通信タイミングを考慮した衝突削減のためのMPIランク配置最適化技術

森江, 善之
九州大学大学院システム情報科学府

末安, 直樹
富士通株式会社ソフトウェア事業本部ミドルウェアコンポーネント事業部

松本, 透
富士通株式会社ソフトウェア事業本部ミドルウェアコンポーネント事業部

南里, 豪志
九州大学情報基盤センター

他

出版情報：Symposium on Advanced Computing Systems and Infrastructures. 2007, pp.283-292, 2007-05-25
バージョン：
権利関係：

通信タイミングを考慮した衝突削減のための MPI ランク配置最適化技術

森江善之[†1] 末安直樹[†2] 松本透[†2]
南里豪志[†3] 石畑宏明[†3,†4]
井上弘士[†5] 村上和彰[†5]

本稿では，通信性能の悪化の主要因である通信の衝突を避けるためのランク配置最適化技術の提案を行う．通信のタイミングを考慮することで，通信の衝突を回避する高精度な MPI ランク配置を行う目的関数の提案を行った．また，本稿提案の目的関数を適用することによる通信時間の削減効果を調べる評価実験を行った．対象プログラムとして recursive doubling の通信パターンや CG 法，umt2000 といったアプリケーションの通信パターンを用いた．評価実験では，通信時間が順配置に対して最大 45%，従来研究によるランク配置に対して最大 24%通信時が削減され，提案手法が有効であることがわかった．

Optimization of MPI rank allocation considering communication timing for reducing contention

Yoshiyuki Morie,[†1] Naoki Sueyasu,[†2] Toru Matsumoto,[†2]
Takeshi Nanri,[†3] Hiroaki Ishihata,[†3] Koji Inoue[†5]
and Kazuaki Murakami [†5]

In this paper, it proposes the optimization of rank allocation technology of avoiding the communication contention that is the key factor of the communication performance degradation. It proposes the objective function for high-quality Optimization of MPI rank allocation to be able to avoid a communication contention by considering the communication-timing of each message. Moreover, the evaluation experiment to check how does this object function cut down communication time. The communication pattern of the recursive doubling algorithm and the communication pattern of the application such as CG and umt2000 are used in the evaluation experiments. The ratio of reduction in the communication time are 45% or less for order rank allocation, 24% or less for previous work rank allocation in the experiment.

1. はじめに

近年の先端科学技術計算においては，さらに性能要件が厳しくなっている．先端科学技術計算は並列計算機を用いて計算されることが多い．このため，この分野では並列計算機に対してより高い性能向上を求めている．このような要求に対して，大量の計算ノードを搭載した分散メモリ型の計算機の導入が進められている．特にスーパーコンピュータの計算ノード数は非常に大きいものとなってきている．例えば，2006 年 11 月発表のスーパーコンピュータ性能ランキング TOP500[6] において 1 位となった IBM の BlueGene/L では約 6 万個の計算ノード (2CPU/1node) を用いている．

しかし，計算ノード数が大きくなるとそのネットワークを多段の Crossbar や Fat-Tree などの高速か

†1 九州大学大学院システム情報科学府
Graduate School of Information Science and Electrical Engineering, Kyushu University
†2 富士通株式会社ソフトウェア事業本部ミドルウェアコンポーネント事業部
Middleware Components Division, Software Unit, FUJITSU LIMITED
†3 九州大学情報基盤センター
Computing and Communications Center, Kyushu University
†4 富士通 (株) サーバシステム事業本部次世代 HPC 開発企画室
Next Generation HPC Development Planning Division, Server Systems Unit, FUJITSU LIMITED
†5 九州大学大学院システム情報科学研究院
Graduate School of Information Science and Electrical Engineering, Kyushu University

つ均質なネットワークで構成することは価格や電力などのコスト，設置面積などの問題で困難となる．そこで，結線数やスイッチ数を削減するため，3D-mesh，3D-Torus や Tree などのネットワークトポロジで大規模並列計算機のネットワークを構築することが増加している．これらのネットワークトポロジでは通信先や通信元が同一でない場合も経路上で複数のメッセージが同時に通信される，つまり通信の衝突が発生する可能性がある．このようなネットワークトポロジを不均質なネットワークトポロジと呼ぶ．

本研究は，不均質なネットワークトポロジを起因とした通信の衝突による通信時間の増大を緩和することが目的である．

本研究では，通信のタイミングを調べ，通信の衝突を回避するようにタスクを計算ノードに割り付けることにより通信時間の削減を図る最適化を提案する．このようにタスクの割り付け方を変更することにより最適化を行うことをタスク配置最適化と呼ぶ．本研究と同様に通信時間の削減を目的とするタスク配置最適化の既存研究も多数存在する[1)~5)]．本稿では，MPI プログラムを対象としているため，タスク配置最適化のことをランク配置最適化と呼ぶ．ランクとは MPI プログラムの実行時にタスク (MPI のプロセス) に付けられた ID のことである．

2 章では，従来研究の紹介を行う．3 章では，通信タイミングを考慮したランク配置最適化の提案を行い，4 章では，提案手法の評価実験について述べる．最後に 5 章でまとめと今後の課題について述べる．

2. 従来研究

T. Hatazaki[2)] や J. L. Traff[3)] の提案したランク配置最適化では，ツリー上のネットワークトポロジを対象としている．このネットワークトポロジでは上段になるほど低速になる階層型のネットワークを想定しており，通信のやり取りを多く行うランク同士が上位のネットワークを使わないように配置する．これにより上位の低速なネットワークを使われることが減り，通信時間の削減が行われる．しかし，これらの研究では，通信の衝突は考慮に入れられていない．

F. Ercal ら[1)]，T. Agarwal ら[4)] や G. Bhanot ら[5)] の提案では，Hypercube，3D-Mesh，3D-Torus や Tree などの不均質なネットワークトポロジを対象としている．これらのネットワークでは，計算ノードやスイッチ間のリンクの通信性能はすべて等しいと仮定する．これらのネットワークトポロジでは，直接接続していない計算ノード同士が多数存在する．これらの計算ノードが通信を行う際は，複数の計算ノード，またはスイッチを通過する通信を行わなければならない．このことは通信遅延の増加や通信の衝突を発生させ，通信時間を増加させる．このため，これらの研究では，通信遅延や通信の衝突を削減するため，通信が計算ノードやスイッチを通過しないようにランクを割り付けるランク配置最適化手法を提案している．このランク配置最適化により，通信遅延や通信の衝突が減少し，通信時間の削減が行われる．近年の高性能ネットワークにおいて，リンクやスイッチを通過することによる通信遅延の増加は，MPI 通信ライブラリやネットワークドライバなどのソフトウェアの実行にかかるオーバーヘッドに比べて小さい．これに対して通信の衝突による性能の低下は，プログラム性能に影響するほど大きい場合があり，通信の衝突に重点を置いて最適化を施すべきと考えられる．これらの従来手法では，スイッチや計算ノードを通過する通信が少なくなるので，結果的に通信が衝突する可能性が低くなるが，通信のタイミングを考慮しないため，通信の衝突の回避を十分に行えない．

3. 通信タイミングを考慮したランク最適化

本研究では，通信タイミングを考慮し，通信の衝突を回避するようにランク配置を決めることでより高速な通信を可能とするランク配置最適化の提案を行う．

3.1 通信タイミングを考慮したランク配置の例

2 つのランク配置を示し，通信タイミングを考慮してランク配置を決定することの重要性を示す．

図 1 のような通信パターンで通信を実行するプログラムがある．はじめのステップでランク 1 とランク 2 の間で通信を行い，次のステップでランク 4 とランク 5 の間で通信を行う．さらに，次のステップでランク 1 とランク 3，ランク 4 とランク 6 の間で通信を行い，最後のステップでランク 1 とランク 6，ランク 2 とランク 4，ランク 3 とランク 5 の間で通信を行う．この時，通信は各ステップにならなければ行えないものとする．

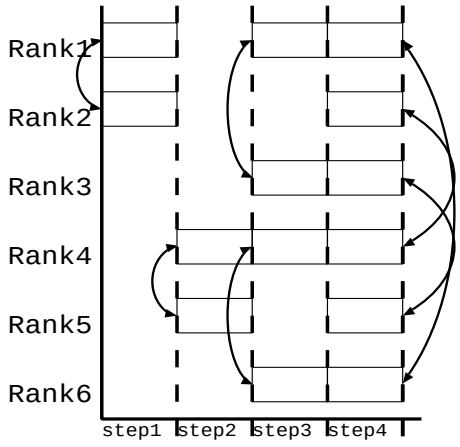

図 1 通信パターン

このような通信パターンを持つプログラムを実行する際に各ランクをツリー型ネットワークを持つ 6 ノードの並列計算機に割り当てる．図 2 のように，スイッチ A に接続する計算ノードにランク 1，2，3，スイッチ B に接続する計算ノードにランク 4，5，6 が割り当てる．このランク配置をランク配置 1 とする．

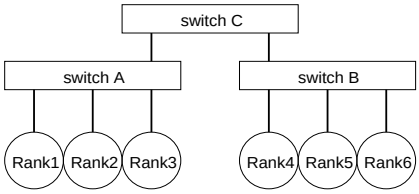

図 2 ランク配置 1

図 3 のように第 1，2，3 ステップではスイッチ C を通過する通信は行われない．第 4 ステップでは，ランク 1 とランク 6 の通信はそれぞれ違うスイッチに接続しているため，スイッチ C を通過する通信を行う．また，ランク 2 とランク 4，ランク 3 とランク 5 の通信も同様にスイッチ C を通過する通信を行う．すなわち，第 4 ステップにおいて 3 つの通信が同時にスイッチ C を通過する通信を行うため，上位のスイッチと下位のスイッチ間の経路で通信の衝突が起き，実行時間が増大する．

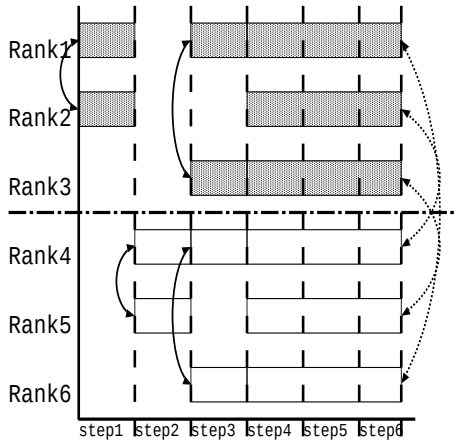

図 3 ランク配置 1 の時の通信の様子

次に通信のタイミングを考慮に入れ，通信の衝突が起きないようにランク配置を行う．図 4 のように，スイッチ A に接続する計算ノードにランク 1，3，5，スイッチ B に接続する計算ノードにランク 2，4，6 を割り当てる．このランク配置をランク配置 2 とする．

このランク配置での通信の様子を図 5 に示す．第 1 ステップでは，ランク 1 とランク 2 がそれぞれ別のスイッチに接続しているため，スイッチ C を通過する通信を行う．しかし，これ以外の通信は行わないため，通信の衝突は発生しない．第 2 ステップも第 1 ステップと同様に通信の衝突は発生しない．また，第 3 ステップでは，通信を行うランク同士がともに同一スイッチに接続しているため，スイッチ C を通過する通信は行われず，通信の衝突は発生しない．最後に第 4 ステップであるが，ランク 1 がスイッチ A，ランク 6 がスイッチ B に接続しているため，このランク間の通信はスイッチ C を通過する通信を行う．しかし，これ以外の通信は同一スイッチ内で行われるため通信の衝突は発生しない．このため，ランク配置 1 に比べ，ランク配置 2 ではより高速に通信を行うことができる．

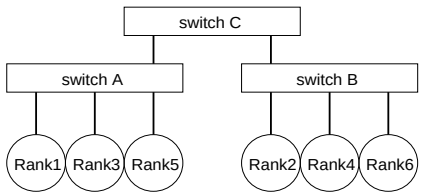

図 4 ランク配置 2

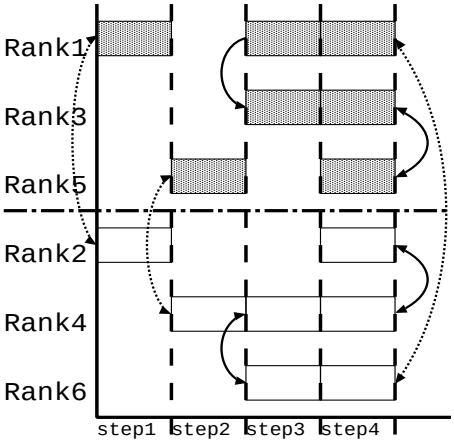

図 5 ランク配置 2 の時の通信の様子

ランク配置 1 とランク配置 2 はどちらもスイッチを通過する通信の数が 3 であるので，T. Agarwal ら[4] や G. Bhanot ら[5] の提案したランク配置最適化で用いられている目的関数では同じ通信コストであるとみなされる．このことから通信の衝突を回避するランク配置最適化では，目的関数を作る際に通信毎のタイミングを考慮する必要があることが分かる．

3.2 通信タイミングを考慮した目的関数

前節のようなランク配置最適化を行うためには，メッセージごとに衝突が起こるかどうか考慮してランク配置最適化を行うことが必要である．このため，式 1 に示す目的関数を提案する．この目的関数では，各通信フェーズにおける予想通信時間の最大値の総和を求める．これにより，各通信フェーズにおける通信の衝突による通信時間の増加を推定する．

通信フェーズとは，同時に実行可能なすべての通信が終了するまでの期間とする．通信フェーズの分割例を図 6 に示す．プログラムから rmsg と smsg2 間には依存関係があることが分かる．これにより，smsg, rmsg を通信する MPI 通信関数と smsg2，rmsg2 を通信する MPI 通信関数は同時に実行することはできないと分かる．同時に実行が不可能な MPI 通信なので，それぞれ通信フェーズ 1，2 とする．

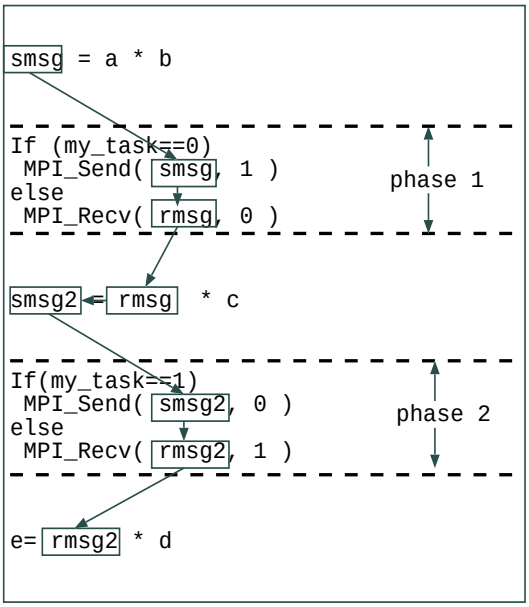

図 6 通信フェーズの分割例

このように通信フェーズという概念を導入することで，通信の衝突の時間変化を検出することができる．通信の衝突の時間変化とは，各通信フェーズおける通信の衝突回数の時間的な変化のことをいう．全ての通信フェーズで通信の衝突回数が等しければ，通信の衝突の時間変化は無いという．

$$\Sigma_t^N \max_i^n (L(\pi(i),\pi(tork(i))) + S(t,i,tork(i)) \times coll(t,\pi(i),\pi(tork(i))) / B(\pi(i),\pi(tork(i)))) \quad (1)$$

t は通信フェーズの順序，N は通信フェーズの総数である．i はランク，n はランク総数である．$\pi(i)$ はランク i が割り付けられている計算ノードを返す関数である．$tork(i)$ はランク i の通信先のランクを返す関数である．$L(p,q)$ と $B(p,q)$ はそれぞれ計算ノード p，q 間の通信遅延と通信帯域幅を返す関数である．$S(t,i,j)$ は通信フェーズ t におけるランク i からランク j へのメッセージサイズを返す関数である．また，$coll(t,p,q)$ は通信フェーズ t における計算ノード p，q 間の各経路で発生した通信の衝突回数の最大値を返す関数である．関数 $coll$ は計算ノード間の経路上で通信の衝突がない場合は 1 を返す．

このように目的関数を定義することで通信フェーズごとに通信の衝突を加味することが可能となる．このため，この目的関数を最小にするランク配置を求めることは，全通信フェーズを通した通信の衝突回数を最小とするランク配置を求めることと同義となり，全通信時間の削減を図ることができる．

3.3 実行方法

提案手法は以下のような実行手順を想定している．まず，MPI プログラムの解析を行い，MPI 通信関数のメッセージサイズ，通信パターンを取得する．この時，通信パターンとプログラムソースコードから各通信の通信フェーズを決定し，通信フェーズごとに通信の情報を管理する．また，システム管理者もしくはユーザが対象並列計算機の計算ノードやスイッチ間の接続関係を与える．これらの情報を目的関数の入力としてこの目的関数の値が最小となるようなランク配置を求める．

4. 実　　験

通信タイミングを考慮したランク配置最適化の有効性を示すための評価実験を行う．

4.1 実験環境

実験環境としては表 1 に示す PC クラスタを用いる．ネットワークは，図 7 のように 5 つのスイッチによる 2 段のツリートポロジである．

表 1 実験環境

CPU	Intel Xeon 3.0GHz
L2 キャッシュ	2MB
RAM	7GB
計算ノード数	16(2CPU/nodes)
ネットワーク	Gigabit Ethernet
トポロジ	2 段のツリー
OS	RedHat Enterprise Linux AS (Linux kernel 2.4.21)
コンパイラ	gcc version 3.2.3
MPI ライブラリ	mpich-1.2.7p1[9]

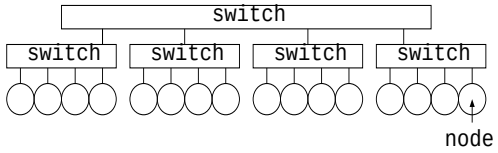

図 7 PC クラスタのネットワークトポロジ

また，対象プログラムとして，全体全の集団通信である allreduce などで用いられる recursive doubling の通信パターン，および実アプリケーションの通信パターンを用いる．実アプリケーションの通信パターンとしては，NAS Parallel Benchmark[8] の CG 法 (Conjugate

Gradient method) と ASCI Purple Benchmark[7] の umt2000(Unstructured Mesh Transport 2000) の通信パターンを用いる．

4.2 比較対象

導入した目的関数の有効性を調べるため，順配置，および従来手法によるランク配置についても通信時間の評価を行う．これらのランク配置と提案手法によるランク配置を適用した際の各プログラムの通信時間の比較を行う．

順配置とは，図 8 に示すようにスイッチごとに 1 から順にランクを計算ノードに割り当てるランク配置のことをいうこととする．順配置は MPICH[9] など MPI ライブラリを導入する際に多く採用されるランク配置である．

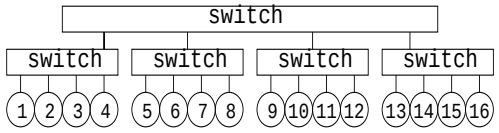

図 8 順配置

また，従来手法では，T. Agarwal ら[4] や G. Bhanot ら[5] が提案している目的関数を用いる．T. Agarwal らや G. Bhanot らの提案では，不均質なネットワークトポロジを対象としており，本研究と対象が同じである．また，通信の衝突を考慮できる点も本研究との比較評価実験を行う際に有用であると考えられる．従来手法で提案している目的関数を式 2 に示す．

$$\Sigma_i^n \Sigma_j^n C(i,j) H(\pi(i), \pi(j)) \qquad (2)$$

i, j はランク，n はランク総数である．$\pi(i)$ はランク i が割り付けられている計算ノードを返す関数である．$C(i,j)$ はランク i からランク j への通信量を返す関数である．また，$H(p,q)$ は計算ノード p から計算ノード q へのホップ数を返す関数である．ホップ数とは，通信がリンクを通過する回数のことをいう．計算ノード間の通信量とホップ数の積をホップバイトと呼ぶ．1 バイトのデータを 1 ホップ分離れた計算ノード間で通信する際の通信コストを 1HB とする．この目的関数は全ての計算ノード間のホップバイトの総和となる．このため，この目的関数を最小とするランク配置では，通信量の多いランク同士をホップ数が少なくなるような計算ノード同士に割り付けることになる．

4.3 実験方法

通信パターンとデータの依存関係から通信フェーズを決定する．まず，recursive doubling では，i ステップ目において各ランクが 2^i 離れたランク同士で送受信を行うアルゴリズムである．ステップ間の通信に依存関係があるため，同時に通信を行うことができない．まず，ここで通信フェーズを区切る．さらにステップ内における送受信は，同時には行えない．そこで，ここではランクの小さい方から大きい方への送信を先に実行するとして通信フェーズを区切る．

次に CG 法では，ランクを正方行列の順にならべ，各ランクを行番号 i と列番号 j に対応付ける．行番号が等しいランク間で recursive doubling を行い，その後，行番号 i と列番号 j をもつランクと行番号 j と列番号 i をもつランクで送受信を行う．recursive doubling とその後の送受信にはメッセージに依存関係があり同時に実行できない．そこで，まず recursive doubling の直後で通信フェーズを区切る．次に同一行番号で行う recursive doubling では，前述の recursive doubling と同様に通信フェーズを区切る．recursive doubling の後の送信と受信も，同時には行えない．このためランクの小さい方から大きい方への送信を先に実行するとして通信フェーズを区切る．

最後に umt2000 では，まず，最初に同期を取った後，すべての通信が非同期に行われる．非同期に行われる通信の通信順序は任意であるので以下の方針で通信フェーズを決める．まず奇数番目の通信フェーズでは小さいランクから優先的に通信を割り当てる．一方偶数番目の通信フェーズでは奇数番目の通信フェーズで送信をしていたランクが受信，受信していたランクが送信を行う．

これらを入力として従来手法によるランク配置最適化と提案手法によるランク配置最適化を行う．

今回は 2 段のツリートポロジを想定しているので，ルーティングは行われず，同一通信フェーズにおいてスイッチ間の同経路を通過する MPI 通信関数が複数あれば，通信の衝突が発生したとして目的関数の値を増加させる．本実験環境では利用するスイッチが双方向通信に対応しているため，送信先のスイッチへ複数の送信が行われた場合に衝突が発生したとする．

ランク配置は，解となりうるランク配置を全て探索し，各手法の目的関数の値を最小とするランク配置を求める．通信の衝突を考える際は同一スイッチに直接接続している計算ノードのランク配置を考慮する必要はないため，これらのランク配置は探索しない．また，評価実験において対象とするネットワークは対称な Tree であるため，ランク配置が逆順となるランク配置の探索も行わない．また，評価実験で対象とする並列計算機の計算ノードが 16 ノードであるため現実的な時間で求めることができる．このように提案手法や従来手法の理論最適解を求めることで，提案手法によるランク配置の従来手法によるランク配置に対する最大の通信時間の削減効果がわかる．

決定したランク配置でプログラムを実行し，全通信フェーズの合計所要時間を計測する．また，計測結果としてはプログラムを 1000 回実行した平均値を用いる．

実行の前に，図 9 のように通信フェーズの先頭に同期関数である MPI_Barrier を挿入する．これはランク配置の変更により各メッセージの通信時間に変化が発生し，その通信以降の通信のタイミングがずれてし

まうことで想定していない通信の衝突が発生しまう状況を防ぐための処置である．MPI_Barrier を挿入することで同一通信フェーズの MPI 通信関数をほぼ同時に実行することが可能となる．

```
smsg = a * b

MPI_Barrier(comm)                    ↑
If (my_task==0)                      |
 MPI_Send( smsg, 1 )          phase 1 |
else                                 |
 MPI_Recv( rmsg, 0 )                 ↓

smsg2 = rmsg  * c

MPI_Barirrer(comm)                   ↑
If(my_task==1)                       |
 MPI_Send( smsg2, 0 )         phase 2 |
else                                 |
 MPI_Recv( rmsg2, 1 )                ↓

e= rmsg2 * d
```

図 9 MPI_Barrier の挿入

通信フェーズ毎に MPI_Barrier の挿入を行うため，1 回の MPI_Barrier にかかる時間と通信フェーズ数の積がオーバーヘッドとして加わることになる．MPI_Barrier のコストはメッセージサイズが十分に大きく，各通信フェーズの時間が長ければ無視できる．今回の評価実験では，サイズの違うメッセージ同士の通信の衝突に対応していないため，メッセージサイズを等しくして実験を行った．ここでは CG 法の問題規模をクラス C，ランク数を 16 個と設定して実行した際のメッセージサイズである 300KB を用いた．

4.4 実験結果

まず，各ランク配置による各プログラムの通信の様子を示す．

recursive doubling に順配置と提案手法によるランク配置を適用した時の通信の様子をそれぞれ図 10，図 11 に示す．図中の点線は，スイッチの境界を示す．また，矢印はランク間の通信を示し，点線をまたぐ矢印はスイッチをまたぐ通信を示す．この recursive doubling を用いた実験では，従来手法によるランク配置は順配置と同一のものとなった．

次に CG 法に順配置と提案手法によるランク配置を適用した時の通信の様子をそれぞれ図 12，図 13 に示す．この時の従来手法によるランク配置は順配置と同一のものとなった．

umt2000 に順配置，従来手法によるランク配置と

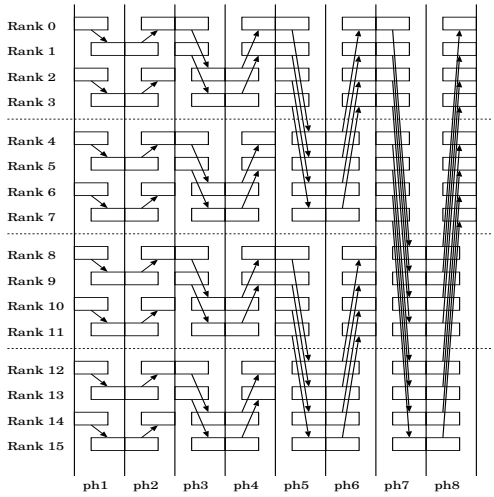

図 10 recursive doubling に順配置を適用した時の通信の様子

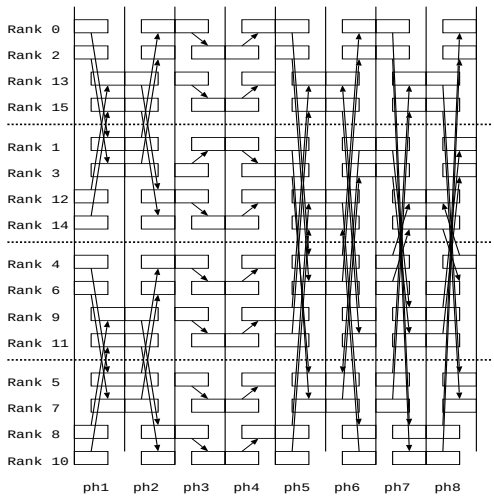

図 11 recursive doubling に提案手法によるランク配置を適用した時の通信の様子

提案手法によるランク配置を適用した時の通信の様子をそれぞれ図 14，図 15，図 16 に示す．

まず，表 2 にそれぞれのランク配置の時の通信時間の実測値，カッコ内に順配置に対する削減率を示す．本稿の提案手法によるランク配置での通信時間は recursive doubling で 41,829usec, CG 法で 25,370usec, umt2000 で 60,006usec となり，通信時間が順配置に対して，それぞれ約 22%，16%，45%削減された．従来手法によるランク配置に対しても，通信時間がそれぞれ約 22%，16%，24%削減された．

recursive doubling や CG 法では，従来手法によるランク配置は順配置と同一であった．これは，recursive doubling や CG 法では，スイッチを通過する通信の数を最小にするランク配置が順配置と一致するからである．一方，提案手法では，recursive doubling

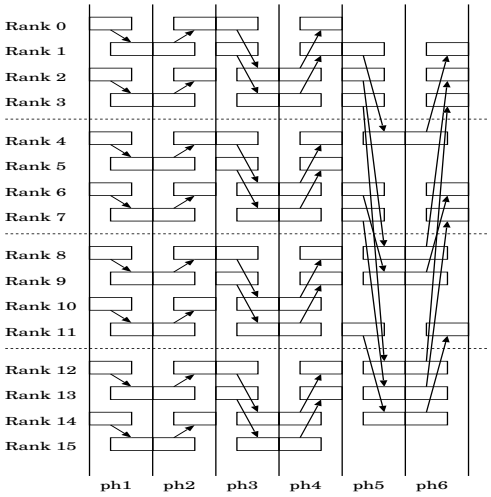

図 12 CG 法に順配置を適用した時の通信の様子

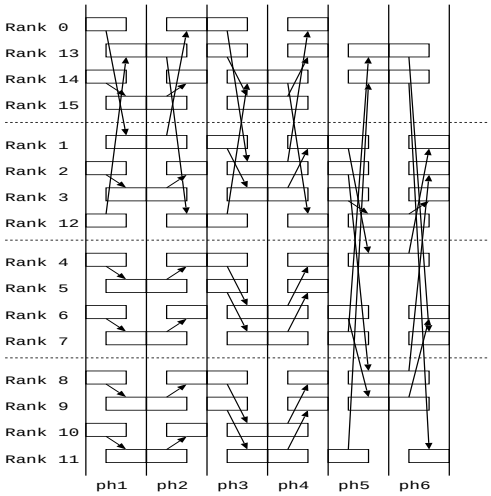

図 13 CG 法に提案手法によるランク配置を適用した時の通信の様子

や CG 法には通信の衝突に時間変化があるため，スイッチを通過する通信の数が少々増えようとも衝突をしないようなランク配置を選ぶことができるので，通信時間の削減につながったと考えられる．

umt2000 では，ランクの通信先に偏りがあるため，その偏りに対応してお互いに通信のあるランクは同一スイッチに接続する計算ノードに割り当てることで高速になる．これは，従来手法でも可能であるため，16%ほどの通信時間の削減が行えている．一方，umt2000 には通信の時間変化もあるため，提案手法を適用した場合は従来手法に対して通信時間がさらに 24%削減された．また，表 2 より，順配置に対する通信時間の削減率の傾向としては通信フェーズ数が多い方が高くなっているといえる．これは，通信フェーズ数が多いプログラムの方が通信の衝突を回避できる可

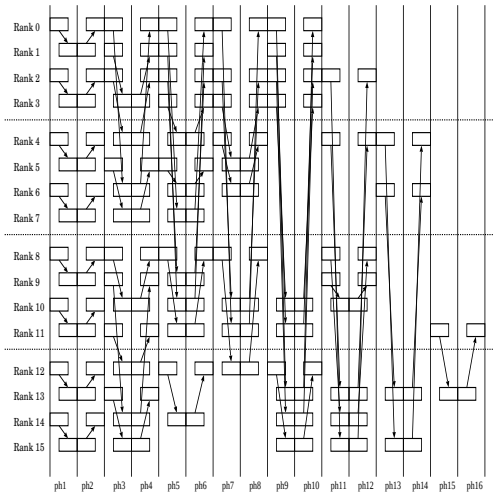

図 14 umt2000 に順配置によるランク配置を適用した時の通信の様子

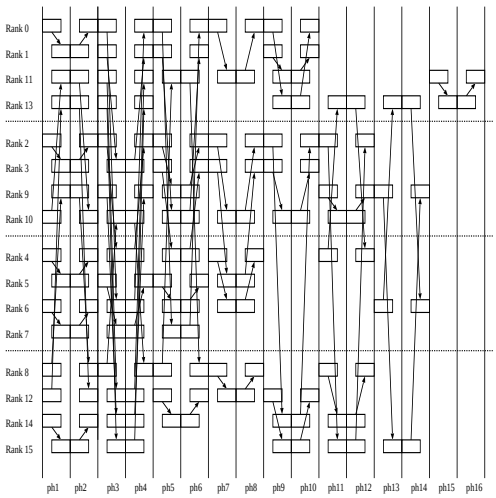

図 15 umt2000 に従来手法によるランク配置を適用した時の通信の様子

表 2 通信時間の実測値

	recursive doubling	CG 法	umt2000
提案手法	41,829μsec (22.00%)	25,370μsec (15.61%)	60,006μsec (45.12%)
従来手法	53,629μsec (0.00%)	30,061μsec (0.00%)	79,065μsec (27.68%)
順配置	53,629μsec (0.00%)	30,061μsec (0.00%)	109,332μsec (0.00%)

能性が高くなっているからだと考えられる．

4.5 考　　察

従来手法が 3.1 節のように提案手法と同じランク配置を出力できる可能性がないか調べる．従来手法によるランク配置最適化において目的関数の値を最小とするランク配置が複数あり，探索順序の関係などにより

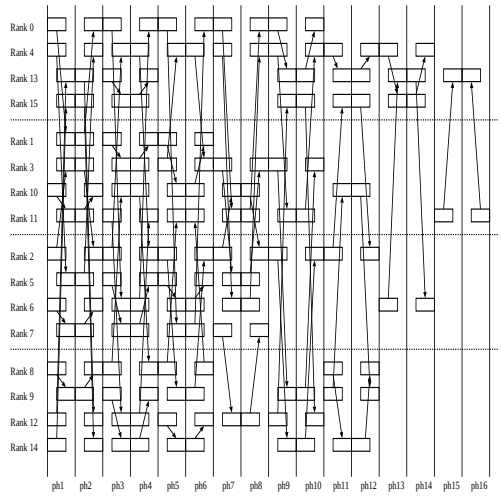

図 16 umt2000 に提案手法によるランク配置を適用した時の通信の様子

表 3 各ランク配置に対する従来手法の目的関数値

	recursive doubling	CG 法	umt2000
提案手法	68,812,800HB (16.67%)	38,092,800HB (10.71%)	77,414,400HB (5.00%)
従来手法	58,982,400HB (0.00%)	34,406,400HB (0.00%)	73,728,000HB (0.00%)
順配置	58,982,400HB (0.00%)	34,406,400HB (0.00%)	76,185,600HB (3.33%)

異なるランク配置を出力されることが考えられる．その中に提案手法によるランク配置と同一のランク配置があるという可能性がある．各手法によるランク配置に対する従来手法の目的関数値を表 3 に示す．カッコ内に従来手法によるランク配置に対する削減率を示す．提案手法は，従来手法によるランク配置に対して，目的関数値が recursive doubling で約 17%，CG 法で約 11%，umt2000 で約 5%増加している．このことから本評価実験において従来手法では，探索順序に関係なく提案手法によるランク配置と同一のランク配置を出力することはないことが分かる．

ここで，提案手法の目的関数である予想通信時間の精度を考えてみる．目的関数の精度が低いとランク配置最適化を行っても通信性能が向上しない，もしくは逆に通信性能が悪化するということが考えられるからである．表 4 に，各手法によるランク配置に対する提案手法の目的関数が出力した予想通信時間を示す．また，カッコ内に順配置に対する削減率を示す．まず，通信時間の絶対値は約 8〜23%ほど誤差があり，目的関数の精度は高いとはいえない．誤差を起こす要因としては，以下のものが考えられる．通信帯域幅を 125MB/sec で一定であると仮定していることが可能性のひとつとして考えられる．実際に Round Trip

表 4 各ランク配置に対する提案手法の予想通信時間

	recursive doubling	CG 法	umt2000
提案手法	34,806μsec (29.76%)	19,961μsec (19.76%)	49,952μsec (40.79%)
従来手法	49,552μsec (0.00%)	24,876μsec (0.00%)	67,155μsec (20.39%)
順配置	49,552μsec (0.00%)	24,876μsec (0.00%)	84,358μsec (0.00%)

Time から通信帯域幅を求めると 100〜125MB/sec であった．また，MPI_Barrier の通信時間を含めていないこともあげられる．本実験環境では MPI_Barrier1 回に 150μsec かかる．umt2000 は通信フェーズ数が 16 であるため，2400μsec のオーバーヘッドが加わる．さらに送信元のスイッチでの衝突を調べていないこともあげられる．誤差に幅がある点については，今回の目的関数では，同一通信フェーズにおいて複数のメッセージが同じスイッチを通過する場合は完全に衝突したと仮定するからであると考えられる．実際の通信では，同一通信フェーズでも完全に衝突する場合もあれば，不完全な衝突になる場合も考えられる．誤差が許容できない範囲になっているとするとこれらの点について調査する必要がある．

ランク配置最適化を考えるにあたっては通信時間の削減の傾向が予想通信時間と実際の通信時間の間で等しければ，実用上は問題ないと考えられる．そこで，順配置に対する提案手法によるランク配置との削減率の差を調べてみた．表 2,4 から各手法によるランク配置に対する順配置の削減率について通信時間の実測と予想通信時間との差を取った．この誤差は，約-7%〜8%が発生していることが分かった．この誤差も実際の通信時間の削減率によっては許容できる誤差とは言えない．今回の評価実験は最小でも約 16%程の通信時間の削減が見られたのでこの誤差は問題ないと考えているが，今後，この誤差により，通信性能が逆に悪化する場合がないかを調べる必要があると考えられる．

今回の評価実験では提案手法においてランク配置の探索を行うのに従来手法の 4 倍程度の時間がかかった．提案手法では通信フェーズのごとに通信の衝突などを調べることが必要だからである．通信フェーズ数が多数ある場合にはランク配置の探索にかなり時間がかかると考えられる．通信フェーズが多数あり，かつ，ランク配置の求解に十分に時間をかけられない状況では提案手法を適用することはできないと考えられる．

最後にランク配置の求解にかかる時間について考える．ランク配置の可能解を探索するための計算量は計算ノード数を n とすると $O(n!)$ となる．計算ノード数が大きくなると爆発的に計算量が増加し，最適解を求めることは難しい．このため，大規模並列計算機への適用を考える際は近似アルゴリズムを用いる必要がある．特に計算ノード数が多い場合は解の精度保証を

することは難しいため，Simulated Annealing や Genetic Algorithm のような発見的手法を用いる必要があると考えられる．発見的手法は局所最適解に容易に陥るが初期のランク配置やパラメータを適切に与えることにより最適解により近い解を得ることが可能だと考えられる．また，ランク配置の求解アルゴリズムを並列に実行することも考えられる．

5. 終わりに

通信タイミングを考慮したランク配置最適化の提案を行い，その有効性を示すための評価実験を行った．評価実験では提案手法を適用することで通信時間が順配置に対して最大 45%，従来手法によるランク配置に対して最大 24%削減された．これにより，提案手法が有効であることが分かった．

今後の課題は，以下のものが挙がられる．まず，他の通信パターンでの追加評価実験を行う．これは，提案手法の適用範囲を通信パターンの面から調べるためである．次に大規模並列計算機での評価を行う．これは提案手法を大規模並列計算機に適用した際の実用性を調べるためである．現在はランク配置を全解探索で求めているため，提案手法を大規模並列計算機に適用することは困難である．このため，大規模並列計算機用のランク配置求解アルゴリズムを考える必要がある．また，大規模並列計算機での評価実験は，実機を用いて行いたいが，現状，多数の計算ノードを持ち，かつネットワークトポロジを自由に変更できる大規模並列計算機を利用することは困難であるため，シミュレータを用いて行うつもりである．

謝辞 本研究は,「ペタスケール・システムインターコネクト技術の開発」プロジェクト (文部科学省 「次世代 IT 基盤構築のための研究開発」の研究開発領域「将来のスーパー コンピューティングのための要素技術の研究開発」（平成 17〜19 年度）の一つ) によるものである．

参 考 文 献

1) F. Ercal, J. Ramanujan, P. Sadayappan, “Task allocation onto a hypercube by recursive Mincut Bipartitioning,” Journal of Parallel and Distributed Computing, v.10, n.1, pp.35-44, in 1990.
2) T. Hatazaki, “Rank Reordering Strategy for MPI Topology Creation Function,” PVM/MPI ’98, LNCS 1497, pp.188-195, in 1998.
3) Jesper Larsson Traff, “Implementing the MPI process topology mechanism,” in Conference on High Performance Networking and Computing, Proceedings of the 2002 ACM/IEEE conference on Supercomputing, pp.1-14, in 2002.
4) T. Agarwal, A.Sharma, L. V. Kale, “Topology-aware task mapping for reducing communication contention on large parallel machines,” Proceedings of IEEE International Parallel and Distributed Processing Symposium 2006, pp.1-10, in 2006.
5) Gyan Bhanot, Alan Gara, Philip Heidelberger, Eoin Lawless, James Sexton, Robert Walkup, “Optimizing task layout on the Blue Gene/L supercomputer.” IBM J. Res. & Dev. 49, No. 2/3, pp.489-500, in 2005.
6) “TOP500 Supercomputer Sites,” `http://www.top500.org/.`
7) “The ASCI Purple Benchmark,” `http://www.llnl.gov/asci/purple/benchmarks/limited/code_list.html`
8) “NAS Parallel Benchmark,” `http://www.nas.nasa.gov/Resources/Software/npb.html`
9) “MPICH Home Page,” `http://www-unix.mcs.anl.gov/mpi/mpich1/.`

(平成 19 年 1 月 22 日受付)
(平成 19 年 3 月 11 日採録)