HPC·AI ADVISORY COUNCIL
NETWORK OF EXPERTISE

AMG
Performance Benchmarking and Profiling

May 2018

AMG

- **AMG is a parallel algebraic multigrid solver**
  - Linear systems arising from problems on unstructured grids
- **AMG is written in ISO standard C**
  - Leverages the LLNL hypre library of high-performance preconditioners
- **Parallelized with both MPI and OpenMP**
  - Setup phase mostly threaded
  - Solve phase fully threaded
- **For more information**
  - https://asc.llnl.gov/coral-2-benchmarks/downloads/AMG_Summary_v1_7.pdf
  - https://github.com/LLNL/AMG

Objectives

- **The following experiments were done to explore higher AMG productivity**
  - AMG performance benchmarking
  - Interconnect performance comparisons
  - Different MPI libraries performance comparisons
  - Understanding AMG profiling and communication patterns

Test Cluster Configuration

- **“Helios” cluster**
  - Sixteen Supermicro SYS-6029U-TR4 nodes and sixteen Foxconn nodes
  - Dual Socket Intel(R) Xeon(R) Gold 6138 CPU @ 2.00GHz
  - Mellanox ConnectX-5 EDR 100Gb/s InfiniBand/VPI adapters
  - Mellanox Switch-IB 2 SB7800 36-Port 100Gb/s EDR InfiniBand switches
  - Memory: 192GB DDR4 2677MHz RDIMMs per node
  - 1TB 7.2K RPM SSD 2.5" hard drive per node

Getting AMG

- **AMG is available at:**
  - https://asc.llnl.gov/coral-2-benchmarks/downloads/AMG-master-5.zip
  - It is self contained

- **Getting started:**
  - wget https://asc.llnl.gov/coral-2-benchmarks/downloads/AMG-master-5.zip
  - unzip AMG-master-5.zip
  - cd AMG-Master

Sample Build

- **Set up build environment:**
  - module load intel/2018.1.163 hpcx/2.1.0
- **Make adjustments to Makefile.include:**
  - For instance, add –x CORE-AVX512 to INCLUDE_CFLAGS for a Skylake build
  - Change –f openmp to –q openmp in both INCLUDE_CFLAGS and INCLUDE_LDFLAGS
- **Build the application**
  - make 2>&1 | tee make_i18h21.log
  - The executable, amg, is built under the test directory

Running AMG

- **AMG runs two types of problems; executing `./amg -help` gives a summary:**

```
Usage: ./amg [<options>]

  -problem <ID>: problem ID
       1 = solves 1 large problem with AMG-PCG (default)
       2 = simulates a time-dependent loop with AMG-GMRES

  -n <nx> <ny> <nz>: problem size per MPI process (default: nx=ny=nz=10)

  -P <px> <py> <pz>: processor topology (default: px=py=pz=1)

  -print       : prints the system
  -printstats  : prints preconditioning and convergence stats
  -printallstats  : prints preconditioning and convergence stats
                    including residual norms for each iteration
```

AMG Problem 1

- **Uses a conjugate gradient solver preconditioned with AMG to solve a linear system with a 3D 27-point stencil**
  - Problem size is `nx*ny*nz*Px*Py*Pz`

- **The program computes three measures of performance, called "Figures of Merit" (FOM)**
  - FOM_Setup: measures the performance of the preconditioner set-up phase
  - FOM_Solve: measures the performance of the solution phase
  - FOM_1: is a weighted average of the above two measures

- **The program also writes a "Final Relative Residual Norm" for verification; it should be smaller than `1.0e-08`**

AMG Problem 2

- **Simulates a time-dependent problem with AMG-GMRES, using a 3D 27-point stencil**
  - Problem size is nx*ny*nz*Px*Py*Pz

- **The program computes a single measure of performance (“Figure of Merit”, or FOM)**
  - FOM_2: computed from the number of iterations across all time steps, the number of time steps (fixed and equal to 6) and the total wall clock time

- **The program also writes a “Final Relative Residual Norm” for verification; it should be smaller than 1.0e-10**

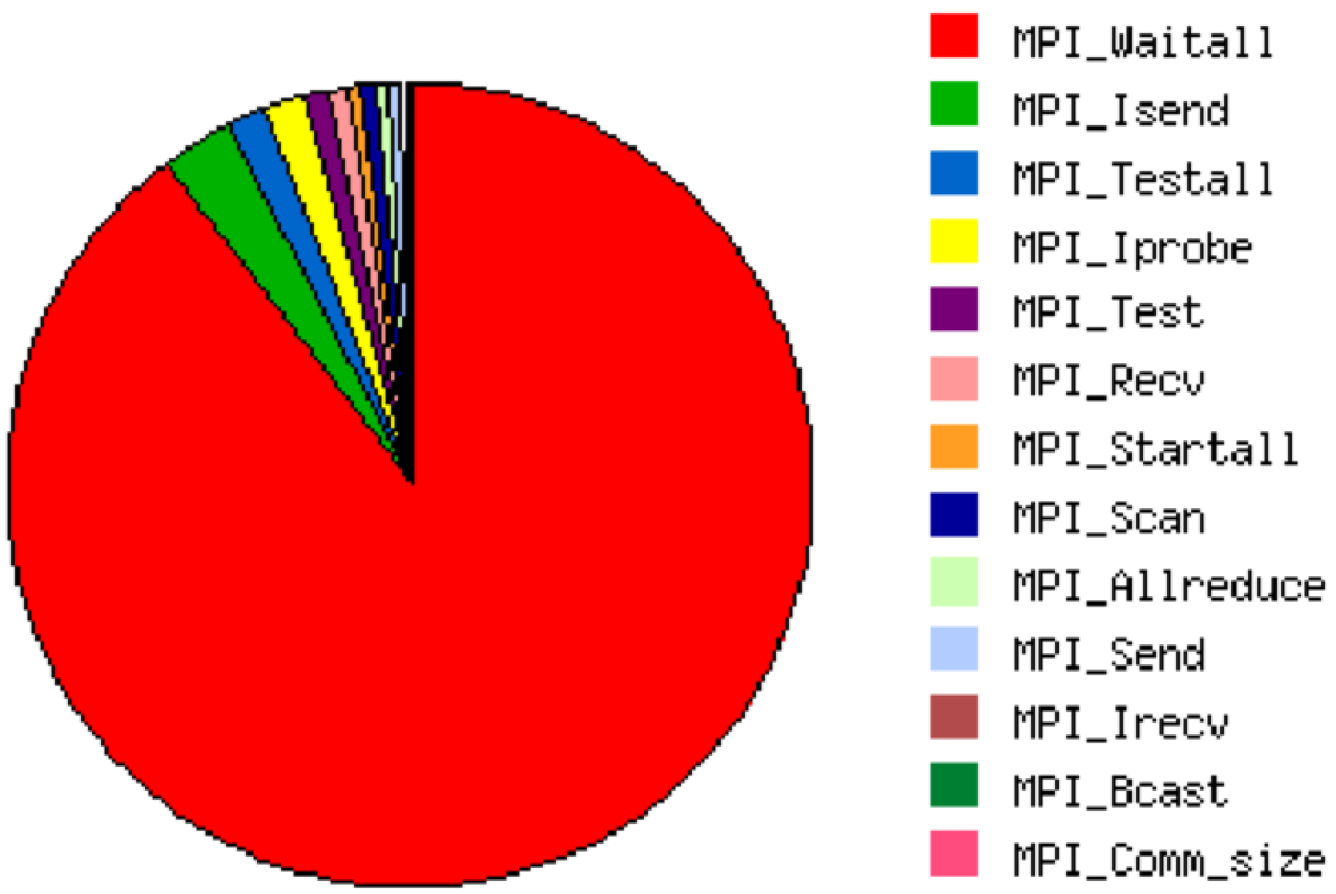
Final Output Samples (640 MPI ranks, 2 OpenMP threads)

```
=============================================
Problem 1: AMG Setup Time:
=============================================
PCG Setup:
  wall clock time = 26.794626 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 26.810000 seconds
  cpu MFLOPS      = 0.000000

FOM_Setup: nnz_AP / Setup Phase Time: 6.221523e+08

=============================================
Problem 1: AMG-PCG Solve Time:
=============================================
PCG Solve:
  wall clock time = 61.392113 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 61.400000 seconds
  cpu MFLOPS      = 0.000000

Iterations = 24
Final Relative Residual Norm = 9.453082e-09

FOM_Solve: nnz_AP * Iterations / Solve Phase Time: 6.516931e+09

Figure of Merit (FOM_1): 5.043236e+09
```

```
=============================================
Problem 2: Cumulative AMG-GMRES Solve Time:
=============================================
GMRES Solve:
  wall clock time = 313.561779 seconds
  wall MFLOPS     = 0.000000
  cpu clock time  = 313.570000 seconds
  cpu MFLOPS      = 0.000000


No. of Time Steps = 6
Cum. No. of Iterations = 215
Final Relative Residual Norm = 1.610531e-14


nnz AP * (Iterations + time_steps) / Total Time:

Figure of Merit (FOM_2): 3.448595e+07
```

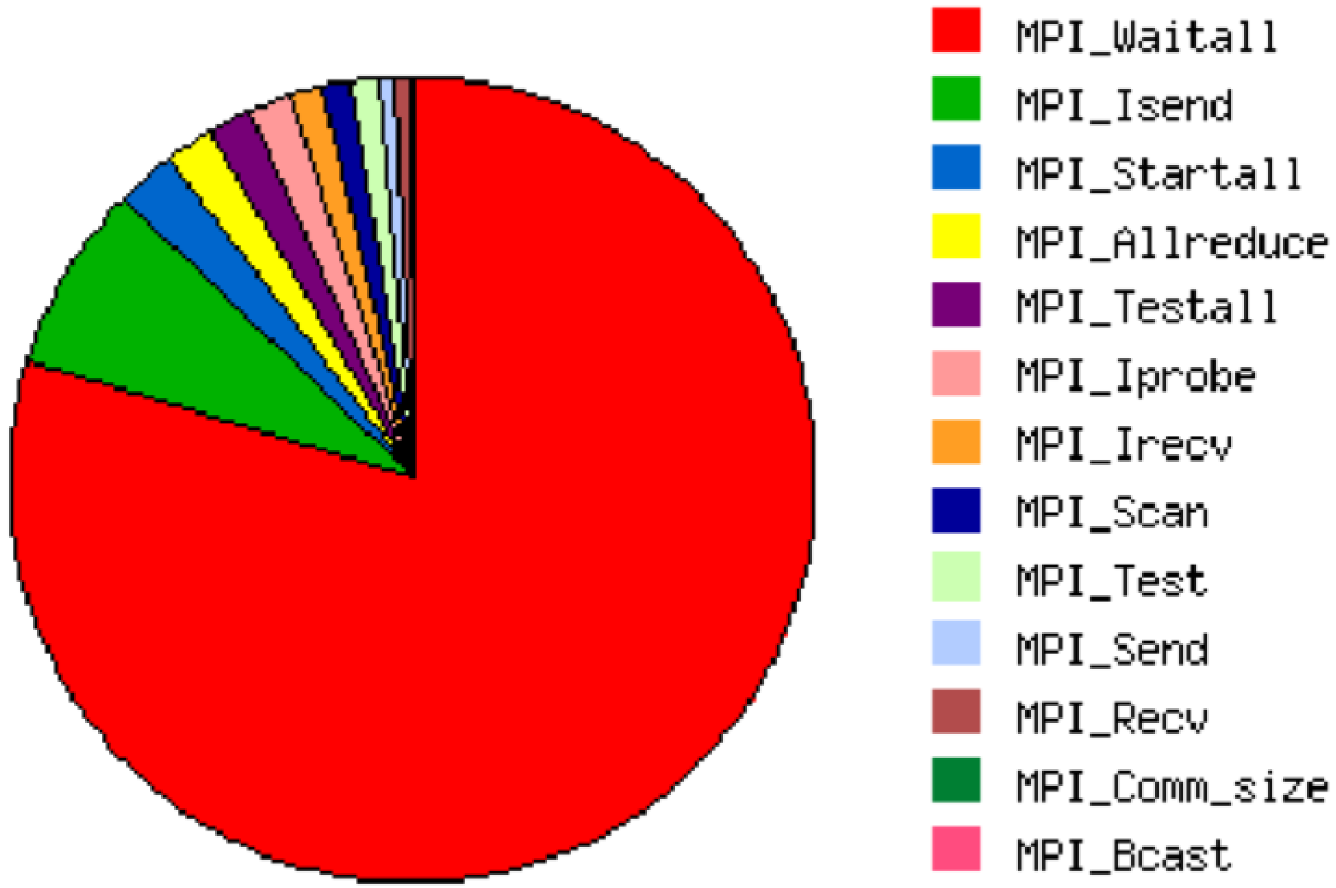
MPI Profiles with 512 MPI ranks (-P 8 8 8), 2 OpenMP Threads

Problem 1 (-n 96 96 96 -P 8 8 8)



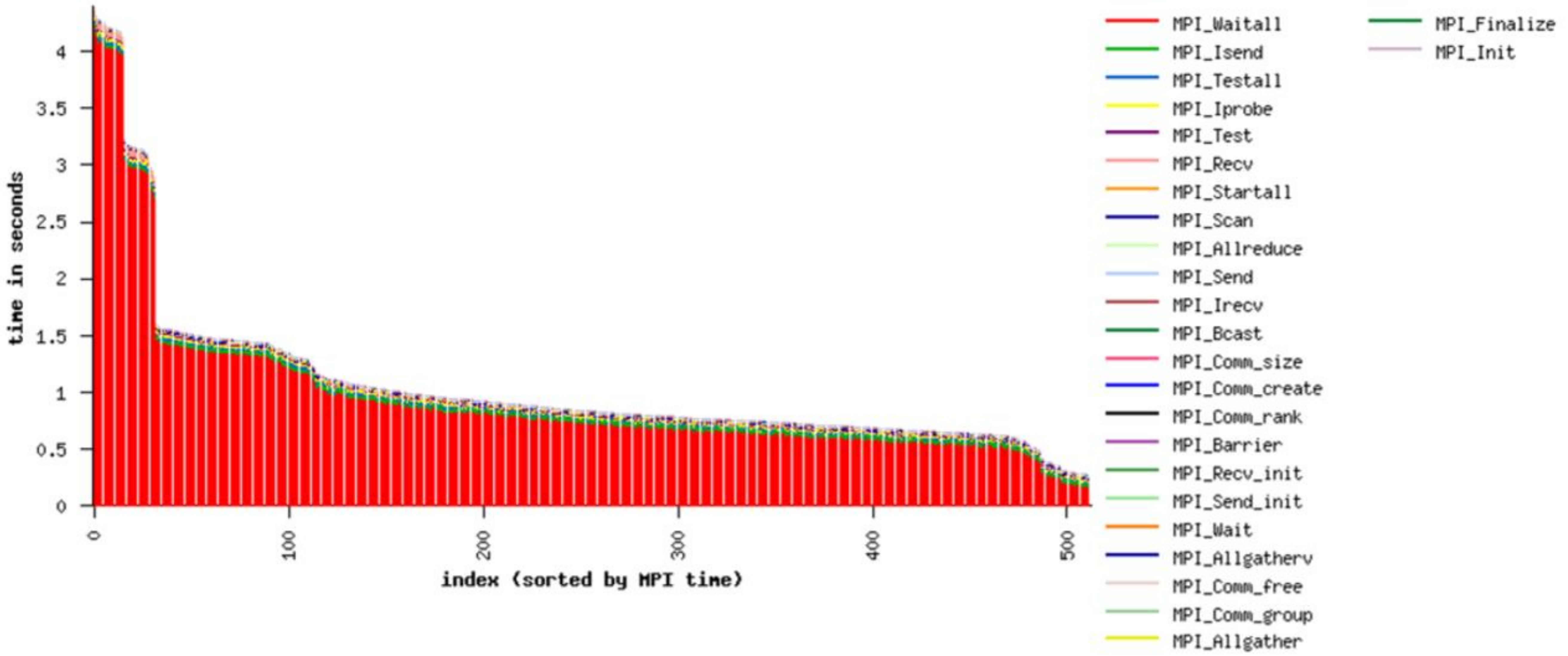
Time spent in MPI was 5.4% of total wall clock time

Problem 2 (-n 40 40 40 -P 8 8 8)

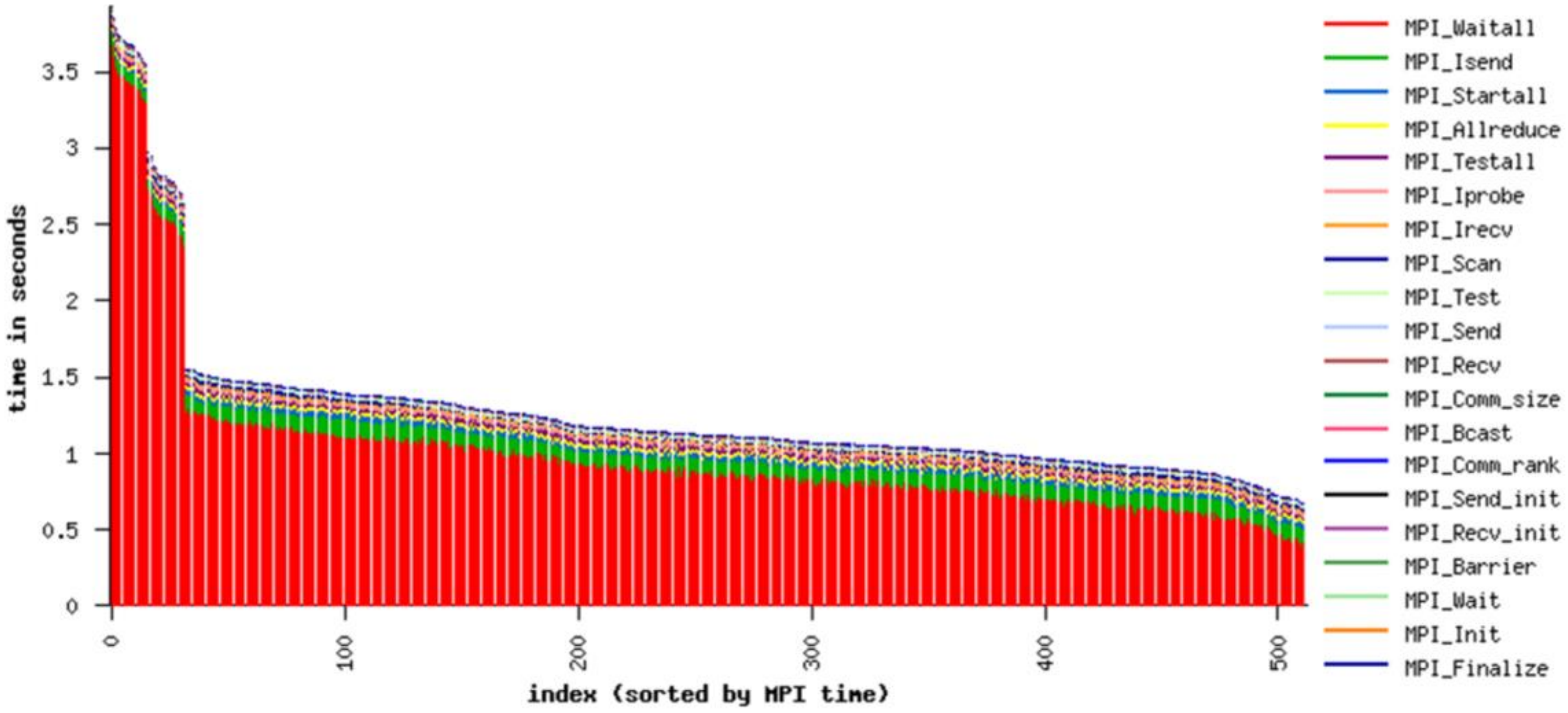


Time spent in MPI was 15.56% of total wall clock time

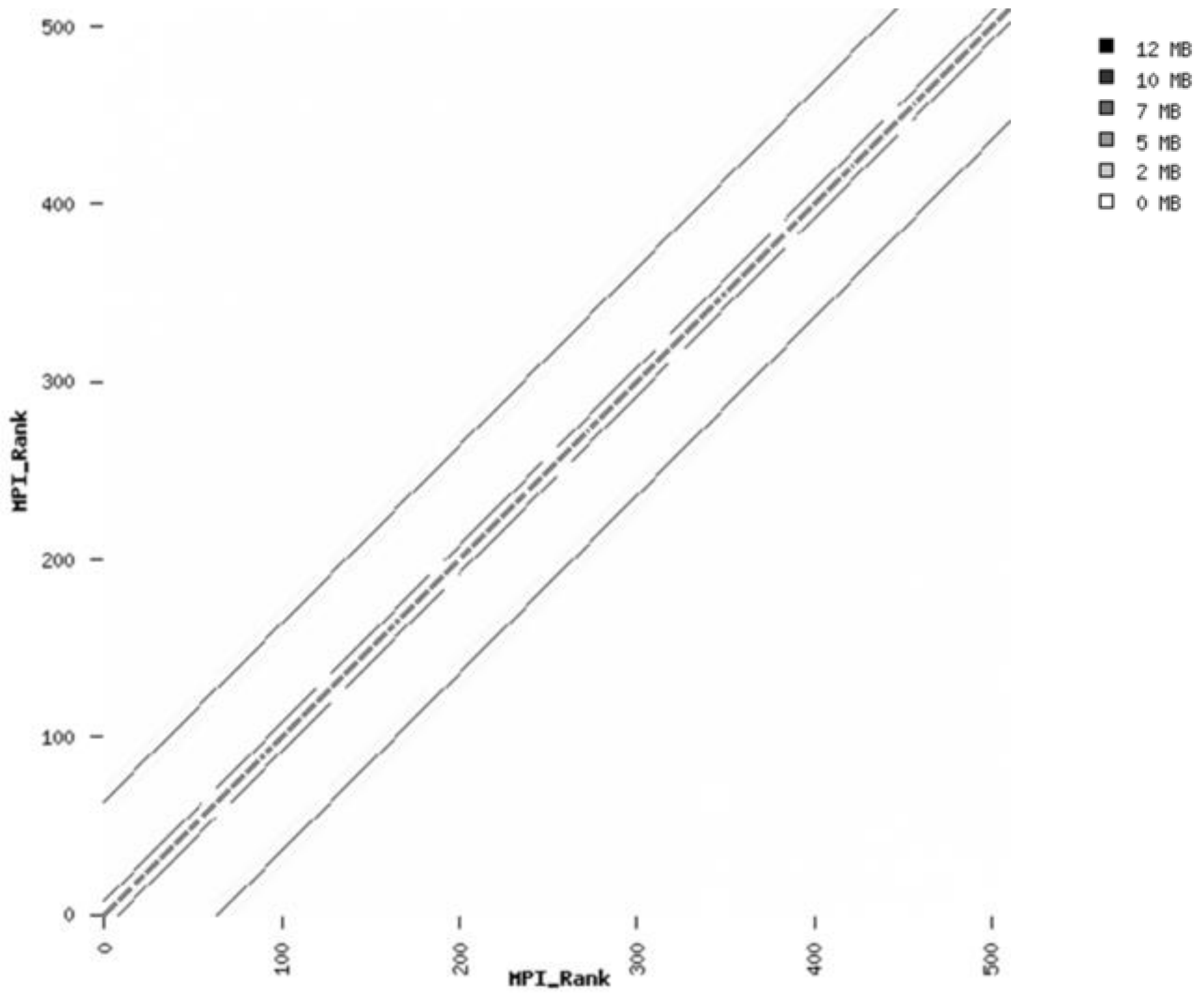
Load Balance Sorted by MPI Rank, Problem 1



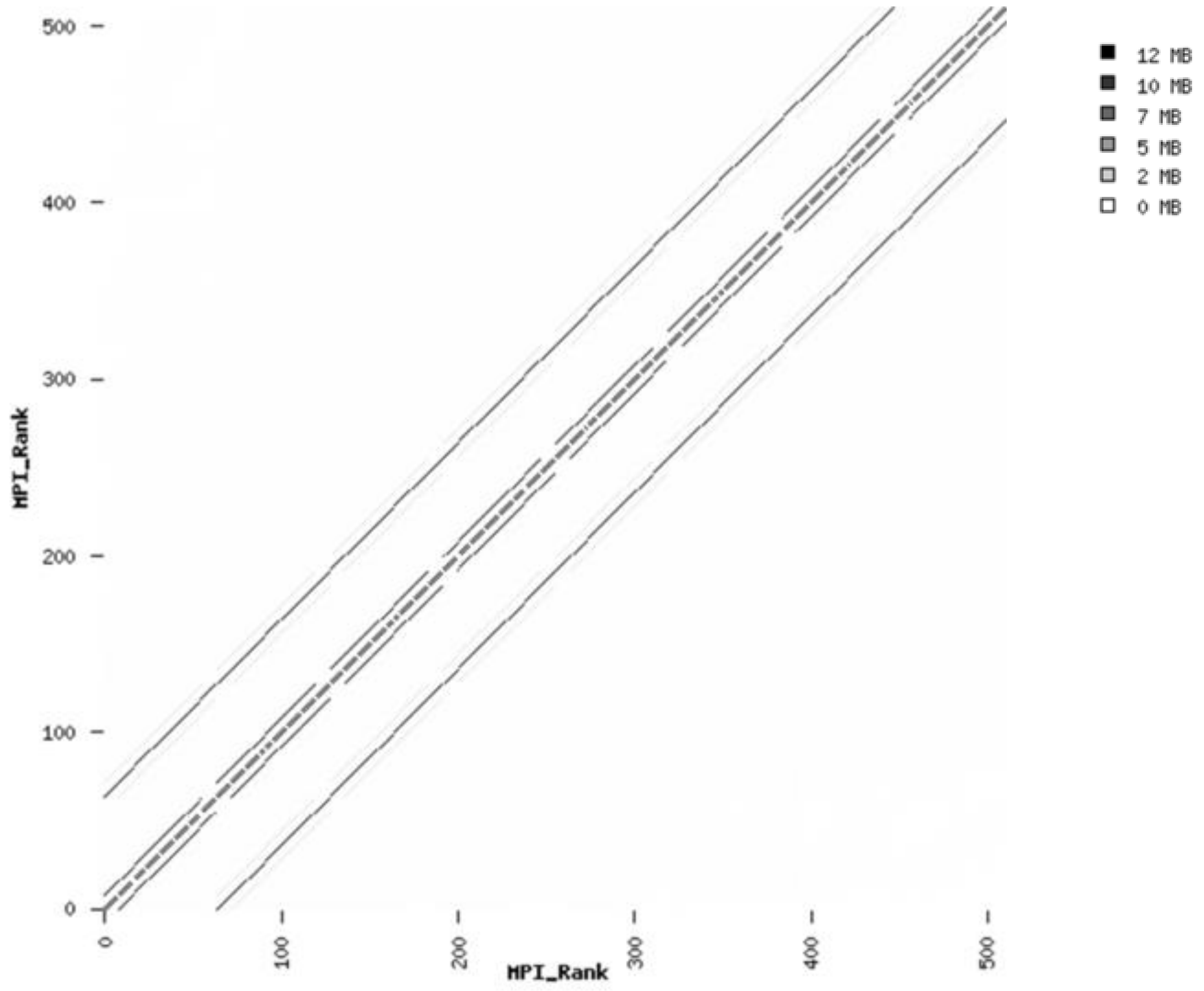
Load Balance Sorted by MPI Rank, Problem 2



Communication Topology (Point to Point, Data Sent)



Problem 1



Problem 2

Performance Measurements

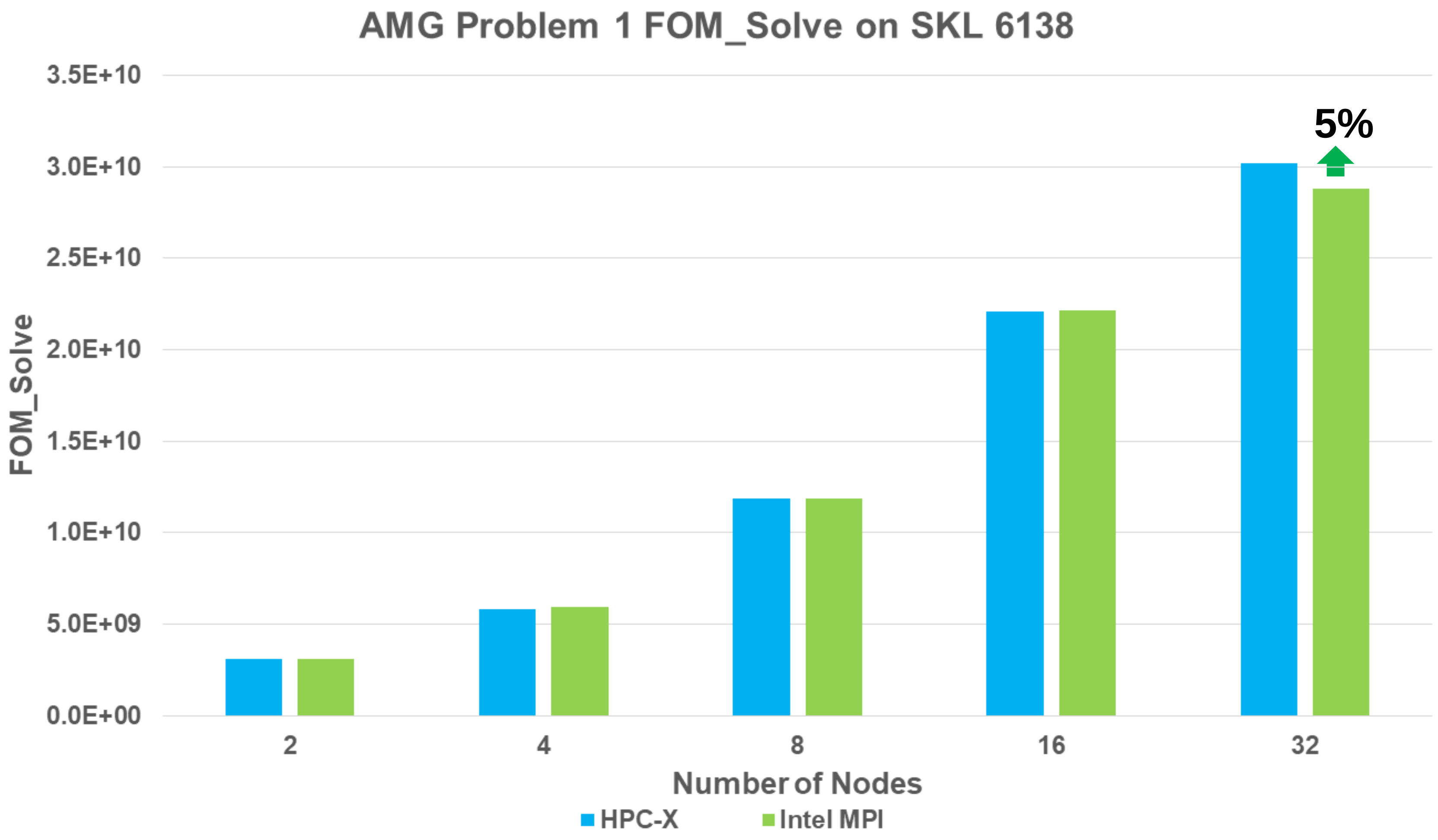
- **In the next few slides, we made several tests comparing the following:**

- **MPI layers**
  - HPC-X 2.1
  - Intel MPI

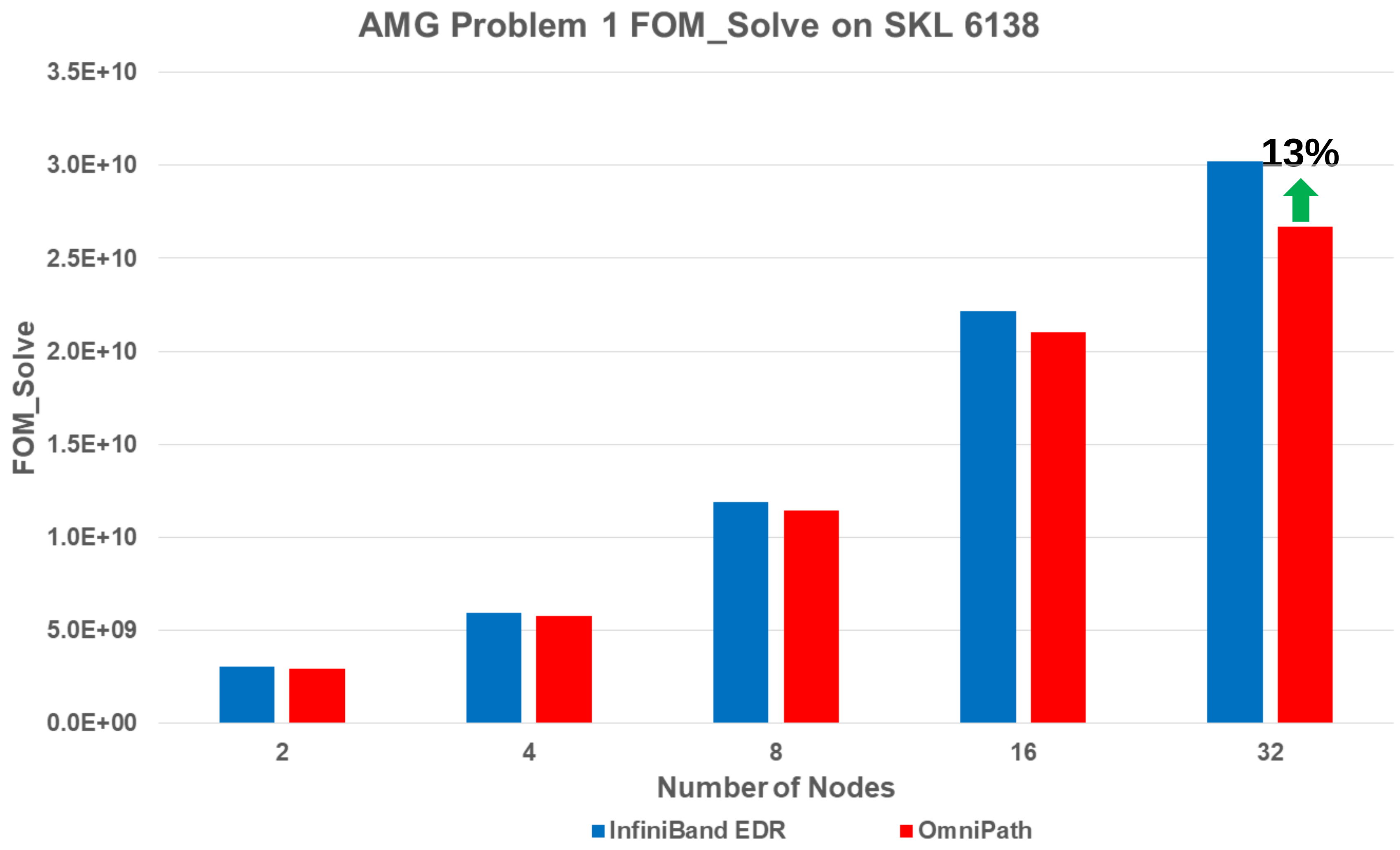
- **Interconnect Technology**
  - InfiniBand EDR (ConnectX-5, Switch-IB2)
  - Intel OPA

- **For both Problem 1 and Problem 2**

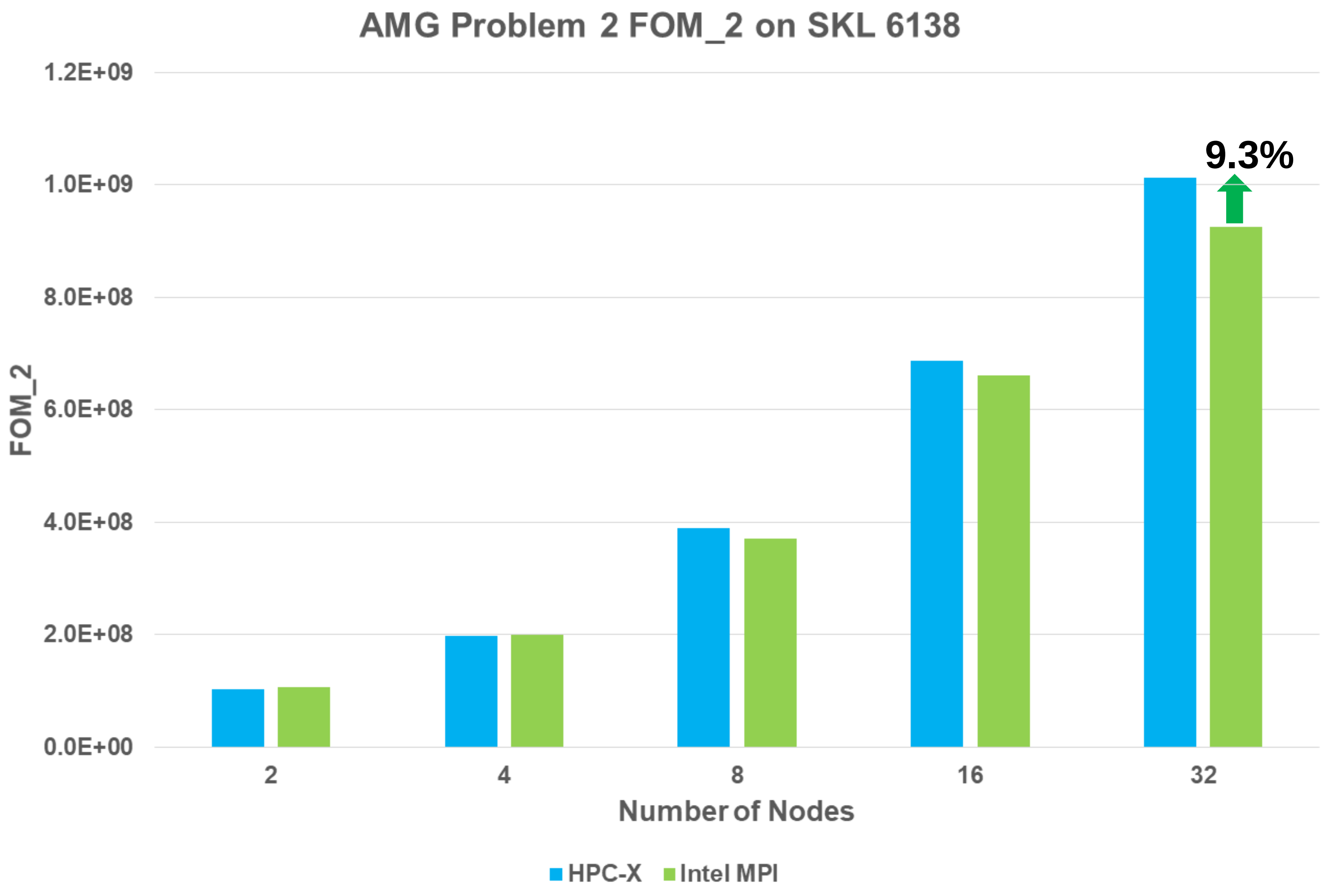
MPI Libraries Performance with Problem 1 (InfiniBand)



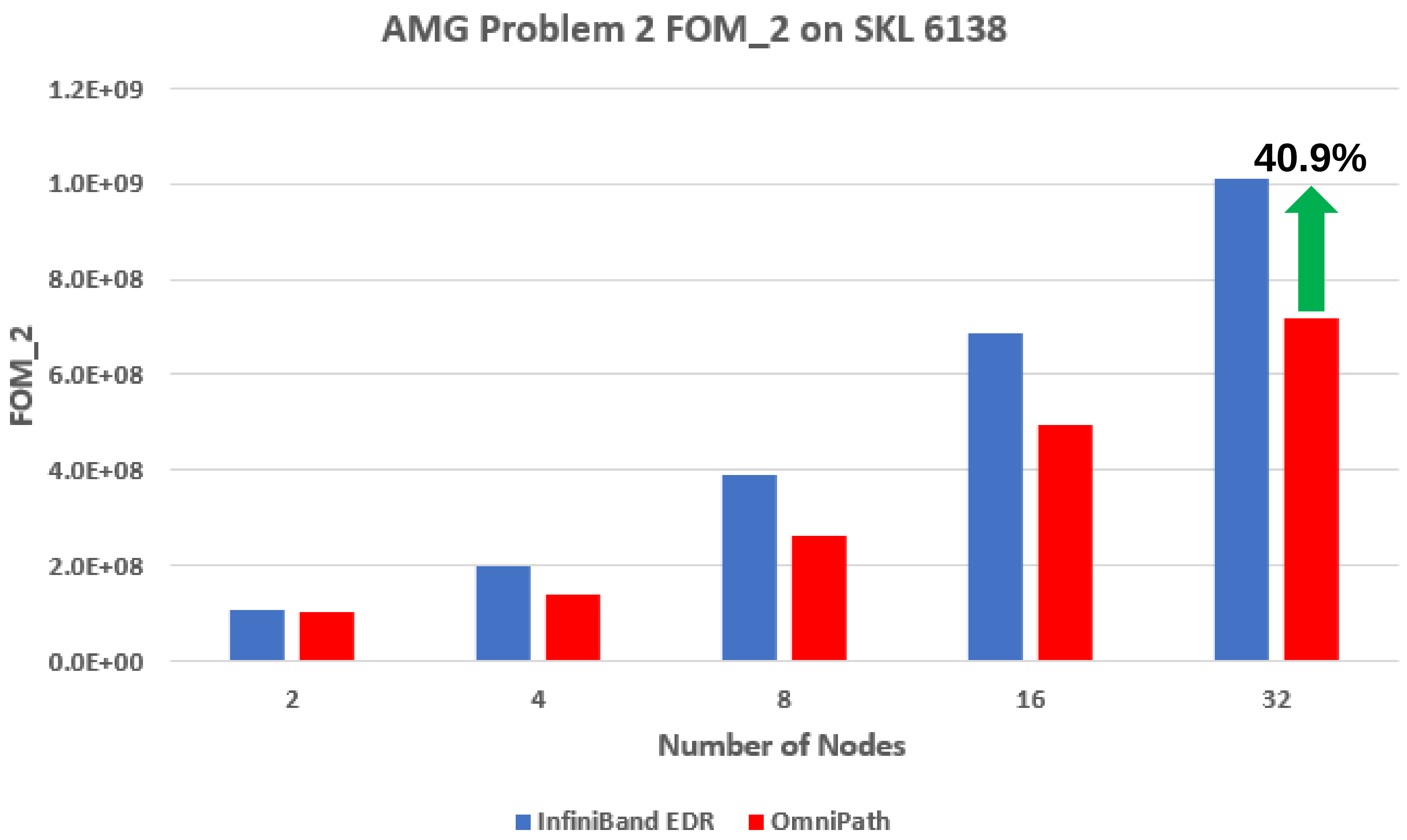
Interconnect Performance with Problem 1



MPI Libraries Performance with Problem 2 (InfiniBand)



Interconnect Performance with Problem 2



Observations

- **Profiling**
  - Mostly MPI_Waitall was used for both problems, due to the unbalanced application
- **HPC-X 2.1 provides higher performance, especially at larger core counts**
  - By 5% on Problem 1 at 1280 cores
  - By 9.3% on Problem 2 at 1280 cores
- **InfiniBand provides higher performance versus OmniPath**
  - By 13% on Problem 1 at 1280 cores
  - By 40.9% on Problem 2 at 1280 cores

Thank You