A Performance Analysis of Profiling Tools on the Fugaku Supercomputer

Samar A. Aseeri,Judit Gimenez, Sameer S. Shende, Benson K. Muite and David E. Keyes

September 24, 2024

Abstract

This study focuses on the overhead of performance monitoring tools when profiling Fast Fourier Transform (FFT)-based solvers for the Klein Gordon equation. Tools TAU, Extrae, FAPP, and FIPP are examined for their ease of use, overhead, and memory consumption. The study finds that minimizing communication interference significantly improves performance.

Keywords

- Performance
- Profiling Tools
- Fast Fourier Transform

Introduction

This study is concerned with the overhead of performance tool measurements when profiling FFT-based solvers for the Klein Gordon equation. The tools examined are TAU and Extrae, along with Fujitsu's FAPP and FIPP. The study highlights the benefits of using the small torus mode on Fugaku for communication-intensive codes.

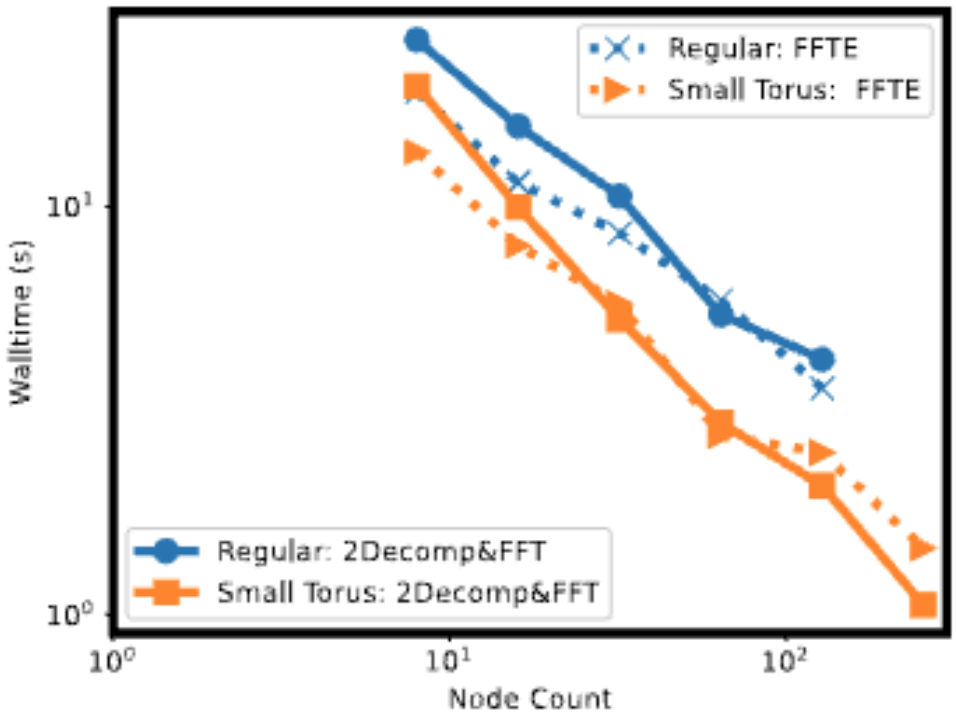
Experimental Setup

- Test Program: Solves the Klein Gordon equation using a three-dimensional wave propagation model.
- Test Platform: Fugaku features a 6D torus network and consists of nodes with Fujitsu A64FX Arm architecture, each having 48 cores.
- Profiling Tools: TAU, Extrae, FIPP, and FAPP.

Profiling Results

- Comparison of small torus and regular modes shows reduced job completion time (see Figure 1a).
- TAU provides detailed profiling, capturing significant time spent in FFTW routines (see Figure 2a).
- Extrae shows better insights into MPI time distribution (see Figure 4a).

Additional Profiling Figures



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

Figure: Comparison of Communication Modes

Additional Profiling Figures

TAU: ParaProf: Statistics for: node 0 - RUN-2decomp/2decomp_fugaku.ppk

Name	Excl...	Incl... ▽	Calls	Chil...
.TAU application	18.435	37.68	1	252
[CONTEXT] .TAU application	0	17.533	579	0
[SAMPLE] UNRESOLVED /lib64/libfftw3.so.3.5.5	12.767	12.767	421	0
[SUMMARY] __float128 const [{/vol0003/hp210193/data/benchmark3/version2decomp/KleinGordon/2decomp_tau/KgSemiImp3d.f90}]	2.52	2.52	84	0
[SAMPLE] __float128 const [{/vol0003/hp210193/data/benchmark3/version2decomp/KleinGordon/2decomp_tau/KgSemiImp3d.f90} {251}]	1.96	1.96	65	0
[SAMPLE] __float128 const [{/vol0003/hp210193/data/benchmark3/version2decomp/KleinGordon/2decomp_tau/KgSemiImp3d.f90} {243}]	0.54	0.54	18	0
[SAMPLE] __float128 const [{/vol0003/hp210193/data/benchmark3/version2decomp/KleinGordon/2decomp_tau/KgSemiImp3d.f90} {266}]	0.02	0.02	1	0
[SUMMARY] enercalc_ [{/vol0003/hp210193/data/benchmark3/version2decomp/KleinGordon/2decomp_tau/enercalc.f90}]	1.04	1.04	35	0
[SUMMARY] storeold_ [{/vol0003/hp210193/data/benchmark3/version2decomp/KleinGordon/2decomp_tau/storeold.f90}]	0.38	0.38	12	0
[SAMPLE] g_dcos [{/opt/FJSVxtclanga/tcsds-1.2.36/lib64/libfj90f.so.1} {0}]	0.14	0.14	3	0
[SUMMARY] decomp_2d_fft.fft_3d_c2c_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/fft_fftw3.f90}]	0.12	0.12	4	0
[SAMPLE] decomp_2d_fft.fft_3d_c2c_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/fft_fftw3.f90} {540}]	0.1	0.1	3	0
[SAMPLE] decomp_2d_fft.fft_3d_c2c_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/fft_fftw3.f90} {498}]	0.02	0.02	1	0
[SAMPLE] mmap [{} {0}]	0.095	0.095	3	0
[SAMPLE] UNRESOLVED /opt/FJSVxos/mmm/lib64/libmpg.so.1	0.08	0.08	2	0
[SUMMARY] decomp_2d.mem_merge_zy_complex_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/./transpo	0.06	0.06	2	0
[SUMMARY] decomp_2d.mem_merge_yx_real_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/./transpose_y	0.06	0.06	2	0
[SAMPLE] __pthread_mutex_lock_full [{} {0}]	0.06	0.06	2	0
[SUMMARY] decomp_2d.mem_split_xy_real_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/./transpose_x_t	0.04	0.04	2	0
[SUMMARY] decomp_2d.mem_split_yx_real_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/./transpose_y_t	0.04	0.04	1	0
[SUMMARY] decomp_2d.mem_merge_zy_real_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/./transpose_z	0.03	0.03	1	0
[SUMMARY] decomp_2d.mem_split_yz_real_ [{/vol0003/hp210193/data/benchmark3/version2decomp/FFT_libraries/2decomp_fft/src/./transpose_y_t	0.02	0.02	1	0
[SAMPLE] UNRESOLVED [vdso]	0.02	0.02	1	0
[SAMPLE] qsort_r [{} {0}]	0.02	0.02	1	0
[SAMPLE] UNRESOLVED /vol0003/hp210193/data/benchmark3/version2decomp/KleinGordon/2decomp_tau/sameer_final_run_384/Kg	0.02	0.02	1	0
[SAMPLE] msort_with_tmp.part.0 [{} {0}]	0.02	0.02	1	0
MPI_Alltoallv()	13.685	13.685	138	0
[CONTEXT] MPI_Alltoallv()	0	13.657	457	0
[SAMPLE] utofu_read_tcq_nocb [{/lib64/libtofucom.so} {0}]	3.37	3.37	113	0
[SAMPLE] ompi_coll_mtofu_read_tcq_cq [{/opt/FJSVxtclanga/tcsds-1.2.36/lib64/libmpi.so.0.0.0} {0}]	3.078	3.078	101	0
[SAMPLE] mca_btl_vader_component_progress [{/opt/FJSVxtclanga/tcsds-1.2.36/lib64/libmpi.so.0.0.0} {0}]	3.05	3.05	103	0
[SAMPLE] ompi_coll_mtofu_alltoallv_doublespread [{/opt/FJSVxtclanga/tcsds-1.2.36/lib64/libmpi.so.0.0.0} {0}]	1.37	1.37	48	0

Figure: TAU Profiling Insights

Additional Profiling Figures

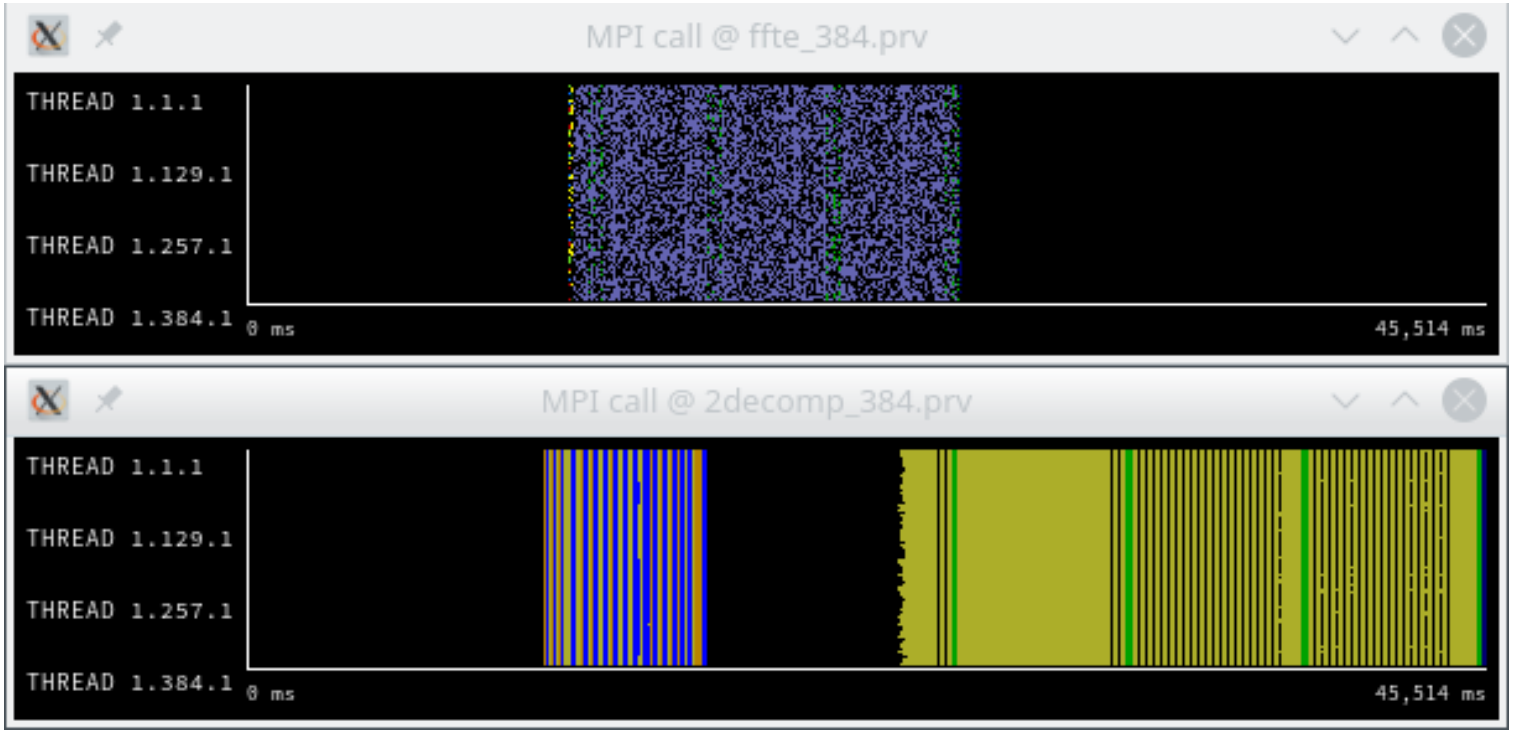
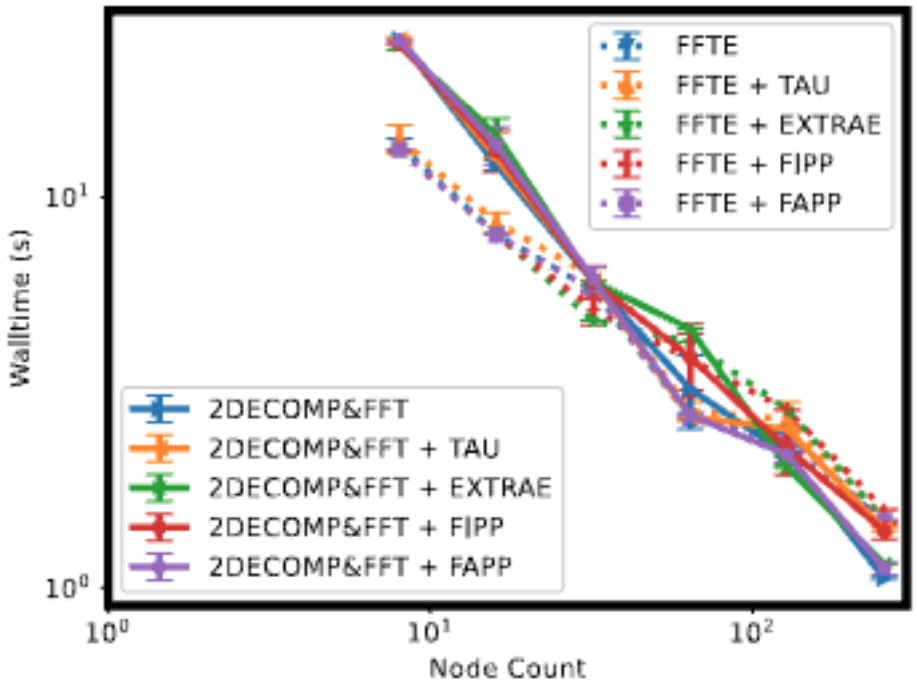


Figure: Extrae Analysis

Tool Overhead, Memory and Data Storage

- Runtime overhead is minimal across tools (see Figure 1b).
- Extrae uses the most memory due to tracing.
- Data storage requirements increase with the number of nodes (see Tables 2 and 3).

Additional Profiling Figures



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

Figure: Runtime Overhead

Data Size with 2Decomp&FFT

Nodes	8	16	32	64	128	256
2Decomp&FFT	0.62	0.63	0.64	0.67	0.72	0.83
+ Extrae	79	281	731	1400	2800	5600
+ Extrae (gzip)	2.8	7.8	16	30	55	104
+ TAU	11	23	37	64	115	220
+ TAU (ppk)	0.46	0.84	1.3	2.3	3.9	7
+ FIPP	9.5	17	29	55	100	198
+ FAPP	1.3	2.0	3.4	6.2	12	23

Table: Data size in MB stored on disk for the test program using the 2Decomp&FFT library.

Data Size with FFTE

Nodes	8	16	32	64	128	256
FFTE	0.39	0.39	0.40	0.43	0.48	0.59
+ Extrae	21	42	92	190	440	920
+ Extrae (gzip)	2.4	4.9	9.5	19	38	-
+ TAU	20	30	43	65	100	170
+ TAU (ppk)	0.57	0.86	1.2	1.8	2.9	4.5

Table: Data size in MB stored on disk for the test program using the FFTE library.

FIPP and FAPP Overview

Brief Description

- **FIPP (Fugaku Instant Performance Profiler)**: A low overhead sample-based profiler that primarily provides insights into load imbalance in annotated code regions. It generates results in a plain text format.
- **FAPP (Fugaku Advanced Performance Profiler)**: A more comprehensive profiling tool that provides detailed performance metrics using sampling. It outputs data in a proprietary binary format, which can be converted to CSV for analysis in Excel.

Operating Mechanism

- Both tools require source code annotation with commands (e.g., fipp_start, fapp_start) for profiling specific regions.
- FIPP focuses on CPU performance, while FAPP offers deeper insights into a broader range of performance metrics.

Unique Insights from FIPP and FAPP

Profiling Insights

- FIPP provides summaries for aggregate performance metrics and identifies tasks with the highest computational time.
- FAPP helps contextualize performance relative to hardware capabilities, offering insights into CPU utilization and memory throughput.

Visuals

- A summary output example from FIPP showing the lines in the source program ordered by computational time (Figure 8).
- A report generated by FAPP detailing CPU performance metrics and execution time (Figure 7).

Additional Profiling Figures

```
Time statistics

        Elapsed(s)        User(s)      System(s)
   ------------------------------------------
          38.5575             --             --   Application
   ------------------------------------------
          38.5575             --             --   Process      5
          38.5567             --             --   Process      7
          38.5563             --             --   Process      4
          38.5561             --             --   Process      3
          38.5557             --             --   Process      0
          38.5540             --             --   Process      2
          38.5540             --             --   Process      1
          38.5419             --             --   Process      6
          38.5378             --             --   Process      8
          38.5216             --             --   Process     12
          38.5183             --             --   Process     11
          38.5058             --             --   Process     13
          38.5055             --             --   Process      9
          38.4910             --             --   Process     10
          38.4784             --             --   Process     14
          38.4605             --             --   Process     16

Performance monitor event:statistics
```

Figure: Head part of FIPP Profiling data for 2decomp&fft

Additional Profiling Figures

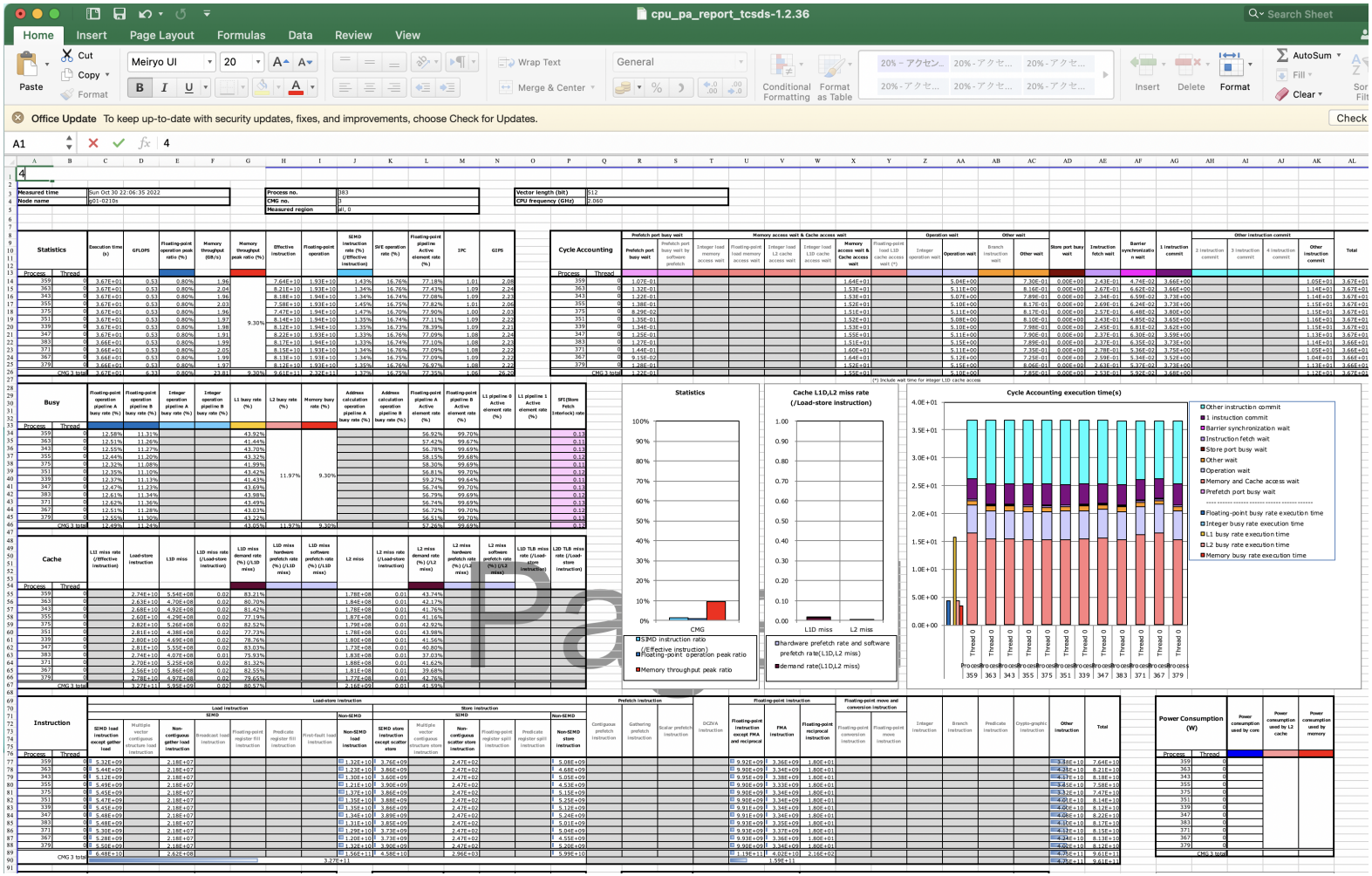


Figure: FAPP CPU Performance Analysis Report for the 2Decomp&FFT executed on 384 tasks for reading the Pa1,Pa2,Pa3,Pa4 and Pa5 reports to obtain the Brief Report

Conclusion

Performance tools are essential for optimizing applications utilizing distributed FFTs. A well-chosen pencil decomposition can improve the performance of MPI calls. Additionally, tools like FIPP and FAPP complement TAU and Extrae by providing targeted insights into CPU performance and memory utilization. This enhances the overall understanding of application behavior on high-performance computing platforms. The study recommends further research into optimizing startup time and exploring non-blocking communication calls.

References

- Aseeri, S., et al. (2015). Solving the Klein-Gordon Equation Using Fourier Spectral Methods.
- Leu, B., et al. (2021). A Comparison of Parallel Profiling Tools for Programs Utilizing the FFT.
- Various other studies and reports as referenced in the paper.