

Performance Benchmarks and Tuning on SM-3

Le Yan



CENTER FOR COMPUTATION
& TECHNOLOGY

Objectives

- Show application performance benchmarks on SuperMike-3
- Show methods that may improve the performance

Disclaimers

- Target audience: users who run (and sometimes compile) applications developed by others
 - Not an in-depth guide for programmers and developers

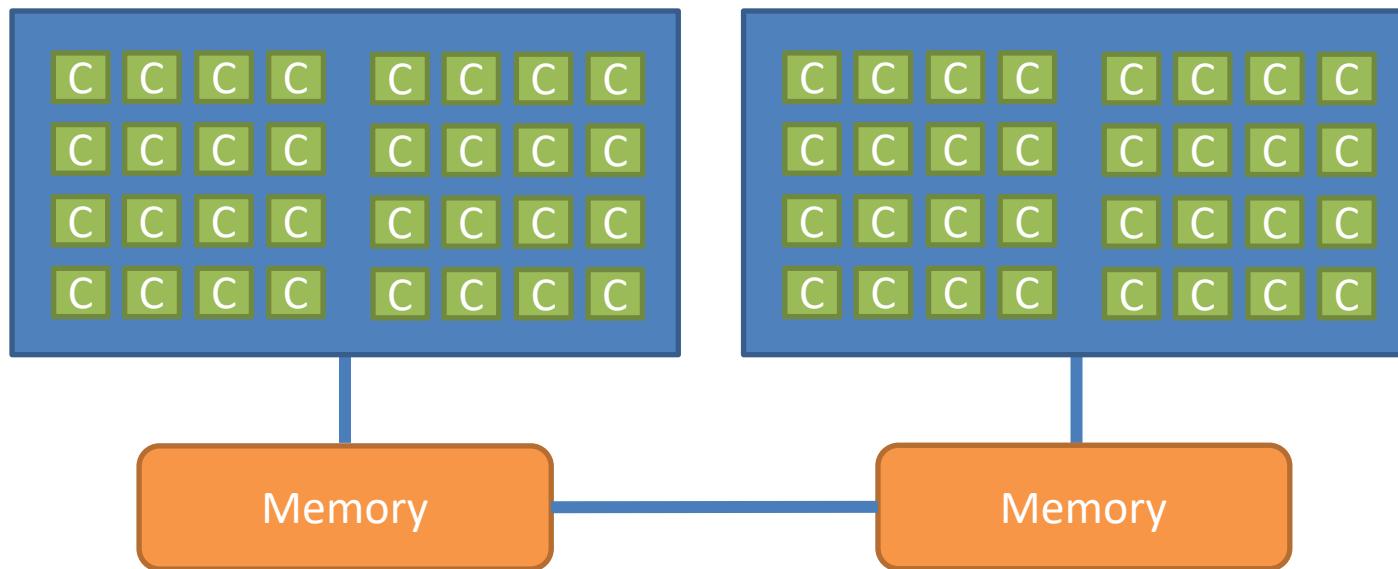
Outline

- Summary of SuperMike-3 architecture
- Single node performance
- Multi-node performance

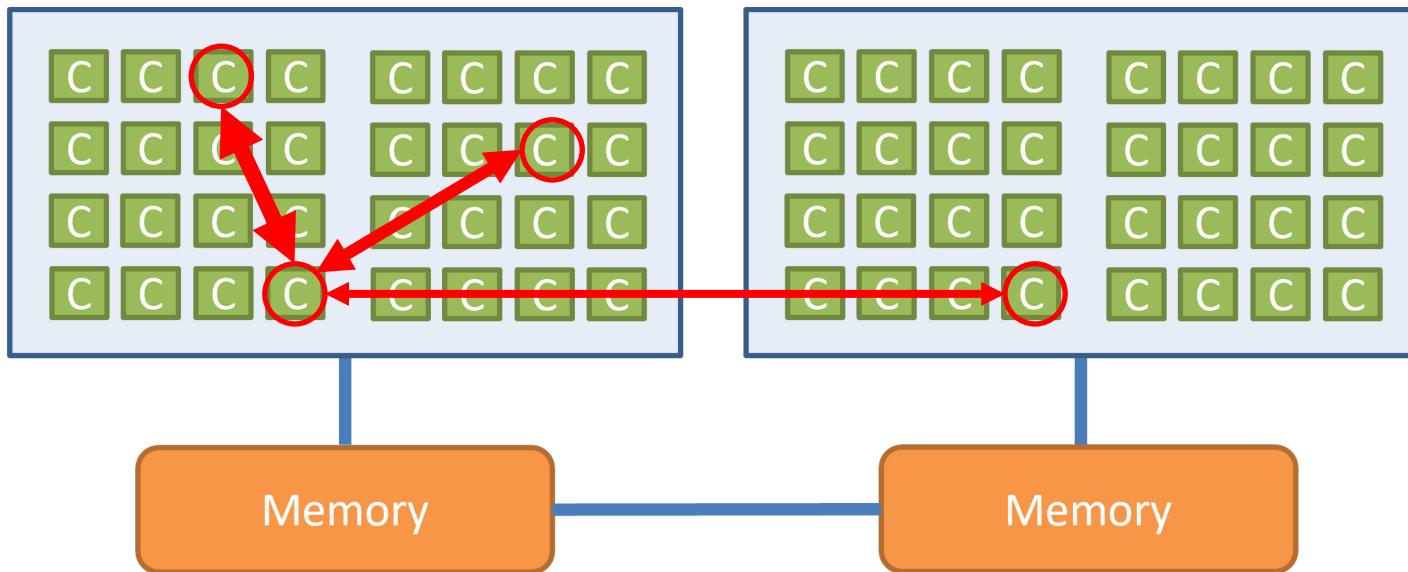
Summary of SM-3 architecture

SM-3 Architecture

Node level view



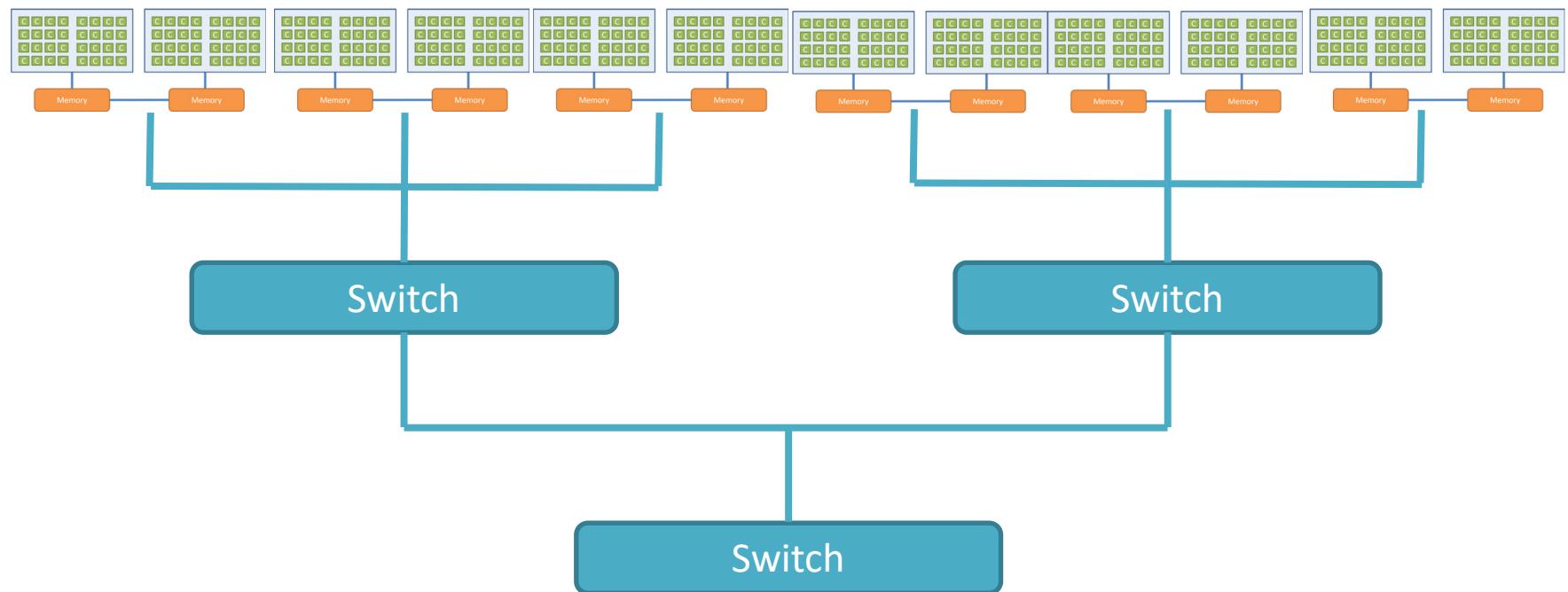
Thread Affinity



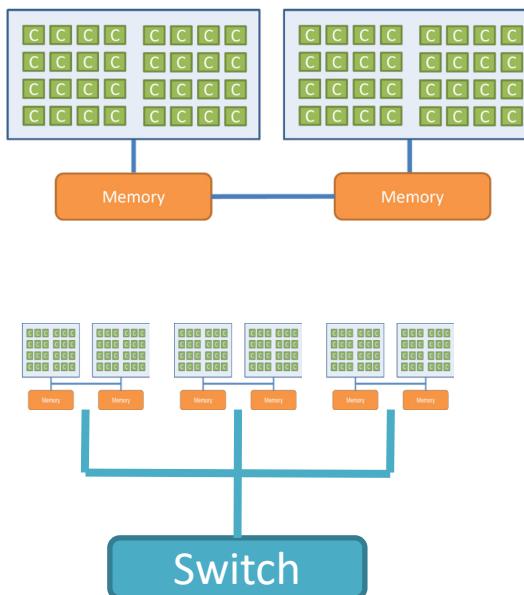
- The 64 cores on a SM-3 node are grouped into 4 sets.
- The data exchange cost is not homogeneous.
- Depending on the data exchange pattern, how the threads are arranged could affect performance significantly.

SM-3 Architecture

Cluster level view



Parallel Paradigms



Pro

- Low latency, high bandwidth
- Implicit communication
- Fine granularity
- Dynamic load balancing

Con

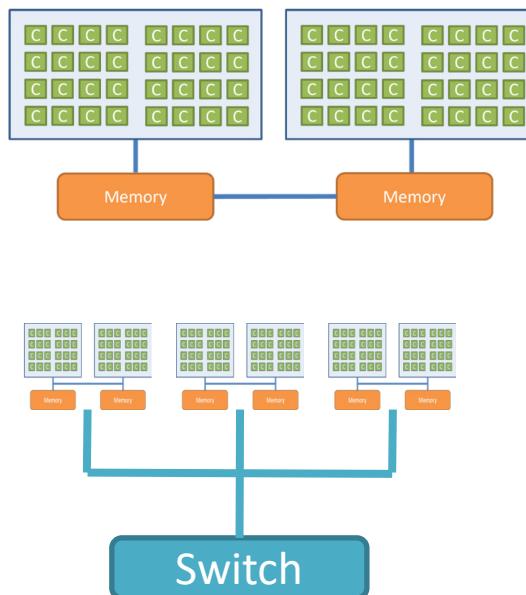
- Shared memory system only (Limited to one node)

Intranode

Internode

- Scalability beyond 1 node
- High latency, low bandwidth
- Explicit communication
- Hard to load balancing

Parallel Paradigms



Pro

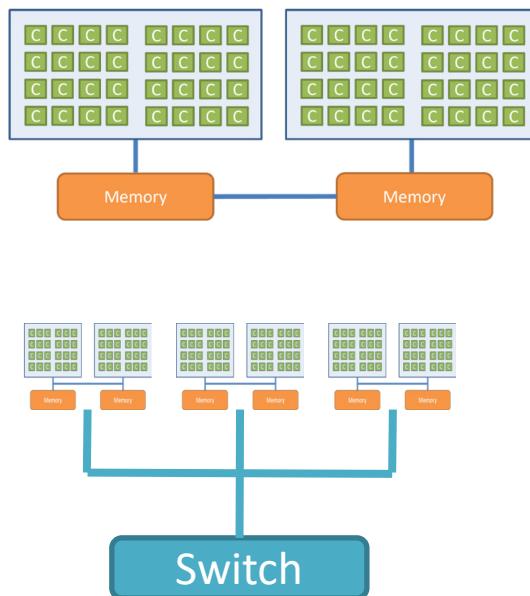
Con

- Intranode
- Low latency, high bandwidth
 - Implicit synchronization
 - Shared memory system
- OpenMP (Multi-thread)**
- Fine granularity
 - Dynamic load balancing

- Internode
- Scalability beyond node boundaries
 - High latency, low bandwidth
 - Hard to load balancing
- MPI (Multi-process)**

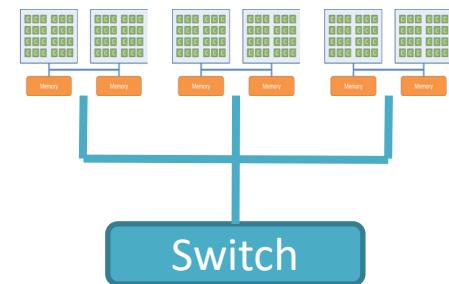
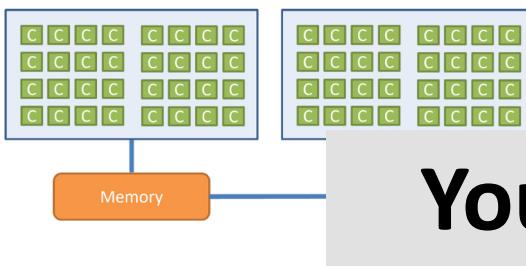
What About MPI+OpenMP Hybrid?

- Getting the benefits from both worlds?
- In theory, yes
- But adding OpenMP to (well-written) MPI programs might hurt the performance
- Hybrid helps to
 - Reduce memory footprint
 - Extend scalability



What About MPI+OpenMP Hybrid?

- Getting the benefits from both worlds?
- In theory, yes



Your mileage may vary! (well-written, well-optimized programs might hurt the performance)

- Hybrid helps to
 - Reduce memory footprint
 - Extend scalability

Single Node Performance

SM-3 Specification

CPU Intel Ice Lake (Xeon Platinum 8358)
(2 sockets *32 cores/socket)

CPU frequency 2.6 G Hz

Floating operation per clock cycle (double precision) 32

Memory 256 GB DDR4

Interconnect Mellanox 200 Gbps Infiniband

SM-3 vs SuperMIC (Node over Node)

	SM-3	SuperMIC
CPU frequency	2.6×10^9	2.8×10^9
CPU cores	64	20
Operation per cycle	32	8
Memory bandwidth	~205 GB/s	~48 GB/s
Interconnect	200 Gbps	56 Gbps

Node (theoretical) peak performance

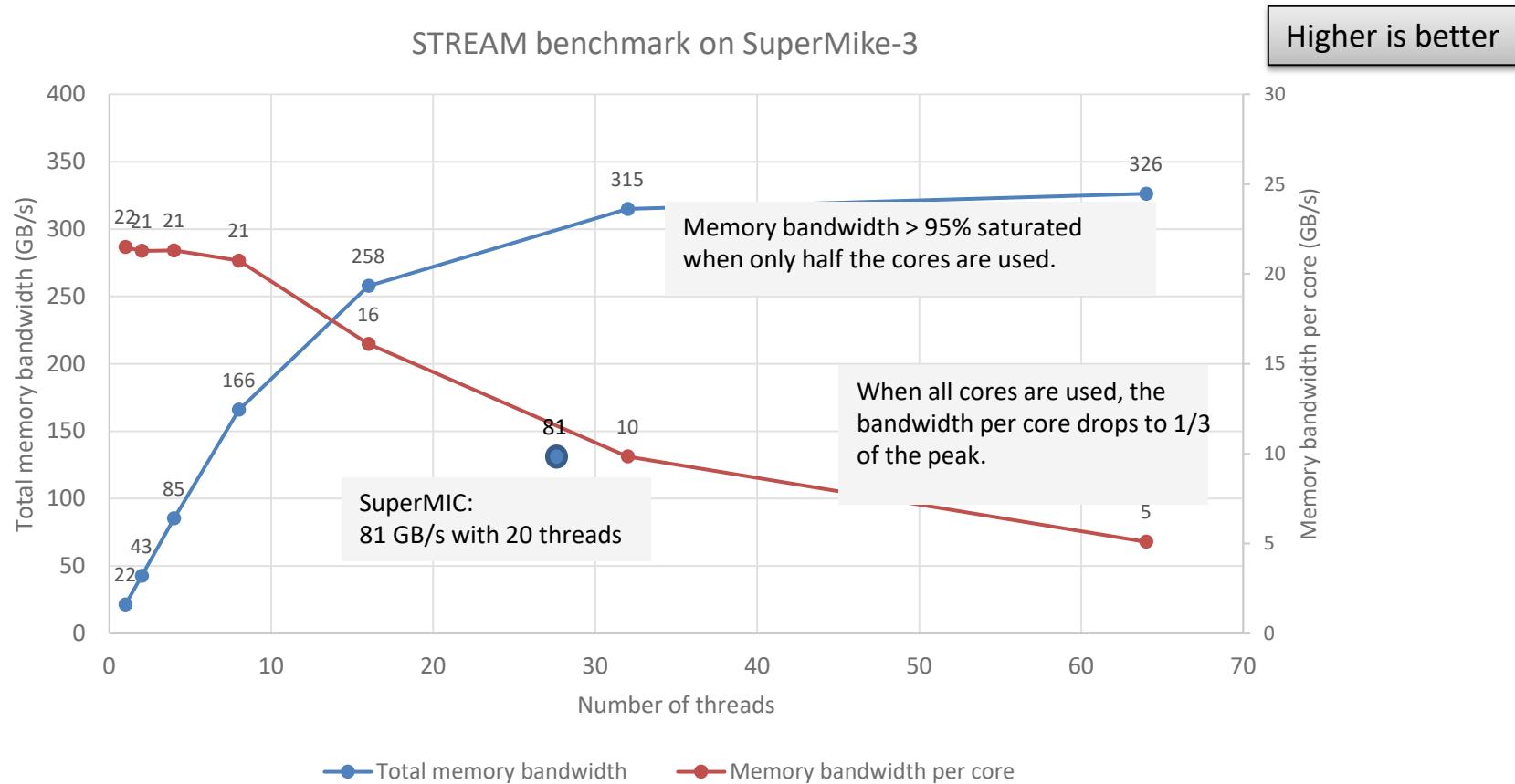
$$\text{SM-3: } 64 \text{ cores/node} * 2.6 \times 10^9 \text{ cycles/second} * 32 \text{ flop/cycle} = 5.32 \times 10^{12} \text{ flops}$$

$$\text{SuperMIC: } 20 \text{ cores/node} * 2.8 \times 10^9 \text{ cycles/second} * 8 \text{ flop/cycle} = 0.45 \times 10^{12} \text{ flops}$$

STREAM Benchmark

- The de facto industry standard benchmark in HPC domain for the measurement of sustainable **memory bandwidth** (in GB/s).

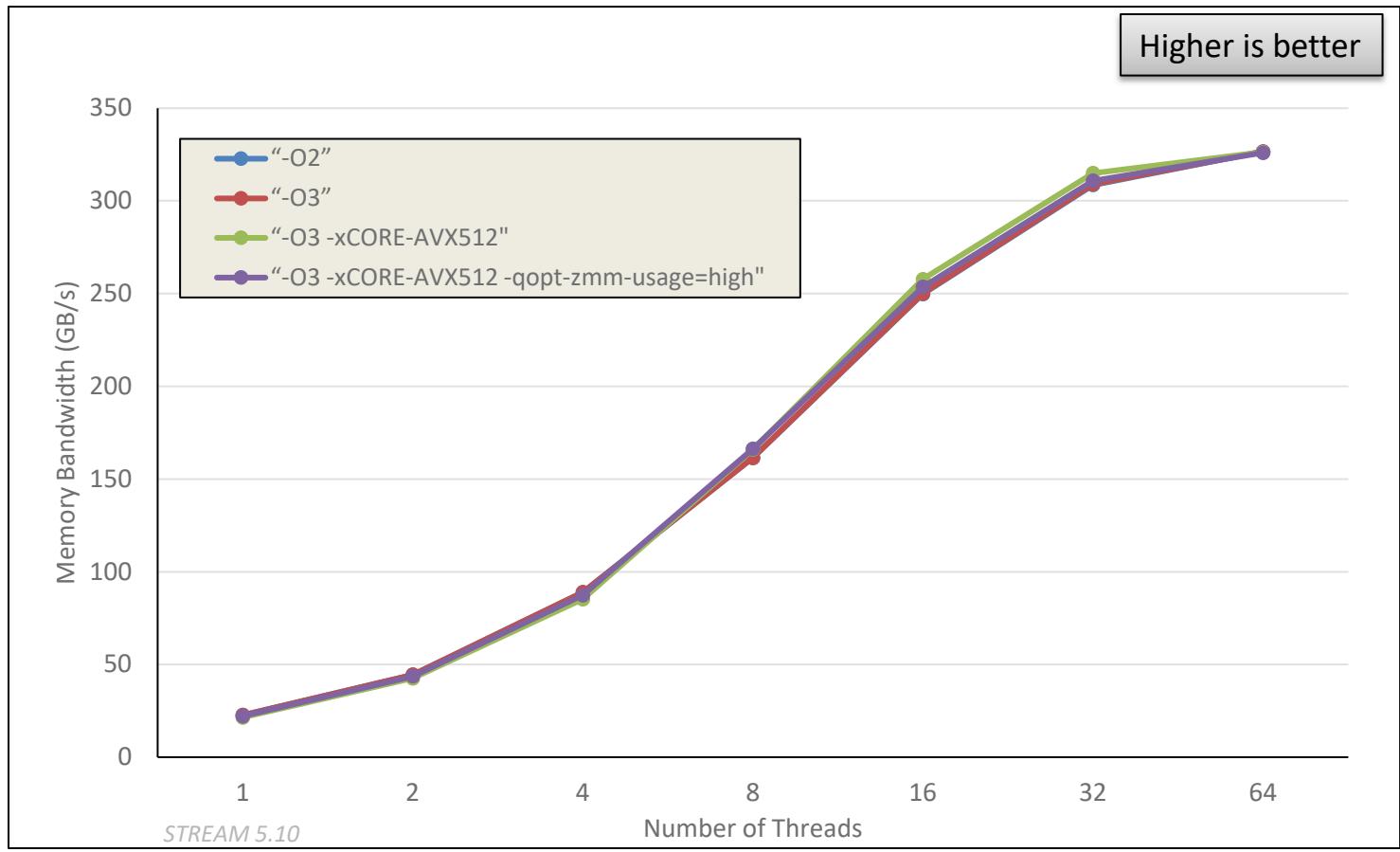
STREAM Benchmark - Results



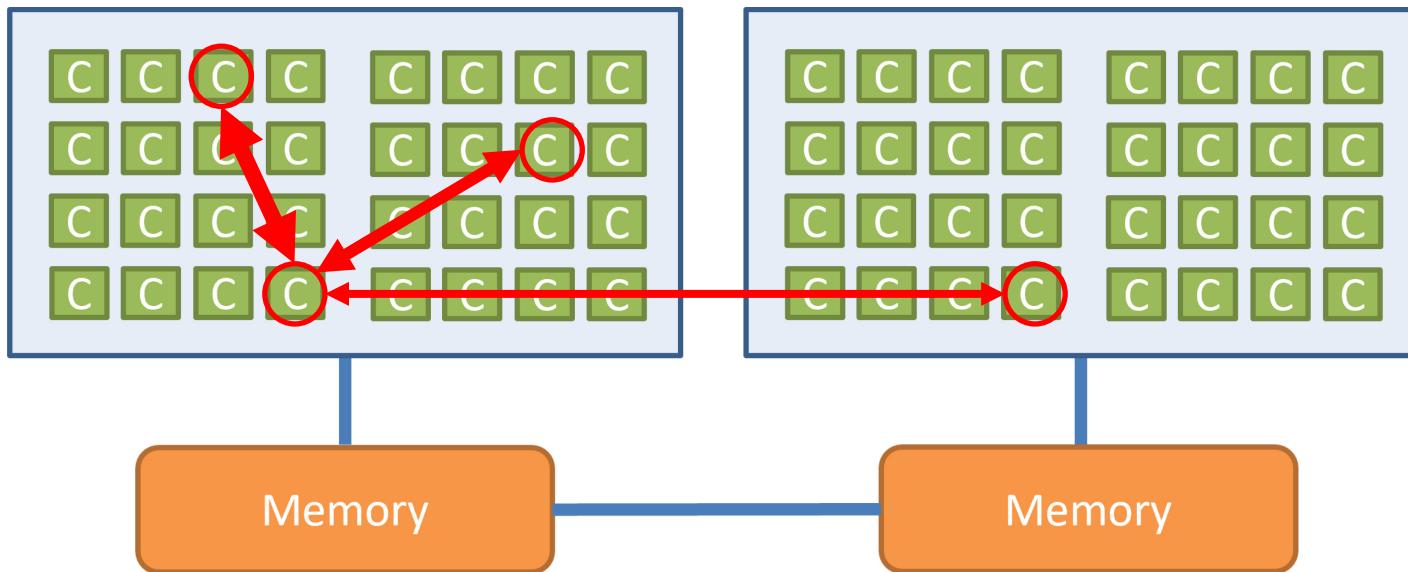
Compiler Flags (Intel)

- `-O2, -O3`
 - Generic, aggregated optimization flags
- `-xCORE-AVX512`
 - Turns on optimization for the Ice Lake (and SkyLake/Cascade Lake) processor
- `-march=icelake-server`
 - Turns on optimization specifically for Ice Lake
- `-qopt-zmm-usage=high`
 - Improves performance for some code

STREAM – Compiler Flags

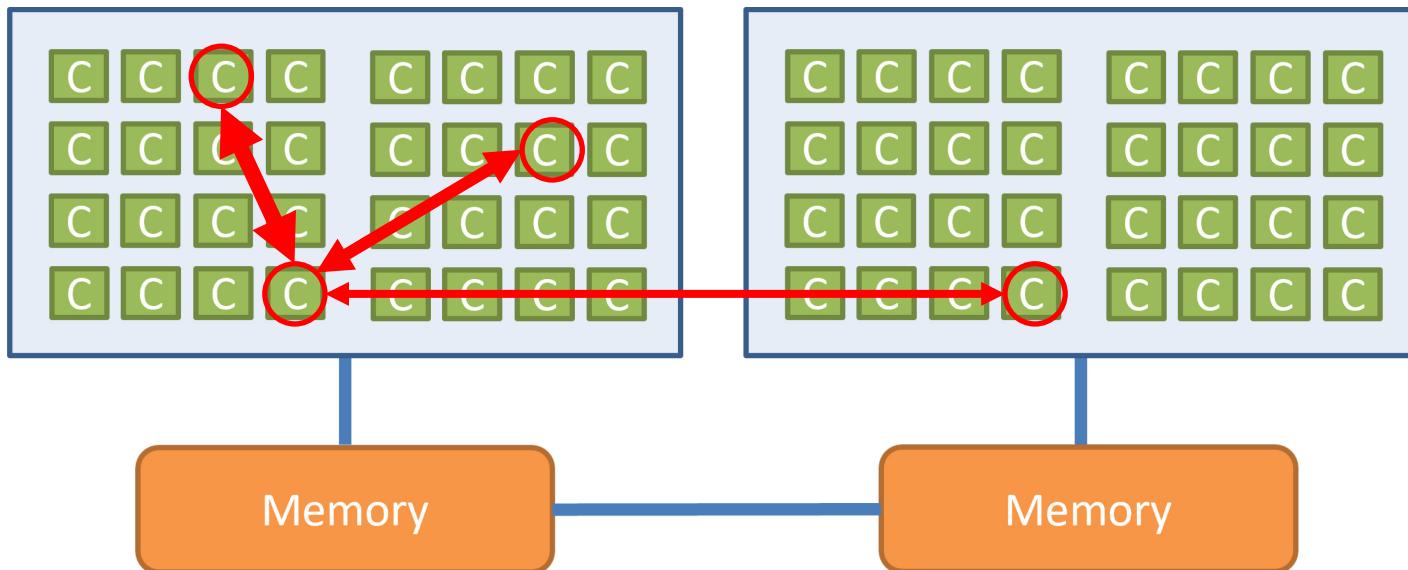


Thread Affinity



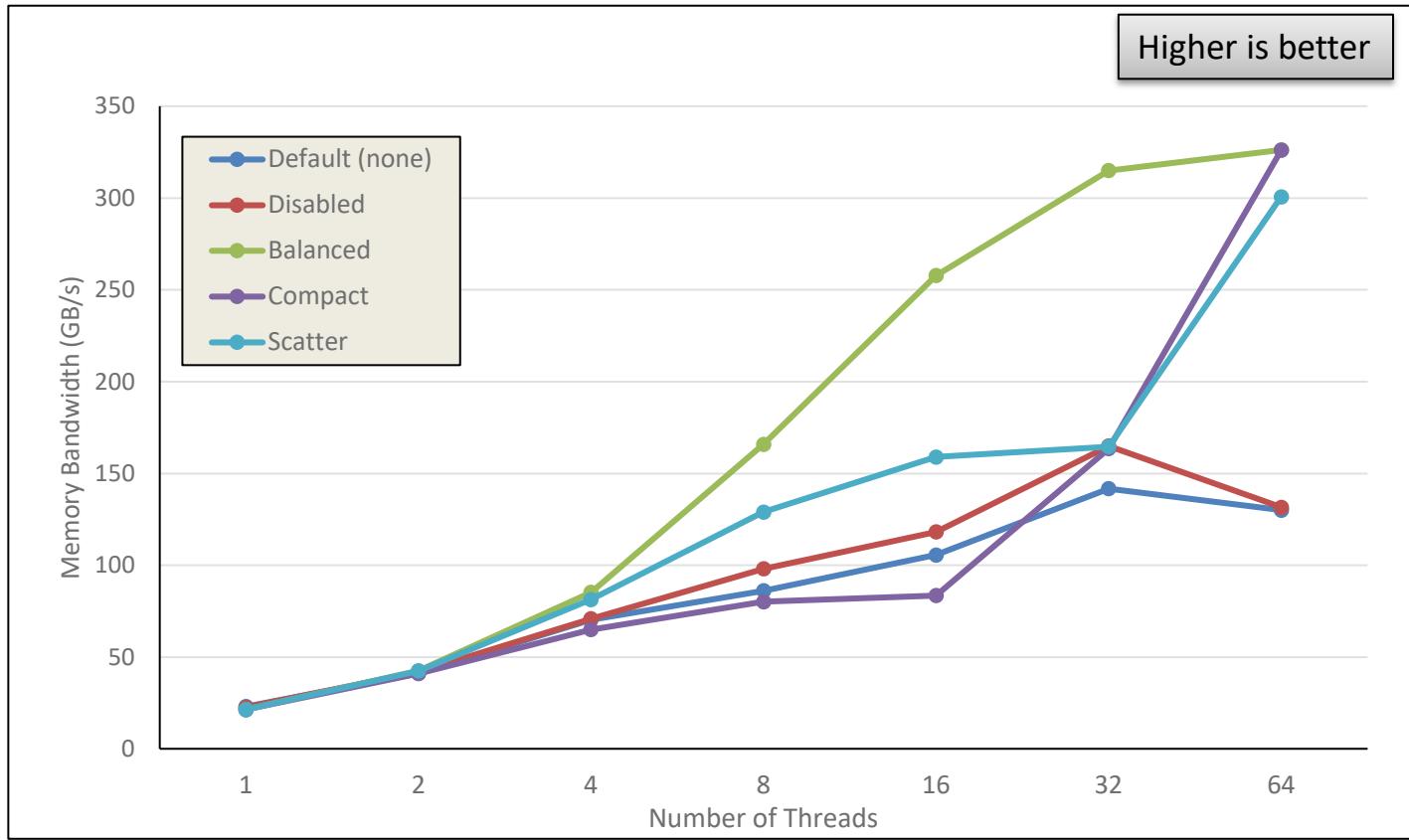
- The 64 cores on a SM-3 node are grouped into 4 sets.
- The data exchange cost is not homogeneous.
- Depending on the data exchange pattern, how the threads are arranged could affect performance significantly.

Thread Affinity



- Use the `KMP_AFFINITY` environment variable to control thread placement/affinity
- The options are “none” (default), “disabled”, “balanced”, “compact”, and “scatter”.

STREAM – Thread Affinity



STREAM 5.10
Array size = 4000 MB
Intel 2021.5.1 with “-O3 -xCORE-AVX512”

HPL Benchmark

- High Performance Linpack
 - Standard benchmark for **CPU-bound** HPC applications
- Results
 - SM-3: 3800 GFLOPS per node
 - SuperMIC: 424 GFLOPS per node
 - Speedup = 9.0 (compared to 11.9 theoretical)

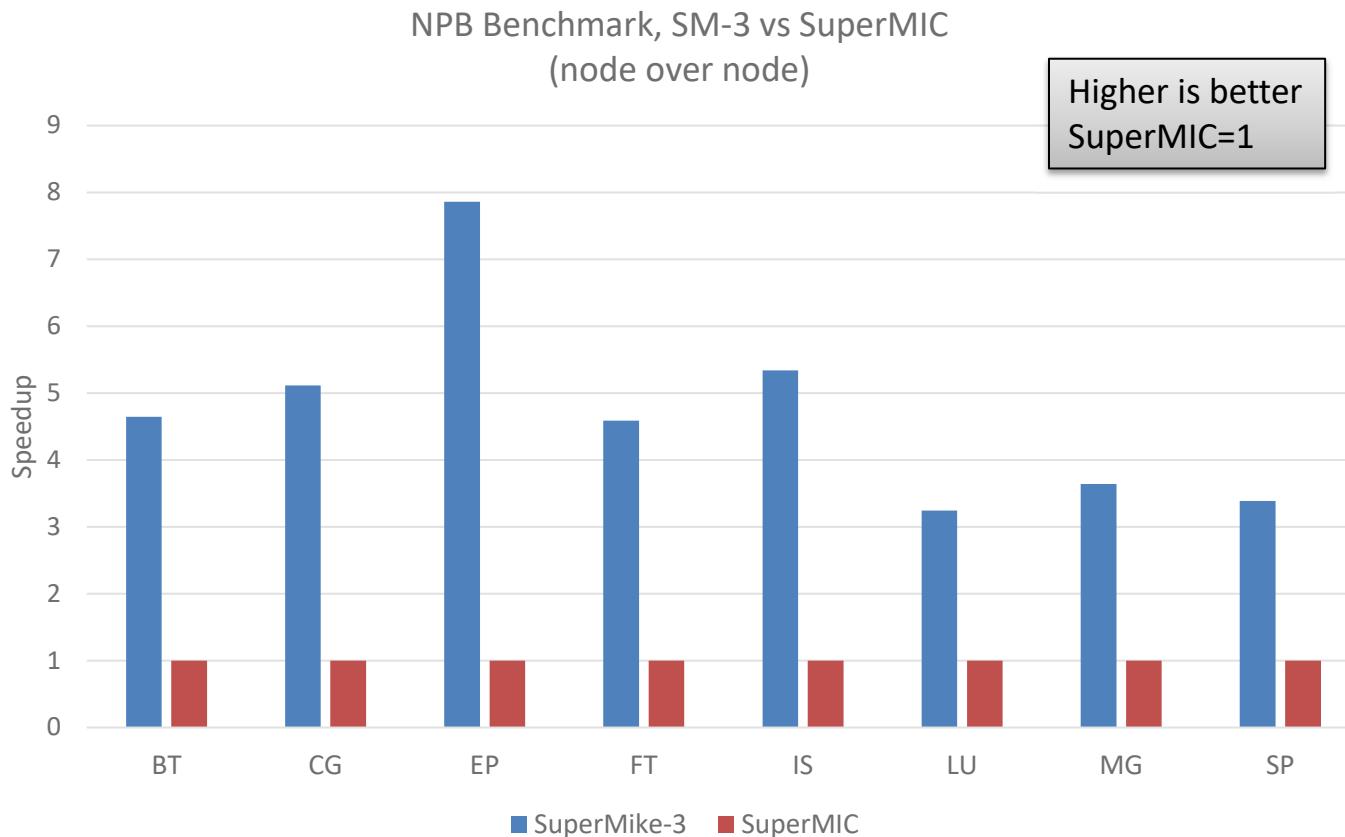
HPCG Benchmark

- High Performance Conjugate Gradient
 - Standard benchmark for **memory-bound** HPC applications
- Results
 - SM-3: 56.8 GFLOPS per node
 - SuperMIC: 14.5 GFLOPS per node
 - Speedup: 3.9 (compared to 11.9 theoretical)

NPB Benchmark Suite

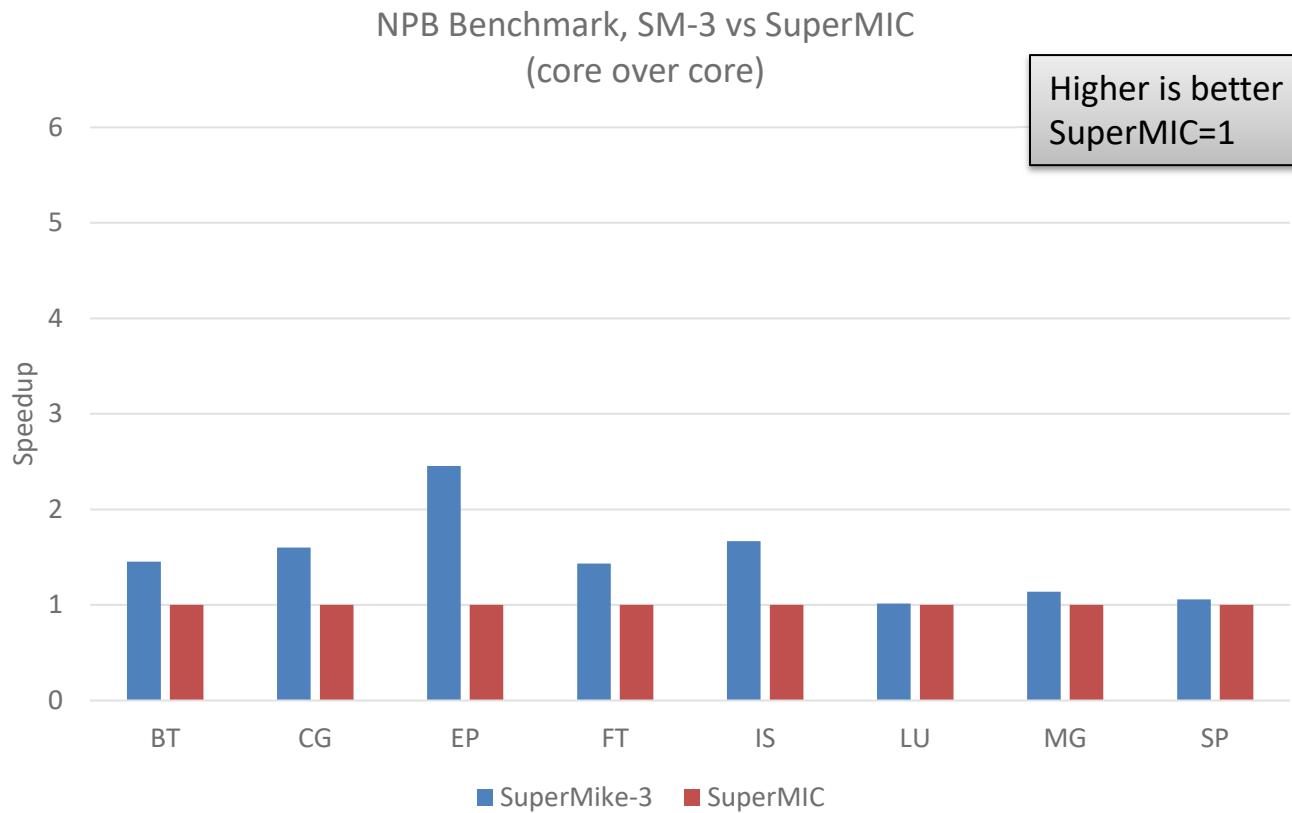
- NAS Parallel Benchmarks
 - a small set of programs derived from computational fluid dynamics applications
- Five kernels and three pseudo-applications
 - IS - Integer Sort, random memory access
 - EP - Embarrassingly Parallel
 - CG - Conjugate Gradient, irregular memory access and communication
 - MG - Multi-Grid on a sequence of meshes, long- and short-distance communication, memory intensive
 - FT - discrete 3D fast Fourier Transform, all-to-all communication
 - BT - Block Tri-diagonal solver
 - SP - Scalar Penta-diagonal solver
 - LU - Lower-Upper Gauss-Seidel solver

NPB Benchmarks – Node over Node



NPB 3.4.2, Class C
Intel 2021.5.1 with “-O3 -xCORE-AVX512”
KMP_AFFINITY=balanced

NPB Benchmarks – Core over Core



NPB 3.4.2, Class C
Intel 2021.5.1 with “-O3 -xCORE-AVX512”
KMP_AFFINITY=balanced

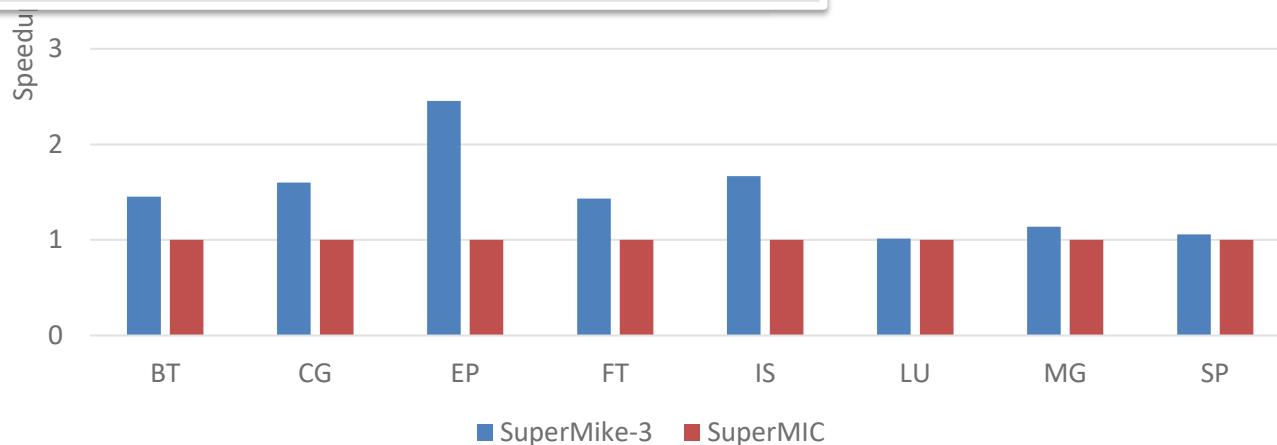
NPB Benchmarks – Core over Core

Takeway

- Performance gain on SM-3 varies a lot from application to application.
- Core-over-core performance gain could be limited.

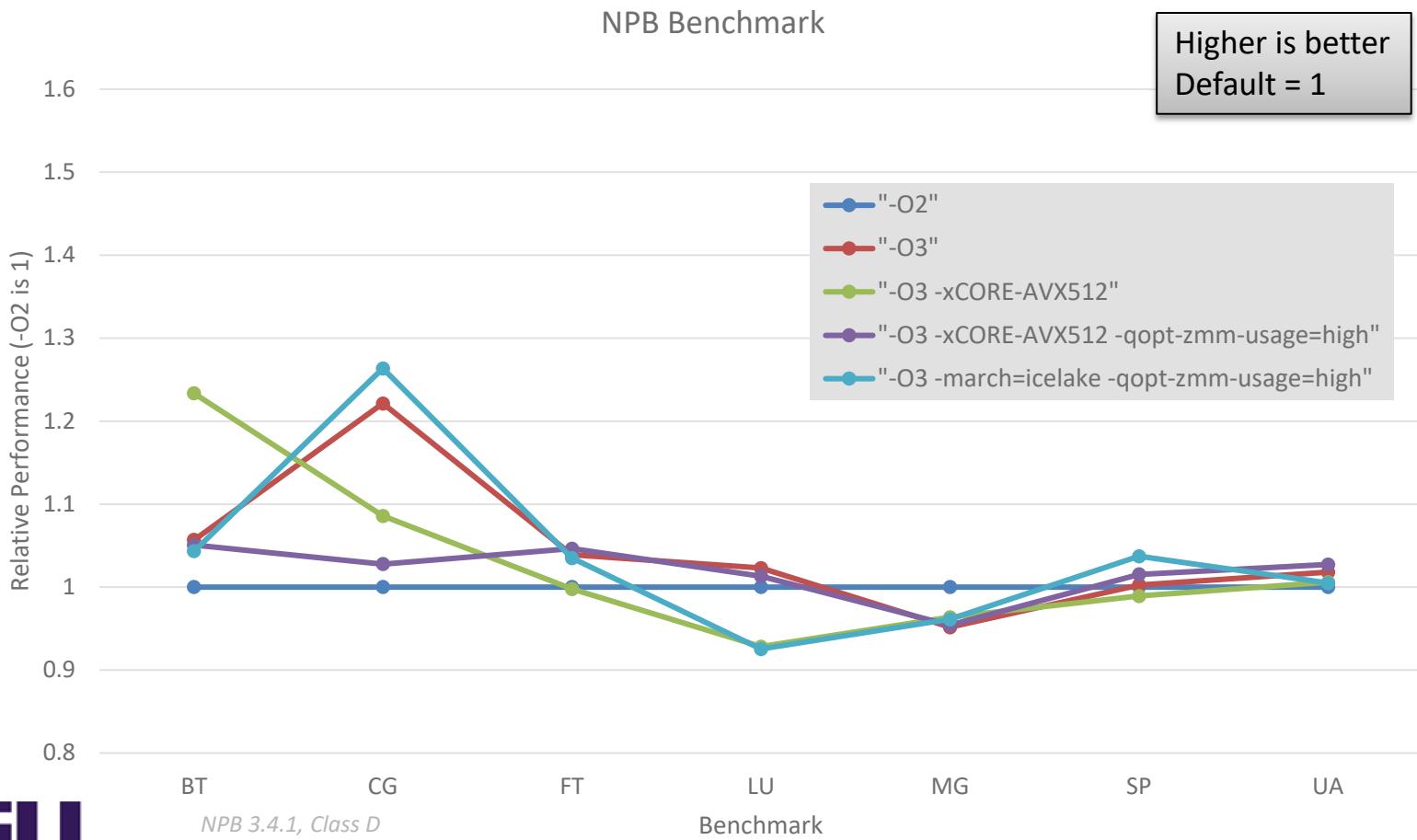
erMIC

Higher is better
SuperMIC=1

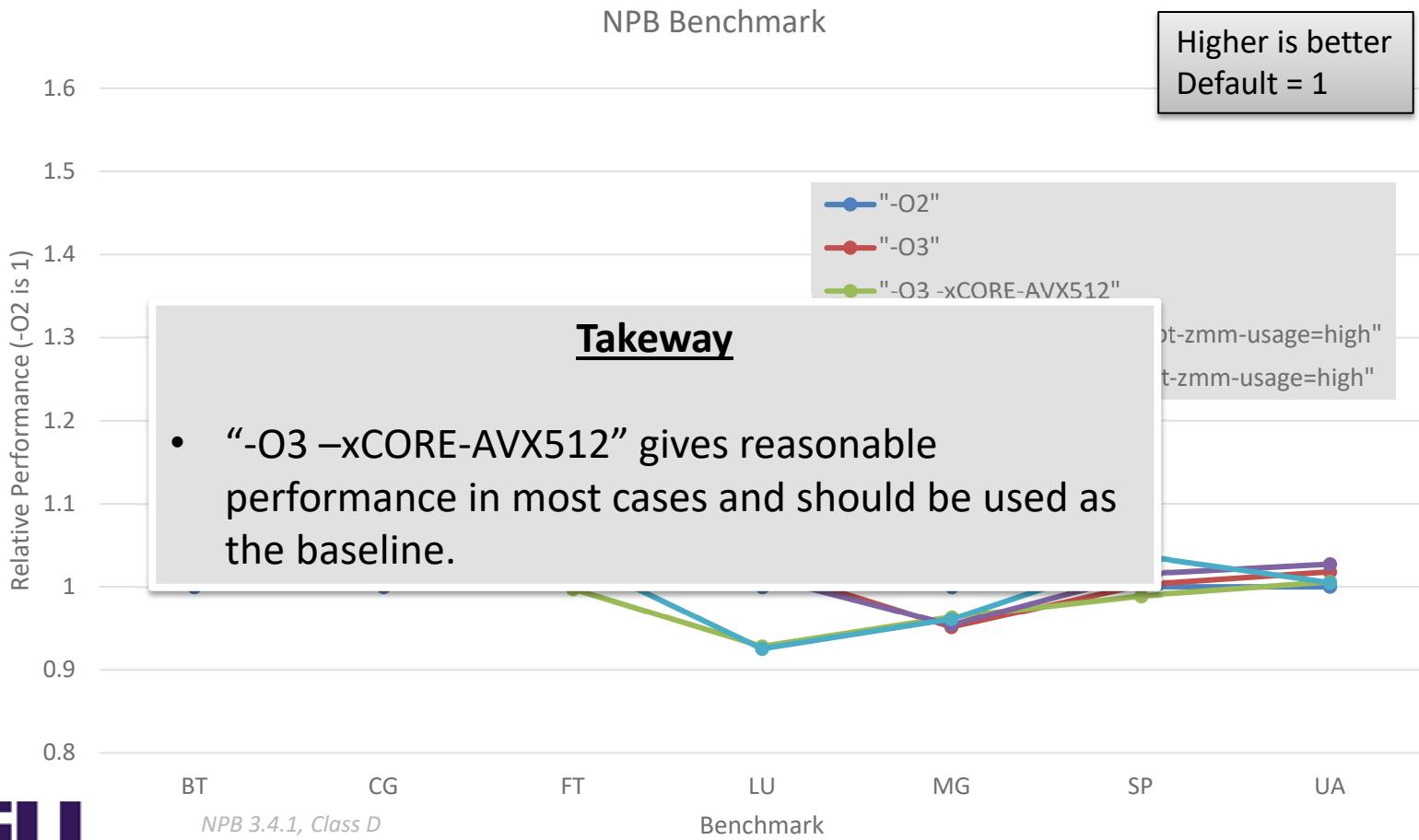


NPB 3.4.2, Class C
Intel 2021.5.1 with “-O3 -xCORE-AVX512”
KMP_AFFINITY=balanced

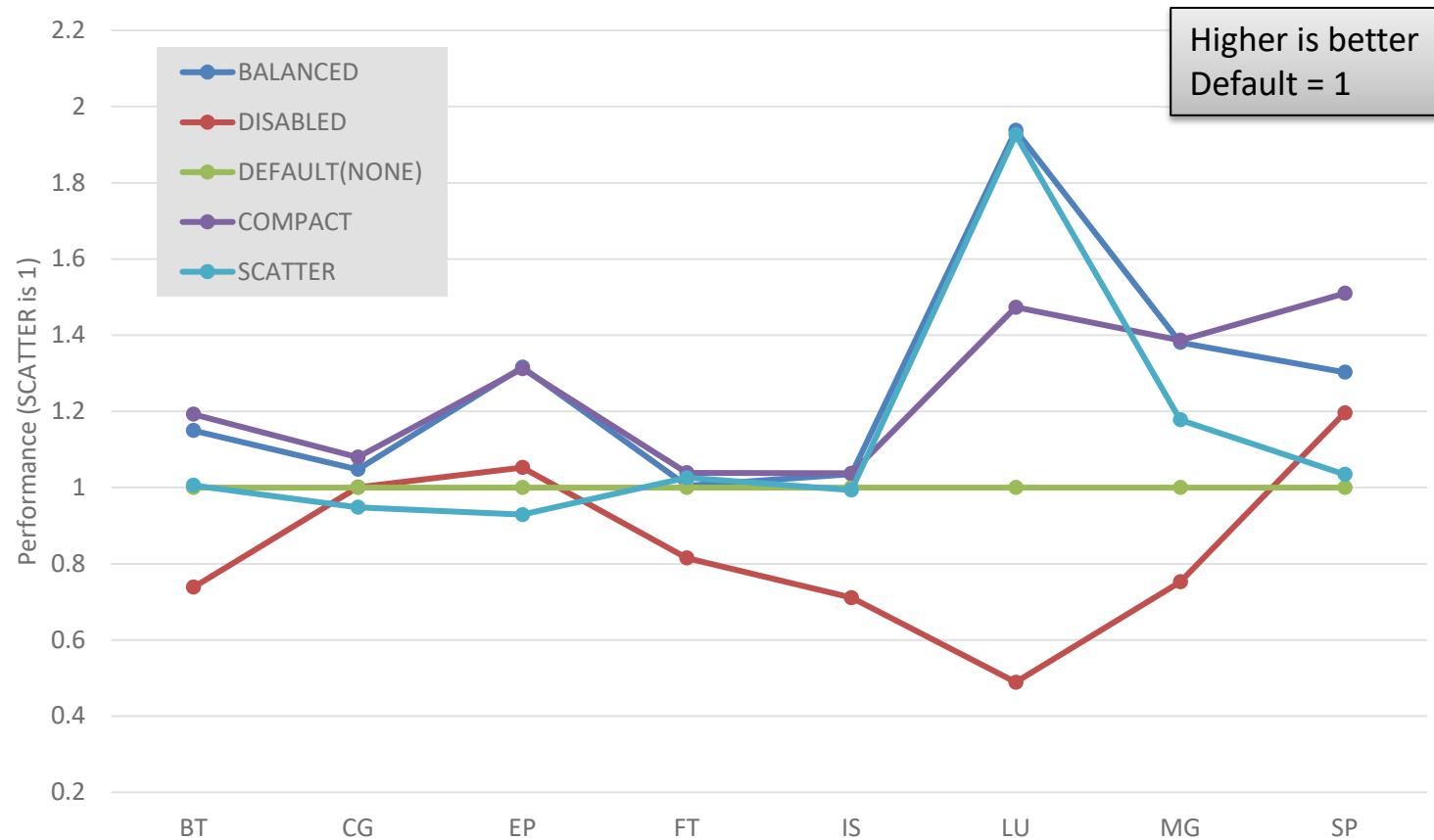
NPB Results – Compiler Flags



NPB Results – Compiler Flags

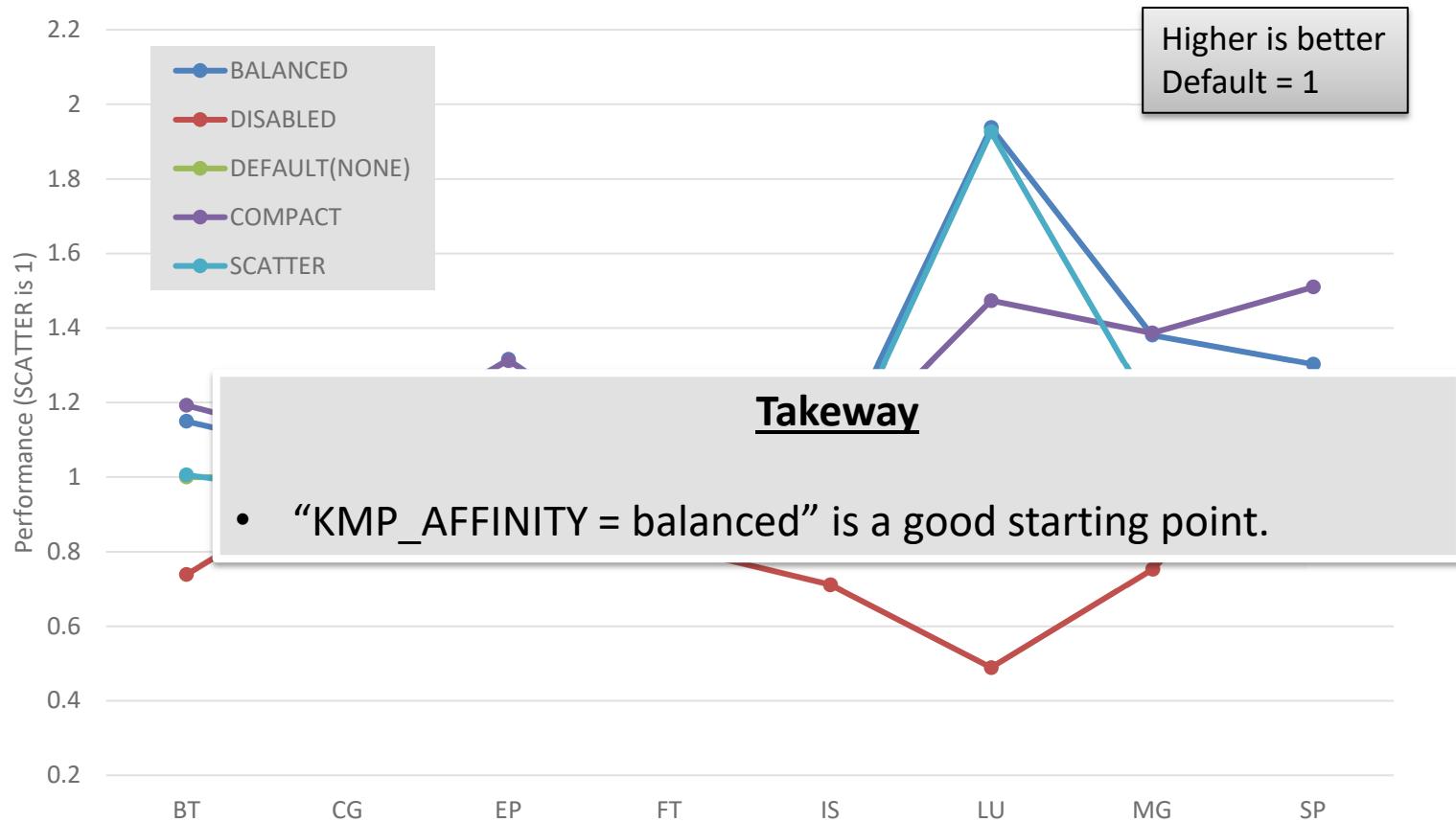


NPB Results – Thread Affinity



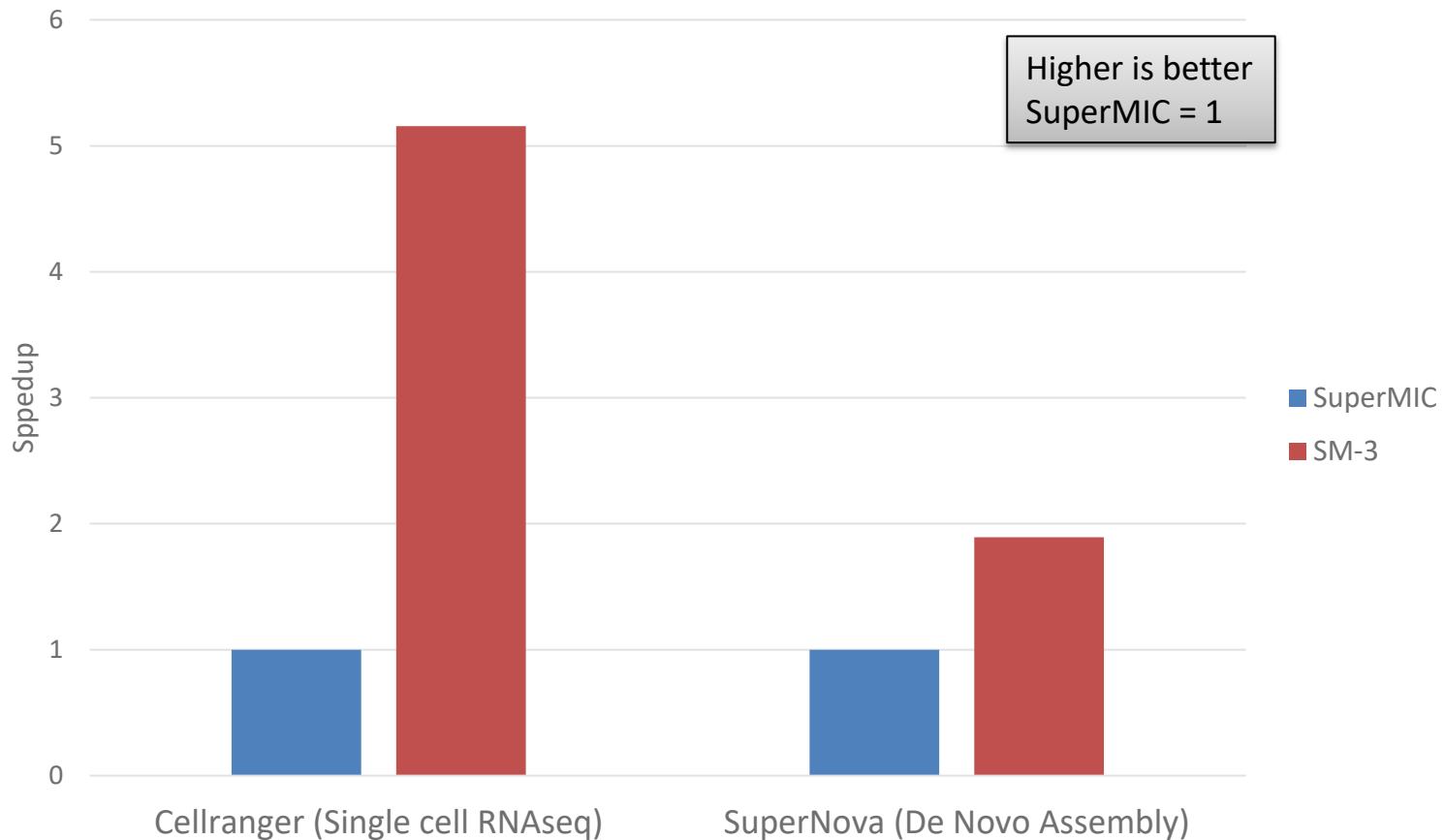
NPB 3.4.2, Class C
Intel 2021.5.1 with “-O3 -xCORE-AVX512”

NPB Results – Thread Affinity



NPB 3.4.2, Class C
Intel 2021.5.1 with “-O3 -xCORE-AVX512”

Bioinformatics

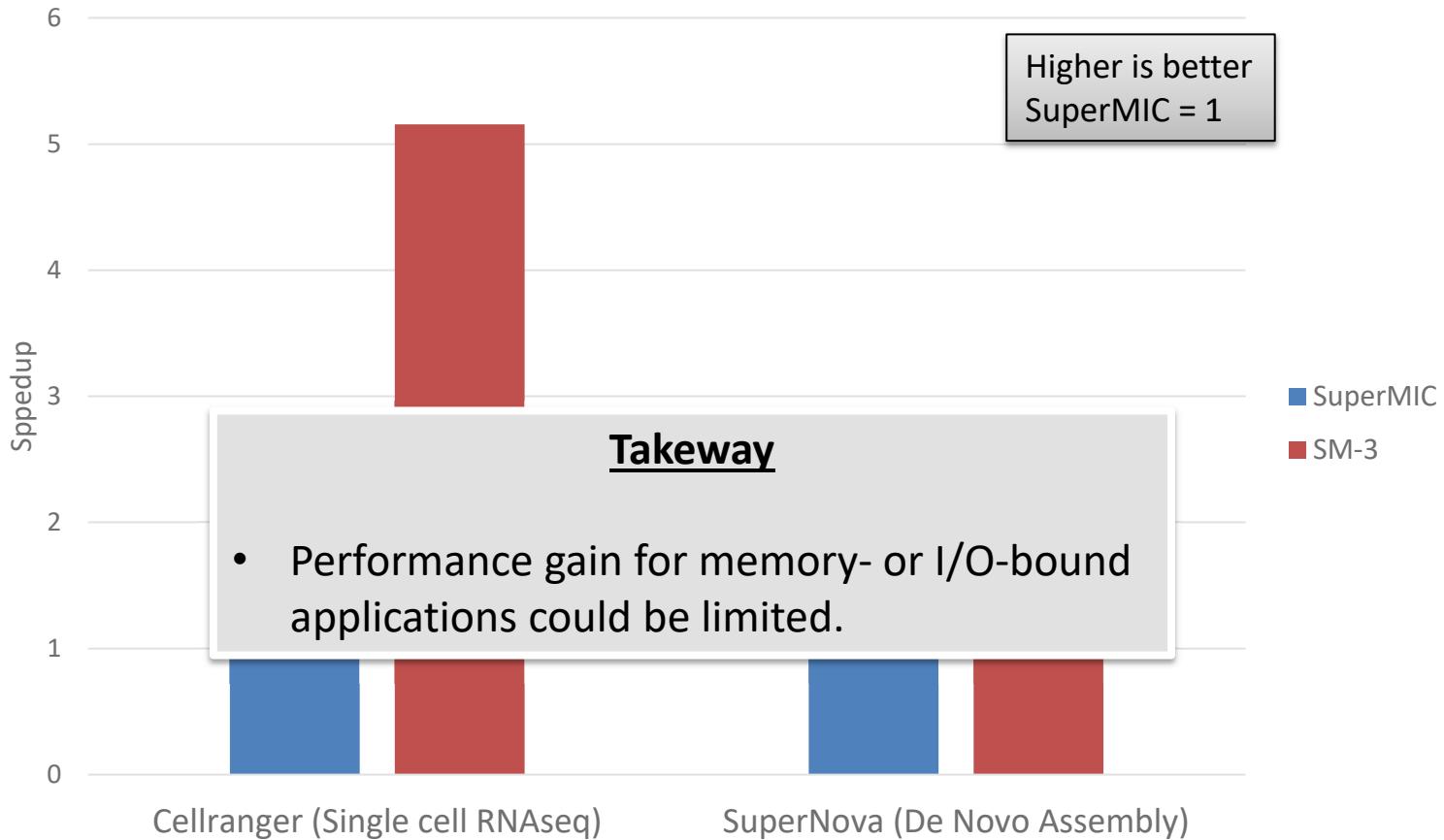


Both are binary distribution from 10x Genomics

Cellranger version 7.0, mouse intestine cell, input data size = 157 GB, reference data size = 22 GB

Supernova version 2.1.1, fruit fly genome, input data size = 259 GB

Bioinformatics



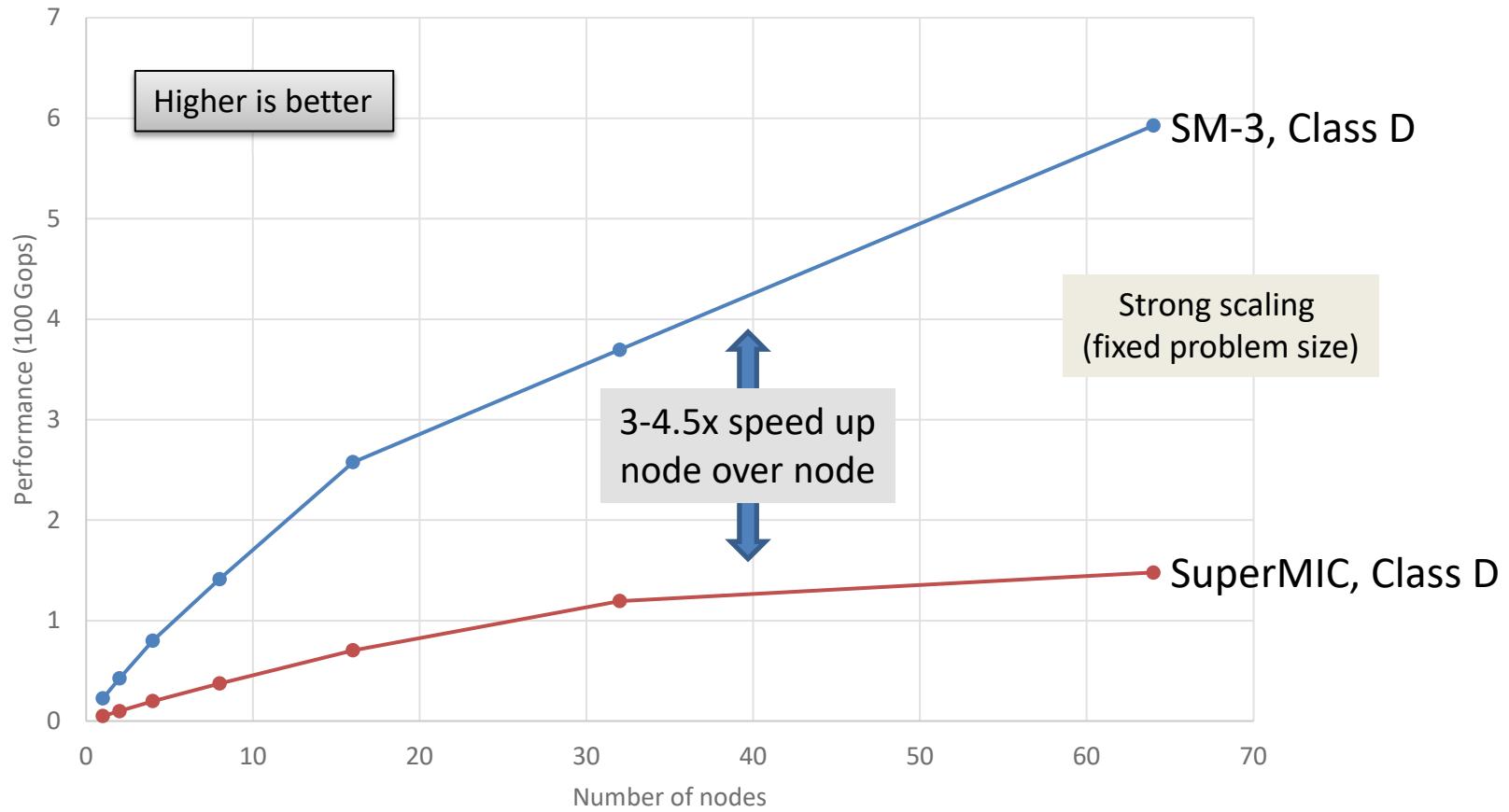
Both are binary distribution from 10x Genomics

Cellranger version 7.0, mouse intestine cell, input data size = 157 GB, reference data size = 22 GB

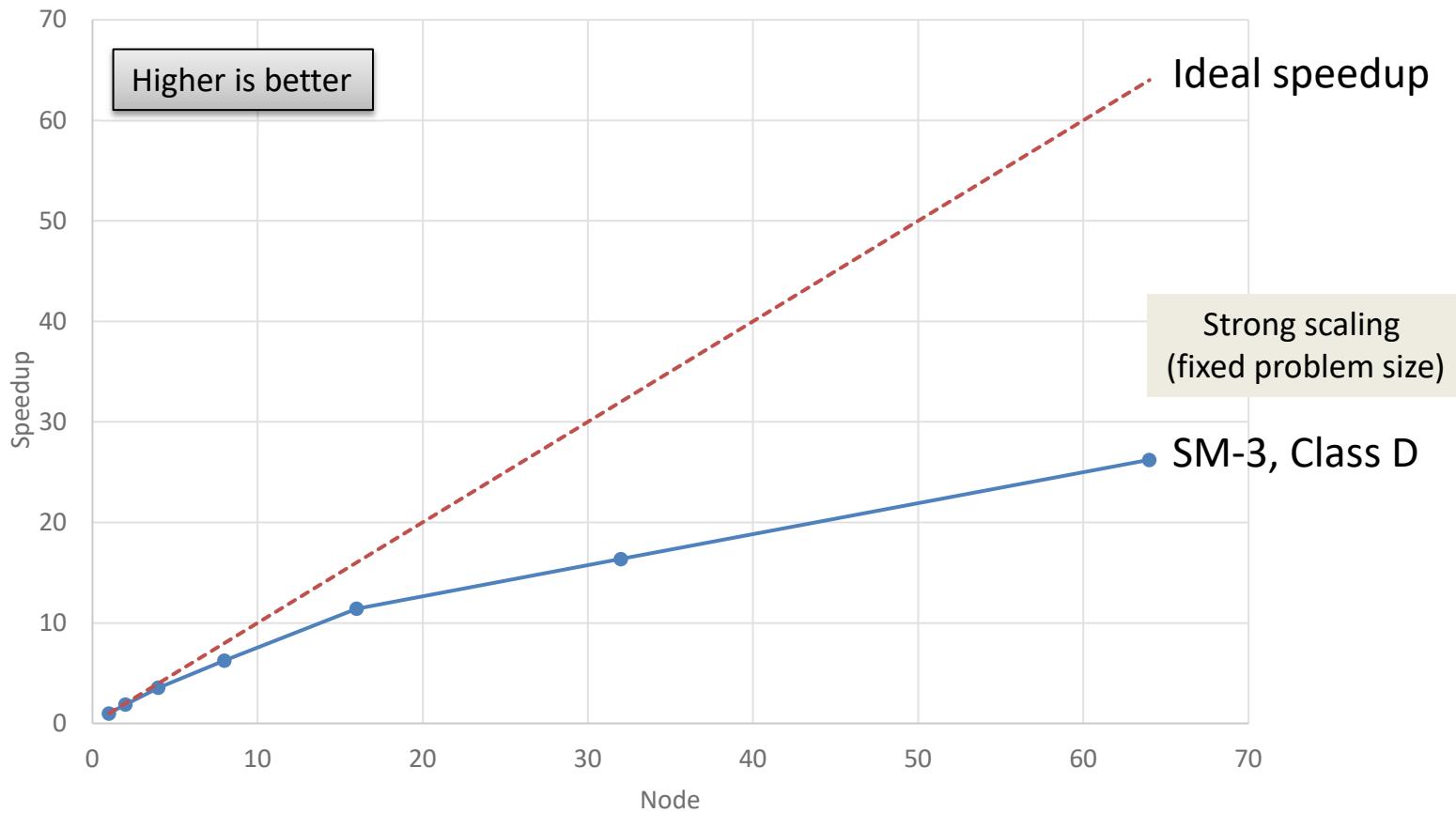
Supernova version 2.1.1, fruit fly genome, input data size = 259 GB

Multi-node Performance

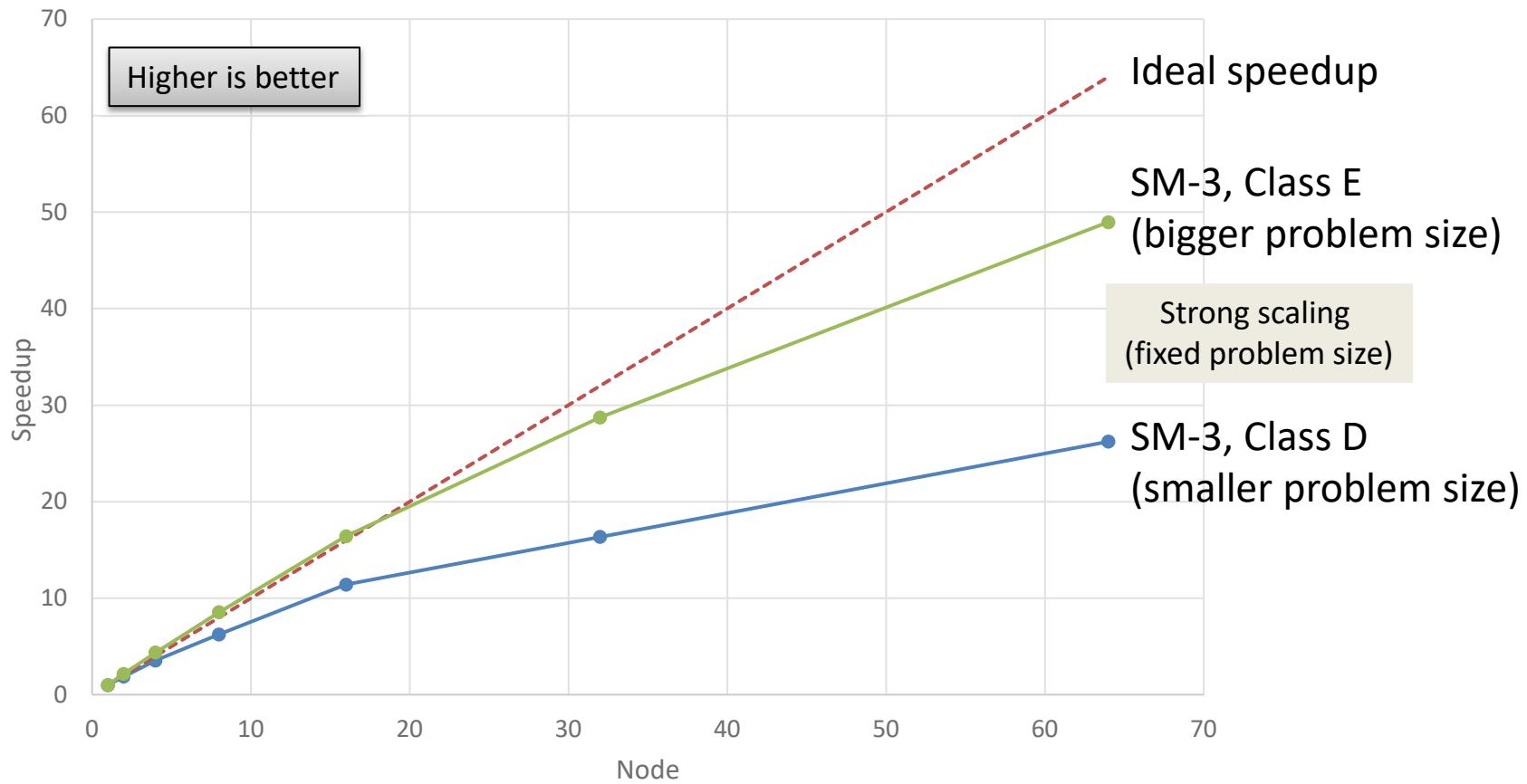
Pure MPI - NPB LU Benchmark



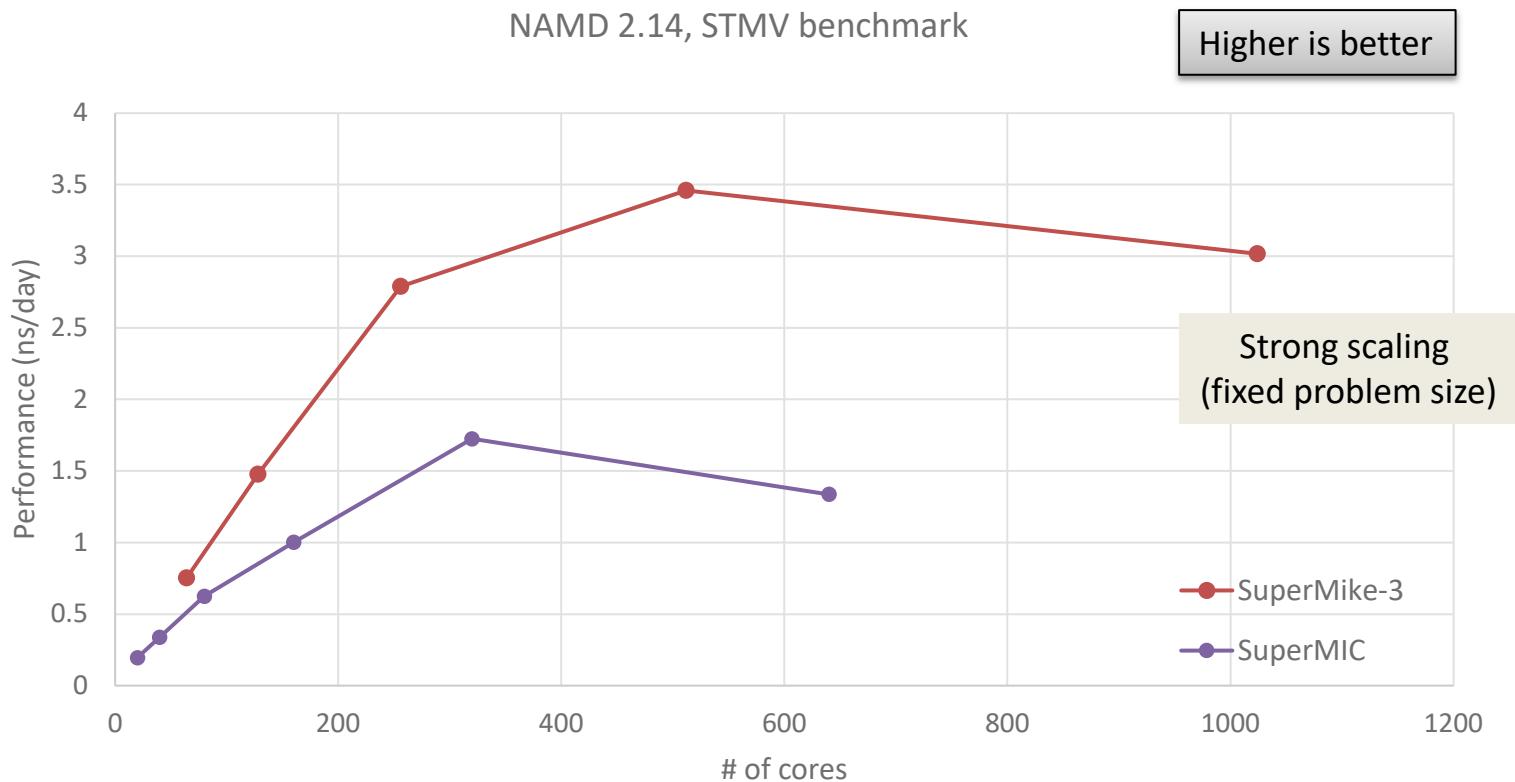
Pure MPI - NPB LU Benchmark



Pure MPI - NPB LU Benchmark

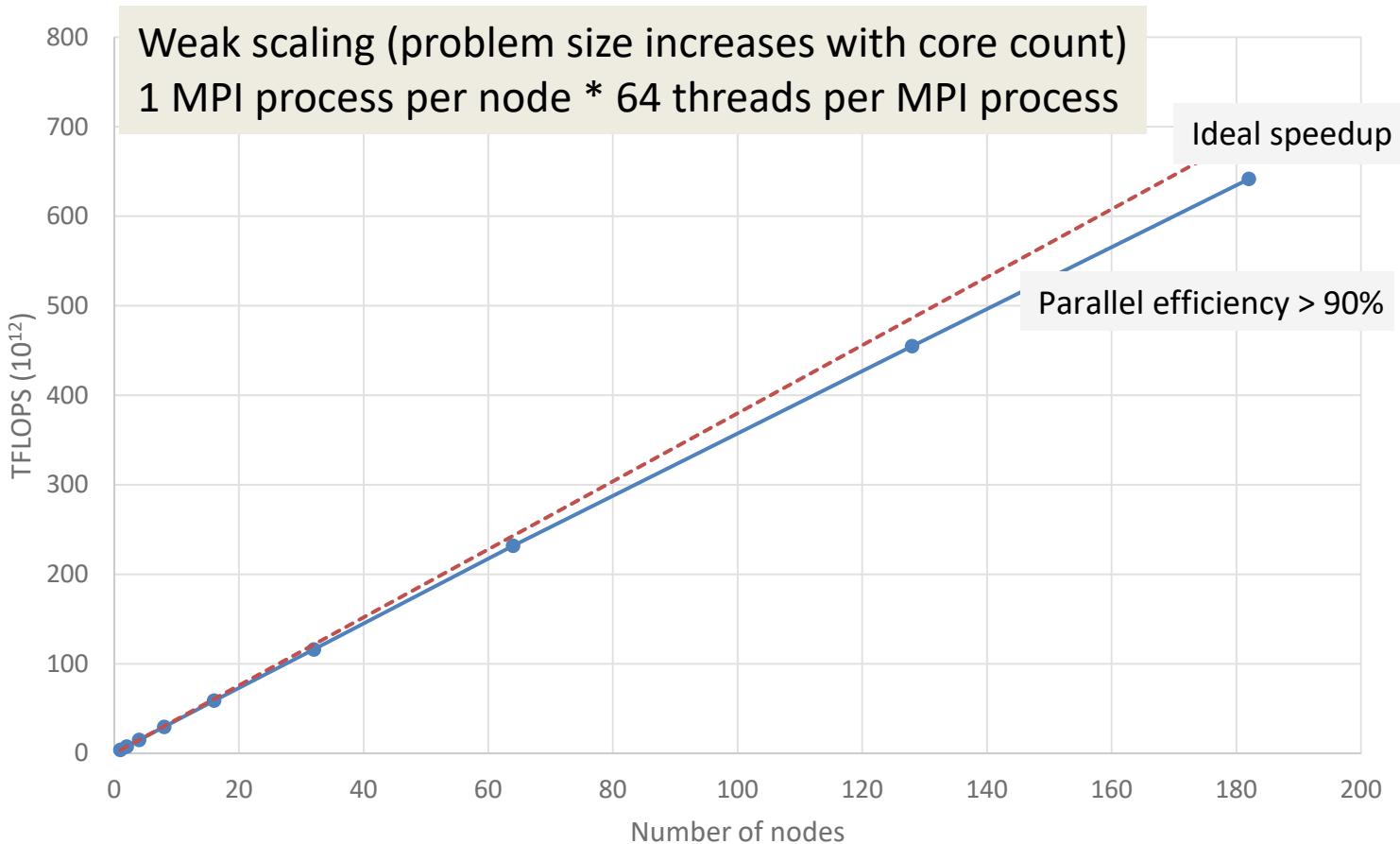


Pure MPI - NAMD



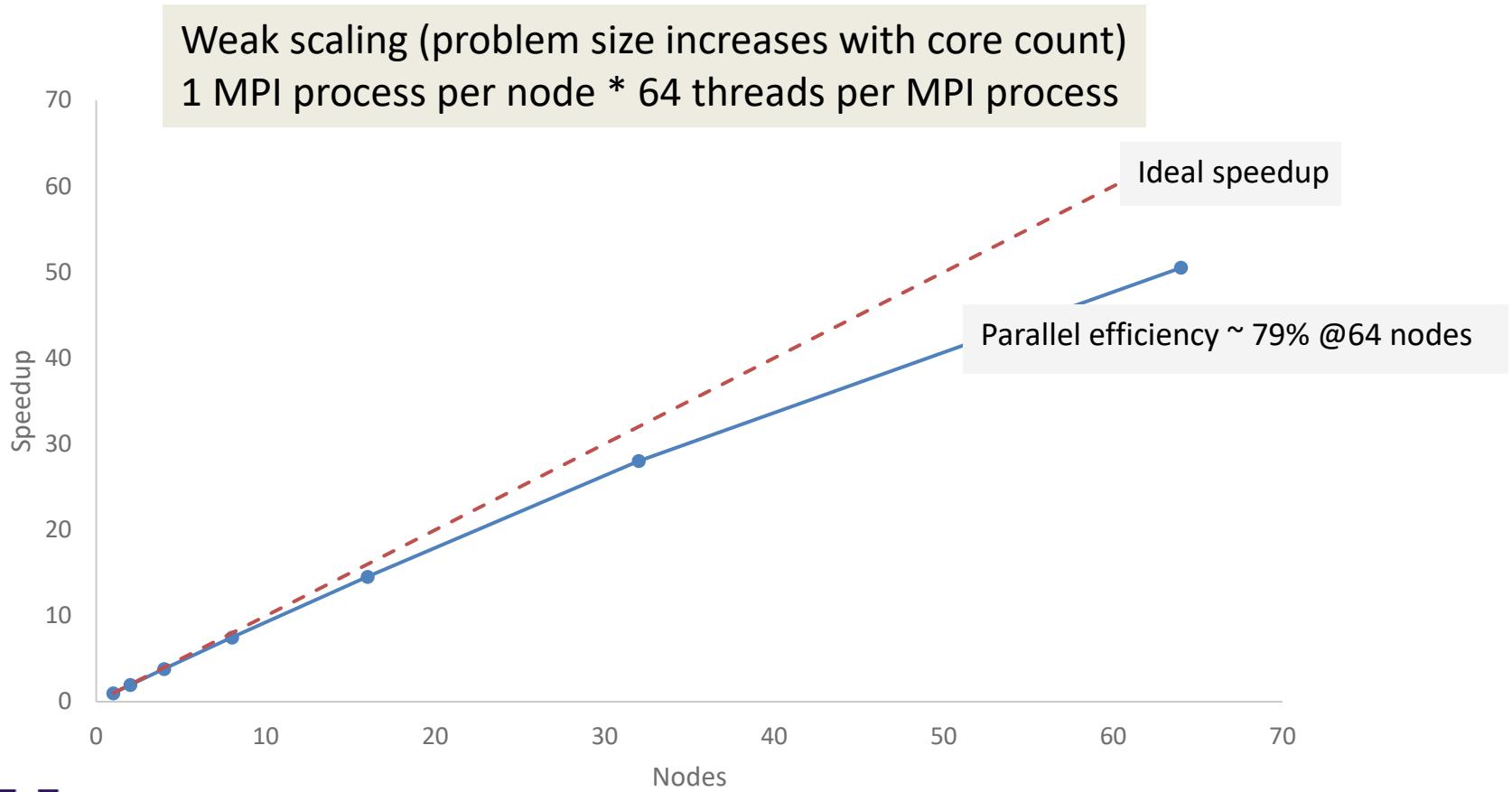
NAMD 2.14
Iverbs version of charmrun
Intel 2021.5.1
Intel MPI 2021.5.0

Hybrid – HPL

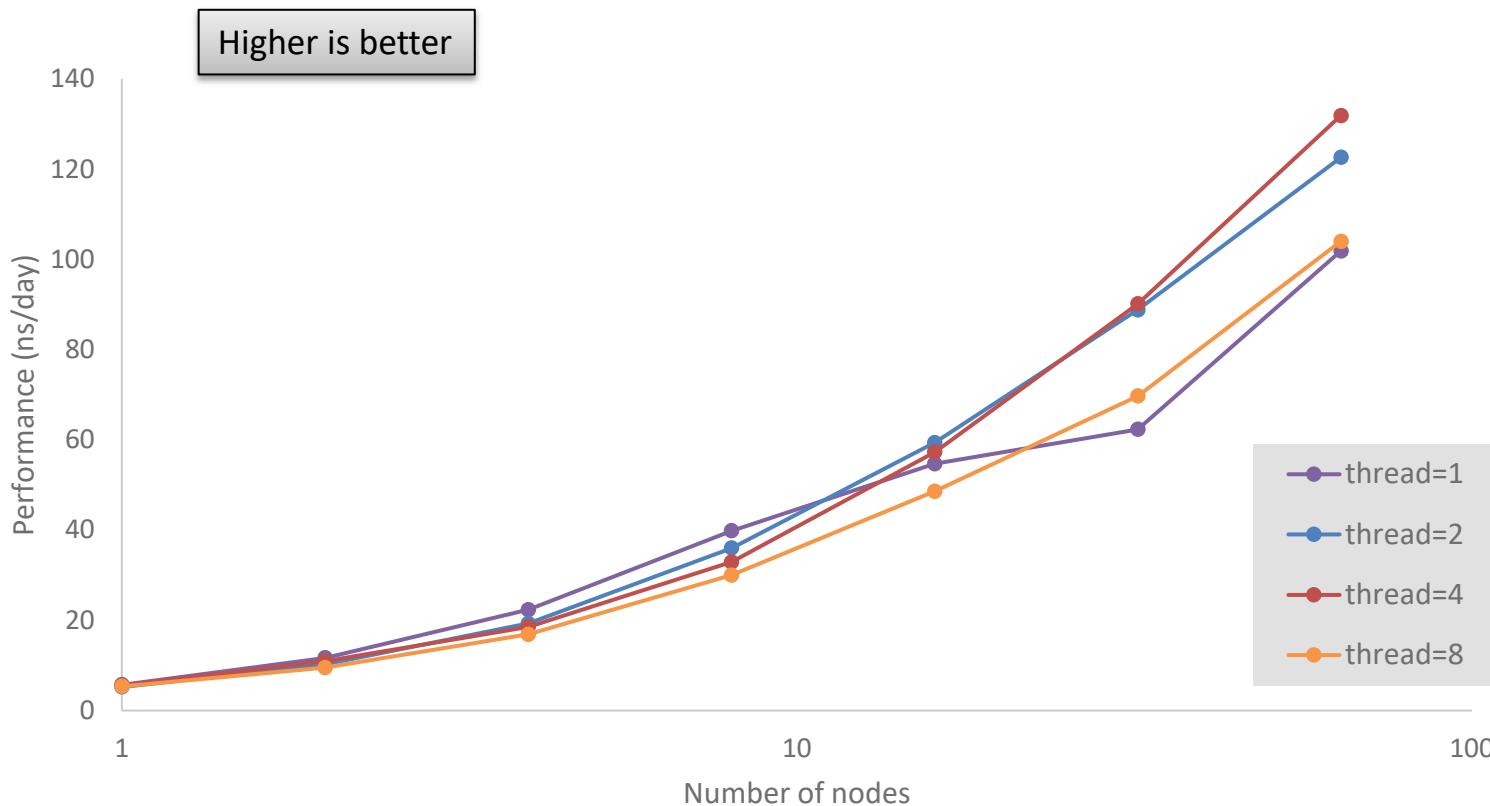


HPL 2.3 (Intel MKL version)
Problem size = 90% installed memory
Intel MPI 2021.5.0

Hybrid - HPCG



Hybrid - GROMACS



I/O Consideration

- You want to avoid letting your program accesses to disk excessively
 - Reading/writing hundreds of GBs to checkpoint or output files frequently
 - Running with thousands of MPI tasks all reading/writing individual files
- What you should and should not do
 - Avoid writing intermediate/checkpoint files unless necessary
 - Look for and use options that allow one or a few big files instead of files per process
 - Reduce the frequency of writing output files
 - Do not use /home for productive jobs – use /work instead

Takeaways

- In most cases, your application will run faster and scale better on SM-3 compared to SuperMIC
- That being said, how much faster depends on a lot of factors
- You need to figure that out before making a (educated) decision whether switch to SM-3
- You need to run your own experiments

Takeaways

- Baseline
 - Use “-O3 -xCORE-AVX512” to compile
 - The default settings work reasonably well in most cases
- Serial programs
 - The performance gain could be limited
- OpenMP programs
 - Try different KMP_AFFINITY settings
- MPI programs
 - Remember the scaling behavior depends on the problem size
- Hybrid programs
 - It does not always help, especially with small node/core count
 - Finding the optimal number of threads could be tricky, which depends on applications as well as problem size

Takeaways

- Baseline
 - Use “-O3 -xCORE-AVX512” to compile
 - The default settings work reasonably well in most cases
- Serial programs
 - The performance gain could be limited
- OpenMP
 - Try different KMP_AFFINITY settings
- MPI programs
 - Remember the scaling behavior depends on the problem size
- Hybrid programs
 - It does not always help, especially with small node/core count
 - Finding the optimal number of threads could be tricky, which depends on applications as well as problem size