An Implementation of Parallel 3-D FFT with 1.5-D Decomposition

Daisuke Takahashi, Alex Yee, Torsten Hoefler, Camille Coti, Jeongnim Kim and Franck Cappello

Outline

- Background
- Related works
- Approach
- 3-D FFT with 4-D formulation
- Parallel 3-D FFT algorithm with 1.5-D decomposition
- 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 2011 TOP500 Supercomputing Sites
  - K computer (SPARC VIIIfx 8-core 2 GHz) 10.51 PFlops (705,024 Cores)
  - Tianhe-1A (X5670 2.93 GHz 6-core, NVIDIA C2050) 2.566 PFlops (186,368 Cores)
  - Jaguar (Cray XT5-HE 6-core 2.6 GHz) 1.759 PFlops (224,162 Cores)
- Recently, the number of cores keeps increasing.

Background (2/2)

- A typical decomposition for performing a parallel 3-D FFT is slab-wise.
  - 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 node counts for a massively parallel cluster of multi-core processors.

Related Works

- Scalable framework for 3-D FFTs on the Blue Gene/L supercomputer [Eleftheriou et al. 03, 05]
  - Based on a volumetric (3-D) decomposition of data.
  - Scale well up to 1,024 nodes for 3-D FFTs of size 128x128x128.
- P3DFFT [Pekurovsky 08]
  - Based on a pencil (2-D) decomposition.
  - Supports forward (real-to-complex) and backward (complex-to-real) 3-D FFTs.
- 2DECOMP&FFT [Li 10]
  - Based on the 2-D decomposition.
  - Supports 3-D FFTs (both complex-to-complex/complex-to-real).

Approach

- Some conventional parallel 3-D FFT algorithms [Pekurovsky 08, Li 10] use the 2-D decomposition.
  - These algorithms allow up to N^2 MPI processes for N^3-point FFT.
  - These schemes require two all-to-all communications for transposed order output.
- We use a "1.5-D" decomposition for 3-D FFT.
  - Our proposed parallel 3-D FFT algorithm allows up to $N^{1.5}$ MPI processes for N^3-point FFT.
  - For example, 4096^3-point FFT can be performed on up to 262,144 MPI processes.
  - This scheme requires only one all-to-all communication for transposed order output.

3-D FFT

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

$$y(k_1,k_2,k_3)=\sum_{j_1=0}^{n_1-1}\sum_{j_2=0}^{n_2-1}\sum_{j_3=0}^{n_3-1}x(j_1,j_2,j_3)\omega_{n_3}^{j_3k_3}\omega_{n_2}^{j_2k_2}\omega_{n_1}^{j_1k_1},$$

where $0 \le k_r \le n_r - 1,\ \omega_{n_r} = \exp(-2\pi i / n_r)$

and $1 \le r \le 3.$

3-D FFT with 4-D Formulation

- If n_2 has factors n_{21} and n_{22} $(n_2 = n_{21} n_{22})$, then the indices j_2 and k_2 can be expressed as:

$$j_2 = j_{21} + j_{22} n_{21}, \quad k_2 = k_{22} + k_{21} n_{22}.$$

- We can derive the following equation:

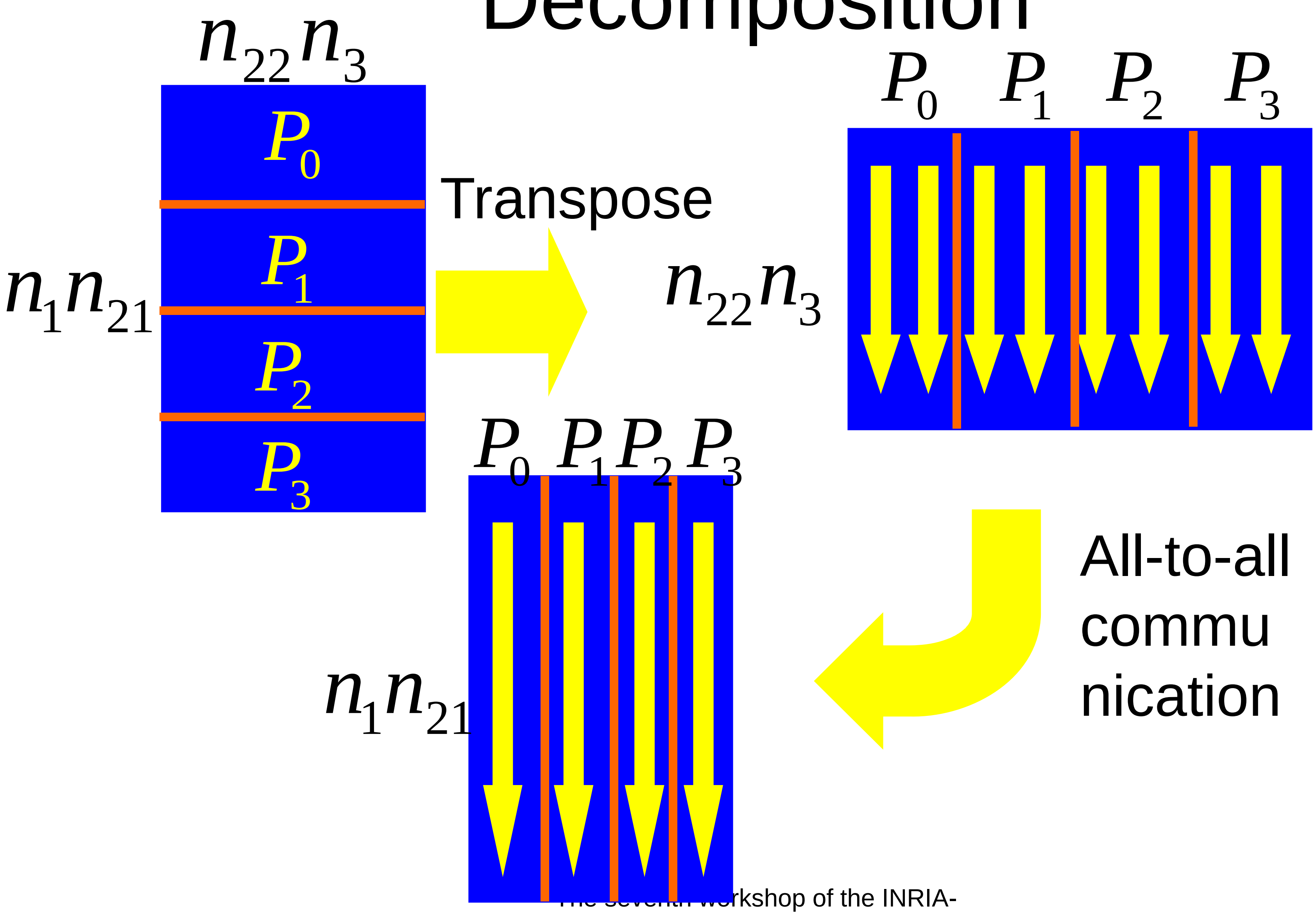
$$y(k_1, k_{22}, k_{21}, k_3) = \sum_{j_1=0}^{n_1-1} \sum_{j_{21}=0}^{n_{21}-1} \sum_{j_{22}=0}^{n_{22}-1} \sum_{j_3=0}^{n_3-1} x(j_1, j_{21}, j_{22}, j_3)$$

$$\omega_{n_3}^{j_3 k_3} \omega_{n_{22}}^{j_{22} k_{22}} \omega_{n_2}^{j_{21} k_{22}} \omega_{n_{21}}^{j_{21} k_{21}} \omega_{n_1}^{j_1 k_1}.$$

3-D FFT Algorithm with 4-D Formulation

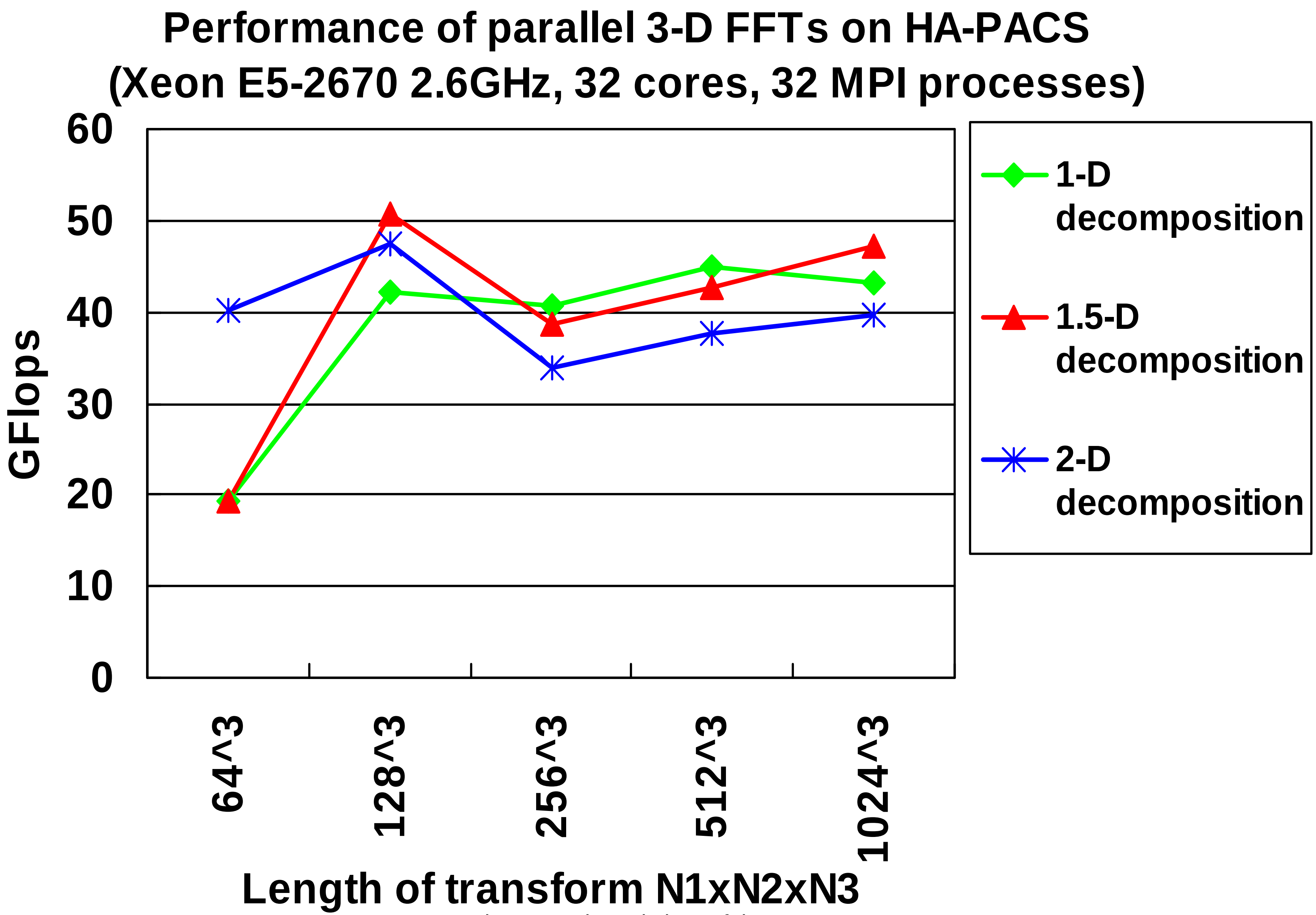
1. Transpose
2. $n_1 n_{21} n_{22}$ individual n_3 -point multicolumn FFTs
3. Transpose
4. $n_3 n_1 n_{21}$ individual n_{22} -point multicolumn FFTs
5. Twiddle factor ($\omega_{n_2}^{j_{21} k_{22}}$) multiplication
6. Transpose
7. $n_{22} n_3 n_1$ individual n_{21}-point multicolumn FFTs
8. Transpose
9. $n_{21} n_{22} n_3$individual n_1 -point multicolumn FFTs

Parallel 3-D FFT Algorithm with 1.5-D Decomposition

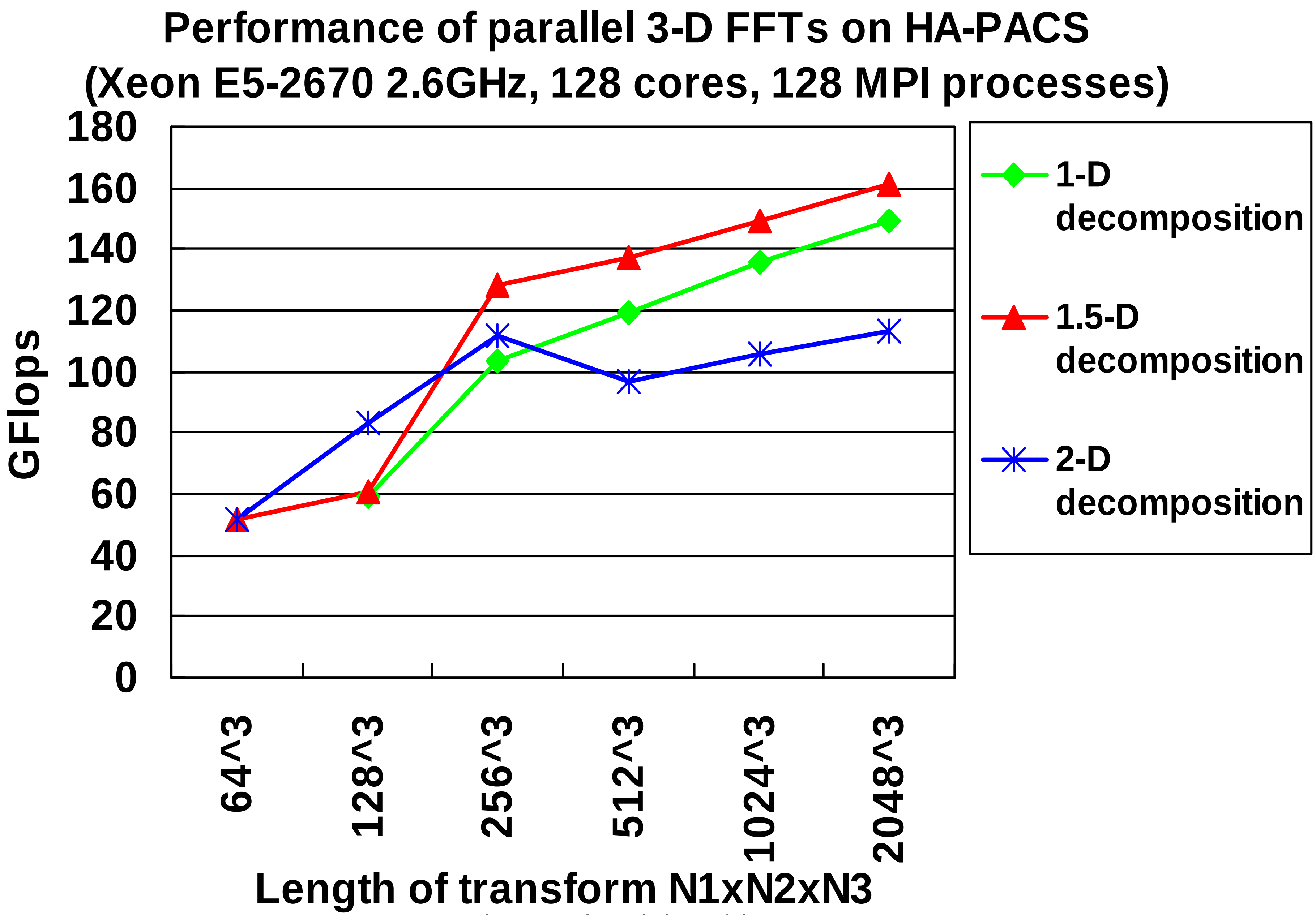


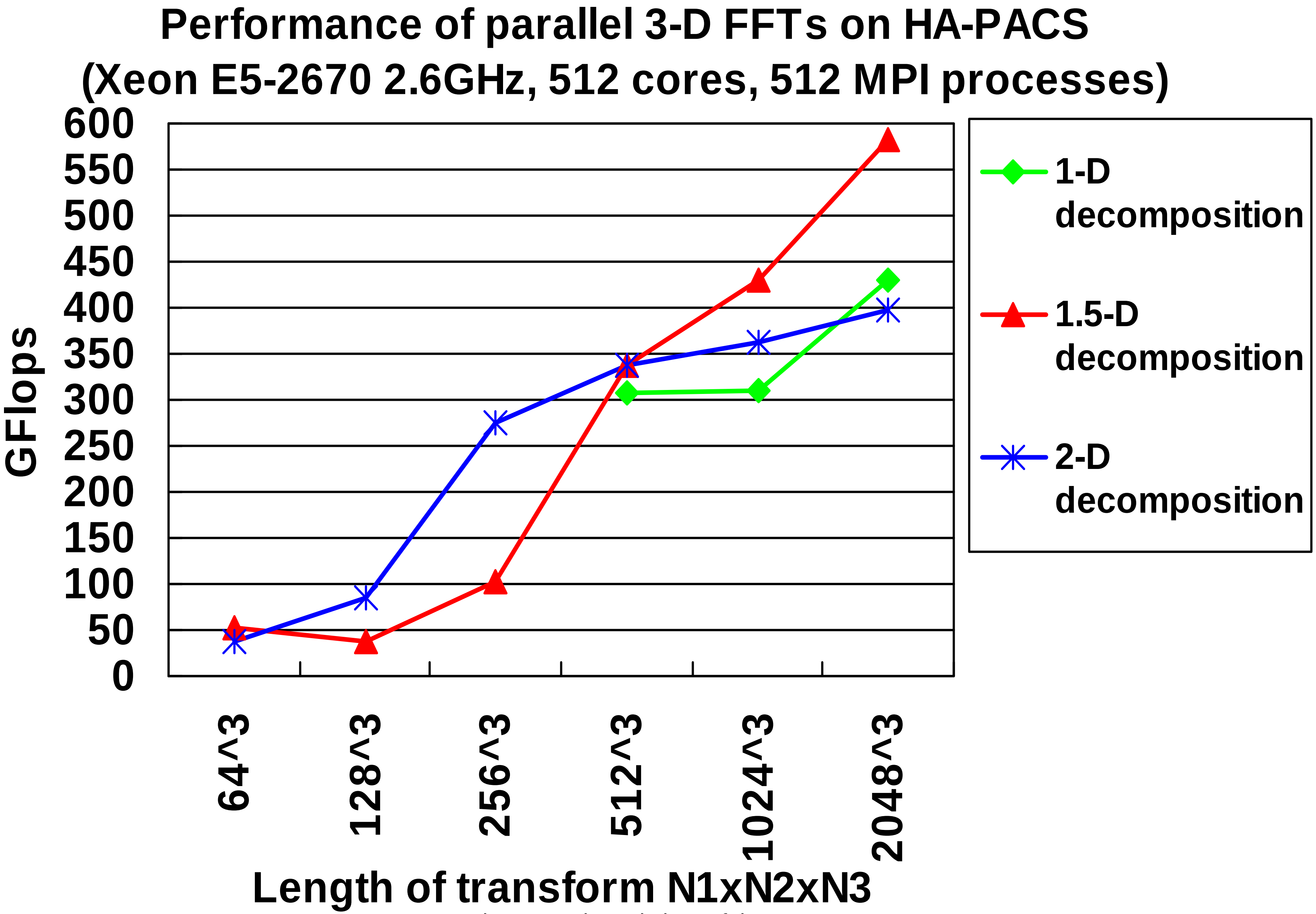
Performance Results

- To evaluate parallel 3-D FFTs, we compared
  - 1-D decomposition (transposed order output)
  - 1.5-D decomposition (transposed order output)
  - 2-D decomposition (transposed order output)
- Target Machine: HA-PACS (268 nodes, GPU: 713 TFlops, CPU: 89 TFlops)
  - Each node is equipped with a two-socket of 8-core Intel Xeon E5-2670 (Sandy Bridge-EP 2.6GHz) and four cards of NVIDIA Tesla M2090.
  - All the nodes are connected through a full-bisectional fat-tree network with a dual-rail Infiniband QDR.
- The flat MPI programming model was used.
- Intel Fortran 12.1 and Intel MPI 4.0 were used.



Performance of parallel 3-D FFTs on HA-PACS
(Xeon E5-2670 2.6GHz, 32 cores, 32 MPI processes)
60
50
40
30
20
10
0
GFlops
64^3
128^3
256^3
512^3
1024^3
Length of transform N1xN2xN3
1-D decomposition
1.5-D decomposition
2-D decomposition


Performance of parallel 3-D FFTs on HA-PACS
(Xeon E5-2670 2.6GHz, 128 cores, 128 MPI processes)
GFlops
180
160
140
120
100
80
60
40
20
0
64^3
128^3
256^3
512^3
1024^3
2048^3
Length of transform N1xN2xN3
1-D decomposition
1.5-D decomposition
2-D decomposition


Performance of parallel 3-D FFTs on HA-PACS
(Xeon E5-2670 2.6GHz, 512 cores, 512 MPI processes)
GFlops
600
550
500
450
400
350
300
250
200
150
100
50
0
64^3
128^3
256^3
512^3
1024^3
2048^3
Length of transform N1xN2xN3
1-D decomposition
1.5-D decomposition
2-D decomposition

Conclusions

- We proposed a parallel 3-D FFT with 1.5-D decomposition.
- The parallel 3-D FFT with 1.5-D decomposition is similar to the parallel 2-D FFT with 1-D decomposition.
- Our proposed parallel 3-D FFT algorithm allows up to $N^{1.5}$ MPI processes for N^3-point FFT.
- This scheme requires only one all-to-all communication for transposed order output.