Joint Institute for Computational Sciences

UT ORNL Computational Sciences

MVAPICH2-MIC on Beacon:

Exploring Performance on a Cluster with 4 Coprocessors per Node

OAK RIDGE National Laboratory

THE UNIVERSITY of TENNESSEE KNOXVILLE

R. Glenn Brook

Chief Technology Officer, JICS

Director, AACE

Mitch Horton & Chad Burdyshaw

Computational Scientists, AACE

Application Acceleration Center of Excellence (AACE)

- Established by The University of Tennessee (UT) in early 2011 alongside the National Institute for Computational Sciences (NICS) within the Joint Institute for Computational Sciences (JICS) at Oak Ridge National Laboratory (ORNL)
- Tasked with exploring the application of emerging computing technologies to the acceleration of science and engineering
- *An essential element of a sustainable software infrastructure for simulation in science and engineering*
- Director: R. Glenn Brook

JICS

The Beacon Project

- Funded by NSF to port and optimize codes to the Intel Xeon Phi, followed by state-funded expansion
- Beacon is #1 on the Nov. 2012 Green 500 List – 36 nodes, 71.4% efficiency, 2.499 GFLOPS/W
- Aggregate usage: 101 projects, 921 users
- Batch usage: 60 projects, 383 users
- More than 55 related publications to date reported by 35 projects
- Open Call for Participation

Hosted Accelerators: Intel MICs

Beacon Cray Xtreme-X Supercomputer Peak Performance: 210.1 TFLOP/s	
Compute Nodes	48
CPU model	Intel Xeon E5-2670
CPUs per node	2 8-core, 2.6GHz
RAM per node	256 GB
SSD per node	2 x 480 GB (RAID 0)
Intel® Xeon Phi Coprocessors per node	4 x 5110P 60-core, 1.053GHz 8 GB GDDR5 RAM
Interconnect	FDR InfiniBand Fat Tree

#1 on November 2012 Green500 List

Beacon's Architecture

MIC-to-MIC Communication Paths

Intramic: MIC0 **– Mem →** MIC0

Intra-Node/Inter-MIC:

MIC0 **–PCIe1→** MIC1

MIC2 **–PCIe2→** MIC3

MIC0 **–PCIe1→** Xeon1 **–QPI→** Xeon2 **–PCIe2→** MIC3

Inter-Node/Inter-MIC:

One Hop: Node1:(MIC0 **–PCIe1→** Xeon1) **–IB→**

Node2:(Xeon1 **–PCIe1→** MIC1)

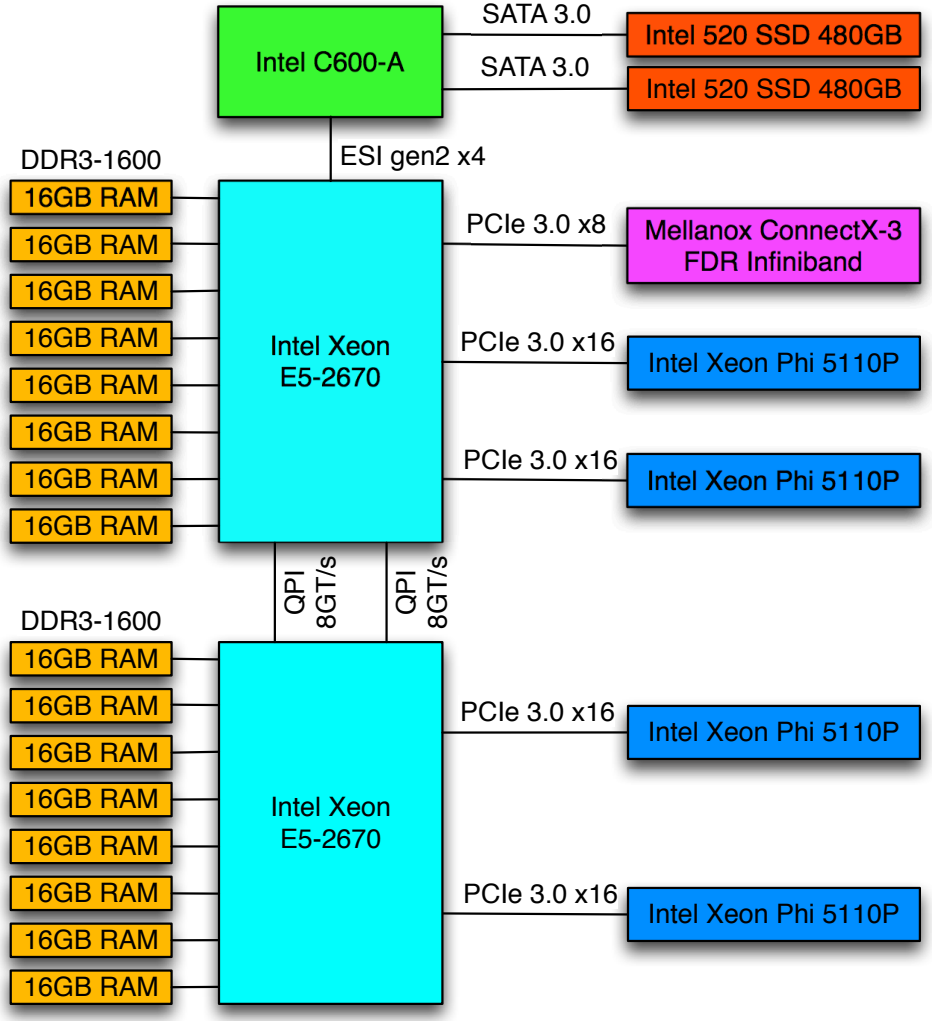
1.5 Hops:

Node1:(MIC0 **–PCIe1→** Xeon1) **–IB→**

Node2:(Xeon1 **–QPI→** Xeon2 **–PCIe2→** MIC3)

Two Hops:

Node1:(MIC2 **–PCIe2→** Xeon2 **–QPI→** Xeon1) **–IB→** Node2:(Xeon1 **–QPI→** Xeon2 **–PCIe2→** MIC3)



Why MVAPICH2-MIC?

- Initial issues with widely varying IMPI performance for different communication paths
  - Appears to be mitigated in later versions of IMPI, but performance still well below peak
- Limitations in SNB chipset affecting RDMA across different PCIe buses
  - Primary cause for widely varying IMPI performance, according to Intel
  - MVAPICH2-MIC offers proxy modes to address this
- Does MVAPICH2-MIC provide a better performance than Intel MPI on Beacon?

Testing Concerns

- Are we using MVAPICH2-MIC appropriately?
- Are we using the best Intel MPI tuning (for a fair comparison)?
- What is required to achieve improved performance with MVAPICH2-MIC?
- Should we recommend MVAPICH2-MIC for performance optimization on Beacon and other machines with similar architectures?
- Can we provide a set of associated best practices?

OSU MicroBenchmarks v4.4.1

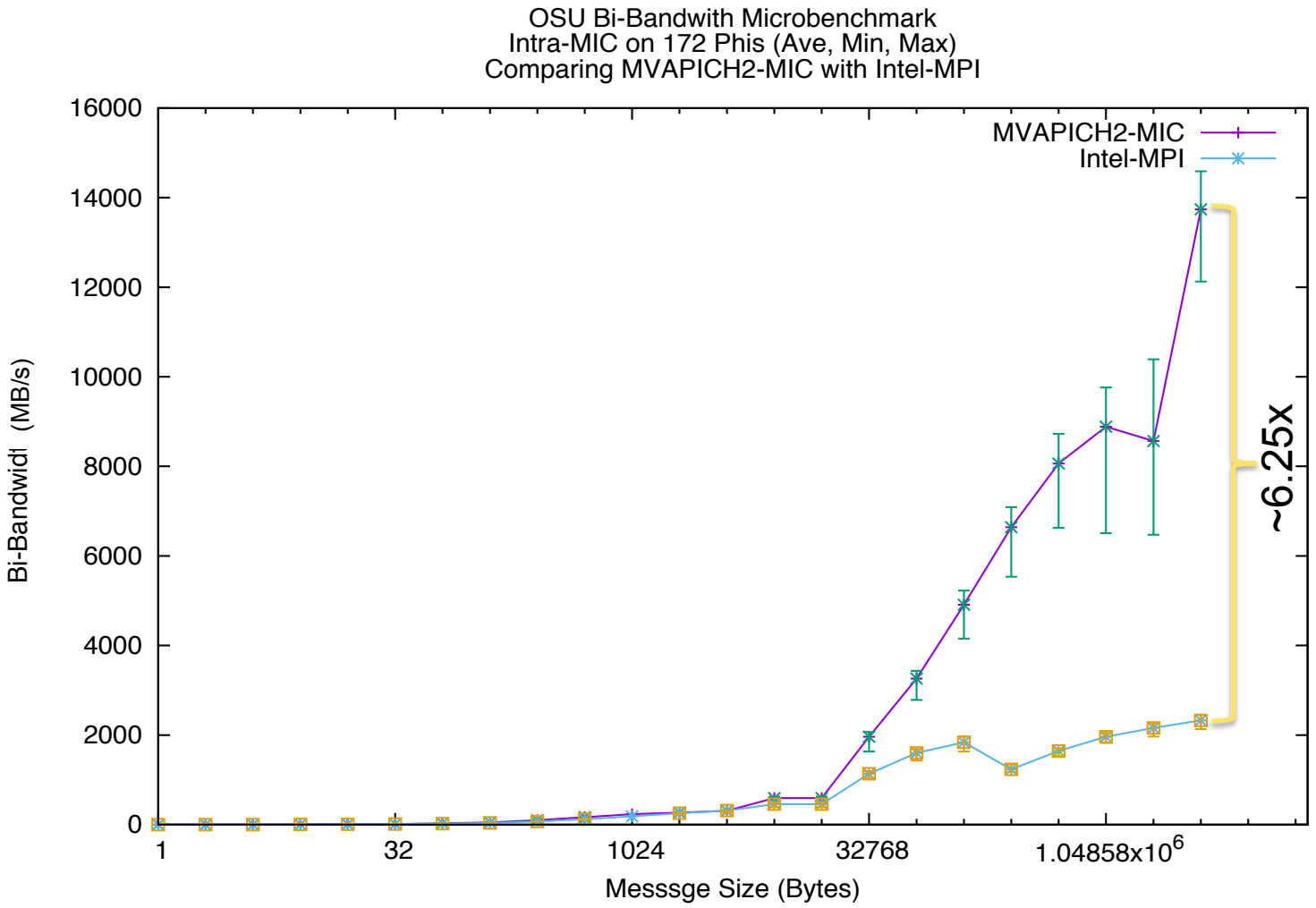
- Can we duplicate reported microbenchmark performance improvements using MVAPICH2-MIC?
- Which benchmarks should we use?
  - osu_bibw
    - Bi-directional bandwidth measures the maximum sustainable aggregate bandwidth by two nodes
  - Also tested the following, but chose to limit presentation to osu_bibw – similar comparison results
    - osu_alltoall
    - osu_latency (point to point)
- Machine-wide regression testing for communication fabric and MPI stack

Bi-Bandwidth Intra-MIC regression test over all Beacon MIC's

An osu_bibw regression test was run over all available MICs.

The plot shows the mean value bracketed by min and max values at each message size.

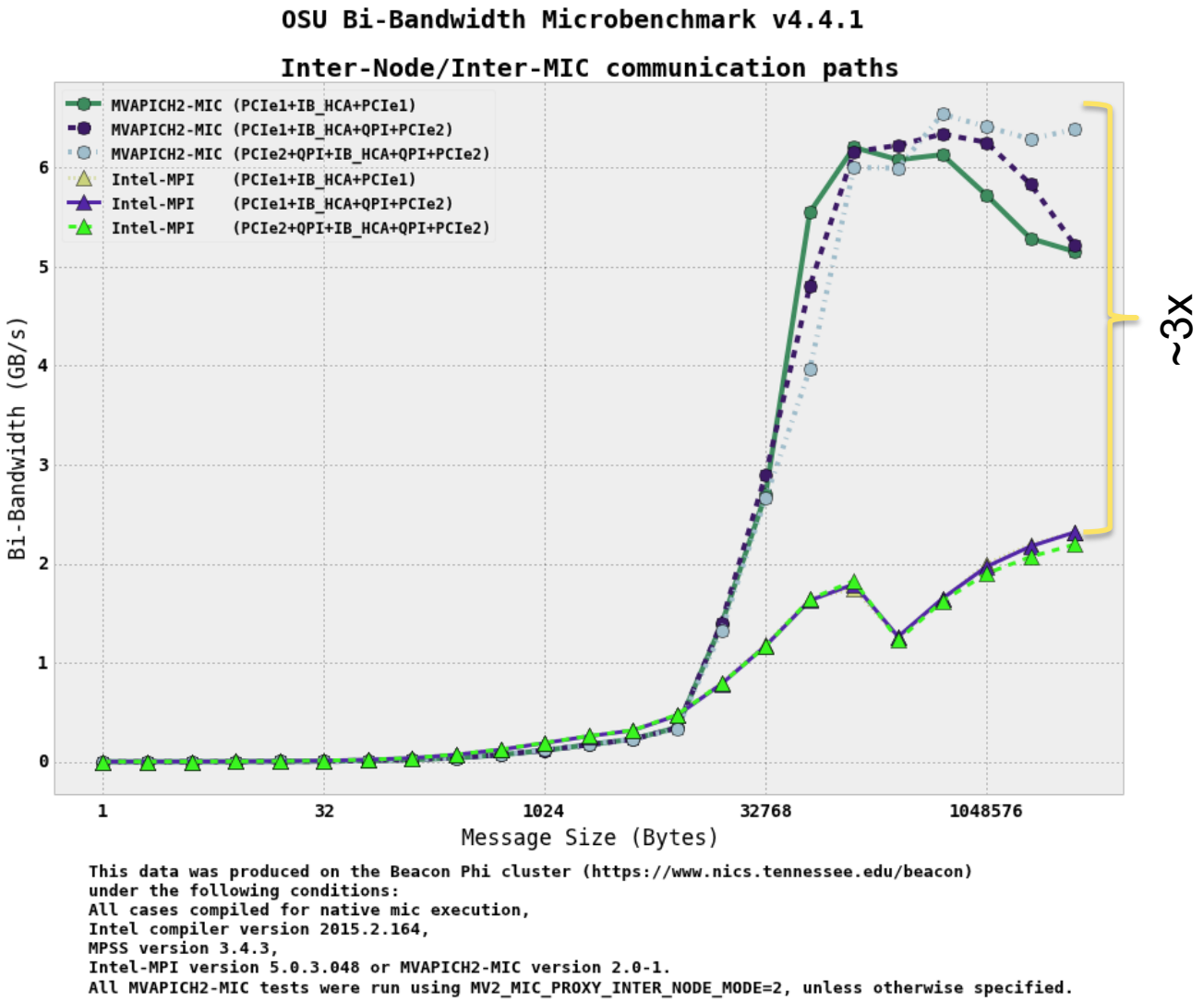
Why is IMPI so much slower within the MIC?



Bi-Bandwidth Inter-Node/Inter-MIC

OSU Bi-Bandwidth microbenchmark profile of Inter-Node/Inter-MIC pathways after tuning MVAPICH2-MIC proxy parameters for best performance on Beacon.

These results were not immediately translated to application performance.



Cartesian Stencil App

To determine why applications were not seeing performance improvements under MVAPIC2-MIC, we implemented a simplified, point-to-point Cartesian communicator (no computation or I/O) as an analog to large-scale parallel field solvers.

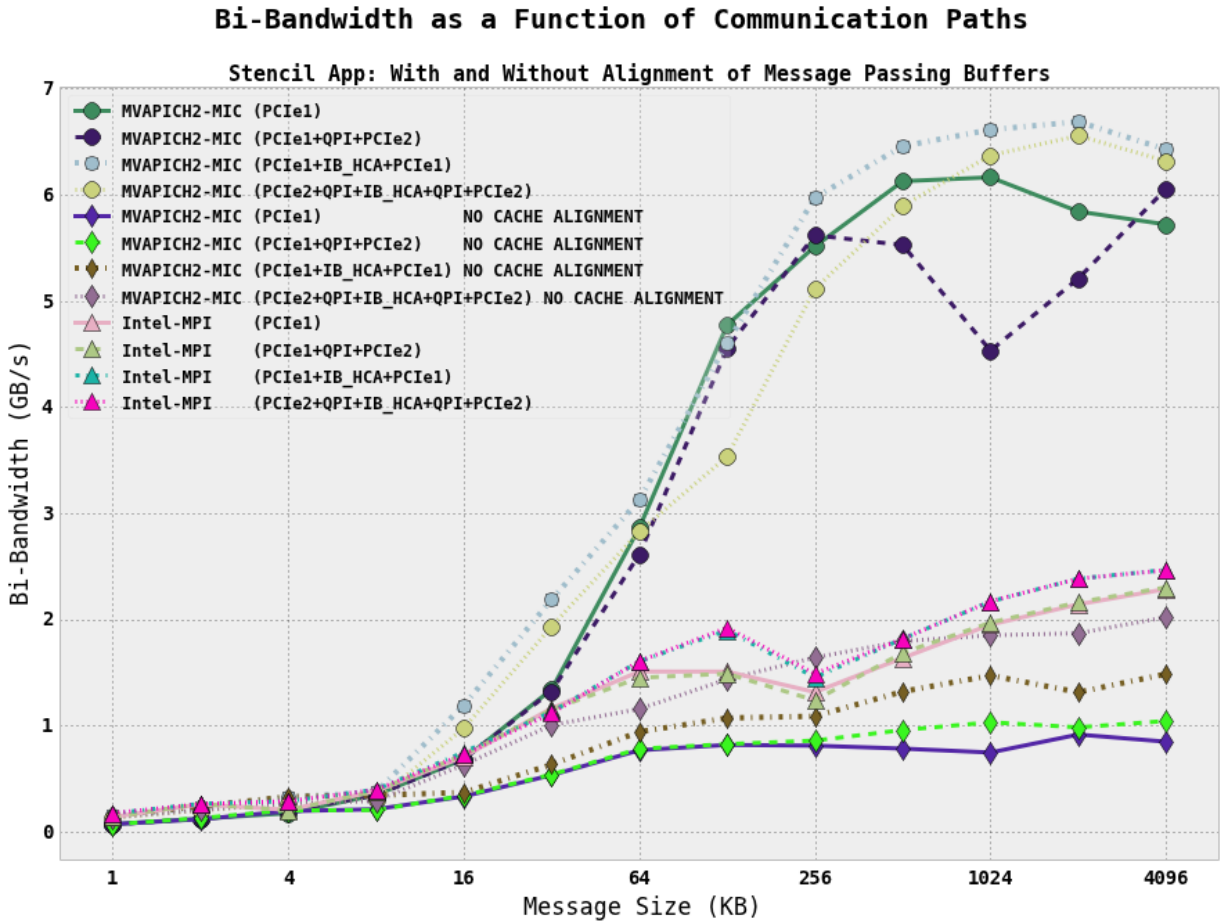
With this experimental application, we investigated the following questions:

- Is cache optimization necessary for MVAPICH performance?
  - Investigated the use of cache optimization through MPI buffer alignment.
- Do number of ranks/MIC have an impact on MVAPICH performance?
  - Reduced the analysis to 1D stencil cases with periodic boundaries (2 neighbors for each rank) to allow for linear scaling of ranks to investigate the effect of rank sizes on MPI performance.
- What inter-node proxy settings are appropriate for Beacons architecture?
  - Experimented with inter-node proxy settings to determine optimal configuration for Beacon

Stencil App: MVAPICH2-MIC Performance With and Without MPI Buffer Alignment

Prior to implementing MPI buffer alignment, MVAPICH2-MIC performance was worse than Intel-MPI for all communication pathways.

After aligning buffers, performance for MVAPICH2-MIC was ~3 times that of IMPI

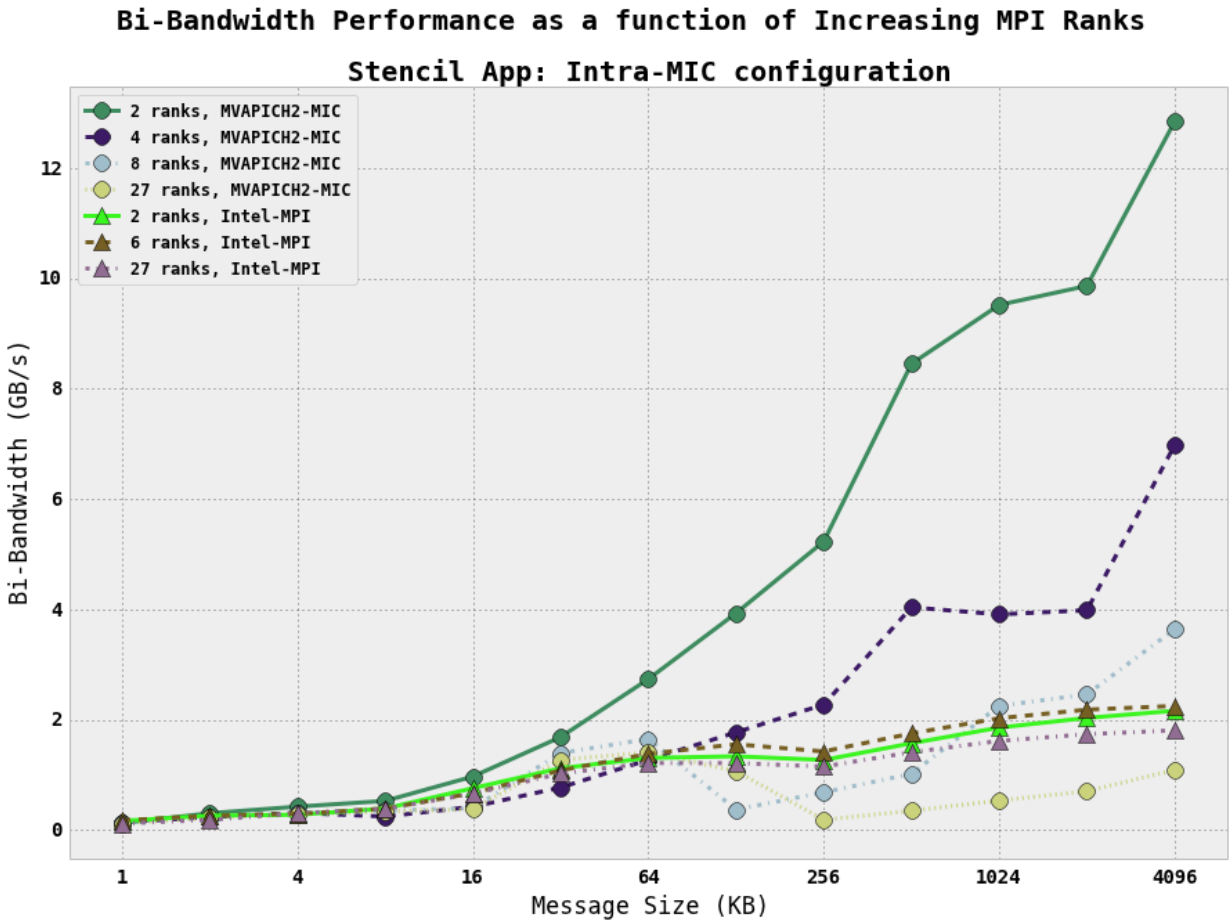


This data was produced on the Beacon Phi cluster (https://www.nics.tennessee.edu/beacon)
under the following conditions:
All cases compiled for native mic execution,
Intel compiler version 2015.2.164,
MPSS version 3.4.3,
Intel-MPI version 5.0.3.048 or MVAPICH2-MIC version 2.0-1.
All MVAPICH2-MIC tests were run using MV2_MIC_PROXY_INTER_NODE_MODE=2, unless otherwise specified.

Effect of Ranks/MIC on Intra-MIC Bi-Bandwidth Performance

Increasing the ranks/MIC quickly degraded MVAPICH2-MIC performance.

Intel-MPI performance was not affected by ranks/mic



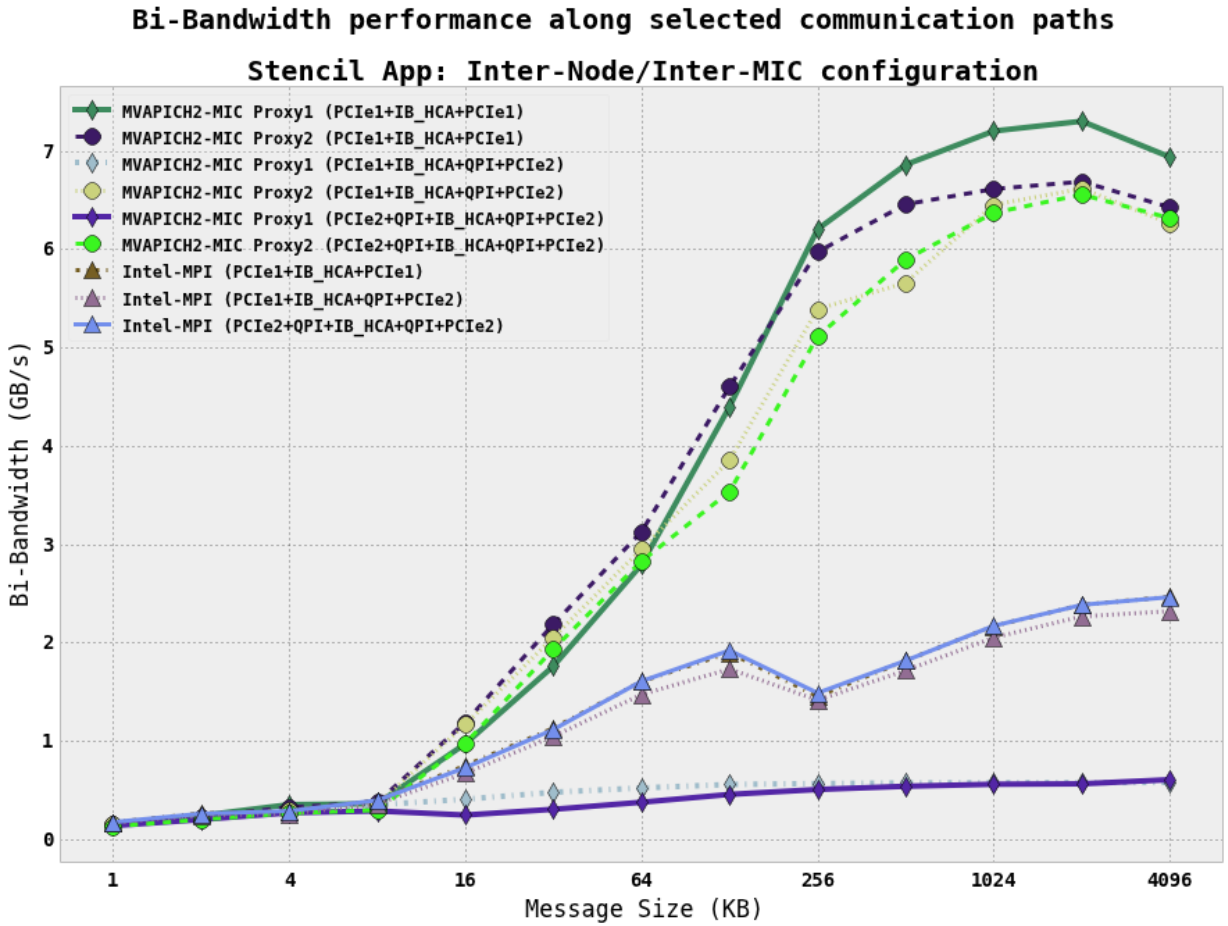
This data was produced on the Beacon Phi cluster (https://www.nics.tennessee.edu/beacon)
under the following conditions:
All cases compiled for native mic execution,
Intel compiler version 2015.2.164,
MPSS version 3.4.3,
Intel-MPI version 5.0.3.048 or MVAPICH2-MIC version 2.0-1.
All MVAPICH2-MIC tests were run using MV2_MIC_PROXY_INTER_NODE_MODE=2, unless otherwise specified.

Inter-Node/Inter-MIC
MVAPICH2-MIC Inter-Node Proxy Comparison

Inter-Node proxy settings for MVAPICH2-MIC were critical to performance on Beacon.

Setting proxy=1 degrades performance on all pathways except One-Hop

Setting proxy=2 did not significantly degrade One-Hop performance and was critical for all other pathways.



This data was produced on the Beacon Phi cluster (https://www.nics.tennessee.edu/beacon)
under the following conditions:
All cases compiled for native mic execution,
Intel compiler version 2015.2.164,
MPSS version 3.4.3,
Intel-MPI version 5.0.3.048 or MVAPICH2-MIC version 2.0-1.
All MVAPICH2-MIC tests were run using MV2_MIC_PROXY_INTER_NODE_MODE=2, unless otherwise specified.

Stencil App Summary

To determine why applications were not seeing performance improvements under MVAPIC2-MIC, we implemented a simplified application which involved only point-to-point communications and no computation.

Lessons learned:

- MPI message buffers must be aligned to page boundaries for best performance.
  - Necessary for Inter-Node and Inter-MIC performance with MVAPICH2-MIC
  - Does not affect Intel-MPI performance
- Intra-MIC performance of MVAPIC2-MIC degrades as MPI ranks increase
  - Must minimize MPI ranks per MIC for good MVAPICH2-MIC performance
  - Intel-MPI is minimally effected
- MV2_MIC_PROXY_INTER_NODE_MODE=2 provides best overall performance on Beacons architecture, with minimal impact to performance for MIC's on the primary PCIe bus.
- In general, IMPI provides more consistent but slower performance than properly tuned MVAPICH2-MIC.

Livermore Unstructured Lagrangian Explicit Shock Hydrodynamics (LULESH)*

LULESH is a highly simplified application, hard-coded to only solve a simple Sedov blast problem with analytic answers – but represents the numerical algorithms, data motion, and programming style typical in scientific C or C++ based applications

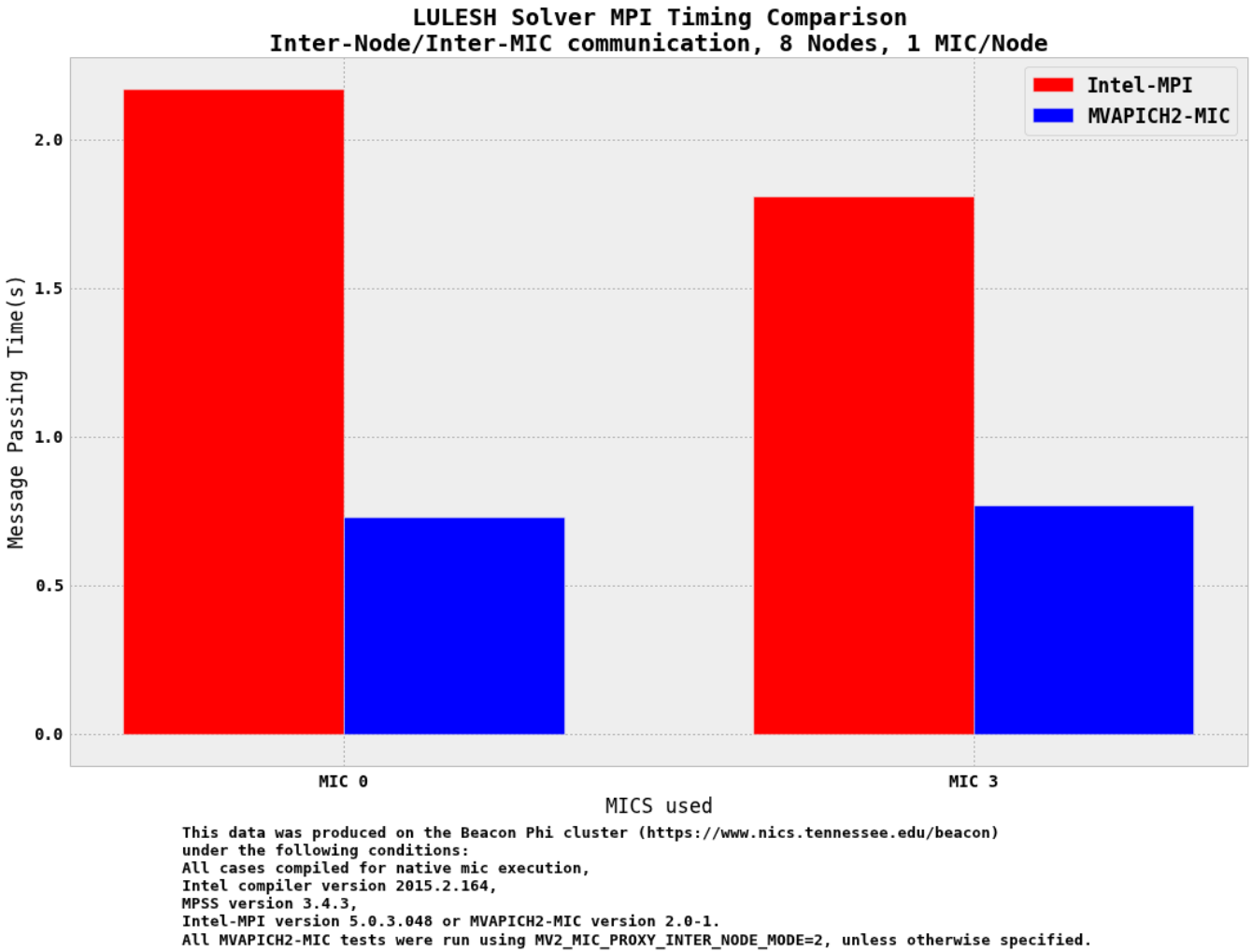
- LULESH code explicitly pads the MPI communication buffers to align with the page size in order to optimize performance
- The case run in this test is a 90x90x90 element cube distributed among 8 nodes using 1 MIC/Node. The average message size is 17 KB
- Characterized on Xeon processors (not MIC's) using Intel-MPI and Intel Trace Analyzer.
  - MPI portion is only 3% of run time:
  - 65% MPI_Wait
  - 26% MPI_Allreduce
  - 5% MPI_Isend

* Karlin, A. Bhatele, B. Chamberlain, J. Cohen, Z. Devito, M. Gokhale, R. Haque, R. Hornung, J. Keasler, D. Laney, E. Luke, S. Lloyd, J. McGraw R. Neely, D. Richards, M. Schulz, C. H. Still, F. Wang, D. Wong. LULESH Programming Model and Performance Ports Overview, December 2012, pages 1-17,

LULESH MPI Timing Comparison One-Hop and Two-Hop cases

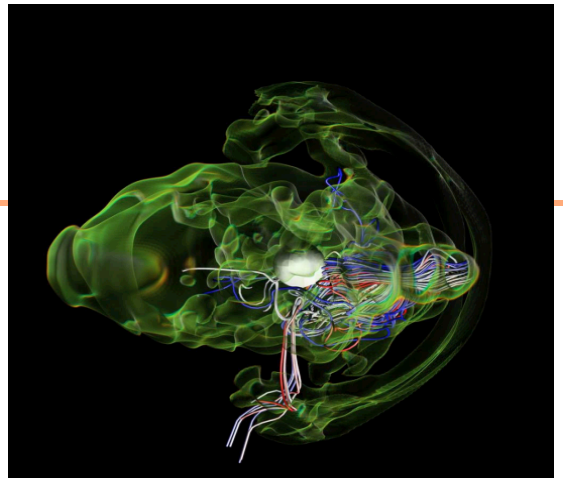
Timing of MPI_Allreduce and MPI_Isend calls

~60% reduction in message passing time for best and worst case internode communication topologies



*These results are taken from a single run per case so some variation is expected

General Astrophysical Simulation System (GenASiS)*



GenASiS is a Multiphysics code developed for large scale astrophysics simulations

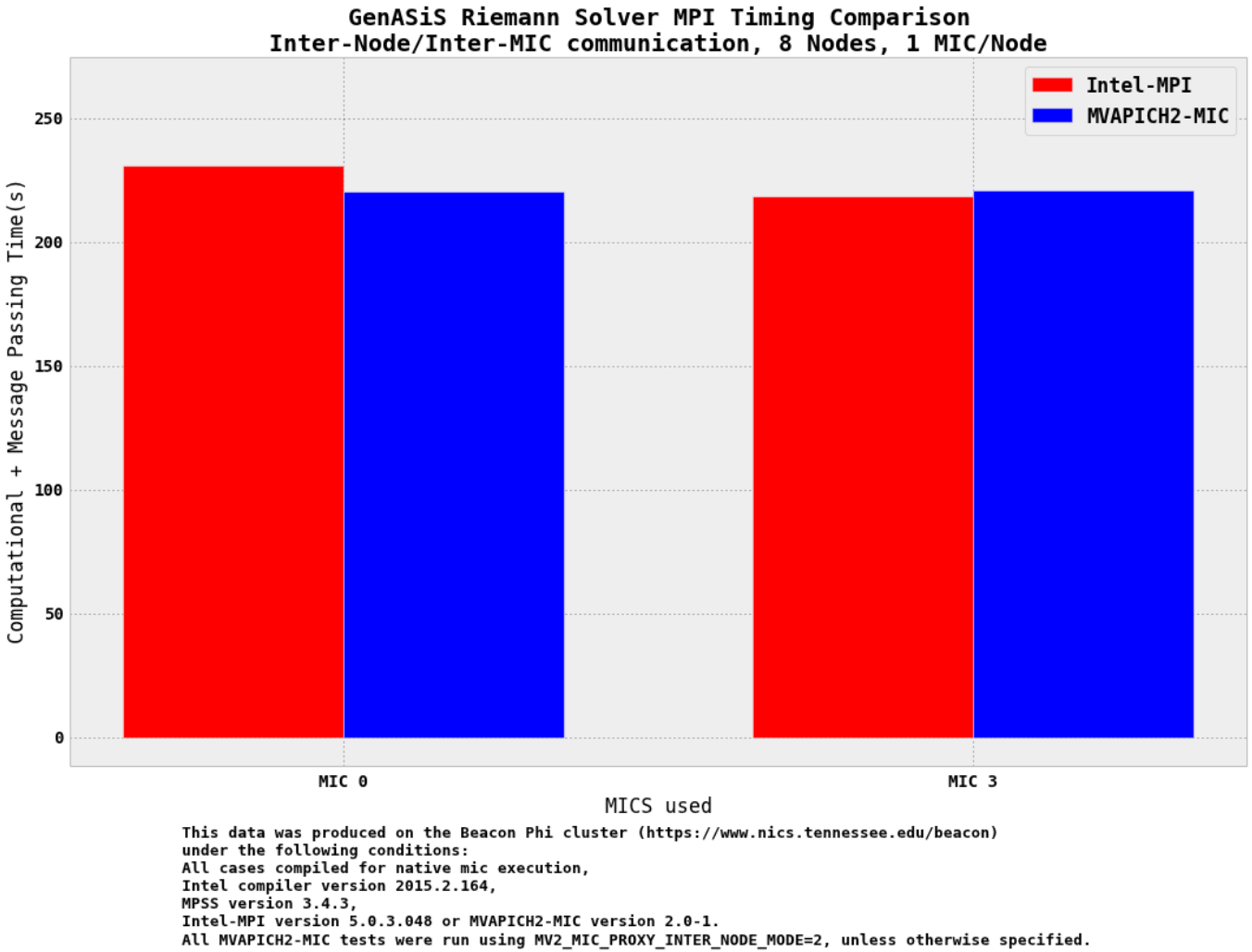
- The case tested here was a finite volume, 2nd order approximate Riemann solver, with 2nd order TVD Runge-Kutta method
  - 1D problem with 4096 cells distributed over 8 nodes using 1 MIC/ Node
  - Nearest neighbor ghost exchange, Allreduce for time step determination
- GenASiS does not currently align MPI buffers for cache optimization.
- Characterized on Xeon processors using Intel-MPI and Intel Trace Analyzer.
  - MPI portion is ~10% of run time.
  - 64% MPI_Allreduce
  - 35% MPI_Wait and MPI_Barrier

*Christian Y. Cardall, Reuben D. Budiardja, Eirik Endeve, Anthony Mezzacappa, "GenASiS: GENERAL ASTROPHYSICAL SIMULATION SYSTEM. I. REFINABLE MESH", The Astrophysical Journal Supplement Series, 210:17 (29pp), 2014 February

GenASiS Timing Comparison One-Hop and Two Hop Cases

GenASiS does not currently align MPI buffers for cache optimization.

This is the likely reason for lack of speedup.



*These results are taken from a single run per case so some variation is expected

Basic Local Alignment Search Tool (mpiBLAST)

NCBI BLAST compares nucleotide or protein sequences to sequence databases and calculates the statistical significance of matches.

BLAST can be used to infer functional and evolutionary relationships between sequences as well as help identify members of gene families.

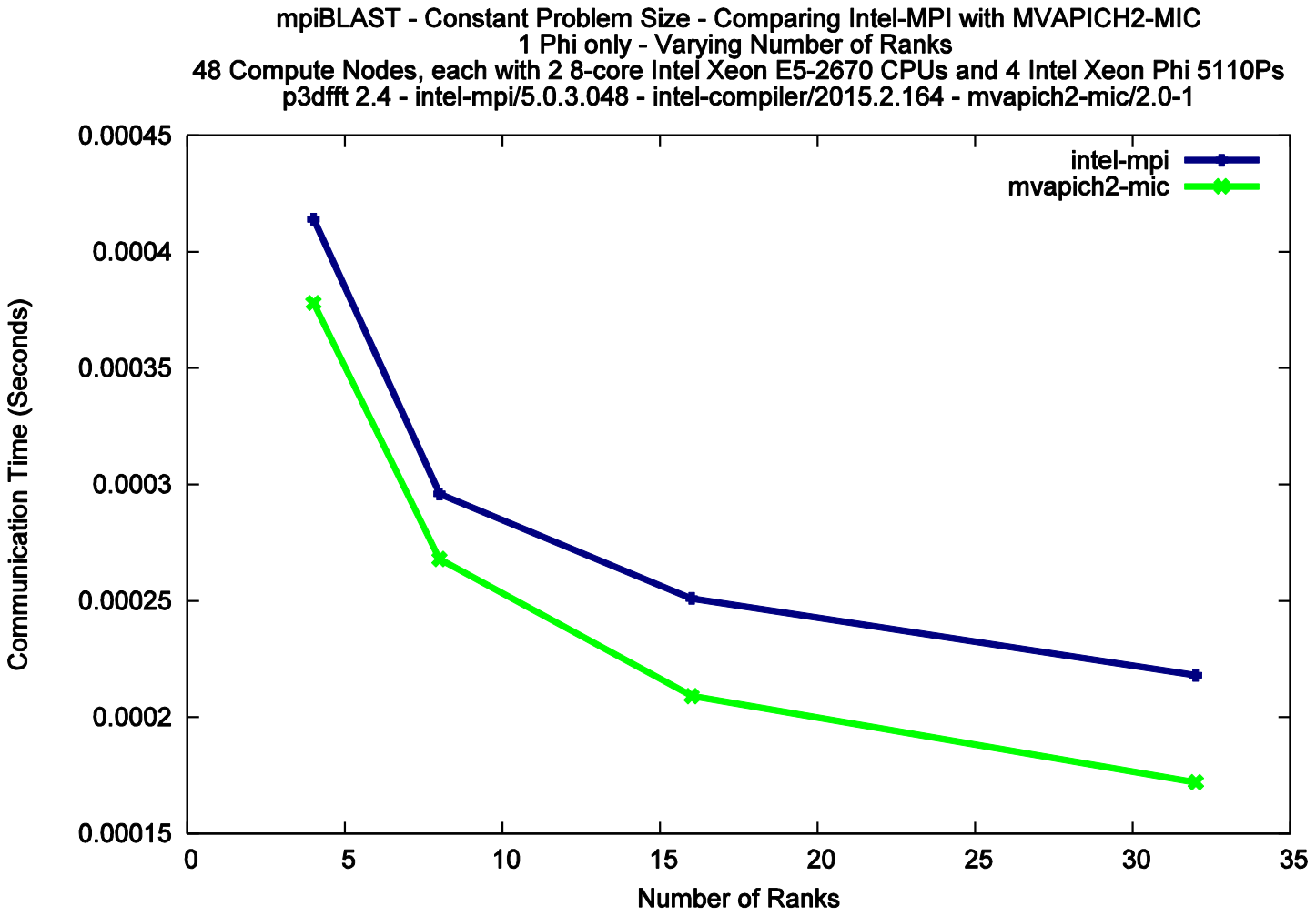
mpiBLAST is a parallel/distributed implementation of NCBI BLAST

- mpiBLAST uses MPI ranks to distribute the tasks in two ways:
  - The search database of biological sequences is partitioned and distributed to the ranks which allows large databases to be held in the collective memory of the MPI ranks (replication groups)
  - input query sequences are distributed between replication groups.
  - A super master is used get queries to and from the master to the workers to distribute the queries
  - communicating search results: different methods include back to a centralized location or writing cooperatively with other search ranks in a replication group with MPI-IO
- mpiBLAST does not optimize MPI buffer for cache, but the testing was done intra-MIC, so this condition will not inhibit performance

mpiBlast MPI Timing Comparison (Intra-MIC)

~20% faster at 32 ranks

Increasing ranks/ MIC does not affect performance in the same way as seen with the stencil app.



Parallel 3D Fast Fourier Transforms (P3DFFT)*

Parallel Three-Dimensional Fast Fourier Transforms, dubbed P3DFFT, is a library for large-scale computer simulations on parallel platforms. 3D FFT is an important algorithm for simulations in a wide range of fields, including studies of turbulence, climatology, astrophysics and material science.

This project was initiated at San Diego Supercomputer Center (SDSC) at UC San Diego by its main author Dmitry Pekurovsky, Ph.D.

- P3DFFT uses 2D, or pencil, decomposition. This overcomes an important limitation to scalability inherent in FFT libraries implementing 1D (or slab) decomposition:
  - the number of processors/tasks used to run this problem in parallel can be as large as N^2, were N is the linear problem size. This approach has shown good scalability up to 131,072 cores. P3DFFT is optimized for large data sets.
- P3DFFT message passing buffers are aligned to optimize cache
- IB errors when using MV2_MIC_PROXY_INTER_NODE_MODE=2

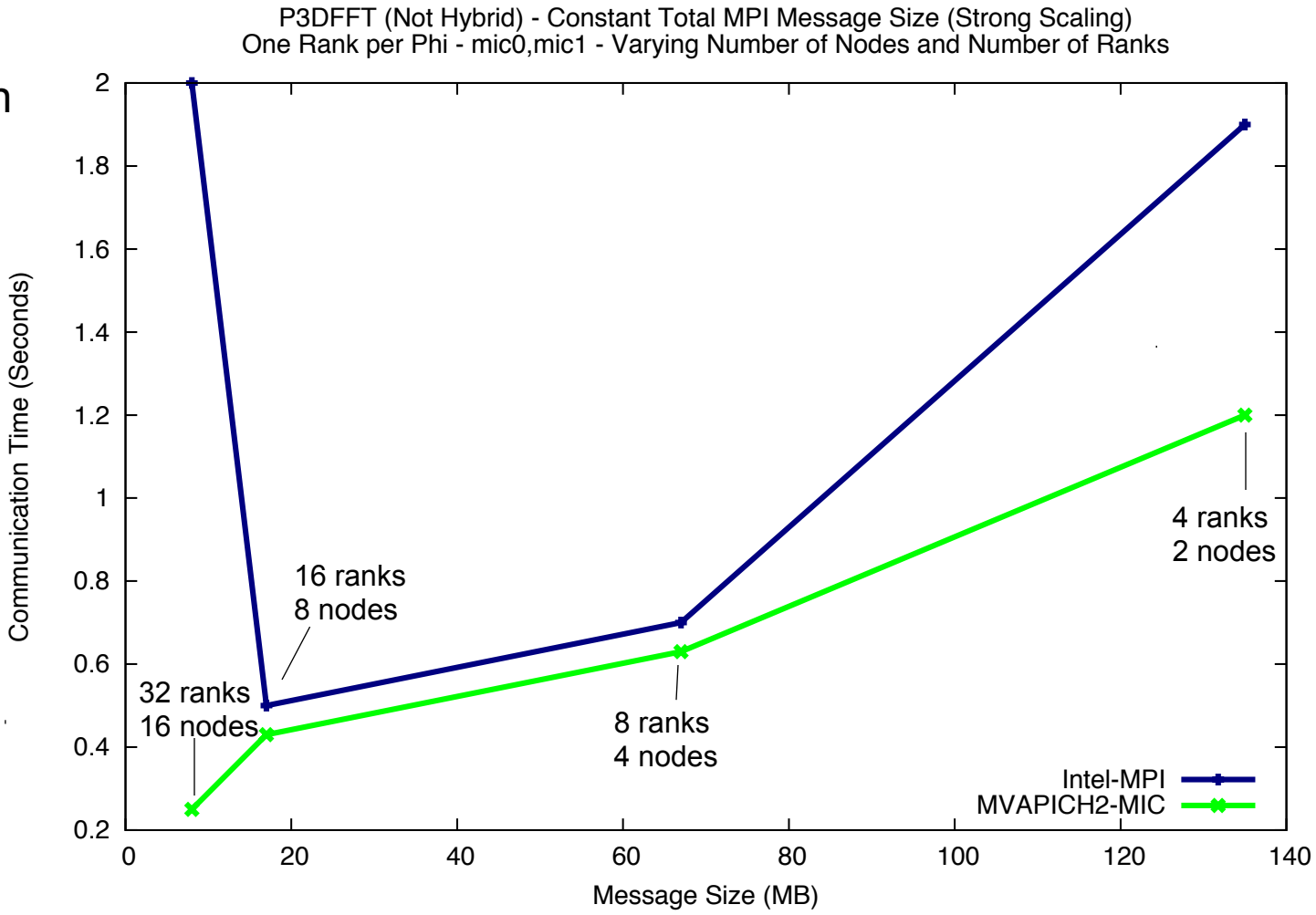
*D. Pekurovsky, "P3DFFT: a framework for parallel computations of Fourier transforms in three dimensions", SIAM Journal on Scientific Computing 2012, Vol. 34, No. 4, pp. C192-C209

P3DFFT MPI Timing Comparison One-Hop configuration

~90% decrease in MPI communication time at message size of 8MB

~40% decrease for message sizes of 135MB

Fixed problem size with increasing ranks (2MIC's/ Node) from right to left



MV2_MIC_PROXY_INTER_NODE_MODE=1

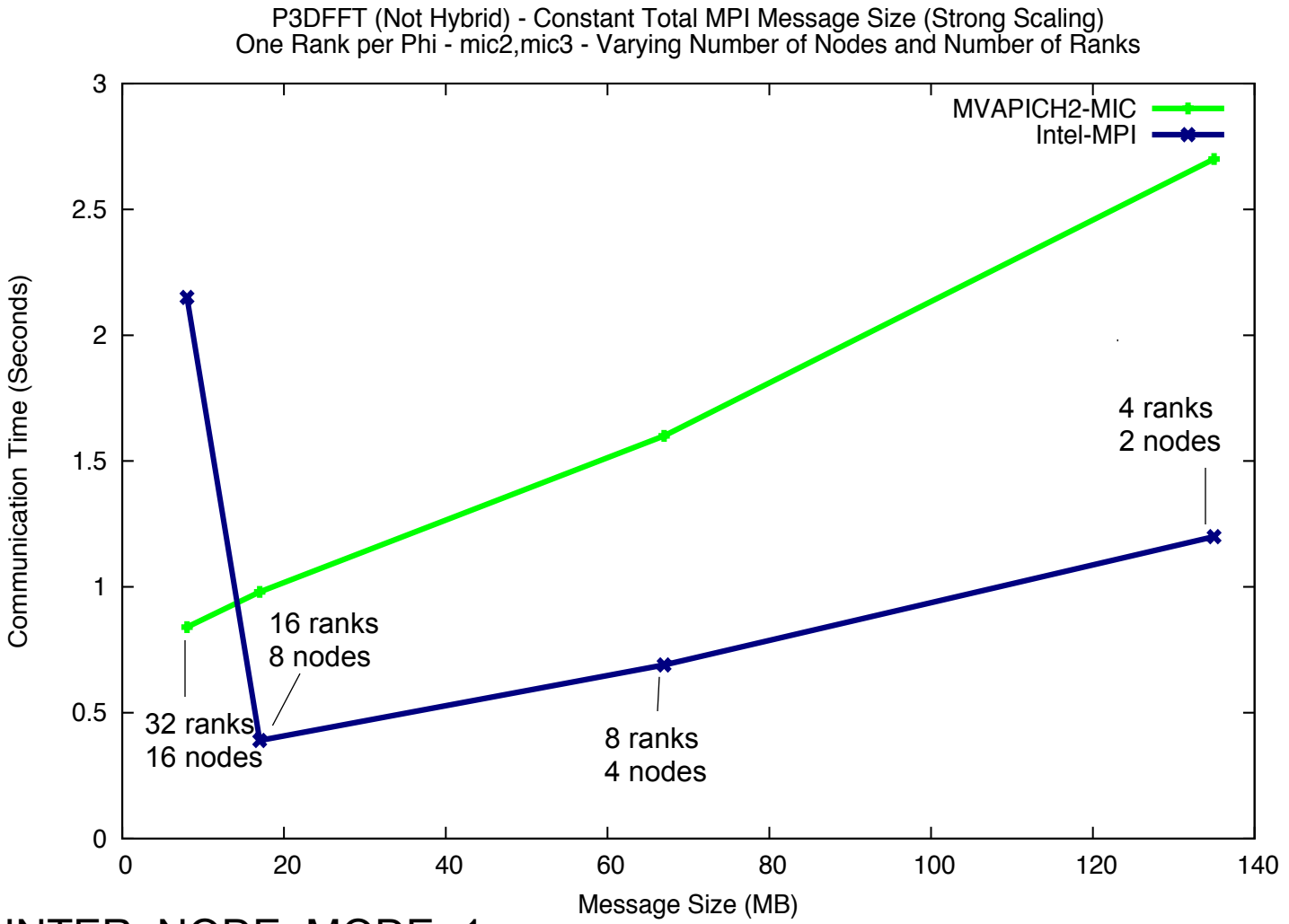
P3DFFT MPI Timing Comparison Two-Hop configuration

MV2-MIC ~65% faster for message size of 8MB

IMPI ~50% faster for 17MB messages

IMPI ~55% faster for 135MB messages

Fixed problem size with increasing ranks (2MIC's/ Node) from right to left



MV2_MIC_PROXY_INTER_NODE_MODE=1

Observations

- MVAPICH2-MIC performs exceptionally well on benchmarks
- MPI performance gains have been achieved for some applications under certain conditions
  - MPI buffer alignment with cache is required for MVAPICH2-MIC performance
  - Appropriate proxy settings must be chosen for a given architecture
  - Ranks/MIC must be kept as small as possible for best performance
- Several test applications did not run under certain conditions with our MVAPICH2-MIC implementation
  - P3DFFT worked Intra-MIC and with Inter-Node proxy=1 but not with Inter-Node proxy=2
  - mpiBLAST did not run for Inter-Node or Inter-MIC cases under MVAPICH2-MIC
  - GROMACS did not run Inter-Node or Inter-MIC under MVAPICH2-MIC

Conclusions

- MVAPICH2-MIC provides significant performance improvements under certain conditions
- Porting application code to achieve performance under MVAPICH2-MIC will probably require additional effort
  - Simply swapping modules and recompiling is not sufficient.
- IMPI works reliably with consistent performance, but it is slower than well-tuned MVAPICH2-MIC
- Unresolved issues with MVAPICH2-MIC on Beacon:
  - Switching between MV2_MIC_PROXY_INTER_NODE_MODE=1 or 2 dynamically is not supported.
  - Errors occur in internode mode for some applications.

Acknowledgements

- Raghu Chandrasekar, Khaled Hamidouche, and MVAPICH core developers – assistance installing, debugging, and using MVAPICH2-MIC on Beacon
- DK Panda – access to beta versions, support from his team, and invitation to speak today
- Intel – various forms of past and present support in deploying and operating Beacon efficiently

Contact Information

R. Glenn Brook, Ph.D.
Chief Technology Officer
Director, Application Acceleration Center of Excellence
Co-Director, Intel Parallel Computing Center
Joint Institute for Computational Sciences
University of Tennessee & Oak Ridge National Laboratory
glenn-brook@tennessee.edu