Automatic Tuning for Parallel FFTs on Intel Xeon Phi Clusters

Daisuke Takahashi

University of Tsukuba, Japan

Outline

- Background
- Objectives
- Six-Step FFT Algorithm
- Overlapping Communication and Computation
- Automatic Tuning for Parallel 1-D FFT on Intel Xeon Phi Clusters
- Performance Results
- Conclusion

Background

- Many FFT implementations work well within Intel Xeon Phi cards, such as Intel MKL's FFT routine.
- However, PCI Express transfer is often a performance bottleneck in FFT because FFT requires a large number of memory accesses per arithmetic operation.
- One goal for parallel FFTs on Intel Xeon Phi clusters is to minimize the PCI Express transfer time and the MPI communication time.
- Several FFT libraries with automatic tuning have been proposed.
  - FFTW, SPIRAL, and UHFFT

Objectives

- An Implementation of parallel 1-D FFT on Intel Xeon Phi cluster has been presented [Park et al. 2013].
- Auto-tuning parallel 3-D FFT for computation-communication overlap has also been presented [Song and Hollingsworth 2014].
- However, to the best of our knowledge, parallel 1-D FFT with automatic tuning on Intel Xeon Phi clusters has not yet been reported.
- We propose an implementation of a parallel 1-D FFT with automatic tuning on Intel Xeon Phi clusters.

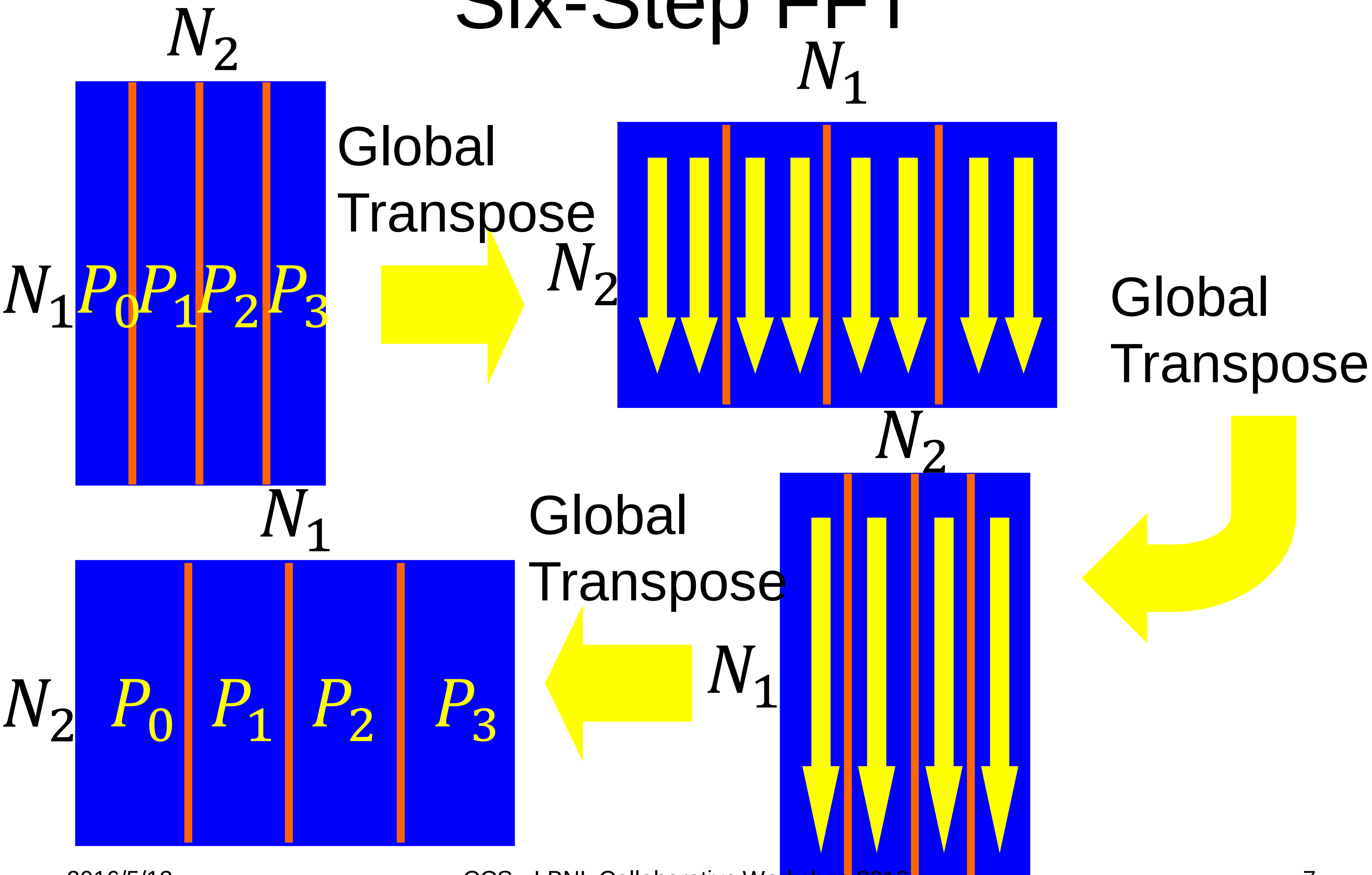
Approach

- Our implemented parallel 1-D FFT is based on the six-step FFT algorithm.
- The six-step FFT algorithm improves performance by utilizing the cache memory effectively.
- Parallel FFTs on Intel Xeon Phi clusters require intensive all-to-all communication, which affects the performance of parallel FFTs.
- How to overlap the computation and the all-to-all communication is one of the issues in automatic tuning for parallel FFTs.

Six-Step FFT Algorithm [Bailey90, VanLoan92]

- Step 1: Transpose
- Step 2: Compute n_1 individual n_2-point multicolumn FFTs
- Step 3: Apply twiddle factor ($\omega_{n_1 n_2}^{j_1 k_2}$) multiplication
- Step 4: Transpose
- Step 5: Compute n_2 individual n_1-point multicolumn FFTs
- Step 6: Transpose

Parallel 1-D FFT Algorithm Based on Six-Step FFT



Optimization of Parallel 1-D FFT on Intel Xeon Phi

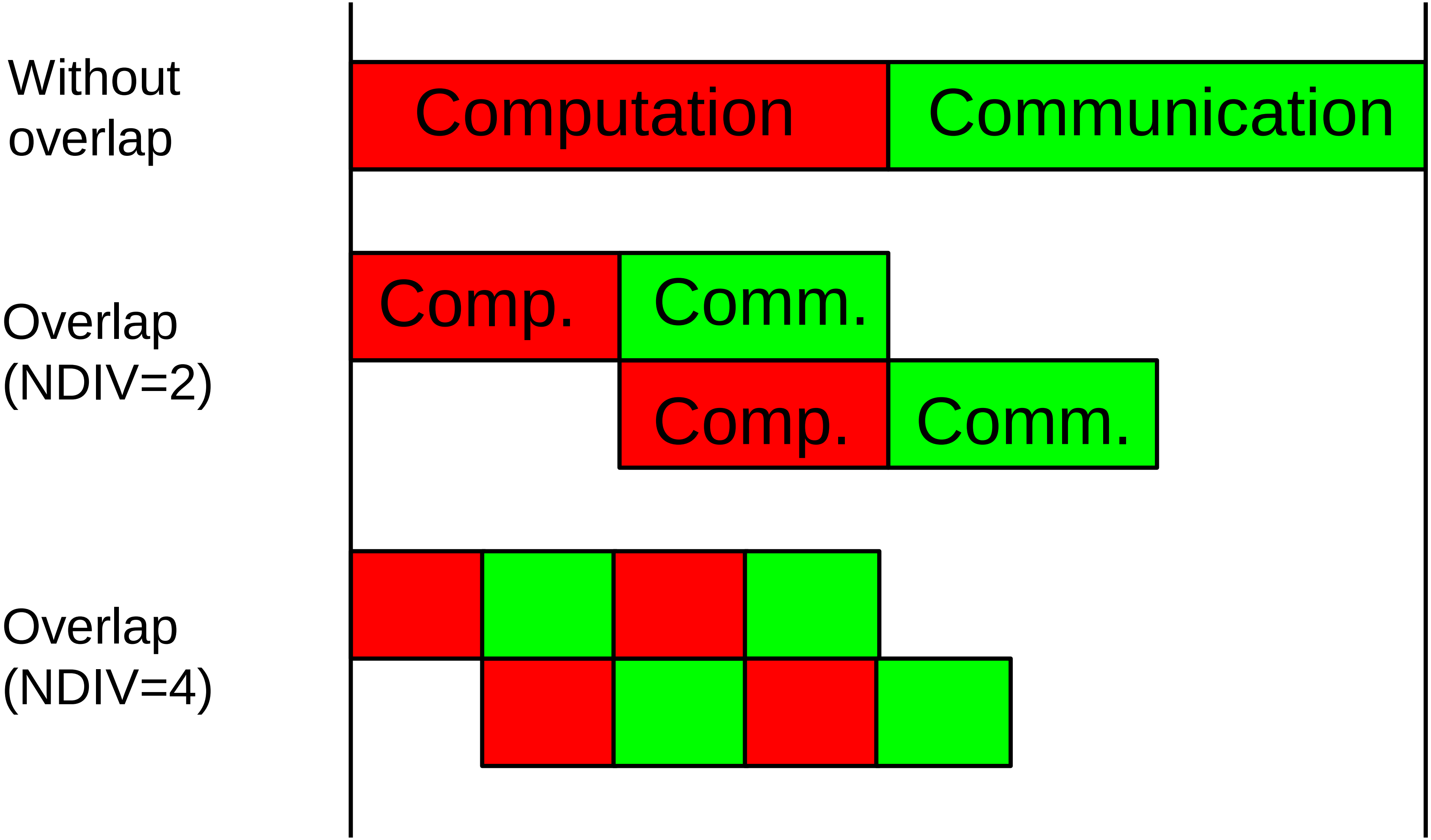
```
      COMPLEX*16 X(N1,N2),Y(N2,N1),WORK2(N1+NP,N2)
!$OMP PARALLEL DO COLLAPSE(2) PRIVATE(I,J,JJ)
      DO II=1,N1,NB
        DO JJ=1,N2,NB
          DO I=II,MIN0(II+NB-1,N1)
            DO J=JJ,MIN0(JJ+NB-1,N2)
              WORK2(J,I)=X(I,J)
            END DO
          END DO
        END DO
      END DO
!$OMP PARALLEL DO
      DO I=1,N1
        CALL IN_CACHE_FFT(WORK2(1,I),N2)
      END DO
      …
```

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

Overlapping Computation and Communication [Idomura et al. 2014]

```
!$OMP PARALLEL
!$OMP MASTER
    MPI communication          ← Communication on master thread
!$OMP END MASTER ← No barrier synchronization
!$OMP DO SCHEDULE(DYNAMIC)
    DO I=1,N
       Computation             ← Computation on other than master thread
    END DO
!$OMP DO ← Implicit barrier synchronization
    DO I=1,N
       Computation using the result of communication
    END DO
!$OMP END PARALLEL
```

Effect of Overlapping Computation and Communication



Automatic Tuning of Parallel 1-D FFT on Intel Xeon Phi Clusters

- An automatic tuning process consists of three steps:
  - Selection of the number of divisions for overlapping communication and computation
  - Selection of the radices (N_1 and N_2)
  - Selection of the block size NB

Selection of the Number of Divisions for Overlapping Communication and Computation

- When we increase the number of divisions for overlapping communication and computation, the overlap ratio also increases.
- On the other hand, the message size is decreased due to split the all-to-all communication.
- Thus, a tradeoff exists between the overlap ratio and the all-to-all communication performance.
- The default overlapping parameter of the original FFTE 6.2beta is NDIV=4.
- In our implementation, the overlapping parameter NDIV is varied with 1, 2, 4, 8 and 16.

Selection of the Radices

- If the condition of $N = N_1 \times N_2$ is satisfied, then we can select the arbitrary N_1 and N_2, where N_1,$N_2 \geq P$.
- We need to select the best combination and order of N_1 and N_2 for computing parallel 1-D FFT.
- If N and P are a power of two, N_1 is varied with $P, 2P, \dots, \sqrt{N}$, then $N_2 = N/\ N_1$.
- In this case, the size of search space is $\log_2(\sqrt{N}/P)$.

Selection of the Block Size

- The default blocking parameter of the original FFTE 6.2beta is NB=8.
- Although the optimal block size may depend on the problem size, the block size NB can also be varied.
- In our implementation, the block size NB is varied with 4, 8, 16, 32 and 64.

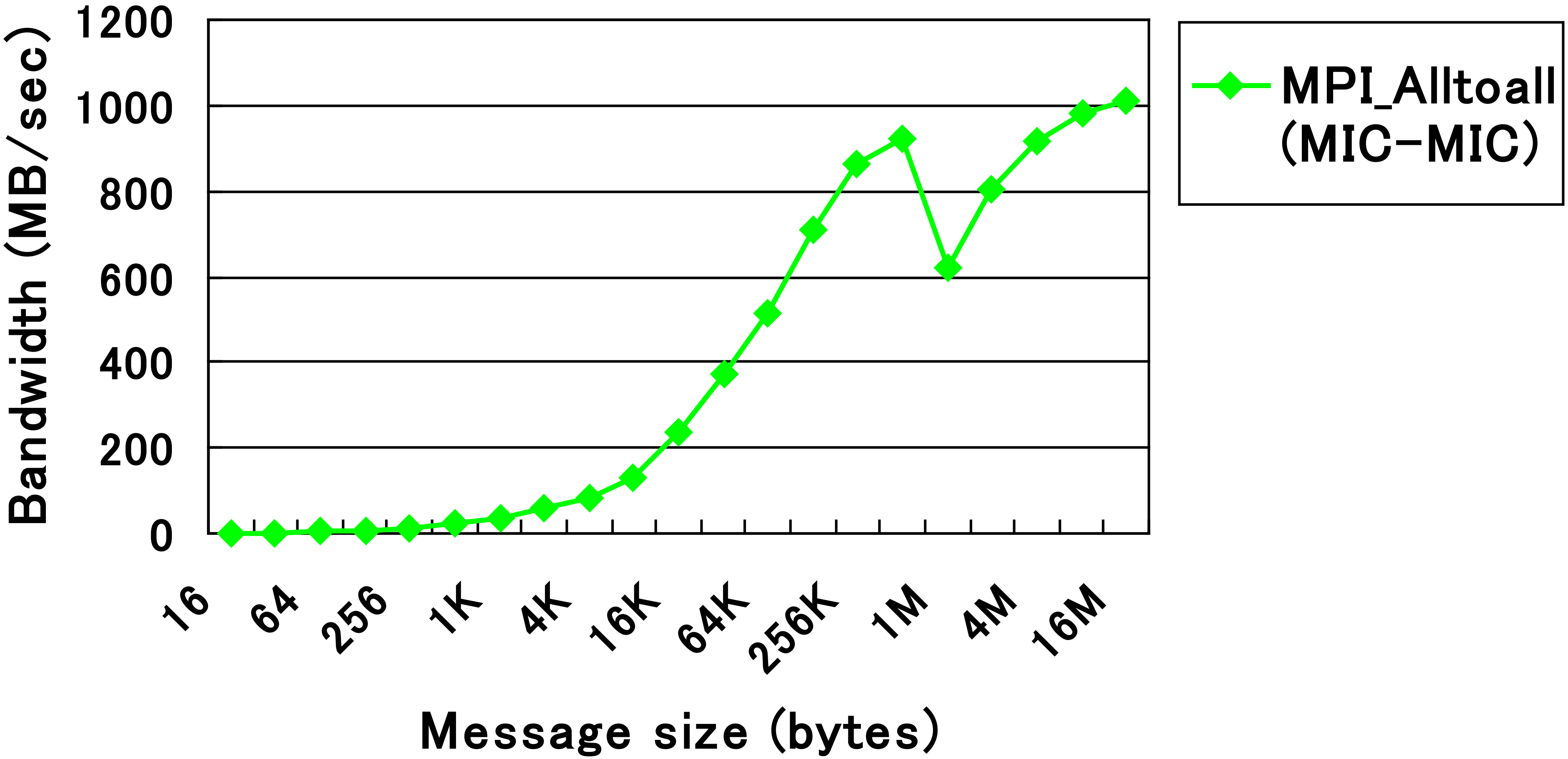
Performance Results

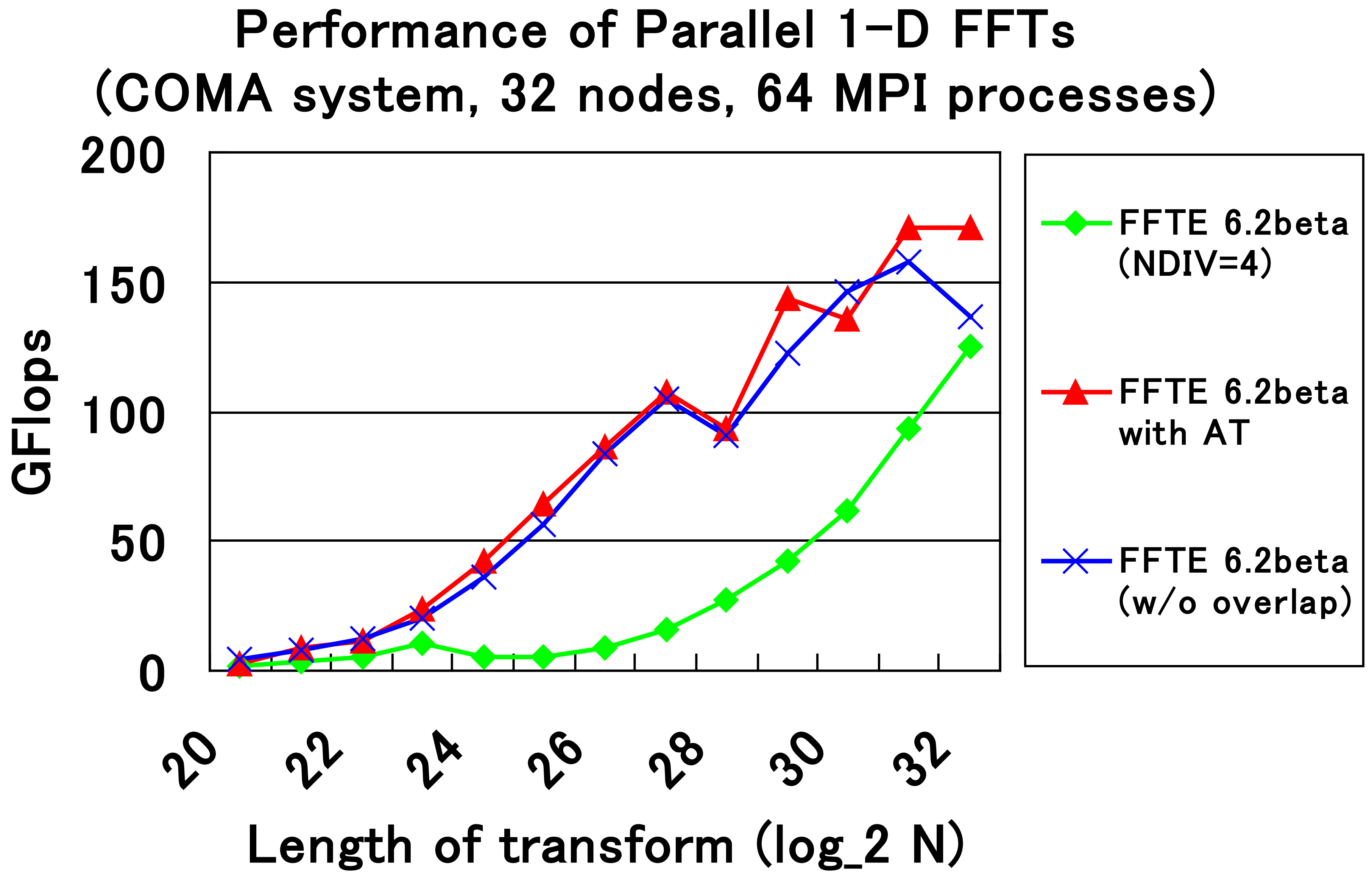
- To evaluate the parallel 1-D FFT with automatic tuning, we compared its performance against that of the FFTE 6.2beta (http://www.ffte.jp/) and that of the FFTE 6.2beta with AT.
- Target parallel machine: COMA system@U. Tsukuba
  - 393 nodes, 7860 cores, 786 Xeon Phis, Peak 1.001 PFlops
  - Intel Xeon E5-2670 v2 (Ivy Bridge-EP 2.5GHz, 10-core) x 2 + Intel Xeon Phi 7110P (1.1GHz, 61-core) x 2
  - All the nodes in the system are connected through a full-bisectional fat-tree network with FDR InfiniBand.
  - Intel MPI 5.1.1 was used as a communication library.
  - Intel Fortran Compiler 15.0.3 with "-O3 -mmic -openmp".
  - 244 threads per Xeon Phi were used in native execution model.

Results of Automatic Tuning of 1-D FFTs on COMA (32 nodes, 64 MPI processes)

	FFTE 6.2beta					FFTE 6.2beta with AT				
# MPI	N1	N2	NB	NDIV	GFlops	N1	N2	NB	NDIV	GFlops
4	16K	16K	8	4	15.3	4K	64K	32	8	15.8
8	16K	32K	8	4	28.6	32K	16K	32	16	29.6
16	32K	32K	8	4	50.6	32K	32K	32	16	57.0
32	32K	64K	8	4	93.0	64K	32K	64	8	107.6
64	64K	64K	8	4	124.7	64K	64K	64	4	171.3

Performance of All-to-All communication (COMA system, 32 nodes, 64 MPI processes)





Performance of Parallel 1-D FFTs
(COMA system, 32 nodes, 64 MPI processes)
GFlops
200
150
100
50
0
20
22
24
26
28
30
32
Length of transform (log_2 N)
FFTE 6.2beta (NDIV=4)
FFTE 6.2beta with AT
FFTE 6.2beta (w/o overlap)

Conclusion

- We proposed an implementation of parallel 1-D FFT with automatic tuning on Intel Xeon Phi clusters.
- An automatic tuning facility for selecting the optimal parameters of the number of divisions for overlapping communication and computation, the radices and the block size is implemented.
- The performance results demonstrate that the proposed implementation of parallel 1-D FFT with automatic tuning is efficient for improving the performance on Intel Xeon Phi clusters.