MS354 and MS386
Next Generation FFT Algorithms in Theory and Practice: Parallel Implementations, Sparse FFTs, and Applications

- **Organizers:**
  - **Daisuke Takahashi**
    University of Tsukuba, Japan
  - **Mark Iwen**
    Michigan State University, U.S.
  - **Samar A. Aseeri**
    King Abdullah University of Science & Technology (KAUST), Saudi Arabia
  - **Benson K. Muite**
    University of Tartu, Estonia

Aim of this minisymposium

- The fast Fourier Transform (FFT) is an algorithm used in a wide variety of applications, yet does not make optimal use of many current hardware platforms.
- Hardware utilization performance on its own does not however imply optimal problem solving.
- The purpose of this minisymposium is to enable exchange of information between people working on alternative FFT algorithms such as sparse and non uniform FFTs, to those working on FFT implementations, in particular for parallel hardware.
- http://www.fft.report

MS354: Part I of II

- **9:45-10:05 Implementation of Parallel 3-D Real FFT with 2-D Decomposition on Intel Xeon Phi Clusters**
  Daisuke Takahashi, University of Tsukuba, Japan
- **10:10-10:30 Discrete Sparse Fourier Transforms: Faster Stable Implementations with Guarantees**
  Mark Iwen, Sami Merhi, and Ruochuan Zhang, Michigan State University, U.S.
- **10:35-10:55 Getting Best Performance of Memory Bandwidth Limited Algorithms with Intel MKL**
  Alexander Kalinkin, Intel Corporation, U.S.
- **11:00-11:20 Implementation of Sparse FFT with Structured Sparsity**
  Sina Bittens, University of Goettingen, Germany; *Ruochuan Zhang and Mark Iwen*, Michigan State University, U.S.

MS386: Part II of II

- **11:30-11:50 FFT Applications and Benchmarks**
  Samar A. Aseeri, King Abdullah University of Science & Technology (KAUST), Saudi Arabia
- **11:55-12:15 High-dimensional Sparse FFT**
  Bosu Choi, University of Texas at Austin, U.S.; *Andrew J. Christlieb*, Michigan State University, U.S.; *Yang Wang*, Hong Kong University of Science and Technology, Hong Kong
- **12:20-12:40 Rank-1 Lattice Based High-dimensional Approximation and FFT**
  Toni Volkmer, Chemnitz University of Technology, Germany
- **12:45-1:05 A Periodic Treecode Method for Electrostatics In Molecular Dynamics Simulations**
  Henry A. Boateng, Bates College, U.S.

Implementation of Parallel 3-D Real FFT with 2-D Decomposition on Intel Xeon Phi Clusters

Daisuke Takahashi

Center for Computational Sciences
University of Tsukuba, Japan

Outline

- Background
- Related Works
- Objectives
- 3-D FFT with 2-D Decomposition
- In-Cache FFT Algorithm and Vectorization
- Performance Results
- Conclusion

Background (1/2)

- The fast Fourier transform (FFT) is an algorithm widely used today in science and engineering.
- Parallel 3-D FFT algorithms on distributed-memory parallel computers have been well studied.
- November 2018 TOP500 Supercomputing Sites
  - Summit: 143,500.0 TFlops (2,397,824 Cores)
  - Sierra: 94,640.0 TFlops (1,572,480 Cores)
  - Sunway TaihuLight: 93,014.6 TFlops (10,649,600 Cores)
- Recently, the number of cores keeps increasing.

Background (2/2)

- A typical decomposition for performing a parallel 3-D FFT is slabwise.
  - A 3-D array $x(N_1, N_2, N_3)$ is distributed along the third dimension N_3.
  - N_3 must be greater than or equal to the number of MPI processes.
- This becomes an issue with very large MPI process counts for a massively parallel cluster of many-core processors.

Related Works

- P3DFFT [Pekurovsky 2012]
  - 3-D real-to-complex/complex-to-real FFT with 2-D decomposition
- 2DECOMP&FFT [Li and Laizet 2010]
  - 3-D complex-to-complex and real-to-complex/ complex-to-real FFT with 2-D decomposition
- PFFT [Pippig 2013]
  - 3-D complex-to-complex and real-to-complex/ complex-to-real FFT with 2-D decomposition

Objectives

- Implementation and evaluation of highly scalable 3-D real FFT with 2-D decomposition on Intel Xeon Phi clusters.
- Reduce the communication time for larger numbers of MPI processes.
- A comparison between 1-D and 2-D decomposition for 3-D real FFT.

3-D DFT

- 3-D discrete Fourier transform (DFT) is given by

$$y(k_1, k_2, k_3) = \sum_{j_1=0}^{n_1-1} \sum_{j_2=0}^{n_2-1} \sum_{j_3=0}^{n_3-1} x(j_1, j_2, j_3)\, \omega_{n_3}^{j_3 k_3} \omega_{n_2}^{j_2 k_2} \omega_{n_1}^{j_1 k_1},$$

$$0 \le k_r \le n_r - 1, \omega_{n_r} = e^{-2\pi i / n_r}, 1 \le r \le 3$$

Real DFT

- When the input data of the DFT are real, two n-point real DFTs can be computed using an n-point complex DFT.
- Let

$$x_j = a_j + ib_j, \qquad 0 \le j \le n-1,$$

where $a_0, a_1, \cdots, a_{n-1}$ and $b_0, b_1, \cdots, b_{n-1}$ are n-point real input data.

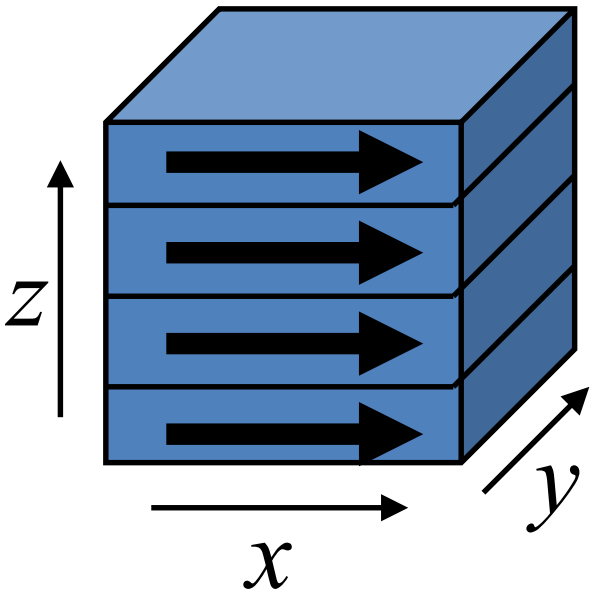
- We obtain two n-point real DFTs as follows:

$$\begin{aligned} X_k &= A_k + iB_k \\ \overline{X_{n-k}} &= A_k - iB_k \\ A_k &= \frac{1}{2}(X_k + \overline{X_{n-k}}) \\ B_k &= -\frac{i}{2}(X_k - \overline{X_{n-k}}), \qquad 0 \le k \le n/2, \end{aligned}$$

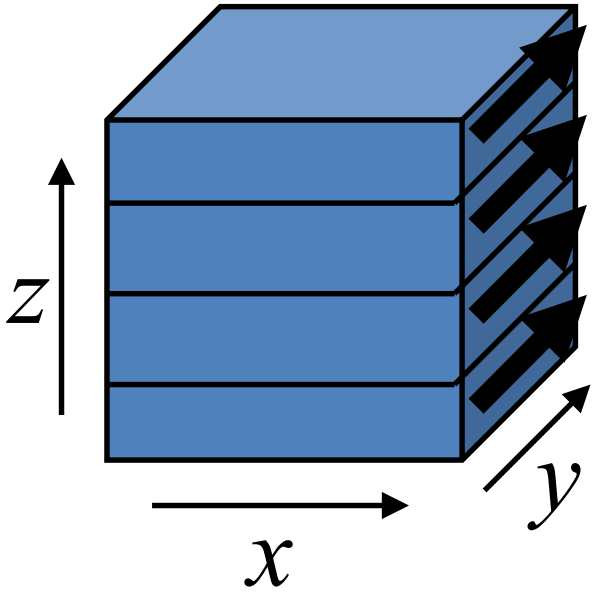
where $A_0, A_1, \cdots, A_{n/2}$ and $B_0, B_1, \cdots, B_{n/2}$ are $(n/2+1)$-point complex output data.

1-D Decomposition along the z-axis

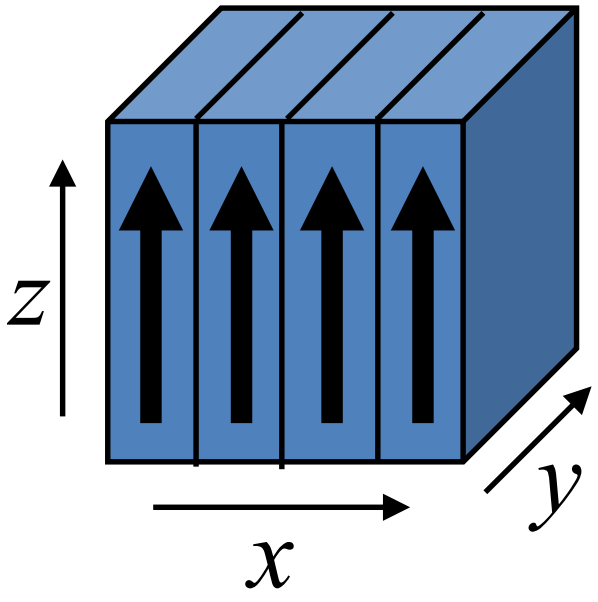
1. FFTs in x-axis



2. FFTs in y-axis



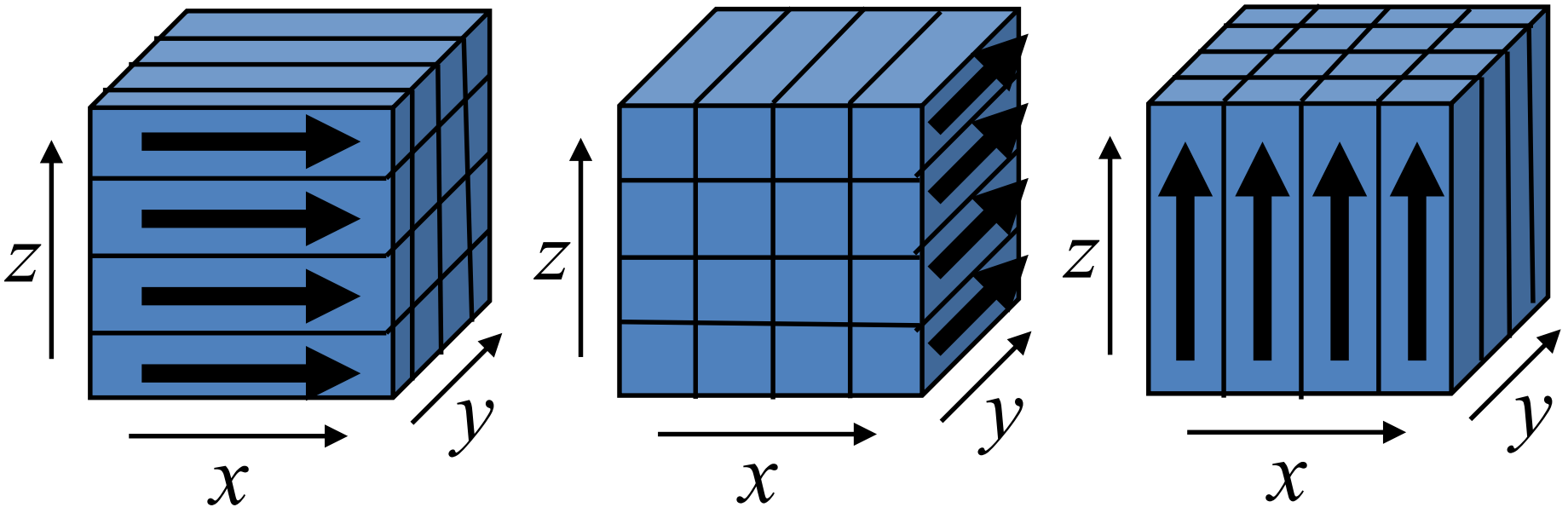
3. FFTs in z-axis



With a slab decomposition

2-D Decomposition along the y- and z-axes

1. FFTs in x-axis 2. FFTs in y-axis 3. FFTs in z-axis



With a pencil decomposition

Communication Time of 1-D Decomposition

- Let us assume for $N = N_1 \times N_2 \times N_3$-point real FFT:
  - Latency of communication: L (sec)
  - Bandwidth: W (byte/sec)
  - The number of MPI processes: $P \times Q$
- One all-to-allv communication among $P \times Q$ MPI processes
- Communication time of 1-D decomposition

$$T_{1\mathrm{dim}} \approx (PQ-1)\left(L + \frac{8N}{(PQ)^2 \cdot W}\right)$$
$$\approx PQ \cdot L + \frac{8N}{PQ \cdot W}\ (\mathrm{sec})$$

Communication Time of 2-D Decomposition

- Q simultaneous all-to-allv communications among P MPI processes in the y-axis.
- P simultaneous all-to-all communications among Q MPI processes in the z-axis.
- Communication time of 2-D decomposition

$$T_{2\mathrm{dim}} \approx (P-1)\left(L + \frac{8N}{P^2Q \cdot W}\right) + (Q-1)\left(L + \frac{8N}{PQ^2 \cdot W}\right)$$

$$\approx (P+Q) \cdot L + \frac{16N}{PQ \cdot W}\ (\mathrm{sec})$$

Comparing Communication Time

- Communication time of 1-D decomposition

$$T_{1\mathrm{dim}} \approx PQ \cdot L + \frac{8N}{PQ \cdot W} \ (\mathrm{sec})$$

- Communication time of 2-D decomposition

$$T_{2\mathrm{dim}} \approx (P + Q) \cdot L + \frac{16N}{PQ \cdot W} \ (\mathrm{sec})$$

- By comparing two equations, the communication time of the 2-D decomposition is less than that of the 1-D decomposition for larger number of MPI processes $P \times Q$ and latency L.

In-Cache FFT Algorithm and Vectorization

- For in-cache FFT, we used radix-2, 3, 4, 5, and 8 FFT algorithms based on the mixed-radix FFT algorithms [Temperton 83].
- Automatic vectorization was used to access the Intel AVX-512 instructions on the Knights Landing processor.
- Although higher radix FFTs require more floating-point registers to hold intermediate results, the Knights Landing processor has 32 ZMM 512-bit registers.

Optimization of Parallel 3-D Real FFT on Knights Landing Processor

```
    COMPLEX*16 A(NNYY*NNZZ,*),B(NX/2+1,*),C(NY,*)
!$OMP PARALLEL DO COLLAPSE(2) PRIVATE(I,J,JJ)
    DO II=1,NX/2+1,NB
       DO JJ=1,NNYY*NNZZ,NB
          DO I=II,MIN(II+NB-1,NX/2+1)
             DO J=JJ,MIN(JJ+NB-1,NNYY*NNZZ)
                A(J,I)=B(I,J)
             END DO
          END DO
       END DO
    END DO
    …
!$OMP PARALLEL DO
    DO K=1,NNZZ*(NNXY/2+1)
       CALL IN_CACHE_FFT(C(1,K),NY)
    END DO
```

To expand the outermost loop, the double-nested loop can be collapsed into a single-nested loop.

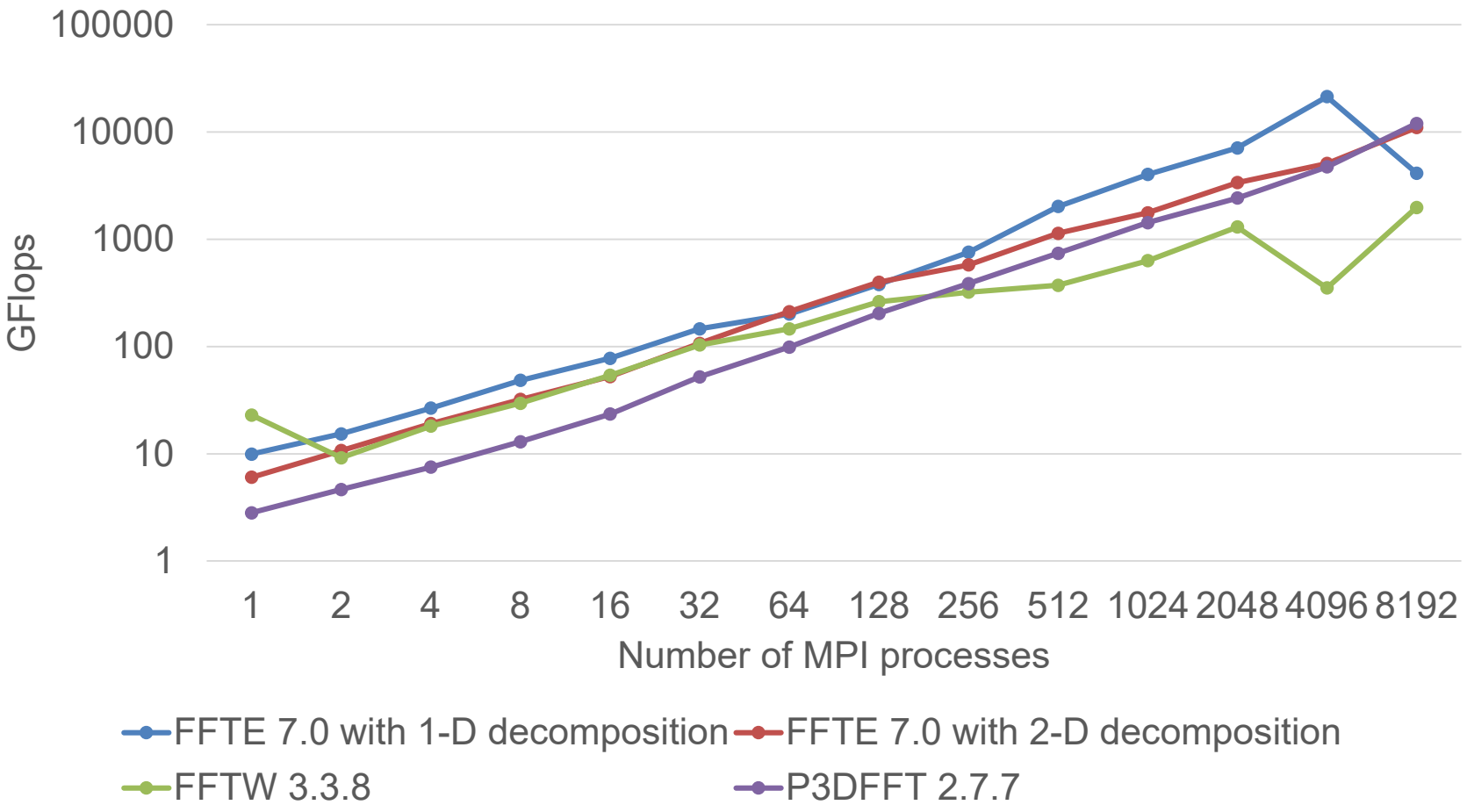
Performance Results

- To evaluate the parallel 3-D real FFT with 2-D decomposition, we compared
  - The implemented parallel 3-D real FFT, referred to as FFTE (version 7.0)
  - FFTW (version 3.3.8)
  - P3DFFT (version 2.7.7)
- Weak scaling ($N = 256 \times 512 \times 512 \times$ MPI processes) and strong scaling ($N = 256 \times 512 \times 512$) were measured.

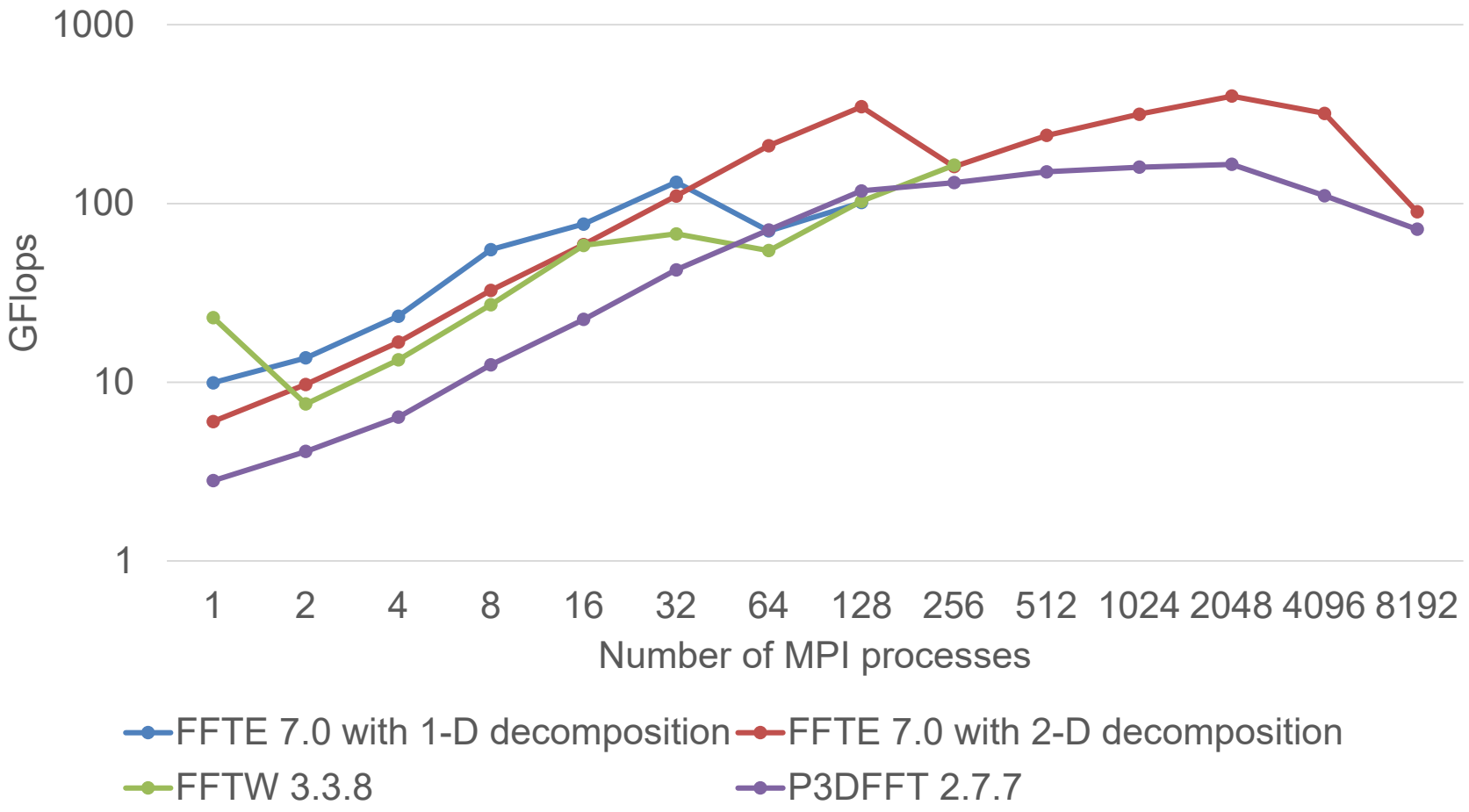
Evaluation Environment

- Oakforest-PACS at Joint Center for Advanced HPC (JCAHPC).
  - 8208 nodes, Peak 25.008 PFlops
  - CPU: Intel Xeon Phi 7250 (68 cores, Knights Landing 1.4 GHz)
  - Interconnect: Intel Omni-Path Architecture
  - Compiler: Intel Fortran compiler 18.0.1.163 (for FFTE and P3DFFT)
    Intel C compiler 18.0.1.163 (for FFTW and P3DFFT)
  - Compiler option: “-O3 -xMIC-AVX512 -qopenmp”
  - MPI library: Intel MPI 2018.1.163
  - flat/quadrant, MCDRAM only, KMP_AFFINITY=balanced
  - Each MPI process has 16 cores and 64 threads,
    i.e. 4 MPI processes per node.

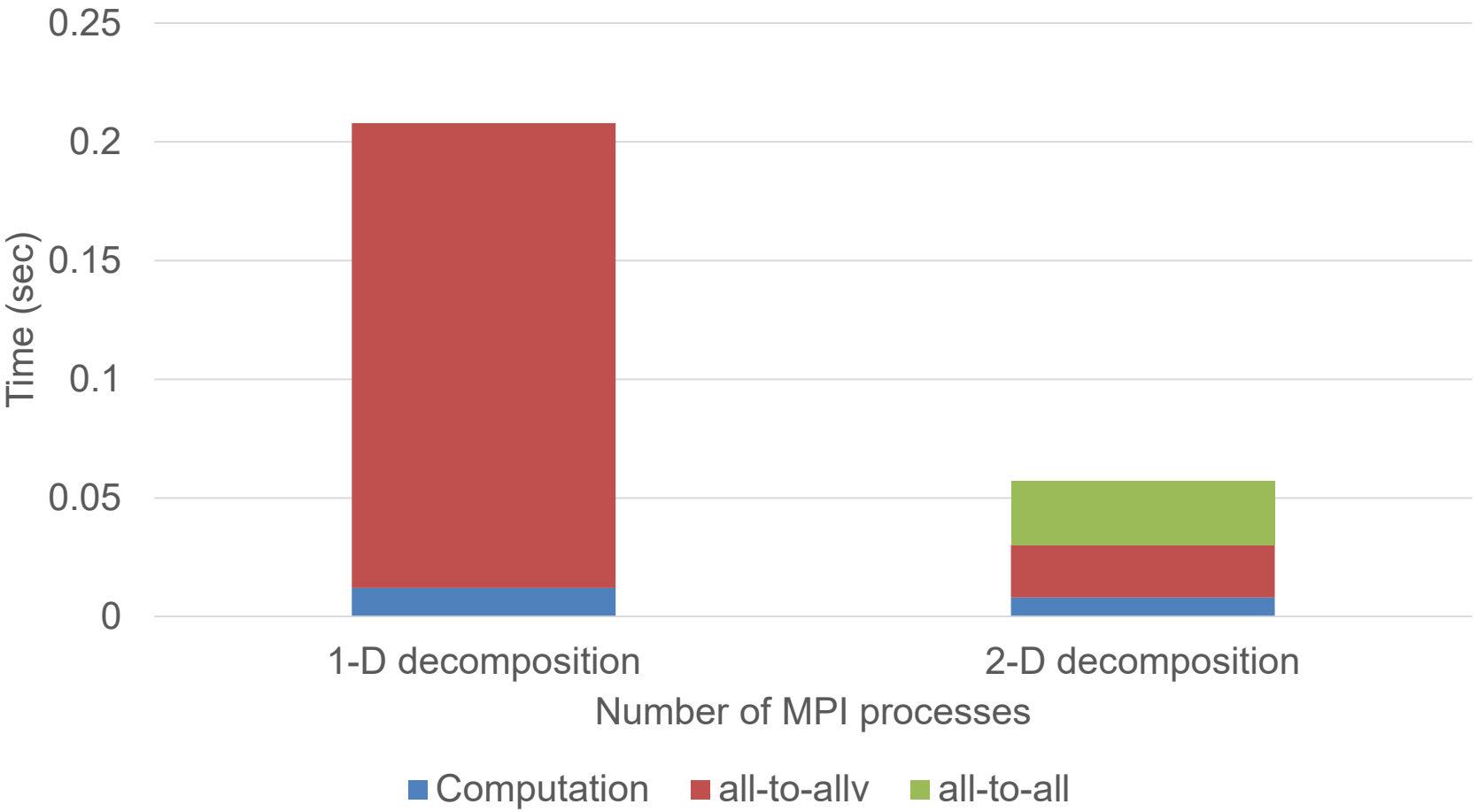
Performance of Parallel 3-D Real FFTs ($N = 256 \times 512 \times 512 \times$ MPI processes)



Performance of Parallel 3-D Real FFTs ($N = 256 \times 512 \times 512$)



Breakdown of Execution Time in FFTE 7.0 ($N = 1024^3$, 512 MPI processes)



Conclusion

- We proposed an implementation of parallel 3-D real FFT with 2-D decomposition on Intel Xeon Phi clusters.
- The proposed parallel 3-D real FFT algorithm is based on the conjugate symmetry property for the DFT and the multicolumn FFT algorithm.
- We showed that a 2-D decomposition effectively improves performance by reducing the communication time for larger numbers of MPI processes.
- The performance results demonstrate that the proposed implementation of a parallel 3-D real FFT with 2-D decomposition is efficient for improving the performance on Intel Xeon Phi clusters.