高次元メッシュ/トーラスネットワークにおける実アプリケーションの通信最適化手法 —「京」上の Tofu ネットワークを例に—

黒田明義[†1]　大井憲行[†2]　井上晃[†3]　村井均[†1]　山崎隆浩[†4]
大野隆央[†4]　今村俊幸[†1]　南一生[†1]

近年ハイパフォーマンスコンピューティングにおいて，システム性能の向上に伴い超並列化が進み，実アプリケーションでは通信コストが増大し，並列性能の限界が問題となっている．超並列に向けてシステムでは，共有メモリを用いたマルチコアや，ノード間の多次元ネットワーク網などが採用されている．スーパーコンピュータ「京」では，6次元メッシュ/トーラスによる直接網を用いて，障害ノードを回避しながらアプリケーション毎に 3 次元トーラスを切り出してユーザに提供している．アプリケーションによる超並列への対応は，ハイブリッド並列や多軸並列などが有効とされる．しかし多軸並列の手法は，一方の通信を最適化すると別の方向の通信が不連続になるなどの現象が発生し，多次元ネットワーク上で通信の最適化が難しいという新たな問題を生み出している．本報では，この問題を解決すべく直接網の多次元性を活かした通信最適化の汎用的な手法について，実アプリケーションを用いて紹介する．

Communication Optimization Method on a High-dimensional Mesh / Torus Network for Real Applications --- Case Study for the Tofu Network of ‘K computer’ ---

AKIYOSHI KURODA[†1]　NORIYUKI OI[†2]　HIKARU INOUE[†3]
HITOSHI MURAI[†1]　TAKAHIRO YAMASAKI[†4]　OHNO TAKAHISA[†4]
TOSHIYUKI IMAMURA[†1]　KAZUO MINAMI[†1]

Recently, there has been an increase in massively parallel computing for the improvement of performance in high-performance computing (HPC) systems. With the increase in communication costs of applications, limitations in parallel performance have become a serious problem. Corresponding to the massively parallel computing systems, the system is configured with multi-core processor using shared memory, and a multi-dimensional network between nodes. Using a 6D mesh torus network on the K computer, users will be able to bypass the three-dimensional torus to avoid a failed node. Hybrid parallelization and multi-axis parallelism are effective methods for massively parallel systems applications. However, multi-axis parallelism has created a new communication optimization problem; while one communication in one direction is optimized, the communication in the opposite direction becomes discontinuous. We introduce communication optimization by taking advantage of a multi-dimensional direct network.

1. はじめに

近年ハイパフォーマンスコンピューティングにおいて，システム性能の向上に併せて，数万～数百万という超並列化が進み，実アプリケーションにおける通信コストの増大が問題となっている．超並列化に向けてシステム側では，共有メモリを用いたマルチコアや，ノード間を接続する多次元ネットワーク網などの技術が採用されており，アプリケーション側の対応としてはハイブリッド並列や多軸並列の手法が有効とされる．しかし多軸並列の手法は，多次元ネットワーク上での通信の最適化が難しいという新たな問題を生み出している．

†1 理化学研究所
RIKEN
†2 株式会社富士通システムズ・イースト
Fujitsu Systems East Limited
†3 富士通株式会社
FUJITSU, LTD.
†4 物質材料研究機構
National Institute for Materials Science (NIMS)

スーパーコンピュータ「京」(以下「京」と呼ぶ)上での通信の最適化については，最適なプロセス配置を探索するツール RMATT がシステムから提供されている[1][2]．これは通信のトレース情報から，Simulated Annealing 法を用いて，プロセス配置の最適解を探索するツールである．また演繹的な手法として隣接通信については，直交座標を用いないアプリケーションでの最適事例がいくつかある．全球を対象とする大気大循環モデル NICAM では，正二十面体メッシュの袖通信での最適化が試みられている[3]．しかし隣接通信の多くは最適化前に既に通信が局所化しており，効果は大きくないことが多い．また集団通信については，実空間密度汎関数法アプリケーション RSDFT にてサブコミュニケータを閉じたプロセス空間に配置し，性能改善に成功している[4]．また高並列汎用分子動力学シミュレーション MODYLAS では，並列数の二巾制限に対応するために，軸入れ替えという独自のプロセス配置を用いており[5]，通信最適化ではないが，本報での最適化の基礎となる試みで

ある．

本報では，ネットワークの多次元性を活かした通信の汎用的な最適化手法について，実アプリケーションを用いて紹介する．2 章では，背景となる評価環境である「京」のシステムについて説明し，3 章では，実アプリケーションで用いられる多軸並列の手法について，その効果と問題点を議論する．4 章では，実際の「京」での通信最適化の手法を紹介し，5 章では，固有値ソルバ，密度汎関数法のアプリケーションでの通信最適化事例を紹介する．6 章では，ハイパフォーマンスコンピューティングの将来を見据えてまとめる．

2. 評価システム

2.1 スーパーコンピュータ「京」

本章では，評価に用いた「京」について紹介する．「京」とは，科学技術の発展を目的として，理化学研究所計算科学研究機構で開発された 10PFLOPS 級のスーパーコンピュータである．2012 年 7 月に完成し，2012 年 9 月からは一般ユーザ向けに共用が開始された．

CPU は「京」向けに新規で開発された富士通社製の SPARC64^{TM} VIIIfx [6][7][8][9][10]であり， 1 CPU 上に 8 コアの演算器を持つ．システムは，システムボードに 4 個のノードを搭載し, 筐体に 24 枚のシステムボードが搭載している．「京」ではこの筐体が 864 台設置され，ユーザが利用可能な計算資源は，82,944 ノード，663,552 コアであり，ピーク性能 10.62PFLOPS，全メモリ量 1.26PB である．

2.2 Tofu インターコネクト

「京」でのネットワーク構成は, 1CPU, 16GB のメモリ，データ転送を行うインターコネクト用 LSI（ICC: Inter-Connect Controller）によりノードが構成される．ノードは ICC を通じて，Tofu インターコネクトと呼ばれる 6 次元メッシュ/トーラスネットワークで結合される[11][12]. Tofu インターコネクトは，24×18×16 の 3 次元メッシュ/トーラスネットワーク(X 軸，Y 軸，Z 軸)と基本単位と呼ばれる 2×3×2 のメッシュ/トーラスネットワーク(A 軸, B 軸, C 軸)を組み合わせから構成される．各ノードからは 10 本のリンクが出ており，6 本は 3 次元メッシュ/トーラスに，残りの 4 本は基本単位の内部結合に使用する．各リンクのバンド幅は 5GB/s(双方向)で, システム全体のバイセクションバンド幅は 30TB/s である．

Tofu インターコネクトは，ユーザから見るとアプリケーション単位で任意の形状の論理 3 次元トーラスを切り出すことが可能である．既存アプリケーションの多くは 3 次元以下での利用を想定しており，またトーラス網により通信に必要なホップ数を半減することが可能となり通信性能が向上し，ロードインバランスが発生しにくいという利点から採用されている．更に B 軸の長さ 3 の冗長性を活かした迂回経路を使用することで，故障ノードを避けて論理 3 次元トーラスネットワークの切り出しが可能になり，利便性だけでなく信頼性も高い．また ICC は，4 個の通信インターフェイスを用いた同時通信や高機能バリアによるハードウェア同期を行うことで，より高速な通信が可能である．

並列化手法は，超並列性からの要請でノード内はスレッド並列，ノード間はプロセス並列というハイブリッド並列が推奨されている．本報での測定結果は，すべてノード内 8 コアは自動並列と OpenMP を併用したスレッド並列を，ノード間は MPI によるプロセス並列を用いて計算したものである．コンパイラやライブラリは,「京」向け言語開発環境が用いられており, 測定は K-1.2.0-13 を用いて行った．評価環境は表 1 の通りである．

表 1 「京」を用いた評価システム

ハードウェア
SPARC64^{TM} VIIIfx, 2GHz, 8 core/CPU, 1CPU/node
浮動小数点演算性能: 128GFLOPS/node
浮動小数点レジスタ: 256 本/core
キャッシュ: L1 - 32KB/core, L2 - 6MB/CPU
メモリ: 16GB/CPU, 0.5B/F
ネットワーク: 6D mesh / torus network, 5GB/s×双方向 (6 方向)
ソフトウェア
OS: Linux (専用 OS)
言語，ライブラリ:「京」向け言語開発環境
K-1.2.0-13（2013 年 4 月～8 月環境）
Fortran, MPI, SSL2（BLAS, LAPACK, ScaLAPACK）
FFTW-3.3（「京」向け最適化済み）

2.3 Tofu 専用アルゴリズム

「京」の MPI ライブラリは Open MPI をベースに Tofu インターコネクトの性能を最大限に引き出すために，低レベル通信や集団通信の拡張が行われている．集団通信については, Open MPI で提供されている通信アルゴリズムのチューニング以外に，Tofu インターコネクトの特性を活かして新たに開発された Tofu 専用アルゴリズムをいくつか用意しており，既存のアルゴリズムと比較して最大で 13.25 倍の性能を実現している[13][14][15].

集団通信における主な Tofu 専用アルゴリズムの特徴並びに適用条件は，表 2 の通りである．4 個の通信インターフェイスを用いた同時通信の機能を活かすために, 1～2 次元の低次元形状のコミュニケータでは適用が難しい．このため alltoall 通信の double spread 以外のアルゴリズムは, コミュニケータ形状が平面形状や棒状形状であってはならず, 厚さや太さが 2 以上の 3 次元直方体形状であることが必要である．6 次元形状のコミュニケータを意識した専用アルゴリズムは，alltoall 通信の blac6d のみであり，適用条件は 3 次元以上の形状であるのものがほとんどである．またサブコミュニケータであっても 3 次元形状であれば適用可能

である．このことは，アプリケーション動作時に Tofu 専用アルゴリズムを有効化するための制限事項として，現状では 3 次元形状指定で実行させる必要があるが, 将来的には，1〜2 次元形状指定での実行時も Tofu 専用アルゴリズムが適用可能かを自動判定できる可能性がある．

また双方向 5GB/s という高スループットを利用しているため，長メッセージ通信向けのものが多い．但し，alltoall 通信の double spread は，短メッセージでも有効なアルゴリズムである．

表 2　主な Tofu 専用アルゴリズムの特徴と適用条件

アルゴリズム	Trinaryx3	Trinaryx3	3dtorus	Blacc3d	Double Spread
通信	MPI_Bcast	MPI_Allreduce , MPI_Reduce	MPI_Allgather , MPI_Allgatherv	MPI_Alltoall	MPI_Alltoall, MPI_Alltoallv
通信サイズ	中〜長	長	長	長	短〜中
適用条件	• 送受信バッファのデータ型が基本データ型. • 送受信バッファの先頭アドレスが4バイト境界. • 送受信のデータ型のサイズと要素数(count/scount/rcount)の積が4の倍数.				
	• ノード内1プロセス. • ジョブ形状が3次元. • コミュニケータ形状が3次元直方体かつ全てのノードにプロセスが配置.				• ノード内1〜2プロセス. • 送受信バッファの各要素4バイト境界.
	• コミュニケータ各軸長が2 以上 .	• MPI_IN_PLACEが未指定. • 定義済み演算子を使用. • コミュニケータ各軸長が2 以上.	• 送信バッファの上限は約16MiB×通信インタフェース数×コミュニケータサイズ. • 送受信バッファ各要素が4 バイト境界.	• コミュニケータ各軸長が偶数. • MPI_COMM_WORLD. • 受信バッファサイズが、約32MiB*コミュニケータサイズまで.	

3. 実問題における通信の特徴

システムの超並列化に伴い，アプリケーション側では，共有メモリを用いたスレッド並列と MPI を用いた分散並列のハイブリッド並列への対応が推奨されている．また分散並列については，多軸並列の手法が有効とされている．本章では，実アプリケーションで用いられている多軸並列について，その特徴と問題点を議論する．

3.1 多軸並列

有限差分法計算などの空間メッシュを用いた計算手法において，1 方向の軸での並列化では，高並列時の並列粒度が細かくなり，並列性能が悪化する問題がある．しかし解きたい問題の多くは 1 次元形状でなく平面や立体形状であるため，これらの方向にも分割が可能である(図 1).

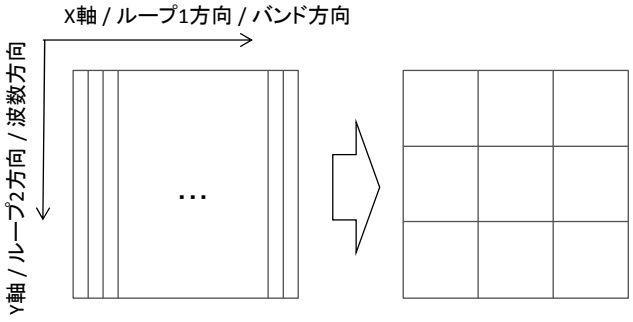

図 1　多軸並列の概念図

多軸並列により，演算は計算順序が変わるのみで演算量の増減はない．しかし分割軸あたりの分割粒度が大きくなり，ループ長が長くなるため，オーバーヘッドが低減されるだけではなく，命令スケジューリングの最適化などが促進されることが多い．隣接通信に関しては，通信時間はプロセスあたりの通信量に比例するため，通信量が一定の物理量を渡すような場合，通信時間は変わらない．通信量が境界の面積(S)に比例するような袖通信の場合, 図 1 の分割例で分割数を n とした時，通信時間は $2S$ から $4S/\sqrt{n}$ へと削減される．集団通信は，通信時間は通信量や通信相手数に比例する．このため，もともと 1 次元分割で通信がある場合，通信相手数が $n-1$ から $\sqrt{n}-1$へと減少し，通信時間の削減が期待できる(図 2 左)．もともとの 1 次元分割で通信がない場合は，新たな通信が発生するが，通信相手数は同様に $\sqrt{n}-1$であるため，通信時間の増分は比較的小さい(図 2 中)．具体的に「京」のノード間 MPI 並列数は $n=82,944$ であるので, 二軸分割におけるコミュニケータサイズは $\sqrt{n}=288$ となり, 一般のクラスタマシンでの並列性能があれば十分対応可能である．

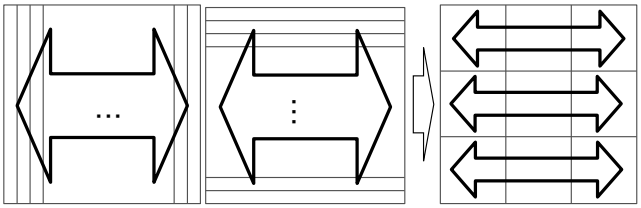

図 2　多軸並列による集団通信.
集団通信が減る例と増える例

本手法は，有限差分法計算での空間方向をループ方向と読み替えることにより一般化が可能である．アプリケーション内に多重ループがある時，各々のループで並列化することに相当する．ループとして異なる物理量を用いることも可能である．

ハミルトニアン行列の固有値問題を解く密度汎関数法では，主な演算が行列-行列積で記述されるため，電子数などの問題規模(N)に対して演算量は $O(N^3)$である．アプリケーションは電子数に相当するバンドのループと波動関数を展開する基底のループと，別の物理量の多重ループで構成される．このように問題規模に対して演算量が増大するアプリケーションでは，演算時間や使用メモリ量が問題規模の制限になることが多い．特に超並列計算では，分割数が問題規模を越えてしまい，単純な 1 ループを用いた分割では, 性能が低下するだけではなく分割不可能な状況に陥り，多軸並列化が不可欠なアプリケーションの代表である[16].

2.2 節で述べたように「京」では，どのような並列形状の計算でも, 障害ノードを避けて任意形状の 1〜3 次元トーラスネットワーク切り出しが可能である．特にコミュニケ

ータが 3 次元形状であると，通信インターフェイスの多方向性を活かした Tofu 専用アルゴリズムの活用が可能になる．

多軸並列したアプリケーションを 3 次元トーラスネットワークに割り当てる場合は，それぞれのアプリケーションの分割軸をトーラスの軸に割り当てる方法がある．一般に，3 次元のトーラス軸のうち 1 次元をアプリケーション分割軸に割り当てると，もう一方のアプリケーション分割軸は各々の軸が直交するため，残りの 2 次元トーラス軸に割り当てることになる(図 3 左)．両通信は連続に配置され，通信が狭い区間内に閉じることで，経路の競合が抑えられる利点はある．しかし扁平形状により通信のホップ数は増加し，両アプケーション軸供に低次元形状への割り当てになるため, Tofu 専用アルゴリズムの適用が不可能になる. Tofu 専用アルゴリズムを適用するために一方のアプリケーション分割軸を 3 次元の直方体に割り当てると，直交するもう一方のアプリケーション分割軸は不連続になるため(図 3 右), 通信のホップ数は増加し, Tofu 専用アルゴリズム適用不可能になる．またサブコミュニケータ間の通信経路の競合も発生するため性能が劣化する可能性がある．

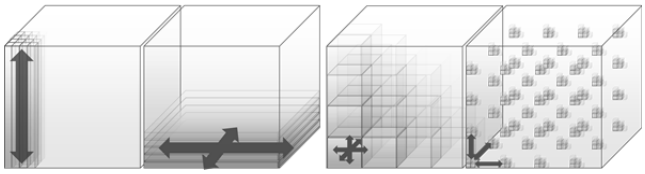

図 3　3 次元トーラスネットワークへ二軸並列アプリケーションを配置した時の直交するサブコミュニケータ形状と通信範囲．左図：1 次元×2 次元．右図：3 次元×不連続

3.2 サブコミュニケータ

数値計算では，通信にサブコミュニケータが利用される．固有値計算などの行列計算では，行と列の両方向に並列化をし，各々の軸方向のみの集団通信をするコミュニケータが使用される．また多軸分散 3 次元 FFT では，ある方向に FFT をした後，別の方向に FFT をする際に，転置操作を必要とし，サブ空間での alltoall 全体通信が発生する．

複数のサブコミュニケータの多くは直交することから，3.1 節の二軸並列と同じ現象が発生する．3 次元直接網で両サブコミュニケータを連続にすると，図 3 左のように，それぞれのコミュニケータは 1,2 次元割り当てになり Tofu 専用アルゴリズムは適用できなくなる．また一方のコミュニケータを 3 次元直方体に割り当てると図 3 右のように，直交するもう一方のコミュニケータ割り当てが不連続になり，どちらの場合も通信性能が劣化する. 他システムの例では，同じ 3 次元トーラスネットワークを持つ TITAN では, 疎行列計算でのサブコミュニケータの形状最適化のためにヒルベルト曲線を用いた割り当てを提案している[17].

一般化すると，直接網での集団通信の最適化とは，「立方体に近い」「連続」な閉じた空間にコミュニケータを割り当てることである．これにより通信経路の競合が防げ，通信のホップ数の削減が期待できる．「京」では，Tofu 専用アルゴリズムの適用も可能になり，更なる効果が期待できる．しかし 3 次元直接網上で多軸並列などによる複数のコミュニケータが同時に最適な条件を満たすのは，不可能である．これは 3 次元上に同じ次元のものを複数割り当てることによって生じた問題である．

4. 通信最適化の手法

本節では，「京」上で多軸並列などにで用いる直交するサブコミュニケータの通信最適化について議論する．

4.1 6 次元ランク割り当てによる最適化

2.2 節で述べたように，「京」はアプリケーションから見ると 3 次元トーラスであるが実際の結線は 6 次元メッシュ/トーラスの直接網である．一般に，6 次元直接網上で通信を最適化するようにアプリケーションを配置するには，6 次元座標軸のいくつかを組み合わせて，アプリケーションに割り当てる方法が妥当である．例えば，一般的な 3 次元分割された単一コミュニケータのアプリケーションでは，XA 軸，YB 軸，ZC 軸をそれぞれ組み合わせて 3 本のアプリケーション分割軸に割り当てる方法が良いと考えられるが，これはまさに現在システムが提供するアプリケーション単位での 3 次元トーラスの切り出しに他ならない．

2.3 節で述べたように Tofu 専用アルゴリズムの適用条件は，サブコミュニケータの厚さ，太さ供に 2 以上の 3 次元直方体形状を必要とするものが多く，軸の方向について制限はない．多軸並列したアプリケーションについては，6 次元直接網の結線を全て用いることで，原理的に直交した 2 つの 3 次元直方体を作ることが可能である. 6 次元を用いて 2 つの直交する 3 次元直方体を作る方法として，以下に 2 つの方法を紹介する．

4.2 マップファイルによる最適化

6 次元直接網上で 2 つの直交する 3 次元直方体を作る方法として,「京」ではシステムから提供されているマップファイルを用いる方法がある．マップファイルとはアプリケーションに割り当てられた 3 次元トーラス座標と MPI のランクとをファイルを介して対応させる方法である．図 4 は，サンプルとして実問題 3,072 並列において，(A 軸, Y 軸, C 軸) = 2×4×2 = 16 並列, (X 軸, Z 軸, B 軸) = 8×8×3 = 192 並列の 2 種類の 3 次元直方体サブコミュニケータをもつマップファイルを作る際の手順である．各サブコミュニケータの分割数は多軸並列を行う多重ループのループ長から決められ，直接網の 6 本の割り当て順序並びに軸長は，その

分割数を素因数分解して決定する．2 種類のコミュニケータが直方体になるように，直方体の軸順，つまり A 軸→Y 軸→・・・→B 軸の順に，0 から軸長まで値を 1 つずつ増やし，ランクとシステムの 6 次元座標との対応表を作成する．更にランク順に並べられた 6 次元座標をアプリケーションが割り当てられる 3 次元トーラス座標に変換することで実現可能になる．6 次元座標からの座標変換には，3 次元トーラス構成規則 *iX(X,A), iY(Y,B), iZ(Z,C)*が必要となり，XA 軸，ZC 軸と YB 軸で，その規則は異なる(図 4 下)．

アプリケーションを実行する際には，作成したマップファイルで決定される 3 次元形状で実行する必要があり，回転割り当ては不可能である．また，この方法はアプリケーション内で並列軸の並列順序が分かっている時に利用可能な方法である．

rank	6次元座標 (*X,Y,Z,A,B,C*)	3次元変換 (*iX(X,A),iY(Y,B),iZ(Z,C)*)	3次元座標 = (*X₃,Y₃,Z₃*)
0	(0,0,0,0,0,0) →	(*iX(0,0),iY(0,0),iZ(0,0)*)	= (0, 0, 0)
1	(0,0,0,1,0,0) →	(*iX(0,1),iY(0,0),iZ(0,0)*)	= (15, 0, 0)
2	(0,1,0,0,0,0) →	(*iX(0,0),iY(1,0),iZ(0,0)*)	= (0, 1, 0)
3	(0,1,0,1,0,0) →	(*iX(0,1),iY(1,0),iZ(0,0)*)	= (15, 1, 0)
4	(0,2,0,0,0,0) →	(*iX(0,0),iY(2,0),iZ(0,0)*)	= (0, 2, 0)
5	(0,2,0,1,0,0) →	(*iX(0,1),iY(2,0),iZ(0,0)*)	= (15, 2, 0)
	・・・		
3,071	(7,3,7,1,2,1) →	(*iX(7,1),iY(3,2),iZ(7,1)*)	= (8, 4, 8)

但し，*iX(X,a),iZ(Z,C)*は，　　*iY(Y,B)*は，

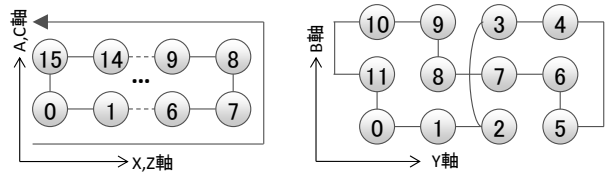

図 4　マップファイルによる 6 次元マッピング方法．
下図：6 次元座標から 3 次元トーラス座標への変換規則

4.3 6 次元ランク取得関数による最適化

アプリケーション内での並列規則は，不明であることが多い．このときサービス関数として提供されている 6 次元相対座標問い合わせ関数を用いるとランク再配置が可能になる．構成する直方体の軸順に 6 軸方向に軸長ループを回し，この関数を用いて 6 次元座標に相当するランク番号を得る．得られたランク番号と 6 次元座標の対応から，集団通信が 3 次元直方体内で収まるようにサブコミュニケータを構成するという方法である．

図 5 は，256 並列の実問題実計算において，行・列方向通信に16×16分割の2つのサブコミュニケータ直方体を作る時のサンプルである．2 つのサブコミュニケータは(A 軸, X 軸, C 軸) = 2×4×2 = 16 並列, (Z 軸, Y 軸, B 軸) = 2×4×2 = 16 並列に割り当て，直方体を形成している．コードでは A 軸→X 軸→・・・→B 軸と，6 次元の直接網の軸順にループを回転し，6 次元座標からアプリケーション内でのランク番号を取得している．得られたランク情報からサブコミュニケータを作成し集団通信を直方体内に閉じさせることが可能となる．図 5 下は，16×16 分割の固有値計算の行・列方向の集団通信がそれぞれ直方体になるように並べ替えた実際のランク配置のサンプルである．行・列方向ともに，ランクの並び順は不規則に見えるが，それぞれ AXC 軸，ZYB 軸で形成される直方体内に集団通信は閉じている．

```
include 'mpif-ext.h'
icount=0
do ib=0,nb                    ┐
  do iy=0,ny                  ┘ 列方向直方体 B
    do iz=0,nz                ┐
      do ic=0,nc              ┘ 行方向直方体 C
        do ix=0,nx
          do ia=0,na
            j=icount/(na*nx*nc)+1
            i=mod(icount,(na*nx*nc))+1
            call
FJMPI_Topology_rel_xyzabc2rank(ix,iy,iz,ia,ib,ic,irank)
            usermap(i,j)=irank
            icount=icount+1
end do
      ・・・
call blacs_gridmap(ictxt,usermap,na*nx*nc,na*nx*nc,nz*ny*nb)
```

usermap(i, j) → j(列)

i(行)＼j(列)	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	256	16	272	32	288	48	304	1536	1792	1552	1808	1568	1824	1584	1840	1600
1	64	320	80	336	96	352	112	368	1600	1856	1616	1872	1632	1888	1648	1904
2	128	384	144	400	160	416	176	432	1664	1920	1680	1936	1696	1952	1712	1968
3	192	448	208	464	224	480	240	496	1728	1984	1744	2000	1760	2016	1776	2032
4	2	258	18	274	34	290	50	306	1538	1794	1554	1810	1570	1826	1586	1842
5	66	322	82	338	98	354	114	370	1602	1858	1618	1874	1634	1890	1650	1906
6	130	386	146	402	162	418	178	434	1666	1922	1682	1938	1698	1954	1714	1970
7	194	450	210	466	226	482	242	498	1730	1986	1746	2002	1762	2018	1778	2034
8	512	768	528	784	544	800	560	816	2048	2304	2064	2320	2080	2336	2096	2352
9	576	832	592	848	608	864	624	880	2112	2368	2128	2384	2144	2400	2160	2416
10	640	896	656	912	672	928	688	944	2176	2432	2192	2448	2208	2464	2224	2480
11	704	960	720	976	736	992	752	1008	2240	2496	2256	2512	2272	2528	2288	2544
12	514	770	530	786	546	802	562	818	2050	2306	2066	2322	2082	2338	2098	2354
13	578	834	594	850	610	866	626	882	2114	2370	2130	2386	2146	2402	2162	2418
14	642	898	658	914	674	930	690	946	2178	2434	2194	2450	2210	2466	2226	2482
15	706	962	722	978	738	994	754	1010	2242	2498	2258	2514	2274	2530	2290	2546

図 5　6 次元ランク取得関数による通信最適化方法．
下図：16×16 分割で作成したランク配置例

5. 実問題での最適化例

本章では，実際のアプリケーションでの 6 次元メッシュ/トーラスを用いた通信最適化事例をいくつか紹介する．

5.1 固有値ソルバ Eigen ライブラリベンチマークの最適化

理化学研究所今村らが開発している高速固有値ライブラリ Eigen [18][19]は，対角化する行列をプロセス並列により行方向と列方向に分割して集団通信を行う．しかし 2 次元トーラスを用いてプロセスを割り当てた場合，コミュニケータが折りたたまれて，6 次元中の 3 次元直方体を構成しても，現状は Tofu 専用アルゴリズムが適用されない．3 次元トーラス形状でアプリケーションを実行させたときに直交するサブコミュニケータを両方とも連続にするのは 3.2 節で議論したように困難である．

本ライブラリのランク割り当ては行・列方向を自由に選択可能であり，割り当て順序をユーザが指定することが可能であるため，マップファイルを用いた 6 次元直接網への最適化が可能である．行・列方向の両サブコミュニケータ

内の集団通信を最適化するための各軸の割り当て例は表 3 の通りである．固有値計算では，2 冪かつ正方形に近い分割形状であるほど余計な通信が発生せず高速であるとされるため，正方分割のみ実測した．分割数に応じて，マップファイルを作成する際に用いる軸が変わるので，作成並びに実行には注意が必要である．また「京」では B 軸長が 3 であるため，分割数の制限で B 軸に 2 を割り当てる必要がある場合は，Y 軸と組み合わせてマップファイルを作成している．

表 3　Eigen ライブラリベンチマークでの 6 次元分割例

Eigen分割数	6次元分割パターン	対応6次元軸	実行形状
8×8	(2×2×2)×(2×2×2)	(A,X,C)(Z,YB)	4x4x4
16×16	(2×4×2)×(4×4)	(A,X,C)(Z,YB)	8x4x8
24×24	(2×3×4)×(2×3×4)	(A,B,X)(C,Z,Y)	8x12x6
32×32	(4×2×4)×(2×16)	(X,A,Z)(C,YB)	8x16x8
48×48	(2×3×8)×(2×4×6)	(A,B,X)(C,Y,Z)	16x12x12

作成したマップファイルを用いた Eigen ライブラリベンチマークの性能は表 4 の通りである．測定条件は，EigenExa 1.0 を用いて，5.3 節で紹介する実問題に対応する比較的小さな 19,000 元の対角化計算である．問題規模が小さいため，並列数が大きくなると通信時間の占める割合が無視できないサンプルである．8×8 分割，16×16 分割，並びに実問題での最速条件として採用している 32×32 分割での測定を行った．マップファイルを用いない場合(NOMAP)と 6 次元マップファイルを使用した場合(6D MAPFILE)での通信関数呼び出し回数の内訳を調べると，8×8 分割では集団通信の大部分は，Tofu 専用アルゴリズム(Tofu [conunt])に変更されている．Tofu 専用アルゴリズムの多くは長メッセージサイズ向けであり，Tofu 専用アルゴリズム未適用の箇所(Non-Tofu [count])は，通信サイズが小さいため適用条件を満たさなかったと判断できる．この傾向は並列数が多いほど顕著であり，16×16 以上の分割では，Tofu 専用アルゴリズムはほとんど適用されない．しかしこの条件でも両サブコミュニケータはより連続化され，通信は小さい直方体内に閉じるため通信時間の削減が期待できる．実際に 8×8 分割でライブラリの通信時間は 7.074[s]→5.445[s]．約 1[s]短縮され，全体で 5%程度であるが，性能は向上した．また 32x32 分割でも 0.2[s]程度とごくわずかであるが，性能向上がみられている．通信改善率が低い理由としては，マップファイルなしの場合でも，サブコミュニケータは比較的コンパクトな連続空間に割り当てられたためと解釈できる．

表 4　EigenExa 1.0 ベンチマークの測定結果

		8×8		16×16		32×32	
		NOMAP	6D MAPFILE	NOMAP	6D MAPFILE	NOMAP	6D MAPFILE
Tofu [Count]	Bcast	5	10813	5	5	5	5
	Allreduce	56	11249	56	56	56	56
	Allgather	133	10278	117	59	85	21
Non-Tofu [Count]	Bcast	36841	26021	38745	38745	38348	38348
	Allreduce	46034	34840	42085	42088	40785	40781
	Allgather	10145	0	2481	2540	63	127
Comm. Time [s]	Bcast	2.159	1.835	1.764	1.658	1.485	1.488
	Allreduce	4.381	3.542	4.380	4.005	3.779	3.604
	Allgather	1.008	0.467	0.405	0.347	0.075	0.093
	Total	7.548	5.844	6.550	6.010	5.339	5.185
Elapse Time [s]	Total	21.582	20.599	14.112	13.874	10.905	10.781

5.2 PHASE のバンド方向と分散 FFT 分割

PHASE とは，密度汎関数法を用いた量子力学計算を行うアプリケーションである[20]．演算量並びメモリ使用量が $O(N^3)$で増大するため超並列では分割数不足になる．超並列に対応するために，電子軌道に相当するエネルギーバンドの添字を持つループと密度汎関数をフーリエ級数展開した波数方向の添字の 2 方向で二軸並列を行っている[16]．

計算条件は SiC 電子軌道収束計算で用いている実問題を使用した．中規模な例として 8,920 原子(19,000 元)を 3,072 並列で，大規模な例として 12,248 原子(25,000 元)を 12,288 並列で，それぞれ 4 種類のソルバを 1 回ずつ計算した．

ランク番号は，アプリケーション内で，まずバンド方向に，次に波数方向に割り当てられており，5.1 節の Eigen ライブラリベンチマーク同様マップファイルでの対応が可能なアプリケーションである．問題は波数方向には二軸分散 3 次元 FFT が行われており，サブコミュニケータが更に分割されている点である．6 次元直接網上で 3 種類のサブコミュニケータ全てに Tofu 専用アルゴリズムを適用するのは不可能なため，通信の大部分を占める FFT に伴う転置による alltoall 通信の 2 種類の直交するサブコミュニケータについてのみ 3 次元直方体になるように再配置を行った．このためバンド方向のサブコミュニケータは不連続な空間(X, Y, Z 軸)に割り当てられる．分割方法は表 5 の通りである．alltoall 通信の 2 種類の分割に相当する FFT1 と FFT2 の分割数が 24×8, 48×8 と均等でない理由は，計算に用いた系の形状が 6:3:1 若しくは 14:7:2 と扁平であり，サブコミュケータ形状がより扁平でなく，FFT 短軸方向の分割数を少なくなるよう構成したためである．

通信時間内訳は表 6 の通りである．マップファイル無し(NOMAP)と，6 次元マップファイル有り(6D MAPFILE)の欄を比較すると，どちらの並列数でも，主要通信時間の合計は，47.2%～49.7%と半分以下に削減している．通信の内訳を見ると，通信最適化によりバンド方向の主要通信の 1 つである Gran-Schmidt の直交化で行われる bcast 通信の時間は 1.81 倍～2.01 倍へと増えたが，二軸分散 FFT 内の主要通信である alltoall 通信は両サブコミュニケータ同時に 19.1%～32.8%と大幅に減少している．

今回 alltoall 通信で適用されている Tofu 専用アルゴリズム double spread は，2 次元形状でも適用されるなど適用条件が緩く，プロセスマッピングを工夫することなく適用さ

れることがある．しかし 6 次元マップファイルを用いることで，より立方体に近い形状に通信が閉じ込めることが可能となり，演繹的に通信のホップ数の削減につながった．

5.3 PHASE 上の対角化計算の最適化

5.2 節で議論した PHASE 計算は，密度汎関数の固有値問題であり，解の収束性を高めるためにバンド方向のみの部分空間対角化計算を行っている．バンド方向は電子数程度の問題規模であるため，数千元～数万元と比較的小さな対角化計算であり，全プロセスを用いて計算すると，オーバーヘッドになる．PHASE では，全プロセスは使用せず，ScaLAPACK では, 行・列方向 16×16 分割の 256 並列[16], 5.1 節で紹介した EigenExa 1.0 ライブラリでは 32×32 分割の 1,024 並列を使用して計算している．この分割はバンド，波数分割とは独立のサブコミュニケータであり，ライブラリ内で実装されている．対角化ライブラリはアプリケーションから間接的に呼び出されるため，ランクの割り当てをファイルで直接指定する方法は困難である．そのため，4.3 節で紹介した 6 次元相対座標ランク番号問い合わせ関数を用いる方法を使用した．具体的には行・列方向各々の通信が表 3 のベンチマークで最速となった直方体のサブコミュニケータ形状になるようにアプリケーション内の 6 次元ループを作成し，相当するランク番号を 6 次元ランク番号問い合わせ関数で算出する．この情報を BLACS の機能である gridmap 関数を介してライブラリに与える方法である．効果は表 6 の通りである．

gridmap による最適化無し(6D MAPFILE)と gridmap による最適化有り(6D MAPFILE/GRIDMAP)を比較すると，対角化計算(Eigen Calc.)が，47.9%～51.2%へと短縮している．この効果は Eigen ライブラリベンチマークでの 5%削減に比べて大きい．但し，3,072 並列でマップファイルによる最適化無し(NOMAP)と比較して 11.306[s]→11.176[s]とベンチマーク同様若干の性能向上率となっている．

対角化計算は一部プロセスを用いて 1,024 並列で行っているため，行・列方向の両サブコミュニケータが 3,072 並列から切り出した時点では，比較的連続であったと推測できる(NOMAP)．しかしマップファイルを用いたランク配置のため，対角化計算のサブコミュニケータは逆に不連続化し，固有値計算が大幅に遅くなった(6D MAPFILE)．しかし不連続化したサブコミュニケータは gridmap により，ランク再配置最適化が施されたため，対角化計算はオリジナルより多少高速化した(6D MAPFILE/GRIDMAP)と解釈が可能である．

12,288 並列では，対角化のための 1,024 並列のサブコミュニケータを切り出した段階で既に連続性が失われ，対角化計算は倍程度遅くなったものと思われる(NOMAP)．この不連続性は alltoall 通信最適化のためのマップファイルで変わることはなく(6D MAPFILE)，gridmap によるランク配置最適化により，2 倍程度の高速化が達成できた(6D MAPFILE/GRIDMAP)と解釈が可能である．

この最適化方法により，他の通信性能に影響を与えることなく，対角化計算は 5.1 節のマップファイルを用いたベンチマーク同様の性能を得ることが可能となった．

表 5　PHASE での 6 次元分割例

PHASE分割数 (バンドxFFT1xFFT2)	6次元分割パターン	対応6次元軸	実行形状
3,072 (16×24×8)	(4×2×2)×(2×3×4)×(2×2×2)	(X,Y,Z)(X,B,Z)(A,Y,C)	16×12×16
12,288 (32×48×8)	(4×2×4)×(4×3×4)×(2×2×2)	(X,Y,Z)(Y,B,Z)(A,C,X)	16×24×32

表 6　PHASE の通信最適化

		3,072 proc.			12,288 proc.		
		NOMAP	6D MAPFILE	6D MAPFILE GRIDMAP	NOMAP	6D MAPFILE	6D MAPFILE GRIDMAP
Main Comm. [s]	Alltoall1	43.39	21.446	21.445	38.481	11.959	11.764
	Alltoall2	74.997	17.347	17.39	74.112	9.542	9.411
	bcast	13.247	26.627	26.703	23.534	42.782	45.043
	Total	131.634	65.42	65.538	136.127	64.283	66.218
Eigen Calc. [s]	Pre/Post	1.652	6.777	1.688	9.648	9.172	7.288
	Eigen Calc.	11.306	23.344	11.176	33.964	31.535	16.157
	Total	12.958	30.121	12.864	43.612	40.707	23.445
Elapse Time [s]	Total	832.974	778.272	755.393	814.456	740.732	666.492

5.4 通信の可視化

「京」には各ノード内の ICC を通過する通信のハードウェアカウンタ情報を採取する機能が用意されている．この機能を用いて，今回の最適化の可視化を行った．図 6 は，5.2 節の表 6 にて計算した 3,072 並列での計算結果について二軸分散 3 次元 FFT で行われる alltoall 通信について，ノード上で送信したデータ量をグレースケールで示したものである．数値は，通信最適化前後の 2 パターンのデータ送信量の最大値で規格化している．左図の通信最適化が行われる前に比べて, 右図の 6 次元通信最適化を行った方が，全体的に通信が少なくなり，色が濃いことが分かる．各ノードの ICC のデータ送信量は，直接網による中継された分も含むため，通信最適化前は，余計に経路を使用していると解釈できる．図中の通信量が多く，白く見える箇所は，最適化前で A=0, X=0 の 192 ノード, 最適化後は, X=0,1, Y=0～3, Z=0,1 の 192 ノードであり，それを 2 次元に射影したものである．この 192 ノードは band 方向のランク番号 0 に相当し, 波数方向のサブコミュニケータ内の通信である．データ送信量が各々約 10 倍程度多い．

表 7 は，同様に表 6 の 3,072 並列の計算における，各々の通信について，通信最適化前と 5.2 節，5.3 節で論じた通信最適化を行った時の各ノードの通信インターフェイスが空でない回数のカウント合計(Zero Credit [count])並びにデータ送信量の合計(Total Send [Byte])である．前者は通信データのつまり具合を示す量である．3,072 並列では，固有値ソルバの性能には大きな変化はなかったが，データのつまり具合，送信量ともに変化は少ない．また通信最適化が行われていないバンド方向の不連続なサブコミュニケータの bcast 通信については，つまり具合が約 250 倍，送信量

が約 4 倍に増え，通信時間も増えている．逆に，図 6 に相当する二軸分散 3 次元 FFT の alltoall 通信は，最適化の効果で，つまり具合で約半分，送信量も約半分に減少することで通信時間が削減されている．

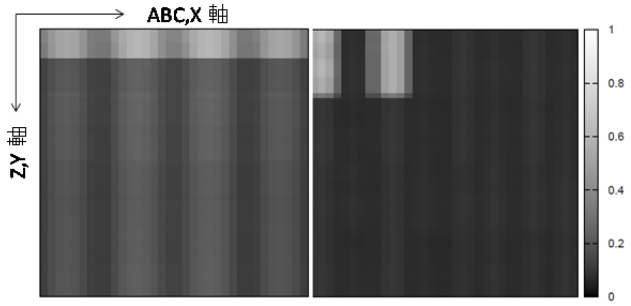

図 6　3 次元 FFT での主要な alltoall 通信について各ノードのデータ送信量合計．左図：最適化前．右図：最適化後．両図の最大値で規格化している

表 7　PHASE における主要通信の通信バッファが空でない回数合計とデータ送信量の全ノード合計

		NOMAP	6D MAPFILE GRIDMAP
Zero credit [count]	eigen	3.475E+10	3.465E+10
	alltoall	8.209E+12	4.850E+12
	bcast	6.156E+09	1.594E+12
Total Send [Byte]	eigen	4.177E+12	4.162E+12
	alltoall	3.704E+14	1.860E+14
	bcast	1.186E+13	4.672E+13

5.5 最適化に伴うスケーラビリティの向上

通信は一般に並列性能の阻害要因となるため，最適化により，高並列時のオーバーヘッドの削減が期待できる．今回，二軸分散 3 次元 FFT の両方向の alltoall 通信と，固有値ライブラリを同時に最適化することにより，通信時間を半分に減らすことが出来た. PHASE では高並列にてこれらの通信時間が並列性能の制限になっていたため，これらの制限が半分になることにより，約倍の並列性能が実現可能となった．PHASE のような収束計算では 1step を短くし，繰り返し数を多くすることが重要である．増大するオーバーヘッドを削減し並列性能が高まることで，より高並列での計算が可能になり 1step の計算時間を短く抑えられるため，反復計算に新たな可能性を生むことが想定される．

6. まとめ

本報で用いた通信最適化の手法は，昨今の多次元直接網，多軸並列という 2 つの潮流に対応した手法であるが，両者ともその起源は超並列にある．超並列に対して結線量の制限などからシステムは，full cross bar, fat tree, binary tree などから直接網への変更を余儀なくされ，並列性能の制限からアプリケーションは多軸並列という手法が必要不可欠になる．このことから，本手法は超並列時代のアプリケーション通信について汎用的な最適化手法といえる．

他システムを見ると，Sequoia では「京」と同じ 10 本のリンクを用いて，等価な 5 次元トーラスを構成している．軸長が均一なため，より立方体に近い形状切り出しが可能と思われ，この多次元性を活かせば，「京」の同様に 3 次元＋2 次元に通信の閉じ込めにより性能向上が見込める．また double spread のような 2 次元形状でも最適化されたアルゴリズムがあれば更なる効果が期待できる．TITAN では 6 本のリンクのうち 5 本を用いた Y 軸非対象の 3 次元トーラスである．通信の最適化には 3 次元中の 3 次元の切り出しが必要になり，複数のサブコミュニケータの同時最適化は比較的難しいと思われる．

一般的にアプリケーションは 3 方向の分割までと思われがちであるが，実アプリケーションを調べると，多重ループ構造の各々の方向で並列化が可能である．ネットワーク直接網を設計する上でも，今後，より多次元なシステム構成の需要が高くなることが想定される．

また「京」で用いられる Tofu ネットワークは，アプリケーション上で 3 次元トーラスを構成する際，故障ノードの回避を主目的として設計されているため，ノード内の分割長は 2×3×2 と小さい．このため 3 次元のサブコミュニケータを 2 つ切り出すと, 短辺サイズが 2～3 の扁平形状のサブコミュニケータや，サイズの異なるサブコミュニケータを構成せざるを得ない．今後，より各軸均等な多次元ネットワーク網が構築できれば，より容易に通信時間の最適化を行うことが可能になる．

また本手法は，冗長構成での使用しない結線を用いた通信最適化手法といえる．故障ノード回避を目的とした結線を使用しているため，実行ノード内に故障があると本最適化は適用不可能である．アプリケーション実行中のノード故障に関しては，本最適化を適用有無にかかわらず異常終了となり，再実行処理が施される．よって，本最適化によるノード故障の影響は，実行までの待ち時間増加につながる．更に実行時 6 次元全ての方向の長さを厳密に設定する必要があるため，ジョブ投入時に 3 次元かつ回転割り付け不可の設定をしなければならない．このことも実行待ち時間に増加につながる．しかし，占有利用など，他のジョブの影響が少ない場合には，その影響は無視できる．

多次元網に対応する最適化アルゴリズムについては，「京」では主に 3 次元向けに Tofu 専用アルゴリズムが用意されているが，将来アプリケーションの多様化に向けて，その他の次元でも開発が進められ，提供できるようになれば，より通信最適化の幅が広がるであろう．

謝辞 本報告に際し，システムソフトウェア開発者の立場で御討論頂いた，富士通株式会社次世代テクニカルコンピューティング開発本部の皆様，並びに理化学研究所計算科学研究機構運用技術部門の諸氏，理化学研究所計算科学研究機構に常駐して「京」の運用支援に携われている佐治隆行氏を始めとする富士通株式会社 SE の皆様に感謝します．本論文の結果は，理化学研究所計算科学研究機構が保有するスーパーコンピュータ「京」によるものです．

参考文献

1) 今出 広明, 平本 新哉, 三浦 健一, 住元 真司: 大規模計算環境のためのラン配置最適化手法RMATT, 先進的計算基盤システムシンポジウム SACSIS2011 論文集, Vol. 2011, pp.340-347 (2011).
2) 今出 広明, 平本 新哉, 三浦 健一, 住元 真司, 黒川 原佳, 横川 三津夫, 渡邊 貞: 大規模計算向け通信時間最適化ツール RMATT における実行時間の高速化, ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集, Vol. 2012, pp.93-100 (2012).
3) 小玉 知央, 寺井 優晃, 野田 暁, 山田 洋平, 佐藤 正樹, 清木 達也, 伊賀 晋一, 富田 浩文, 南 一生: 正二十面体格子におけるノードマッピング手法の開発と評価, 日本気象学会 2012 年秋季大会, 2012.10.03-05, C206, 札幌 (2012).
4) Hasegawa, Y., Iwata, J., Tsuji, M., Takahashi, D., Oshiyama, A., Minami, K., Boku, T., Shoji, F., Uno, A., Kurokawa, M., Inoue, H., Miyoshi, I. and Yokokawa, M.: First principles calculation of electronic states of a silicon nanowire with 100000 atoms on the K computer, SC '11 Proceedings of 2011 International Conference for High Performance Computing Networking Storage and Analysis, 2011.11.14-17, Washington State Convention Center Seattle WA, ACM (2011).
5) Andoh, Y., Yoshii, N., Fujimoto, K., Mizutani, K., Kojima, H., Yamada, A., Okazak, S., Kawaguchi, K., Nagao, H., Iwahashi, K., Mizutani, F., Minami, K., Ichikawa, S., Komatsu, H., Ishizuki, S., Takeda, Y. and Fukushima, M.: MODYLAS: A Highly Parallelized General-Purpose Molecular Dynamics Simulation Program for Large-Scale Systems with Long-Range Forces Calculated by Fast Multipole Method (FMM) and Highly Scalable Fine-Grained New Parallel Processing Algorithms, Journal of Chemical Theory and Computation 9, pp.3201-3209 (2013).
6) Maruyama, T.: SPARC64 VIIIfx: Fujitsu's New Generation Octo-core Processor for Peta Scale Computing., Hot Chips 21 (2009).
7) Maruyama, T.: SPARC64 VIIIFX: A New-Generation Octocore Processor for Petascale Computing, IEEE micro, Vol.30, No.2, pp30-40 (2010).
8) SPARC International, The SPARC Architecture Manual (Version 9), Prentice-Hall (1994).
9) Sparc Joint Programming Specification (JPS1): Commonality, architecture manual., Sun Microsystems and Fujitsu Ltd. (2002).
10) SPARC64VIIIfx Extensions, Fujitsu Ltd., architecture manual (2008).
11) Ajima, Y., Sumimoto, S. and Shimizu, T.: Tofu: A 6D Mesh/Torus Interconnect for Exascale Computers., IEEE Computer, pp.36-40 (2009).
12) Toyoshima, T., ICC: An interconnect controller for the Tofu interconnect architecture., Hot Chips 22 (2010).
13) 住元 真司, 川島 崇裕, 志田 直之, 岡本 高幸, 三浦 健一, 宇野 篤也, 黒川 原佳, 庄司 文由, 横川 三津夫: 「京」のための MPI 通信機構の設計, 先進的計算基盤システムシンポジウム論文集, 2012, pp.237-244 (2012-05-09).
14) 松本 幸, 安達 知也, 田中 稔, 住元 真司, 曽我 武史, 南里 豪志, 宇野 篤也, 黒川 原佳, 庄司 文由, 横川 三津夫: MPI_Allreduce の「京」上での実装と評価, 研究報告計算機アーキテクチャ(ARC), 2011-ARC-197(6), pp.1-10 (2011-11-21).
15) Adachi, T., Shida, N., Miura, K., Sumimoto, S., Uno, A., Kurokawa, M., Shoji, F. and Yokokawa, M.: The design of ultra-scalable MPI collective communication on the K Computer, Proc. of ISC'12, June 17-21, Hamburg, Germany, pp.1-8 (2012).
16) 黒田 明義, 長谷川 幸弘, 寺井 優晃, 井上 俊介, 市川 真一, 小松 秀実, 大井 憲行, 安藤 琢也, 山崎 隆浩, 大野 隆央, 南 一生: ナノ材料第一原理分子動力学プログラム PHASE の京速コンピュータ「京」上の計算性能最適化, ハイパフォーマンスコンピューティングと計算科学シンポジウム論文集, Vol. 2012, pp.144-152 (2012).
17) Ribeiro, C. P., Hutter, J., Vondele, J.V.: Improving Communication Performance of Sparse Linear Algebra for an Atomistic Simulation Application, International Conference on Parallel Computing - ParCo2013, Munich, Germany (2013).
18) Yamada, S., Imamura, T., Kano, T. and Machida, M.: High-Performance Computing for Exact Numerical Approaches to Quantum Many-Body Problems on the Earth Simulator, ACM/IEEE SC'06, Tampa, USA (2006).
19) Imamura, T., Yamada, S., and Machida, M.: Development of a high performance eigensolver on the petascale next generation supercomputer system, Proceedings of Joint International Conference on Supercomputing in Nuclear Applications and Monte Carlo 2010 (SNA+MC2010), (2010).
20) PHASE, http://www.ciss.iis.u-tokyo.ac.jp/riss/project/device/