社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2009－ARC－182 (4)
2009－HPC－119 (4)
2009／2／26

マルチコア超並列クラスタにおける Volumetric 並列三次元 FFT の実現と評価

高 橋 大 介†

本論文では，volumetric 並列三次元 FFT をマルチコア超並列クラスタにおいて実現し評価した結果について述べる．提案する volumetric 並列三次元 FFT アルゴリズムは，multicolumn FFT アルゴリズムに基づいている．二次元分割により通信時間を削減することで，MPI プロセス数が多い場合に性能が改善されることを示す．実現した volumetric 並列三次元 FFT を T2K 筑波システムの 4,096 コアを用いて性能評価を行った結果，256^3 点 FFT において 401 GFlops を越える性能が得られた．

Implementation and Evaluation of Volumetric Parallel 3-D FFT on Massively Parallel Cluster of Multi-Core Processors

DAISUKE TAKAHASHI†

In this paper, we propose an implementation of a volumetric parallel three-dimensional fast Fourier transform (FFT) on massively parallel cluster of multi-core processors. Our proposed volumetric parallel three-dimensional FFT algorithm is based on the multicolumn FFT algorithm. We show that a two-dimensional distribution improves performance effectively by reducing the communication time for larger numbers of MPI processes. We successfully achieved performance of over 401 GFlops on the T2K-Tsukuba system with 4,096 cores for 256^3-point FFT.

1. はじめに

高速 Fourier 変換（fast Fourier transform，以下 FFT）[1] は，科学技術計算において今日広く用いられているアルゴリズムである．FFT において大量のデータを高速に処理するために，分散メモリ型並列計算機における並列三次元 FFT アルゴリズムが多く提案されている[2)~8)]．

これまでに提案されてきた並列三次元 FFT における典型的な配列の分散方法としては，三次元（x，y，z 方向）のうちの一次元のみ（例えば z 方向）を分割して配列を格納することが多かった．この場合，z 方向のデータ数はプロセッサ数以上となる必要がある．

近年の超並列クラスタにおいては，性能を向上させるためにコア数やプロセッサ数が増える傾向にある．例えば，PFlops を超えるピーク性能を有する超並列システムである Roadrunner や Jaguar においては，コア数が 10 万個を超えている．

このようなシステムにおいては，MPI と OpenMP のハイブリッド実行を行い，MPI プロセス数を少なくすることが通信時間を削減する上で有効であるが，それでも MPI プロセス数は最大で 1 万個以上になる．したがって z 方向に一次元分割した場合，このようなシステムでは z 方向のデータ数が 1 万点以上でなければならないことになり，三次元 FFT の問題サイズに制約を受けることになる．

この問題点を改善する方法として，x，y，z 方向に三次元分割する方法[5),8)] が提案されている．三次元分割では，x，y，z 方向に FFT を行う際に，その都度全対全通信でデータを交換する必要がある．

これに対して，二次元分割においては，x，y，z 方向のうち，分割されていない方向が一つあるため，全対全通信の回数を減らすことができるという利点がある．

本論文では，y，z 方向に二次元分割した volumetric 並列三次元 FFT アルゴリズムを提案する．この volumetric 並列三次元 FFT アルゴリズムを用いることで，比較的少ないデータ数でも高いスケーラビリティを得ることが可能になる．

この volumetric 並列三次元 FFT アルゴリズムをマルチコア超並列クラスタである T2K 筑波システムに実現し，性能評価を行う．

† 筑波大学大学院システム情報工学研究科
Graduate School of Systems and Information Engineering, University of Tsukuba

以下，2章で三次元FFTについて説明する．3章で提案するvolumetric並列三次元FFTアルゴリズムについて述べる．4章で一次元分割および二次元分割の場合の通信時間について検討する．5章で本論文で提案したvolumetric並列三次元FFTのアルゴリズムの性能評価の結果を示す．最後の6章はまとめである．

2. 三次元FFT

FFTは，離散Fourier変換（discrete Fourier transform，以下DFT）を高速に計算するアルゴリズムとして知られている．三次元DFTは次式で定義される．

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

ここで，$\omega_{n_r} = e^{-2\pi i/n_r}$ $(1 \le r \le 3)$，$i = \sqrt{-1}$ である．

式(1)から次に示されるような，multicolumn FFTアルゴリズム[9]に基づく三次元FFTが導かれる．

Step 1: $n_2 n_3$ 組の n_1 点 multicolumn FFT

$$x_1(k_1, j_2, j_3) = \sum_{j_1=0}^{n_1-1} x(j_1, j_2, j_3) \omega_{n_1}^{j_1 k_1}$$

Step 2: 転置

$$x_2(j_2, j_3, k_1) = x_1(k_1, j_2, j_3)$$

Step 3: $n_3 n_1$ 組の n_2 点 multicolumn FFT

$$x_3(k_2, j_3, k_1) = \sum_{j_2=0}^{n_2-1} x_2(j_2, j_3, k_1) \omega_{n_2}^{j_2 k_2}$$

Step 4: 転置

$$x_4(j_3, k_1, k_2) = x_3(k_2, j_3, k_1)$$

Step 5: $n_1 n_2$ 組の n_3 点 multicolumn FFT

$$x_5(k_3, k_1, k_2) = \sum_{j_3=0}^{n_3-1} x_4(j_3, k_1, k_2) \omega_{n_3}^{j_3 k_3}$$

Step 6: 転置

$$y(k_1, k_2, k_3) = x_5(k_3, k_1, k_2)$$

この三次元FFTにおいては，3回のmulticolumn FFTがStep 1，3と5で行われる．それぞれのcolumn FFTはメモリ参照の局所性が高く，キャッシュメモリを搭載したプロセッサに適している．また，行列の転置が3回必要になる．

3. Volumetric並列三次元FFTアルゴリズム

これまでに提案されてきた並列三次元FFTにおける典型的な配列の分散方法である，初期データを z 方向に一次元ブロック分割した場合の並列三次元FFTを**図1**に示す．

本論文では，y，z 方向に二次元分割したvolumetric並列三次元FFTアルゴリズムを提案する．

データ数 N を $N = N_1 \times N_2 \times N_3$ とし，プロセッサが $P \times Q$ の二次元にマッピングされるとする．$P \times Q$ 個のプロセッサを持つ分散メモリ型並列計算機では，三次元配列 $x(N_1, N_2, N_3)$ は二次元目（N_2）と三次元目（N_3）に沿って分散される．N_2 が P で割り切れ，N_3 が Q で割り切れるとすると，各プロセッサには $N_1 \times (N_2/P) \times (N_3/Q)$ 個のデータが分散されることになる．

やや複雑になるが，ここで $\hat{N}_r \equiv N_r/P$ および $\hat{\hat{N}}_r \equiv N_r/Q$ の記法を導入する．そして，インデックス J_r に沿ったデータが P 個のプロセッサに分散されることを示す記法を $\hat{J}_r$ とし，インデックス J_r に沿ったデータが Q 個のプロセッサに分散されることを示す記法を $\hat{\hat{J}}_r$ とする．なお，r は次元 r にインデックスが属しているという意味である．

これより，分散された三次元配列は $\hat{x}(N_1, \hat{N}_2, \hat{\hat{N}}_3)$ と表すことができる．ブロック分割によると，y 方向で l 番目のプロセッサにおけるローカルインデックス $\hat{J}_r(l)$ は，次のようなグローバルインデックス J_r に一致する．

$$J_r = \hat{J}_r(l) \times P + l, \quad 0 \le l \le P-1, \quad 1 \le r \le 3 \quad (2)$$

また，z 方向で m 番目のプロセッサにおけるローカルインデックス $\hat{\hat{J}}_r(m)$ は，次のようなグローバルインデックス J_r に一致する．

$$J_r = \hat{\hat{J}}_r(m) \times Q + m, \quad 0 \le m \le Q-1, \quad 1 \le r \le 3 \quad (3)$$

ここで全対全通信を示すために，$\tilde{N}_i \equiv N_i/P_i$ および $\tilde{\tilde{N}}_i \equiv N_i/Q_i$ の記法を導入する．この記法を用いると，N_i は $\tilde{N}_i$ と P_i および $\tilde{\tilde{N}}_i$ と Q_i の二次元表現に分解される．なお，P_i，Q_i はそれぞれ P，Q と同じものを示しているが，このインデックスが次元 i に属していることを示している．

初期データを $\hat{x}(N_1, \hat{N}_2, \hat{\hat{N}}_3)$ とすると，volumetric並列三次元FFTアルゴリズムは次のようになる．

Step 1: $(N_2/P) \cdot (N_3/Q)$ 組の N_1 点 multicolumn FFT

$$\hat{x_1}(K_1, \hat{J}_2, \hat{\hat{J}}_3) = \sum_{J_1=0}^{N_1-1} \hat{x}(J_1, \hat{J}_2, \hat{\hat{J}}_3) \omega_{N_1}^{J_1 K_1}$$

Step 2: 転置

$$\hat{x_2}(\hat{J}_2, \hat{\hat{J}}_3, \tilde{K}_1, P_1) \equiv \hat{x_2}(\hat{J}_2, \hat{\hat{J}}_3, K_1) = \hat{x_1}(K_1, \hat{J}_2, \hat{\hat{J}}_3)$$

Step 3: y 方向の P 個のプロセッサ間で全対全通信を

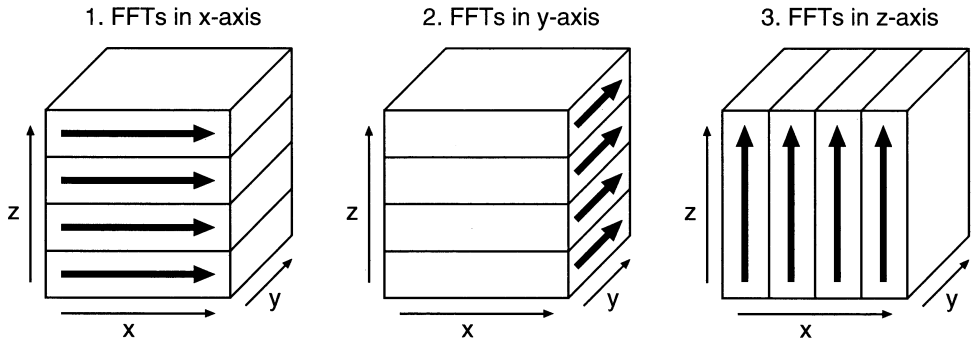

図 1 一次元ブロック分割した場合の並列三次元 FFT

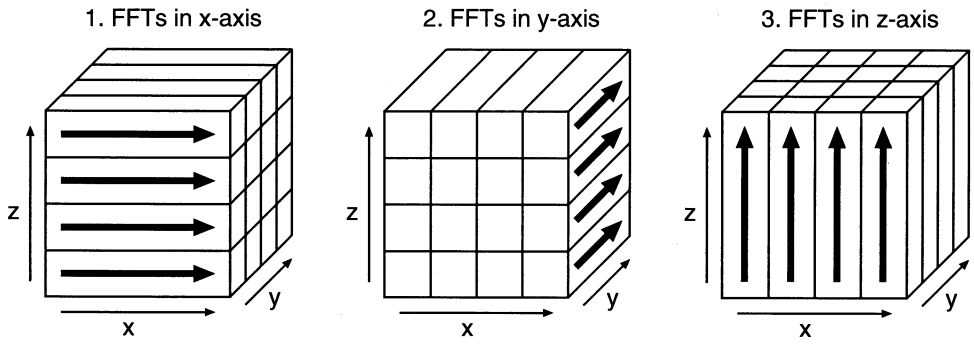

図 2 二次元ブロック分割した場合の並列三次元 FFT

Q 組行う

$$\hat{x}_3(\tilde{J}_2, \hat{\tilde{J}}_3, \hat{K}_1, P_2) = \hat{x}_2(\hat{J}_2, \hat{\tilde{J}}_3, \tilde{K}_1, P_1)$$

Step 4: プロセッサ内再配置

$$\begin{aligned}\hat{x}_4(J_2, \hat{\tilde{J}}_3, \hat{K}_1) &\equiv \hat{x}_4(\tilde{J}_2, P_2, \hat{\tilde{J}}_3, \hat{K}_1)\\ &= \hat{x}_3(\tilde{J}_2, \hat{\tilde{J}}_3, \hat{K}_1, P_2)\end{aligned}$$

Step 5: $(N_3/Q)\cdot(N_1/P)$ 組の N_2 点 multicolumn FFT

$$\hat{x}_5(K_2, \hat{\tilde{J}}_3, \hat{K}_1) = \sum_{J_2=0}^{N_2-1} \hat{x}_4(J_2, \hat{\tilde{J}}_3, \hat{K}_1)\omega_{N_2}^{J_2K_2}$$

Step 6: 転置

$$\begin{aligned}\hat{x}_6(\hat{\tilde{J}}_3, \hat{K}_1, \hat{\tilde{K}}_2, Q_2) &\equiv \hat{x}_6(\hat{\tilde{J}}_3, \hat{K}_1, K_2)\\ &= \hat{x}_5(K_2, \hat{\tilde{J}}_3, \hat{K}_1)\end{aligned}$$

Step 7: z 方向の Q 個のプロセッサ間で全対全通信を P 組行う

$$\hat{x}_7(\hat{\tilde{J}}_3, \hat{K}_1, \hat{\tilde{K}}_2, Q_3) = \hat{x}_6(\hat{\tilde{J}}_3, \hat{K}_1, \tilde{\tilde{K}}_2, Q_2)$$

Step 8: プロセッサ内再配置

$$\begin{aligned}\hat{x}_8(J_3, \hat{K}_1, \hat{\tilde{K}}_2) &\equiv \hat{x}_8(\tilde{\tilde{J}}_3, Q_3, \hat{K}_1, \hat{\tilde{K}}_2)\\ &= \hat{x}_7(\tilde{\tilde{J}}_3, \hat{K}_1, \hat{\tilde{K}}_2, Q_3)\end{aligned}$$

Step 9: $(N_1/P)\cdot(N_2/Q)$ 組の N_3 点 multicolumn FFT

$$\hat{x}_9(K_3, \hat{K}_1, \hat{\tilde{K}}_2) = \sum_{J_3=0}^{N_3-1} \hat{x}_8(J_3, \hat{K}_1, \hat{\tilde{K}}_2)\omega_{N_3}^{J_3K_3}$$

Step 10: 転置

$$\hat{y}(\hat{K}_1, \hat{\tilde{K}}_2, K_3) = \hat{x}_9(K_3, \hat{K}_1, \hat{\tilde{K}}_2)$$

初期データを y，z 方向に二次元ブロック分割した場合の volumetric 並列三次元 FFT を**図 2** に示す．

この volumetric 並列三次元 FFT においては，$N_1 = N_2 = N_3 = N^{1/3}$ とした場合，$(N^{1/3}/P)\cdot(N^{1/3}/Q)$ 組の $N^{1/3}$ 点 multicolumn FFT が Step 1，5 と 9 で実行される．また，入力データ $\hat{x}(J_1, \hat{J}_2, \hat{\tilde{J}}_3)$ が y，z 方向に二次元ブロック分割されているのに対し，Fourier 変換された出力データ $\hat{y}(\hat{K}_1, \hat{\tilde{K}}_2, K_3)$ は x，y 方向に二次元ブロック分割されていることに注意する．このように，入力と出力で異なるデータ分散を行うことで，y，z 方向の全対全通信が Step 3，7 の 2 回で済む．

なお，入力と出力で同じデータ分散にする場合には，

さらに y 方向および z 方向の全対全通信がそれぞれ 1 回ずつ必要になる．

4. 一次元分割および二次元分割の場合の通信時間

全データ数を N，プロセッサ数を $P \times Q$，プロセッサ間通信性能を W （Byte/s），通信レイテンシを L （sec）とする．以下，一次元分割および二次元分割の場合の通信時間について検討する．

4.1 一次元分割の場合の通信時間

一次元分割の場合には，各プロセッサは $N/(PQ)^2$ 個の倍精度複素数データを，自分以外の $PQ-1$ 個のプロセッサに送ることになる．

したがって，一次元分割の場合の通信時間 $T_{1\mathrm{dim}}$ （sec）は，

$$T_{1\mathrm{dim}} = (PQ-1)\left(L + \frac{16N}{(PQ)^2 \cdot W}\right) \approx PQ \cdot L + \frac{16N}{PQ \cdot W} \quad (4)$$

と表される．

4.2 二次元分割の場合の通信時間

二次元分割の場合には，y 方向の P 個のプロセッサ間で全対全通信を Q 組行うことから，y 方向の各プロセッサは $N/(P^2Q)$ 個の倍精度複素数データを，y 方向の $P-1$ 個のプロセッサに送ることになる．また，z 方向の Q 個のプロセッサ間で全対全通信を P 組行うことから，z 方向の各プロセッサは $N/(PQ^2)$ 個の倍精度複素数データを，z 方向の $Q-1$ 個のプロセッサに送ることになる．

したがって，二次元分割の場合の通信時間 $T_{2\mathrm{dim}}$ （sec）は，

$$T_{2\mathrm{dim}} = (P-1)\left(L + \frac{16N}{P^2Q \cdot W}\right) + (Q-1)\left(L + \frac{16N}{PQ^2 \cdot W}\right) \approx (P+Q) \cdot L + \frac{32N}{PQ \cdot W} \quad (5)$$

と表される．

4.3 一次元分割および二次元分割の場合の通信時間の比較

一次元分割および二次元分割の場合の通信時間は式 (4)，式 (5) でそれぞれ表される．

この二つの式を比較すると，全通信量としては式 (4) の一次元分割が式 (5) の二次元分割の約 1/2 になる．ところが，全プロセッサ数 $P \times Q$ が大きく，かつレイテンシ L が大きい場合には式 (5) の二次元分割の方が通信時間が短くなることが分かる．

ここで，式 (5) で表される通信時間 $T_{2\mathrm{dim}}$ が式 (4) で表される通信時間 $T_{1\mathrm{dim}}$ よりも少なくなる条件を求める．

```
#!/bin/bash
MYRANK=$PMI_ID
MYVAL=$(expr $MYRANK / 4)
SOCKET=$(expr $MYVAL % 4)
numactl --cpunodebind=$SOCKET \
--membind=$SOCKET $@
```

図 3 numactl コマンドを用いた flat MPI 向け実行スクリプトの例（MVAPICH 1.2.0 の場合）

式 (4)，(5) より，

$$(P+Q) \cdot L + \frac{32N}{PQ \cdot W} < PQ \cdot L + \frac{16N}{PQ \cdot W} \quad (6)$$

を解くと，

$$N < \frac{(LW \cdot PQ)(PQ - P - Q)}{16} \quad (7)$$

を得る．

例えば，$L = 10^{-5}$ （sec），$W = 10^9$ （Byte/s），$P = Q = 64$ を式 (7) に代入すると，$N < 10^{10}$ の範囲では二次元分割の通信時間が一次元分割に比べて少なくなることが分かる．

5. 性能評価

性能評価にあたっては，提案する二次元分割を行った volumetric 並列三次元 FFT と，一次元分割を行った並列三次元 FFT の性能比較を行った．測定に際しては，32^3，64^3，128^3，256^3 点の順方向 FFT を連続 10 回実行し，その平均の経過時間を測定した．なお，FFT の計算は倍精度複素数で行い，三角関数のテーブルはあらかじめ作り置きとしている．2 章で述べた multicolumn FFT において，各 column FFT がキャッシュに載る場合の in-cache FFT には Stockham アルゴリズム[10] を用いた．in-cache FFT は基数 2，3，4，5，8 の組合せで実現しており，一次元分割と二次元分割で同じプログラムを用いた．

また，全対全通信の回数を減らすために，三次元 FFT の入力と出力で異なるデータ分散形式を用いている．具体的には，一次元分割では入力は z 方向，出力は x 方向にブロック分割し，二次元分割では入力は y，z 方向，出力は x，y 方向にブロック分割している．

マルチコア超並列クラスタとしては，T2K 筑波システムを用いた．T2K 筑波システムは，「T2K オープンスパコン仕様」に基づいた，Appro Xtreme-X3 Server が 648 ノード，10,368 コアからなるマルチコア超並列クラスタである．ノード間は multi-rail の InfiniBand を用いた Fat Tree 相互結合網で接続されている．T2K 筑波システムの諸元を**表 1** に示す．

通信ライブラリとしては，MVAPICH 1.2.0[11] を用いた．なお，今回の性能評価では 1 コアあたり 1 MPI

表 1　T2K 筑波システムの諸元

ノード台数	648
理論ピーク性能	95.39 TFlops
ノード構成	4 ソケット/ノード
CPU	Quad-Core AMD Opteron 8356 （Barcelona，2.3 GHz）
L1 キャッシュ	64 KB×4（命令）+64 KB×4（データ）（各コア独立）
L2 キャッシュ	512 KB×4（各コア独立）
L3 キャッシュ	2 MB（コア間共有）
ノード当たりのメモリ容量	32 GB（DDR2 667 MHz）
総メモリ容量	20 TB
ノード当たりの最大メモリバンド幅	42.7 GB/s
ローカルディスク容量	250 GB×4（SATA-II，RAID-1）
ファイルサーバディスク容量	800 TB（RAID-6）
ネットワークインタフェース	DDR InfiniBand Mellanox ConnectX HCA×4
ネットワークトポロジ	Fat Tree（full-bisection bandwidth）
最大リンクバンド幅	8 GB/s
OS	Red Hat Enterprise Linux version 4 WS (Linux kernel 2.6)
メッセージ通信ライブラリ	MVAPICH 1.2.0

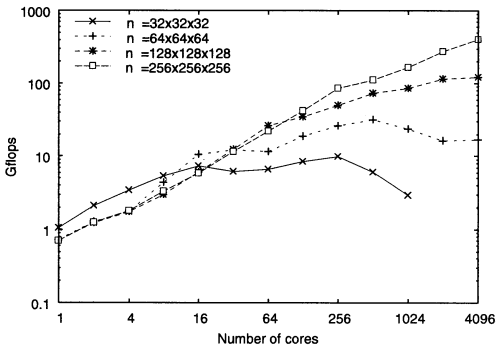

図 4　二次元分割を行った volumetric 並列三次元 FFT の性能（T2K 筑波システム，4096 コア）

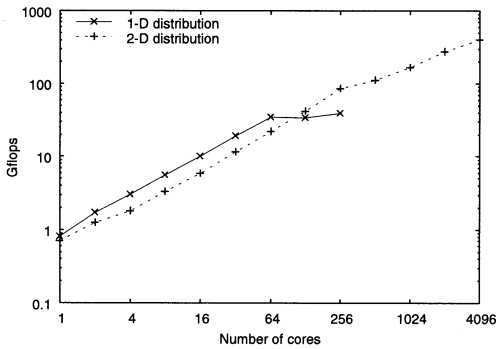

図 5　256^3 点並列三次元 FFT における一次元分割と二次元分割の性能の比較（T2K 筑波システム，4096 コア）

プロセスを用い，flat MPI 実行としている．これは，今回の評価が MPI プロセス数が大きい場合のスケーラビリティに重点を置いているためである．

コンパイラは Intel Fortran Compiler 10.1 を用い，コンパイルオプションは "`ifort -O3 -xO`" を用いた．

T2K 筑波システムの各ノードは，Quad-Core AMD Opteron から構成されており，NUMA アーキテクチャである．したがって，`numactl` コマンドを用いることで，メモリアロケーションをコントロールすることが可能である．

`numactl` コマンドを用いた flat MPI 向け実行スクリプトの例（MVAPICH 1.2.0 の場合）を**図 3** に示す．この実行スクリプトでは，MPI のランクを環境変数 `$PMI_ID` から受け取り，MPI のランクを 1 ソケット当たりのコア数（= 4）で割った値 `MYVAL` をソケット数（= 4）で割った余りである `SOCKET` を求めることで，どの番号のソケットにメモリアロケーションするのかを決定している．

並列三次元 FFT を実行するにあたっては，この実行スクリプトを用いて，1 MPI プロセスが 1 コアに割り当てられるようにしている．

二次元分割を行った volumetric 並列三次元 FFT の性能を**図 4** に示す．ここで，$N = 2^m$ 点 FFT の GFlops 値は $5N \log_2 N$ より算出している．

図 4 から分かるように，32^3 点 FFT では良好なスケーラビリティが得られていない．これは問題サイズが小さい（1 MB）ことから，全対全通信が全実行時間のほとんどを占めているためであると考えられる．それに対して，256^3 点 FFT では 4,096 コアまで性能が向上していることが分かる．なお，4,096 コアにおける性能は約 401.3 GFlops であった．

256^3 点並列三次元 FFT において一次元分割と二次元分割の性能を比較したものを**図 5** に示す．図 5 から分かるように，64 コア以下の場合には，通信量の少ない一次元分割が二次元分割よりも性能が高くなっているが，128 コア以上では通信時間を少なくできる二次元分割が一次元分割よりも性能が高くなっていることが分かる．

表 2 二次元分割を行った 256^3 点 volumetric 並列三次元 FFT における通信時間の割合

Cores	実行時間 (sec)	通信時間 (sec)	通信時間の割合 (%)
1	2.84448	0.33756	11.87
2	1.60708	0.25705	15.99
4	1.11583	0.26579	23.82
8	0.61128	0.16514	27.02
16	0.34474	0.11832	34.32
32	0.17320	0.06372	36.79
64	0.09039	0.03737	41.34
128	0.04823	0.02176	45.11
256	0.02346	0.01320	56.28
512	0.01793	0.01461	81.48
1024	0.01199	0.01079	89.98
2048	0.00736	0.00652	88.64
4096	0.00502	0.00482	96.07

なお，一次元分割を用いた場合，256^3 点 FFT は 256 MPI プロセス以下でのみ実行が可能であるが，二次元分割を用いた場合，256^3 点 FFT は 65,536 MPI プロセスまで実行が可能である．

二次元分割を行った 256^3 点 volumetric 並列三次元 FFT の実行時間における通信時間の割合を**表 2** に示す．表 2 から分かるように，4,096 コアにおいては 96%以上が通信時間に費やされている．この理由としては，全対全通信において各プロセッサが一度に送る通信量がわずか 1 KB となるため，通信時間においてレイテンシが支配的になるからであると考えられる．

通信時間をさらに削減するためには，文献 5) で行っているように，全対全通信に `MPI_Alltoall` 関数を使わずに，より低レベルな通信関数を用いて，レイテンシを削減するなどの工夫が必要になる．

6. ま と め

本論文では，volumetric 並列三次元 FFT をマルチコア超並列クラスタにおいて実現し評価した結果について述べた．

並列三次元 FFT において，二次元分割により通信時間を削減することで，MPI プロセス数が多い場合に性能が改善されることを示した．本論文で提案した volumetric 並列三次元 FFT は，プロセッサ数が 1 万個を超えるような超並列クラスタにおいてより効果的であると考えられる．

実現した volumetric 並列三次元 FFT を T2K 筑波システムの 4,096 コアを用いて性能評価を行った結果，256^3 点 FFT において 401 GFlops を越える性能が得られた．

謝辞 日頃より議論して頂いている筑波大学 HPCS 研究室の皆様に感謝致します．本研究における T2K 筑波システムの利用は，筑波大学計算科学研究センター学際共同利用プログラムによるものです．関係各位に謝意を表します．

参 考 文 献

1) Cooley, J. W. and Tukey, J. W.: An Algorithm for the Machine Calculation of Complex Fourier Series, *Math. Comput.*, Vol. 19, pp. 297–301 (1965).
2) Brass, A. and Pawley, G. S.: Two and Three Dimensional FFTs on Highly Parallel Computers, *Parallel Computing*, Vol. 3, pp. 167–184 (1986).
3) Agarwal, R. C., Gustavson, F. G. and Zubair, M.: An Efficient Parallel Algorithm for the 3-D FFT NAS Parallel Benchmark, *Proceedings of the Scalable High-Performance Computing Conference*, pp. 129–133 (1994).
4) Takahashi, D.: Efficient implementation of parallel three-dimensional FFT on clusters of PCs, *Computer Physics Communications*, Vol. 152, pp. 144–150 (2003).
5) Eleftheriou, M., Fitch, B. G., Rayshubskiy, A., Ward, T. J. C. and Germain, R. S.: Scalable framework for 3D FFTs on the Blue Gene/L supercomputer: Implementation and early performance measurements, *IBM J. Res. Dev.*, Vol. 49, pp. 457–464 (2005).
6) Frigo, M. and Johnson, S. G.: The Design and Implementation of FFTW3, *Proc. IEEE*, Vol. 93, pp. 216–231 (2005).
7) Takahashi, D.: A Hybrid MPI/OpenMP Implementation of a Parallel 3-D FFT on SMP Clusters, *Proc. 6th International Conference on Parallel Processing and Applied Mathematics (PPAM 2005)*, Lecture Notes in Computer Science, Vol. 3911, Springer-Verlag, pp. 970–977 (2006).
8) Fang, B., Deng, Y. and Martyna, G.: Performance of the 3D FFT on the 6D network torus QCDOC parallel supercomputer, *Computer Physics Communications*, Vol. 176, pp. 531–538 (2007).
9) Van Loan, C.: *Computational Frameworks for the Fast Fourier Transform*, SIAM Press, Philadelphia, PA (1992).
10) Swarztrauber, P. N.: FFT Algorithms for Vector Computers, *Parallel Computing*, Vol. 1, pp. 45–63 (1984).
11) MVAPICH: MPI over InfiniBand and iWARP: `http://mvapich.cse.ohio-state.edu/`