A Comparison of Intel and OSU All-to-all Benchmarks for Next Generation FFT Algorithms : Insights from a Comparative Study of Profiling Tools on the Fugaku Supercomputer

Samar A. Aseeri, King Abdullah University of Science Technology (KAUST), Saudi Arabia; Benson Muite, Kichakato Kizito, Kenya; David E. Keyes, KAUST, Saudia Arabia and Columbia University, U.S.

SIAMPP'24 Conference
Baltimore, Maryland, U.S.

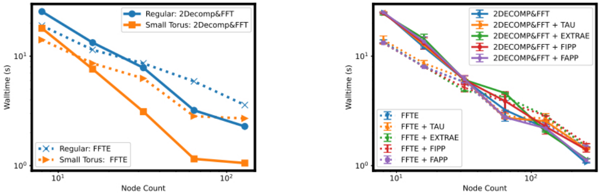
Comparing with MPI Benchmarks

- We delve into comparing the profiling tools as discussed in the paper.
- This comparison will focus on key performance metrics such as run-time overhead, memory utilization, and data storage.
- In this presentation we focus on the Comparing of the MPI Benchmarks.

Definitions

- MPI Benchmark: A benchmark designed to measure the performance of Message Passing Interface (MPI) implementations.
- Run-Time Overhead: Additional time or resources consumed during execution by a tool or process.
- Memory Utilization: The amount of memory used by a tool or process for its operations.

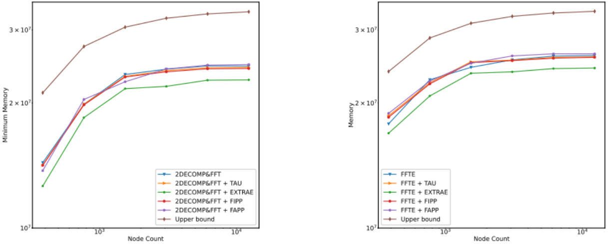
Run-Time Overhead Comparison



(a) A comparison of the small torus and regular communication modes for solvers using the 2Decomp&FFT and FFTE libraries.

(b) Performance tools comparison with small torus mode. Error bars show standard deviation in runtime.

Fig. 1. Strong scaling comparisons for 30 time steps of FFT based Klein Gordon equation solvers using discretizations of 768^3 grid points.



(a) With 2Decomp&FFT

(b) With FFTE

Fig. 9. Comparison of Klein Gordon solver memory utilization and memory utilization bounds

Disk Space Utilization Comparison

Table 2. Data size stored to disk for the test program using the 2Decomp&FFT library. The gzip and ppk values are only for compressed data needed for performance analysis.

Nodes	2Decomp&FFT only	+ Extrae	+ Extrae (gzip)	+ TAU	+ TAU (ppk)	+ FIPP	+ FAPP
8	623.23KB	79.18MB	2.8MB	11.29MB	461KB	9.54MB	23.00MB
16	629.42KB	281.12MB	7.8MB	22.80MB	840KB	17.42MB	11.80MB
32	642.29KB	730.95MB	16MB	37.13MB	1.3MB	28.94MB	6.20MB
64	668.33KB	1.42GB	30MB	64.37MB	2.3MB	55.17MB	3.40MB
128	720.44KB	2.82GB	55MB	115.35MB	3.9MB	99.73MB	2.00MB
256	826.97KB	5.59GB	104MB	219.98MB	7.0MB	197.52MB	1.30MB

Table 3. Data size stored to disk for the test program using the FFTE library. The gzip and ppk values are only for compressed data needed for performance analysis.

Nodes	FFTE only	+ Extrae	+ Extrae (gzip)	+ TAU	+ TAU (ppk)	+ FIPP	+ FAPP
8	385.13KB	20.91MB	2.4MB	20.31MB	574KB	3.11MB	17.15MB
16	391.28KB	42.11MB	4.9MB	30.32MB	862KB	5.12MB	8.76MB
32	404.13KB	91.72MB	9.5MB	42.91MB	1.2MB	8.54MB	4.56MB
64	430.24KB	188.33MB	19MB	65.09MB	1.8MB	15.20MB	2.47MB
128	482.45KB	439.29MB	38MB	101.76MB	2.9MB	28.28MB	1.42MB
256	589.08KB	920.25MB	-	172.14MB	4.5MB	57.52MB	916.75KB

Obtaining MPI Benchmarks

- **OSU Micro-Benchmarks:** (`https://mvapich.cse.ohio-state.edu/download/mvapich/osu-micro-benchmarks-7.0.1.tar.gz`) contains scripts used to build the specific versions of the OSU Micro-Benchmarks employed in this study.
- **Intel MPI Benchmarks:** Included with Intel MPI libraries * Intel MPI Benchmarks:(`https://github.com/intel/mpi-benchmarks/archive/refs/tags/IMB-v2021.3.tar.gz`) builds the Intel MPI Benchmarks used here. * Offers a comprehensive suite of MPI operation benchmarks, potentially including all-to-all communication.

Setting Up MPI Benchmarks

- **OSU Micro-Benchmarks:** * Follow the compilation and execution instructions provided in the repository's README file. * Requires an MPI library like Intel MPI to be installed and configured.
- **Intel MPI Benchmarks:** * Refer to the Intel MPI documentation for detailed setup instructions. * Typically integrated with the installation process of Intel MPI libraries.

Study Considerations

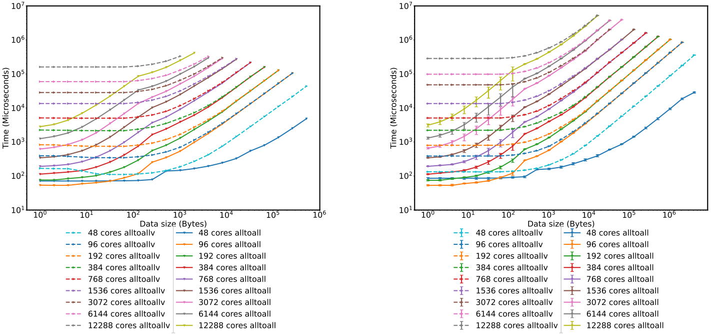
- **Original Study Considerations:** The original study did not directly utilize these specific benchmark scripts due to: * Focus on all-to-all communication patterns within the implemented 3D FFT with pencil decomposition. * The benchmarks may not perfectly capture the communication patterns used in this specific FFT implementation (as noted by collaborator Benson).
- Despite not using the benchmarks directly, the insights gained from understanding these communication patterns (latency, bandwidth) were crucial for optimizing the FFT solver.

Benefits of MPI Benchmarks

MPI benchmarks offer valuable insights for optimizing communication patterns in FFT solvers:

- **Comprehensive Evaluation:** Employing both OSU Micro-Benchmarks and Intel MPI Benchmarks provides a well-rounded analysis of communication relevant to FFT.
- **Targeted Insights:** OSU benchmarks isolate specific communication patterns (MPI_Alltoall, MPI_Alltoallv) crucial for understanding potential bottlenecks.
- **Performance Bottleneck Identification:** Benchmark results pinpoint latency and bandwidth limitations impacting FFT solver efficiency.
- **Optimization Strategies:** Insights from the benchmarks can guide efforts to reduce communication overheads and improve data exchange patterns in the FFT solver.

Figure: OSU and Intel MPI Benchmark Results



(a) Intel Alltoall and Alltoallv.

(b) OSU Alltoall and Alltoallv.

Fig. 7. MPI benchmarks.

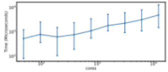


Fig. 8. OSU MPI Init benchmark. Error bars indicate standard deviations.

Conclusions

- Utilizing both OSU Micro-Benchmarks and Intel MPI Benchmarks provided a comprehensive assessment of communication efficiency in FFT solvers.
- It was observed in both the OSU and Intel benchmarks that AlltoallV incurs more overhead than Alltoall for small message sizes.
- Identified latency limitations in MPI_Alltoallv operations as a key factor impacting performance, highlighting the need for optimization.
- Bandwidth constraints in MPI_Alltoall operations suggest scalability challenges for parallel computations on high-core count systems.
- Benchmark trends underscore the importance of optimizing communication strategies for efficient FFT solver execution.
- Addressing latency and bandwidth limitations is crucial for enhancing communication efficiency and scalability.
- Benchmark results provide valuable insights for optimizing FFT solvers by reducing communication overheads and improving data exchange patterns.

Acknowledgements

This work was made possible by the valuable contributions of:

- Sameer Shende (University of Oregon)
- Judit Gimenez (Barcelona Supercomputing Center)

We sincerely appreciate their support and expertise.