB system available for MVAPICH tuning

Ranger MPI Comparisons





Ranger: Bisection BW Across 2 Magnums



- **Using MVAPICH, we were able to sustain ~73% bisection bandwidth efficiency with all nodes communicating (82 racks)**

- **Subnet routing was key! – Using special fat-tree routing from OFED 1.3 which had cached routing to minimize the overhead of remaps**

Clustering History at TACC

- Ranger's production lifespan was extended for one extra year
  - went offline in January 2013
  - we supported both MVAPICH and MVAPICH2 on this resource
- Our next deployment was **Lonestar4** in 2011:
  - 22,656 Intel Westmere cores
  - QDR InfiniBand
  - joint NSF and UT resource
  - first TACC deployment with MVAPICH2 only (v 1.6 at the time)
  - *only real deployment issue encountered was MPI I/O support for Lustre*

☛ Debuted at #28 on Top500

Clustering History at TACC

- Our latest large-scale deployment began in 2012: *Stampede*

- A follow-on NSF Track 2 deployment targeted to replace *Ranger*

- Includes a heterogeneous compute environment

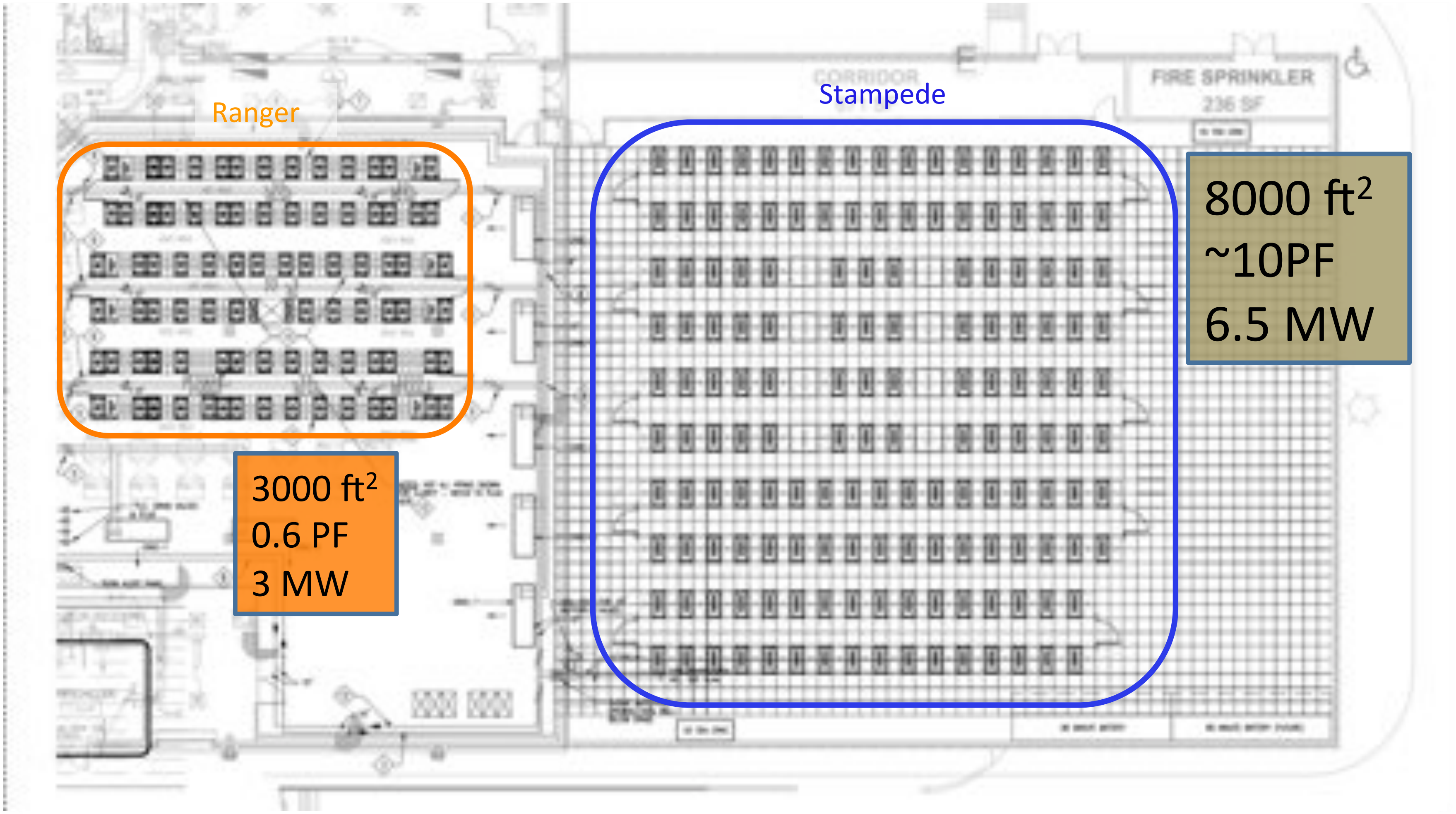
☛ Currently #6 on Top500

Stampede - High Level Overview

- Base Cluster (Dell/Intel/Mellanox):
  - Intel Sandy Bridge processors
  - Dell dual-socket nodes w/32GB RAM (2GB/core)
  - 6,400 compute nodes
  - 56 Gb/s Mellanox FDR InfiniBand interconnect
  - More than 100,000 cores, 2.2 PF peak performance
- Co-Processors:
  - Intel Xeon Phi "MIC" Many Integrated Core processors
  - Special release of "Knight's Corner" (61 cores)
  - All MICs were installed on site at TACC
  - 7.3 PF peak performance
- ***Entered production operations on January 7, 2013***

TACC

Stampede Footprint



Ranger

3000 ft^2
0.6 PF
3 MW

Stampede

8000 ft^2
~10PF
6.5 MW

Machine Room Expansion
Added 6.5MW of additional power

Innovative Component

- One of the goals of the NSF solicitation was to *"introduce a major new innovative capability component to science and engineering research communities"*

- We proposed the Intel Xeon Phi coprocessor (MIC or KNC)
  - one first generation Phi installed per host during initial deployment
  - in addition, **480** of of these 6400 hosts now have 2 MICs/host
  - project also has a confirmed injection of 1600 future generation MICs in 2015

☛ *Note: base cluster formally accepted in January, 2013. The Xeon Phi co-processor component just recently completed acceptance.*

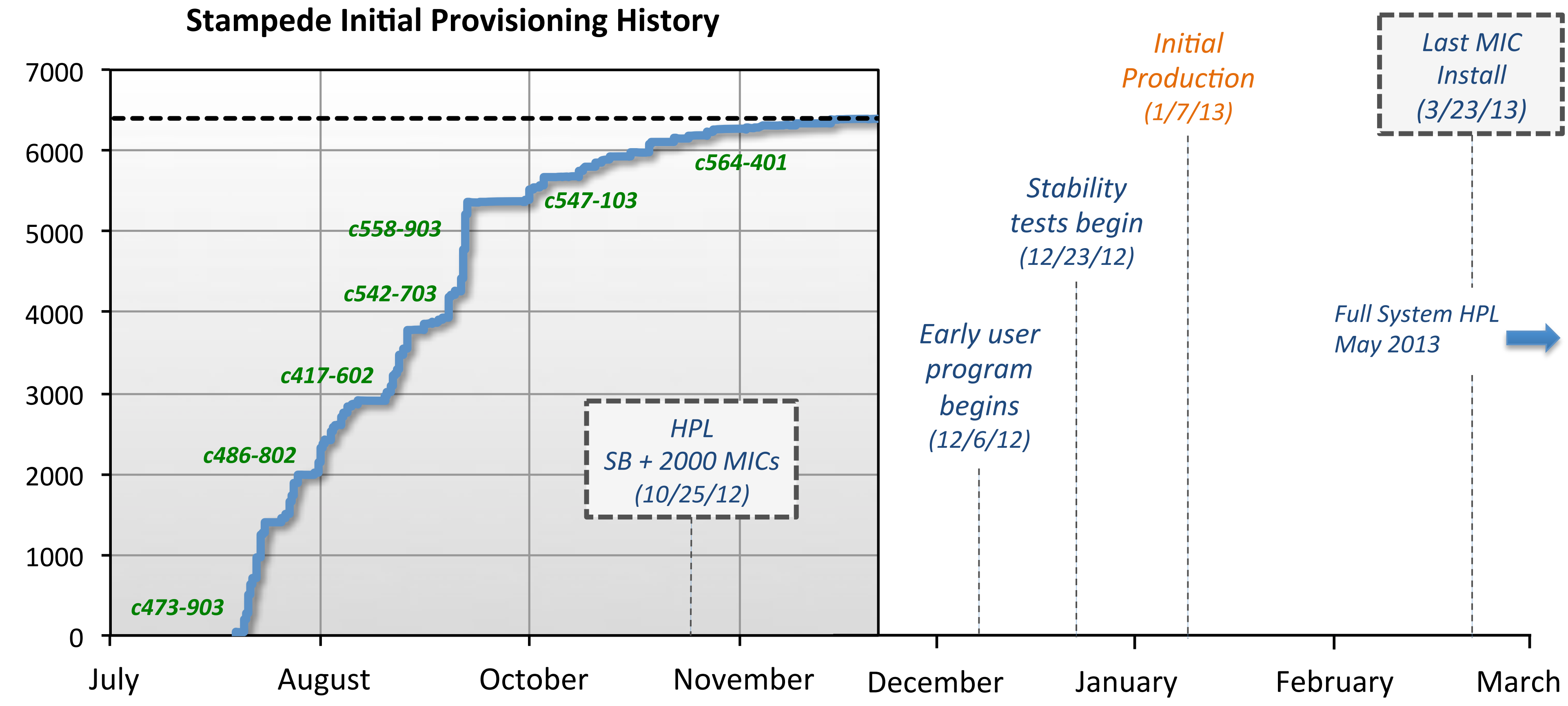
☛ *MVAPICH team involved in both facets*

MVAPICH

Additional Integrated Subsystems

- Stampede includes 16 **1TB** Sandy Bridge shared memory nodes with dual GPUs
- 128 of the compute nodes are also equipped with NVIDIA Kepler K20 GPUs for visualization analysis (and also include MICs for performance bake-offs)
- 16 login, data mover and management servers (batch, subnet manager, provisioning, etc)
- Software included for high throughput computing, remote visualization
- Storage subsystem (Lustre) driven by Dell H/W:
  - Aggregate Bandwidth greater than 150GB/s
  - More than 14PB of capacity

System Deployment History



Stampede Initial Provisioning History

HPL
SB + 2000 MICs
(10/25/12)

Early user program begins (12/6/12)

Stability tests begin (12/23/12)

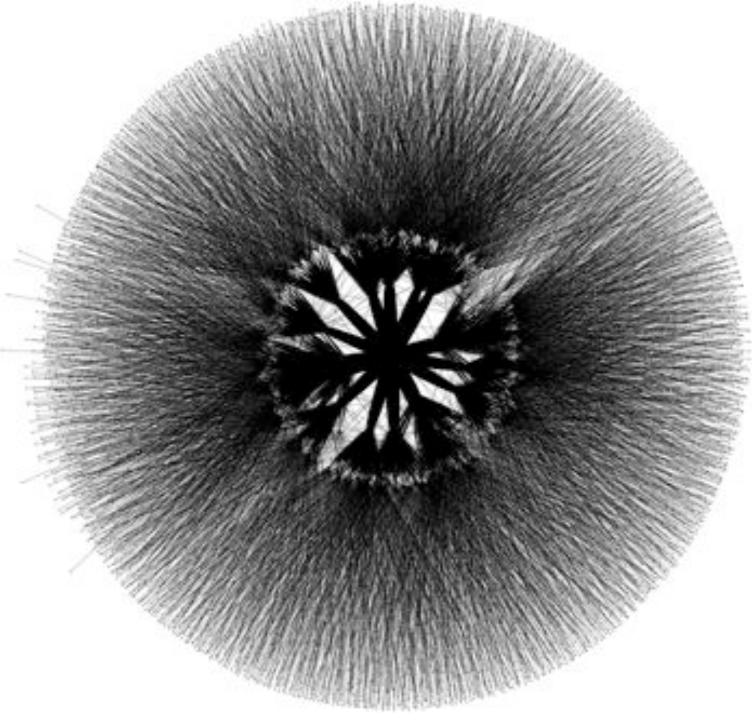
Initial Production (1/7/13)

Last MIC Install (3/23/13)

Full System HPL May 2013

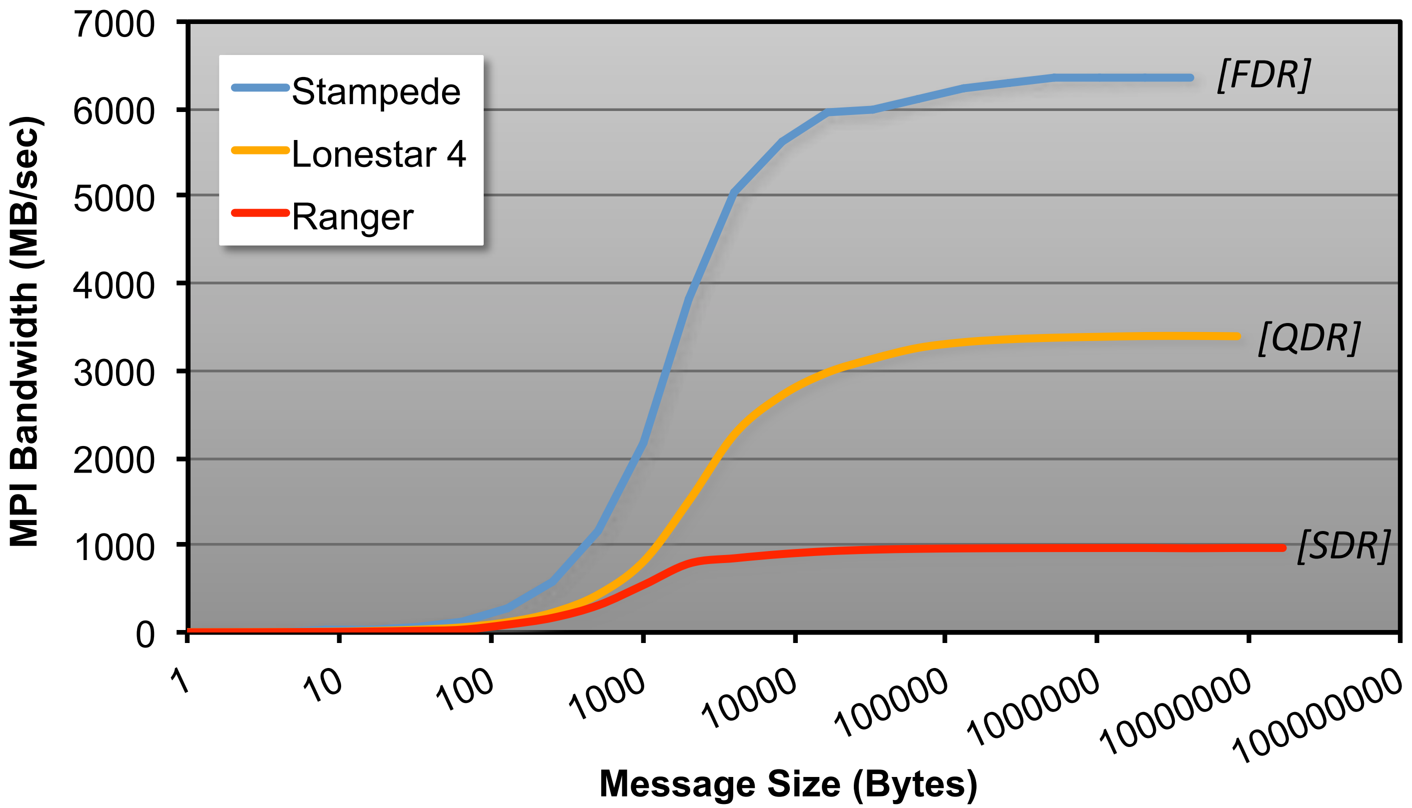
Stampede InfiniBand Topology

8 Core Switches



Stampede InfiniBand (fat-tree)
~75 Miles of InfiniBand Cables

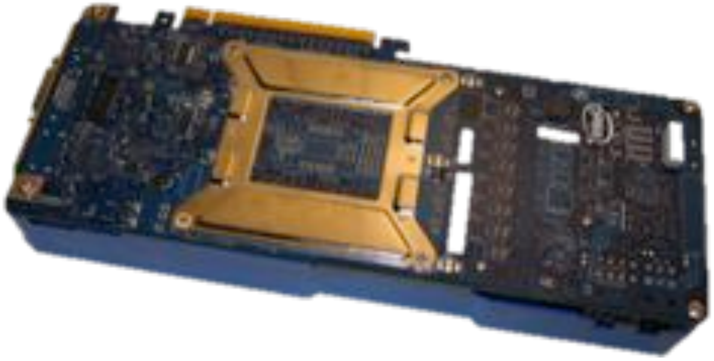
MPI Data Movement
- Historical Perspective Across Platforms -



What is this MIC thing?

Basic Design Ideas:

- Leverage x86 architecture (a CPU with many cores)
- Use x86 cores that are simpler, but allow for more compute throughput
- Leverage existing x86 programming models
- Dedicate much of the silicon to floating point ops., keep some cache(s)
- Keep cache-coherency protocol
- Increase floating-point throughput per core
- Implement as a separate device
- Strip expensive features (out-of-order execution, branch prediction, etc.)
- Widened SIMD registers for more throughput (512 bit)
- Fast (GDDR5) memory on card



Programming Models for MIC

- MIC adopts familiar X86-like instruction set (with 61 cores,244 threads in our case)
- Supports full or partial offloads (offload everything or directive-driven offload)
- Predominant parallel programming model(s) with **MPI**:
  - Fortran: OpenMP, MKL
  - C: OpenMP/Pthreads, MKL, Cilk
  - C++: OpenMP/Pthreads, MKL, Cilk, TBB
- Has familiar Linux environment
  - you can login into it
  - *you can run "top", debuggers, your native binary, etc*

Example of Native Execution

```
login1$  srun –p devel --pty /bin/bash –l
c401-102$ cat hello.c
#include<stdio.h>
int main()
{
  printf("Hook 'em Horns!\n");

#ifdef __MIC__
  printf(" --> Ditto from MIC\n");
#endif
}

c401-102$ icc hello.c
c401-102$ ./a.out
Hook 'em Horns!

c401-102$ icc –mmic hello.c
c401-102$ ./a.out
bash: ./a.out: cannot execute binary file

c401-102$ ssh mic0 ./a.out
Hook 'em Horns!
 --> Ditto from MIC
```

Interactive Hello World

- Interactive programming example
  - Request interactive job (srun)
  - Compile on the compute node
  - Using the Intel compiler toolchain
  - Here, we are building a simple hello world…
- First, compile for SNB and run on the host
  - note the __MIC__ macro can be used to isolate MIC only execution, in this case no extra output is generated on the host
- Next, build again and add “-mmic” to ask the compiler to *cross-compile* a binary for native MIC execution
  - note that when we try to run the resulting binary on the host, it throws an error
  - ssh to the MIC (mic0) and run the executable out of $HOME directory
  - this time, we see extra output from within the guarded__MIC__ macro

Example of Offload Execution

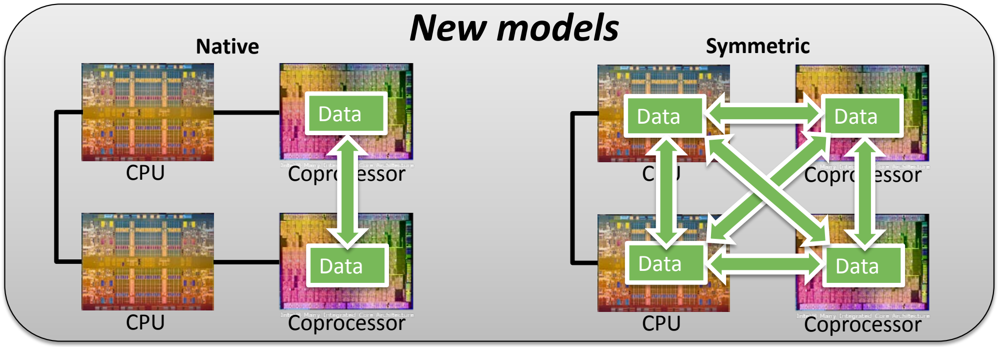
```
!dec$ offload target(mic:0) in(a, b, c) in(x) out(y)
!$omp parallel
!$omp single
      call system_clock(i1)
!$omp end single
!$omp do
      do j=1, n
        do i=1, n
          y(i,j) = a * (x(i-1,j-1) + x(i-1,j+1) + x(i+1,j-1) + x(i+1,j+1)) + &
                   b * (x(i-0,j-1) + x(i-0,j+1) + x(i-1,j-0) + x(i+1,j+0)) + &
                   c * x(i,j)
        enddo
        do k=1, 10000
          do i=1, n
            y(i,j) = a * (x(i-1,j-1) + x(i-1,j+1) + x(i+1,j-1) + x(i+1,j+1)) + &
                     b * (x(i-0,j-1) + x(i-0,j+1) + x(i-1,j-0) + x(i+1,j+0)) + &
                     c * x(i,j) + y(i,j)
          enddo
        enddo
      enddo
!$omp single
      call system_clock(i2)
!$omp end single
!$omp end parallel
```

Kernel of a finite-difference stencil code (f90)

Stampede Data Movement

- One of the attractive features of the Xeon Phi environment is the ability to utilize MPI directly between host and MIC pairs
  - leverage capability of existing code bases with MPI+OpenMP
  - requires extensions to MPI stacks in order to facilitate
  - reacquaints users with MPMD model as we need:
    - MPI binary for Sandy Bridge
    - MPI binary for MIC
  - provides many degrees of tuning freedom for load balancing

- With new software developments, we can support symmetric MPI mode runs



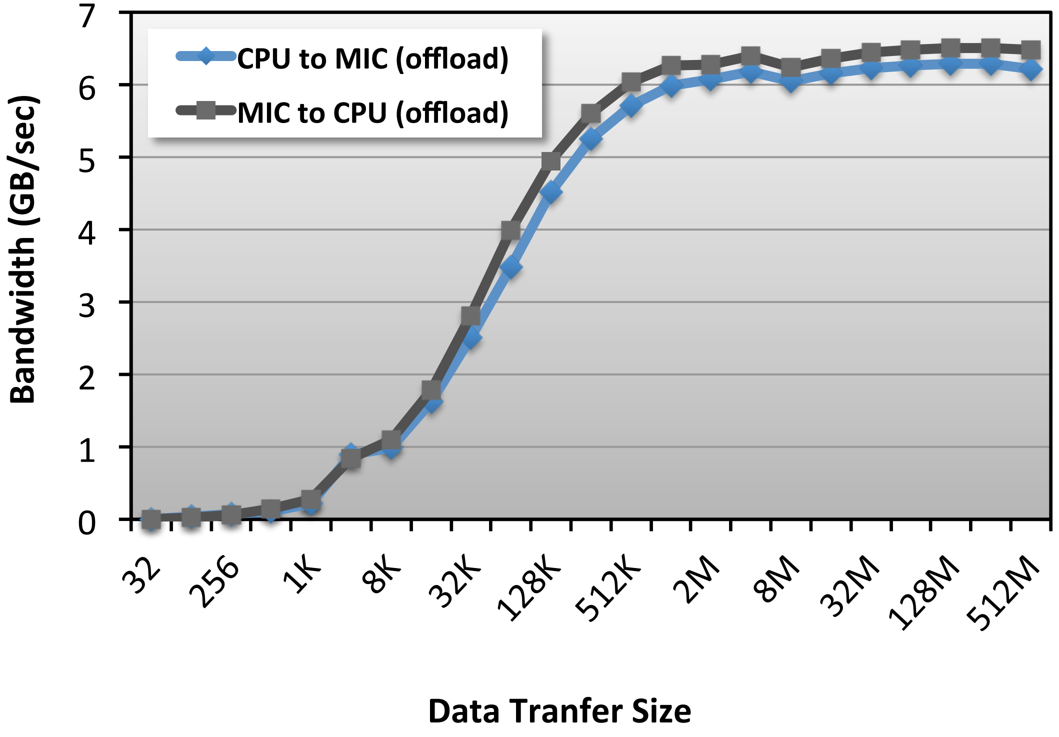
[from Bill Magro Intel MPI Library, OpenFabrics 2013]

- But, let's first compare some basic performance…

Stampede Data Movement

- Efficient data movement is critical in a heterogeneous compute environment (SB+MIC)

- Let’s look at current throughput between host CPU and MIC using standard “offload” semantics
  - bandwidth measurements are likely what you would expect
  - symmetric data exchange rates
  - capped by PCI XFER max



Stampede Host/MIC MPI Example

```
login1$  srun –p devel -n 32 --pty /bin/bash –l

$ export MV2_DIR=/home1/apps/intel13/ mvapich2-mic/76a7650/
$ $MV2_DIR/intel64/bin/mpicc -O3 -o hello.host hello.c
$ $MV2_DIR/k1om/bin/mpicc -O3 -o hello.mic hello.c
```

Compilation
(2 binaries)

```
$ cat hosts
c557-503
c557-504
c557-503-mic0
c557-504-mic0
$ cat paramfile
MV2_IBA_HCA=mlx4_0
$ cat config
-n 2 : ./hello.host
-n 2 : ./hello.mic
```

Configuration
Files

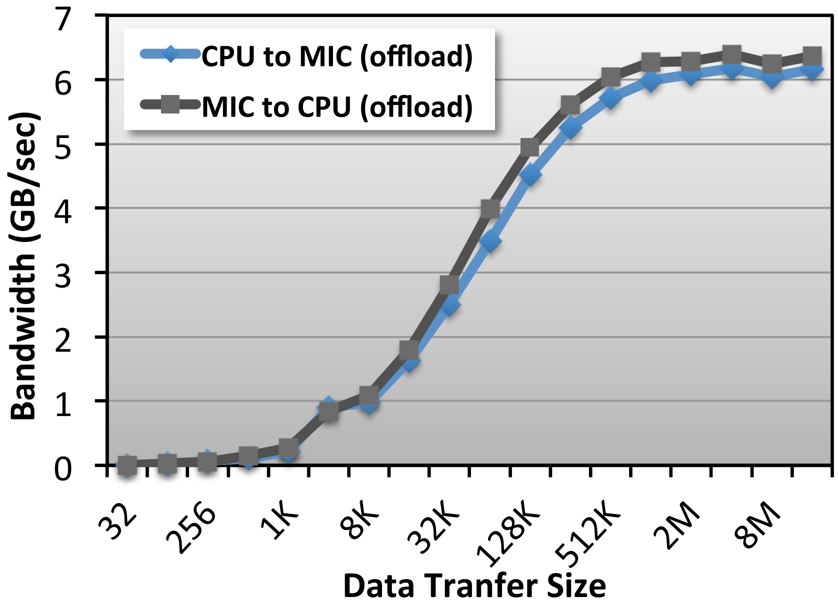
```
$ MV2_MIC_INSTALL_PATH=$MV2_DIR/k1om/ MV2_USER_CONFIG=./paramfile $MV2_DIR/intel64/
bin/mpirun_rsh -hostfile hosts -config config

 Hello, world (4 procs total)
    --> Process #   0 of   4 is alive. ->c557-503.stampede.tacc.utexas.edu
    --> Process #   1 of   4 is alive. ->c557-504.stampede.tacc.utexas.edu
    --> Process #   2 of   4 is alive. ->c557-503-mic0.stampede.tacc.utexas.edu
    --> Process #   3 of   4 is alive. ->c557-504-mic0.stampede.tacc.utexas.ed
```

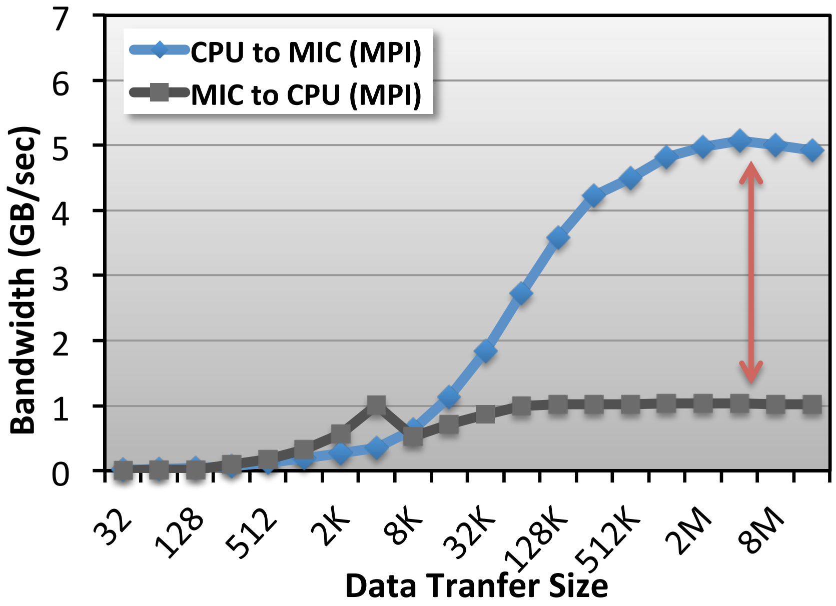
Execution

Phi Data Movement

Offload Test (Baseline)



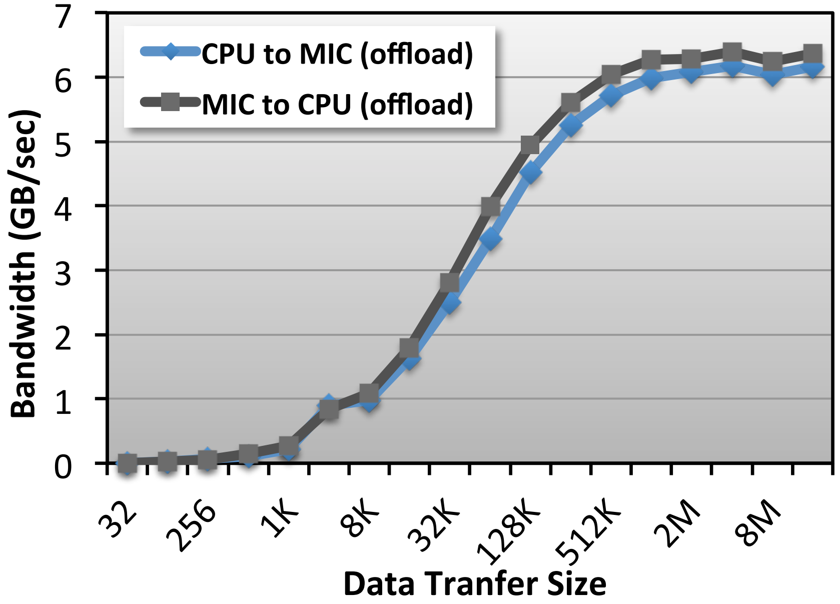
OSU Bandwidth Test
Intel MPI 4.1.0.030 (Feb 2013)
DAPL: **ofa-v2-mlx4_0-1u**



asymmetry undesired for tightly coupled scientific applications…

Phi Data Movement (improvement)

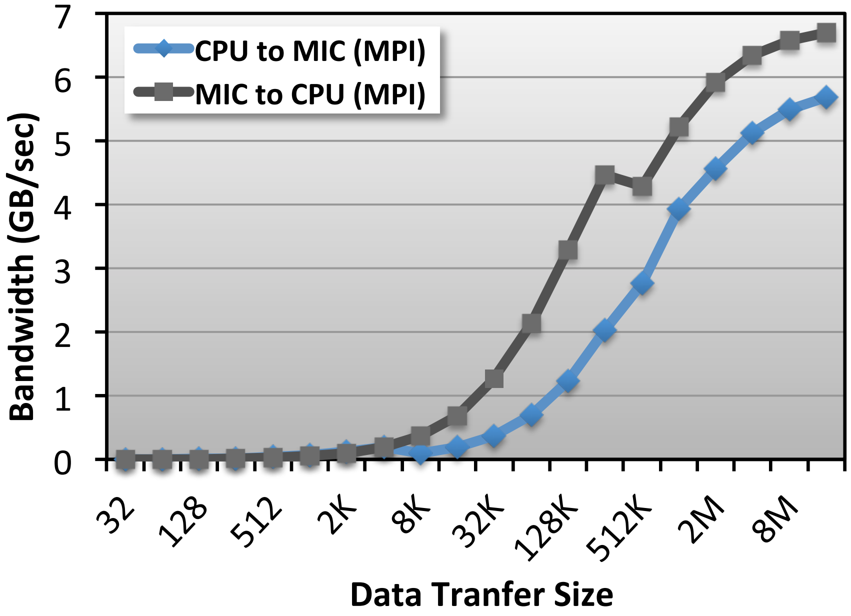
Offload Test (Baseline)



OSU Bandwidth Test

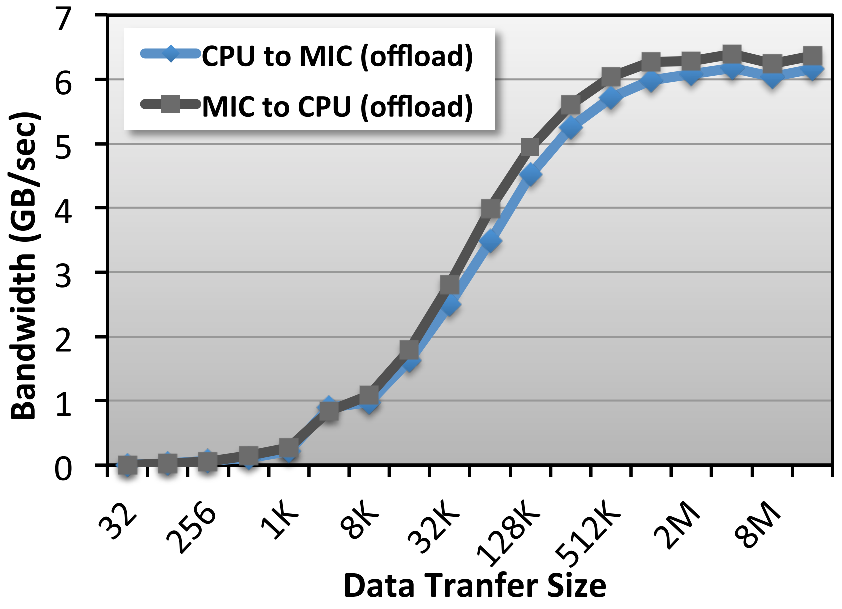
Intel MPI 4.1.1.036 (June 2013)

DAPL: **ofa-v2-scif0**



Phi Data Movement (improvement)

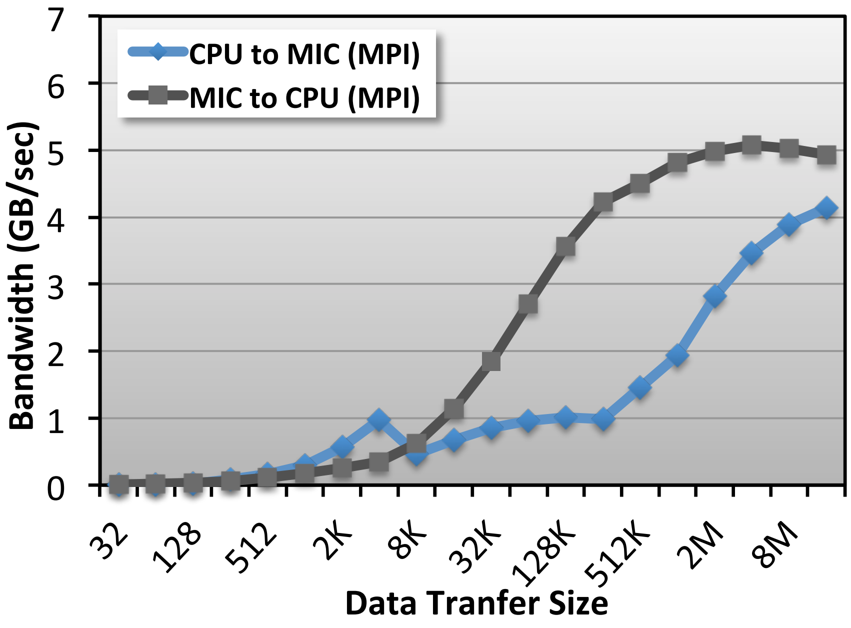
Offload Test (Baseline)



OSU Bandwidth Test

Intel MPI 4.1.1.036 (June 2013)

DAPL: **ofa-v2-mlx4_0-1,ofa-v2-mcm-1**

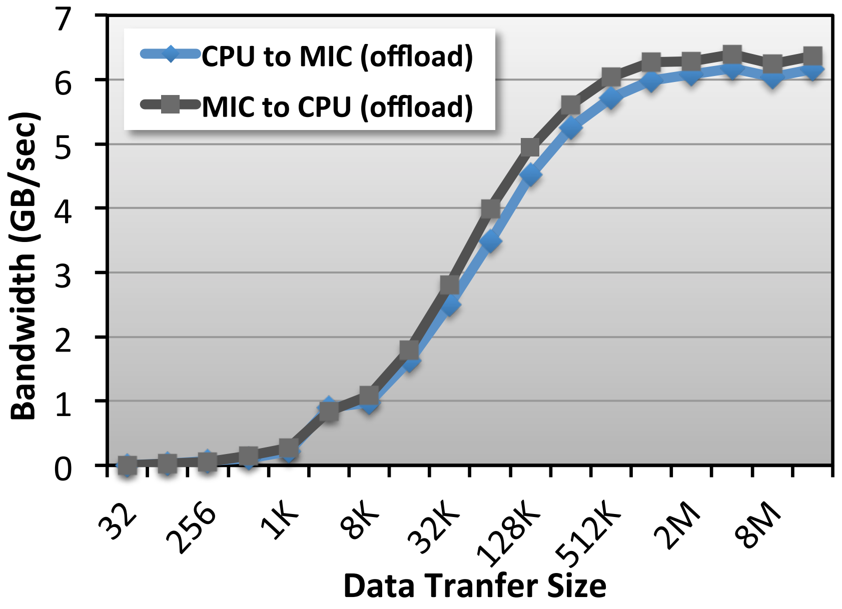


New developments to improve data transfer paths:

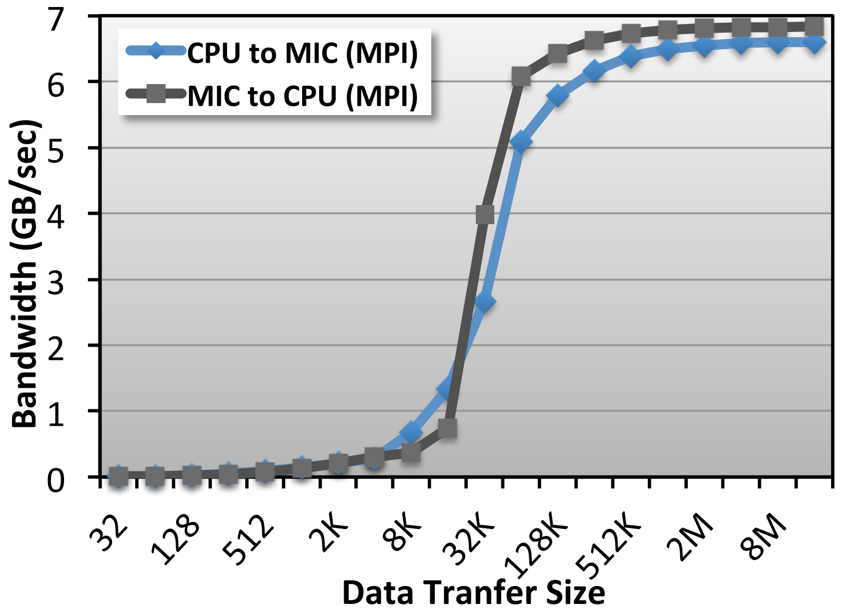
- *CCL Direct*
- *CCL-proxy (hybrid provider)*

Phi Data Movement (improvement)

Offload Test (Baseline)



OSU Bandwidth Test
MVAPICH2 Dev Version (July 2013)

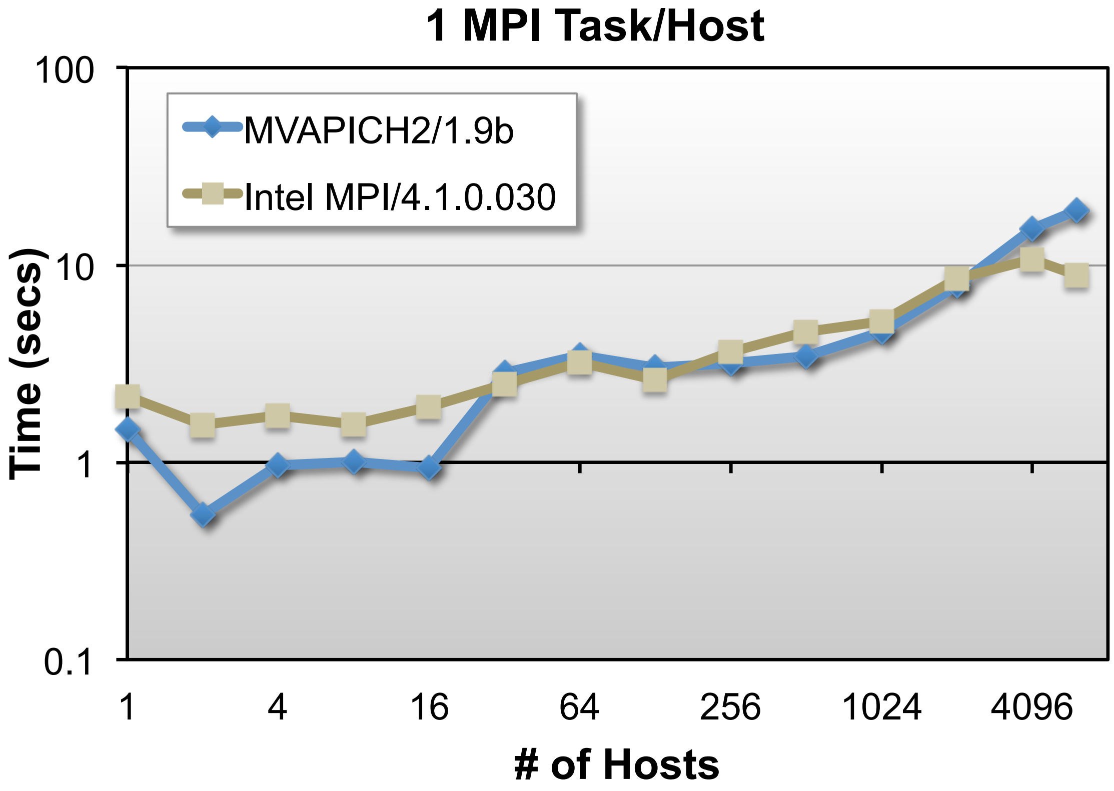


New developments to proxy messages through HOST

Additional MPI Considerations

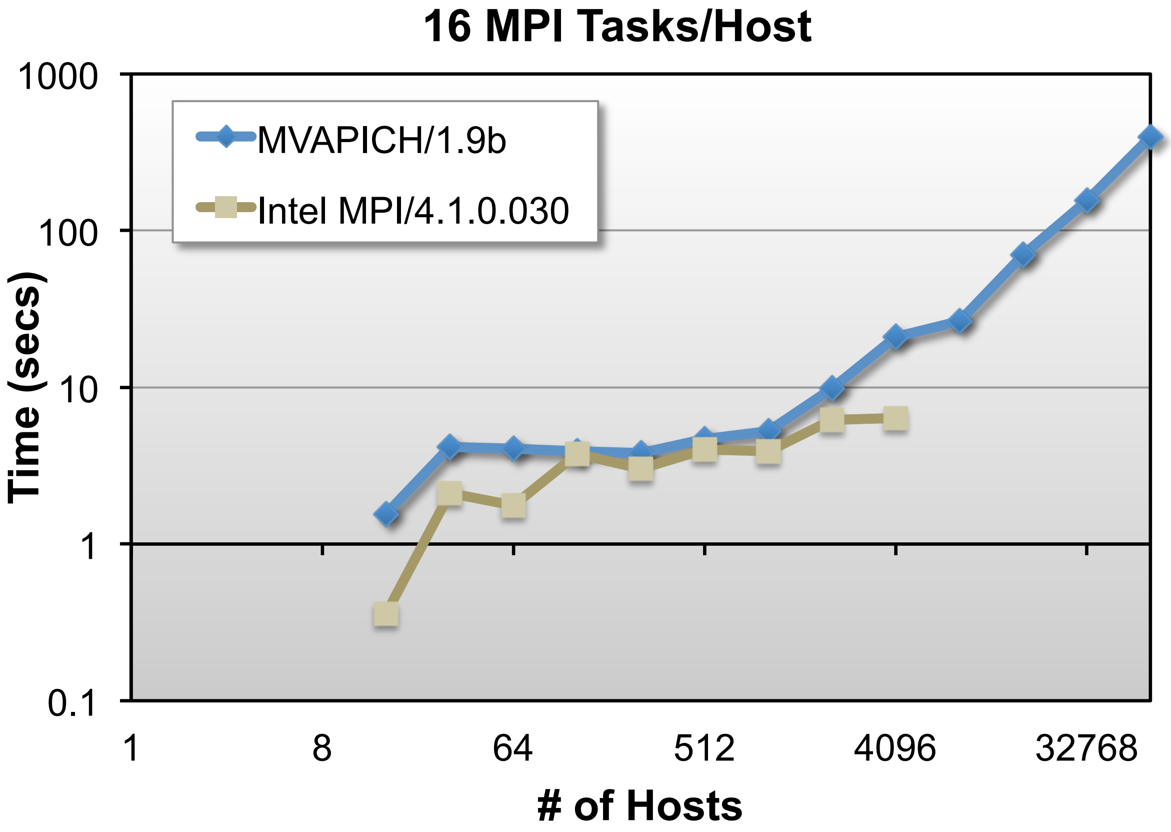
Job Startup Scalability Improvements (1 Way)

- Less than 20 seconds to launch MPI across all 6K hosts

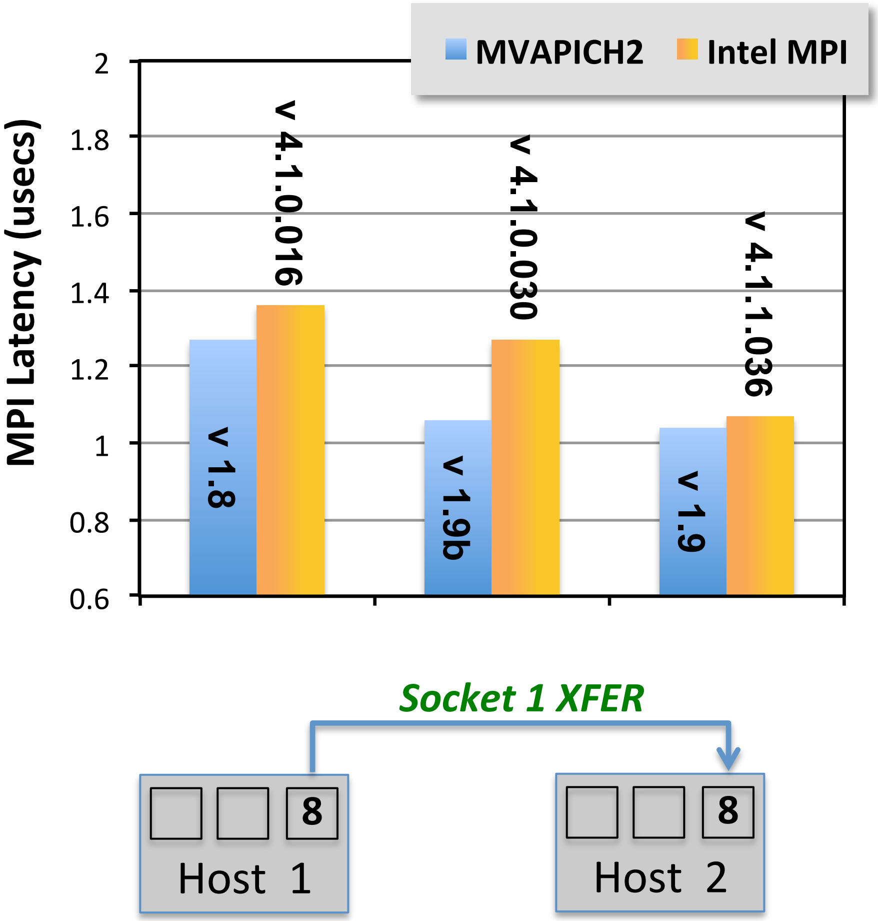


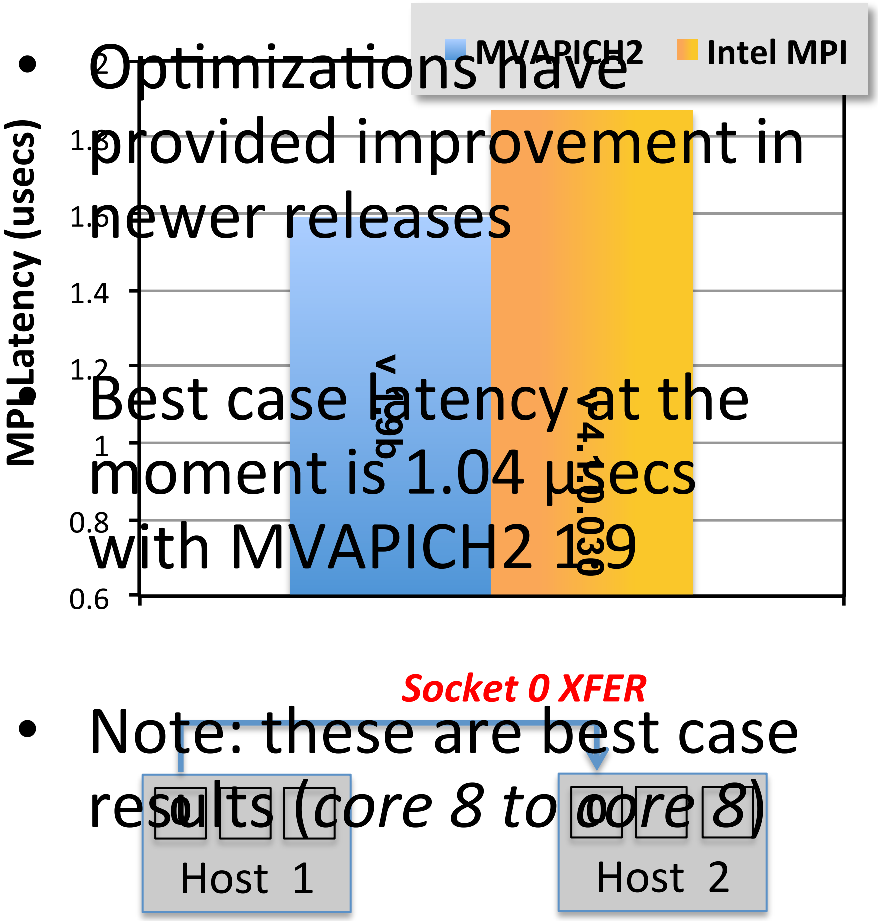
Job Startup Scalability - 16 Way

- Repeat the same process with 16-way jobs
- Majority of our users use 1 MPI task/core
- 2.5 minutes to complete at 32K (but this is still improving)



MPI Latency Improvements





- Optimizations have provided improvement in newer releases
- Best case latency at the moment is 1.04 µsecs with MVAPICH2 1.9
- Note: these are best case results (*core 8 to core 8*)

OSU Microbenchmarks (v3.6)

Performance Characteristics: MPI Latencies

- Minimum value approaching 1 microsecond latencies

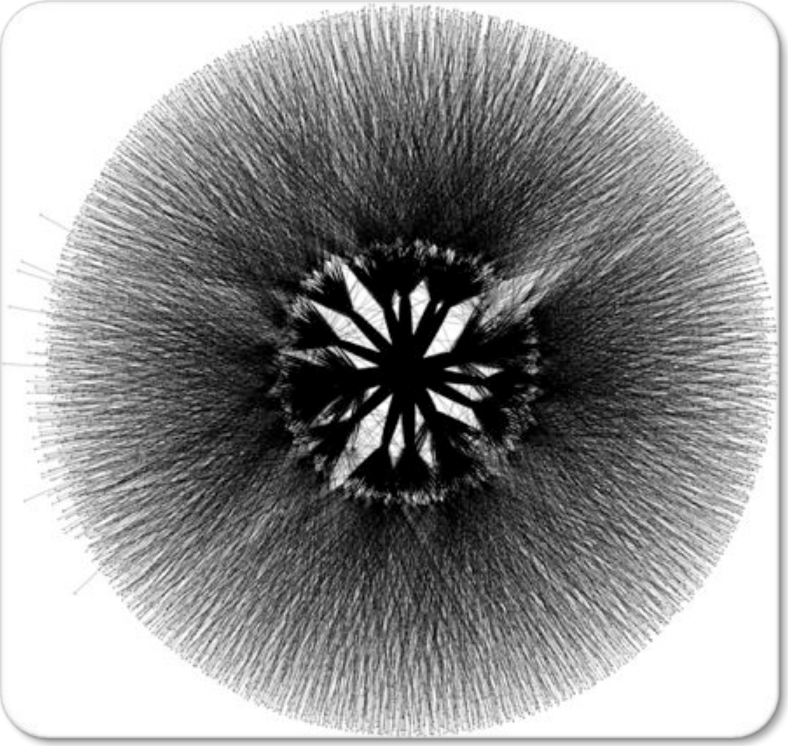
- Notes:
  - switch hops are not free
  - maximum distance across Stampede fabric is 5 switch hops

- These latency differences continue to motivate our topology-aware efforts

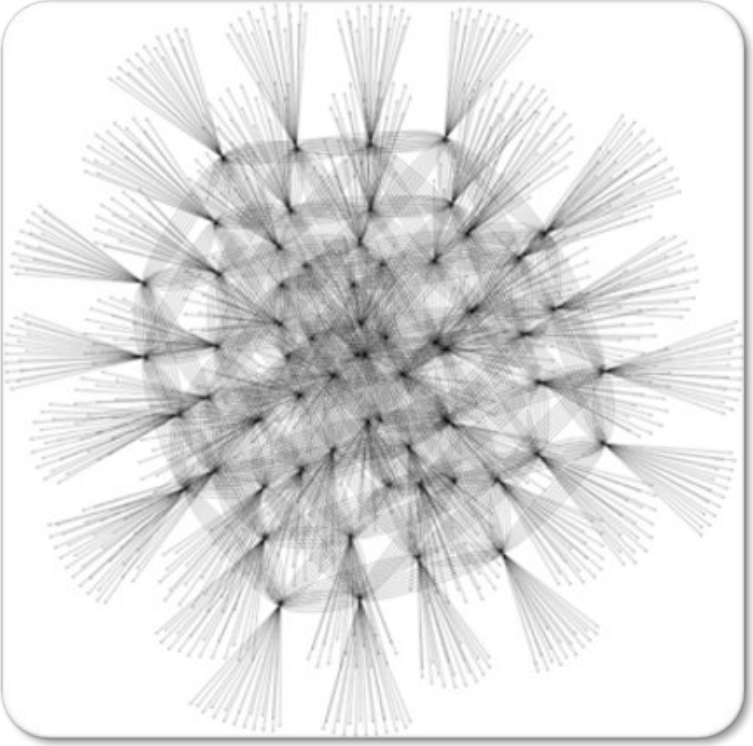
# switch hops	Avg Latency (µsec)
1	1.07
3	1.76
5	2.54

Topology Considerations

- At scale, process mapping with respect to topology can have significant impact on applications



Fat-tree (Stampede, TACC)

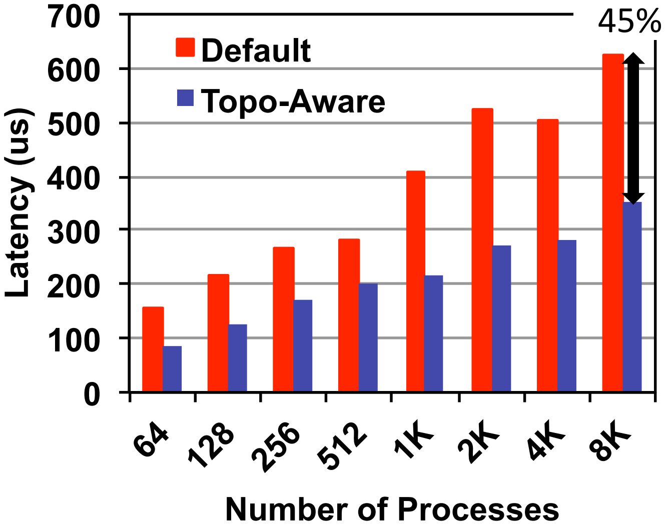


4x4x4 3D Torus (Gordon, SDSC)

Topology Considerations

- Topology query service (now in production on Stampede) - NSF STCI with OSU, SDSC
  - caches the entire linear forwarding table (LFT) for each IB switch - via OpenSM plugin or *ibnetdiscover* tools
  - exposed via network (socket) interface such that an MPI stack (or user application) can query the service remotely
  - can return # of hops between each host or full directed route between any two hosts

Nearest neighbor application benchmark from Stampede [courtesy H. Subramoni, SC 12]

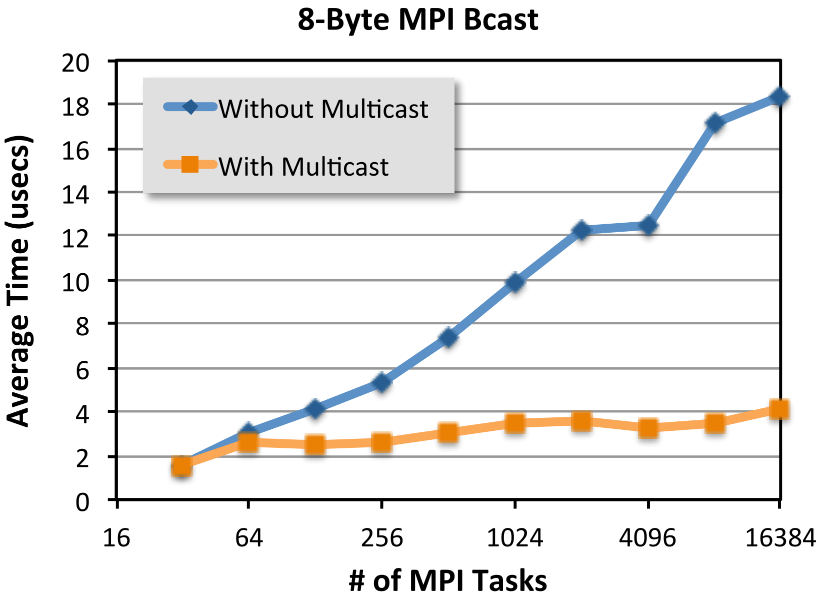


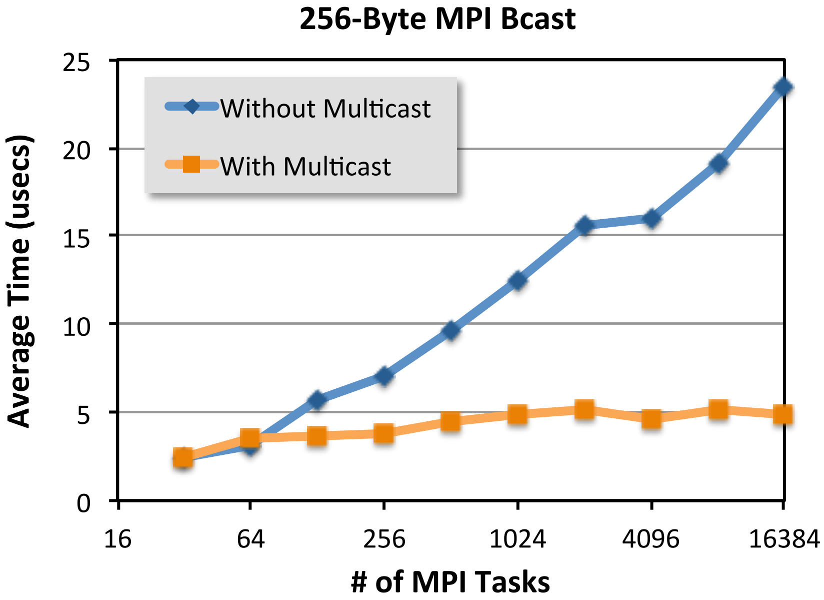
```
query c401-101:c405-101

c401-101 0x0002c90300776490 0x0002c903006f9010 0x0002c9030077c090 c405-101
```

Stampede/MVAPICH2 Multicast Features

- Hardware support for multi-cast in this new generation of IB
  - MVAPICH2 has support to use this
  - means that very large MPI_bcasts() can be much more efficient
  - dramatic improvement with increasing node count
  - factors of 3-5X reduction at 16K cores





Use **MV2_USE_MCAST=1** on Stampede to enable

A Community Thank You

- DK and his team have consistently gone well above and beyond the role of traditional academic software providers
- MVAPICH has evolved into production software that is supporting science in virtually all disciplines on systems around the world
- Performance is critical and the team consistently delivers novel methods to improve performance on fast-changing hardware
- The open-source HPC community benefits tremendously from this effort:

```
MPI_Send(&THANK_YOU,1000,MPI_INT,OSU,42,MPI_COMMUNITY);
```

TACC