直接網において複数の通信デバイスを有効に使用する隣接通信アルゴリズムの提案

森江 善之[1,2,a)]　南里 豪志[1,2,b)]

概要：近年，計算機の大規模化が進む中で，局所的な通信パターンを持つ隣接通信が注目を集めている．しかし， ネットワークアーキテクチャを考慮した隣接通信アルゴリズムの研究は進んでいない．そこで，本研究では，メッシュ/トーラスのような直接網で接続された複数の通信デバイスを持つ並列計算機を対象とした隣接通信アルゴリズムの提案を行った．提案した隣接通信アルゴリズムでは，隣接プロセス数と通信デバイス数の関係を考慮してメッセージをセグメントに分割し，同量のメッセージを全通信デバイスに割り付ける．これにより，全ての通信デバイスの通信帯域幅を使い切ることができ，通信性能を向上させる．そこで，提案した隣接通信アルゴリズムの有効性を示すため，RDMA インターフェースおよび MPI 関数による実装を行い，京コンピュータ互換の並列計算機である FX10 において性能評価実験を行った．FX10 は，通信デバイスを 4 基搭載し，仮想 3 次元トーラストポロジとしてアクセスが可能であるため，提案アルゴリズムによる性能向上が期待できる．性能評価実験では，MPI 関数による実装において既存の隣接通信アルゴリズムに対して 2 倍の性能向上を示した．また，同様に RDMA インターフェースによる実装において 25% の性能向上を示した．また，これらの評価結果やアルゴリズムについて考察を行った．

キーワード：隣接通信アルゴリズム，複数通信デバイス，直接網

A Neighboring Communication Algorithm Using Effective Multiple Communication Devices on Direct Connection Network

Yoshiyuki Morie[1,2,a)]　Takeshi Nanri[1,2,b)]

Abstract: Recently large scale parallel computers have been developed, neighboring communication that is the local communication pattern is remarkable. But neighboring communication algorithm is not developed with considering a network architecture. In this paper, neighboring communication algorithm for machine having multiple communication devices on mesh/torus is proposed. This algorithm allocates same size data into all communication devices for using their communication bandwidth fully. Therefore, it divides a message into segments in order to equally allocate them into all of them. To show effectiveness of the proposed algorithm, implementation of the proposed algorithm by both MPI function and RDMA interface is examined to compare existing one on FX10 that is family of K computer and equips 4 communication devices on virtual 3D torus. In evaluation experiments, the performance of the proposed algorithm that is made by MPI function is two times faster than the existing one. Similarly, the proposed algorithm that is made by RDMA interface improves 25% performance over existing one. It is also considered about an evaluation result of experiments and the proposed algorithm.

Keywords: Neighboring Communication Algorithm, Multiple Comminication Devices, Direct Connection Network

[1] 九州大学情報基盤研究開発センター
Research Institute for Information Technology, Kyushu University, Higashi-ku, Fukuoka, 812–8581, Japan
[2] 独立行政法人科学技術振興機構, CREST
Japan Science and Technology Agency, CREST, Chiyoda, Tokyo 102–0076, Japan
[a)] morie.yoshiyuki.404@m.kyushu-u.ac.jp
[b)] nanri@cc.kyushu-u.ac.jp

1. はじめに

近年，並列計算機は，飛躍的に大規模化した．その著しい計算ノード数の増加により，全対全の大域的な通信を行うことが困難に成りつつある．このため，局所的な通信を行う数値計算の注目度は高い．

このような局所的な通信を行う計算として分子動力学計算や流体計算などを代表とするステンシル計算が挙げられる．これらの計算では，そのシミュレーション対象を領域ごとに分割し，分割した領域の計算を各プロセスに割り当てる．このとき，隣接する領域間では，互いに必要なデータを定期的に交換するための通信が行われる．このように隣接領域を担当するプロセスのみと通信を行うため，局所的な通信が行われる．このような多対多の通信は隣接通信と呼ばれる．隣接通信は並列計算機が大規模化してもその通信コストが隣接プロセス数のみに依存するため，スケーラビリティが高い．このような隣接通信は重要性が増しており，それに伴い，事実上の標準通信ライブラリであるMPI–3[1] において隣接通信のインターフェイスが仕様化された．

しかし，隣接通信の実装については，ネットワークアーキテクチャを考慮した研究開発がまだ進んでいるとは言えない．先に述べたとおり並列計算機の大規模化が進み，そのネットワークトポロジとしては，メッシュ/トーラスが多く採用されている．そのような直接網の並列計算機における各計算ノードでは，定数の計算ノードに隣接する．これらの計算ノードに対する通信が同時発行可能となるよう複数の通信デバイスが搭載されることが多い．この複数の通信デバイスを有効に使用することとネットワークトポロジを考慮して通信衝突の発生を回避することは通信性能を向上させる上で重要となる．したがって，隣接通信アルゴリズムを実装する際には対象となる並列計算機のネットワークアーキテクチャを考慮する必要がある．

本稿では，複数の通信デバイスをもつ並列計算機において通信帯域幅を使い切る隣接通信アルゴリズムを提案する．また，提案した隣接通信アルゴリズムを MPI 関数および RDMA インターフェースを用いて実装し，それらを用いて性能評価実験を行う．提案するアルゴリズムは，隣接通信において隣接プロセス数と通信デバイス数が一致しない場合に注目する．このとき，全ての通信デイバスの通信帯域幅を使い切ることが出来ない場合がある．このような場合には，複数の通信デバイスへ適切にメッセージ割り付けを行うことが通信性能を向上させるために重要となる．これに対して提案する隣接通信アルゴリズムでは，隣接プロセスへのメッセージをセグメントに分割し，通信デバイスへ平等に割り付けることで全通信デバイスの通信帯域幅を使い切る．

本稿は以下のような構成となる．まず，第 2 節では，隣接通信の関連研究の紹介を行う．次に，第 3 節で直接網において複数の通信デバイスの通信帯域幅を有効に使用する隣接通信アルゴリズムの提案を行い，第 4 節で性能評価のための実験を行う．第 5 節で，提案した隣接通信アルゴリズムや評価実験の結果について考察を行い，最後にまとめと今後の課題を述べる．

2. 関連研究

まず，既存の MPI ライブラリ [2][3] では，MPI–3 の規格に基づいて隣接通信が実装されている．しかし，これらは，すべての隣接プロセスへの送受信を非同期に発行する通信アルゴリズムが実装されるのみとなっており，ネットワークアーキテクチャを意識したアルゴリズムの実装は存在しない．

また，Hoefler ら [4] は，隣接通信インターフェースの定義やインターフェースの実装をいくつか提案した．これらの実装では，プロセスの論理的なトポロジの次元数を考慮したアルゴリズムが提案された．しかし，通信デバイス数や物理的なネットワークトポロジを考慮したもではない．

一方，筆者ら [5][6] は，隣接通信の通信パターンを実行するプロセス群をファットツリーなどのネットワークトポロジを持つ並列計算機に通信衝突を回避するようにマッピングするプロセス配置最適化を提案し，通信性能が向上することを示した．しかし，適用対象がプロセスの論理トポロジとネットワークの物理トポロジが一致していない場合に限られており，本稿での適用対象とは異なる．また，複数の通信デバイスをもつ並列計算機を対象とした隣接通信の開発も行っている．ここでは，隣接プロセス数と通信デバイス数を考慮して通信帯域幅を使い切る隣接通信を実装した．しかし，特定の隣接プロセス数および通信デバイス数のみでの実装となっており，汎用的に使用することができない．

さらに，畑中ら [7] は，MPI の集団通信や永続通信の呼び出しによって与えられる複数の通信要求に対して，複数の通信デバイスとリンクを持つネットワーク上で，最適な RDMA コマンド列を生成するスケジューラの提案を行った．実際に京コンピュータ上で気象アプリケーションの隣接 4 方向および 8 方向の隣接通信の通信パターンを生成することを示した．しかし，本稿が対象としている各計算ノードの全通信デバイスの通信帯域幅が同一でかつ隣接プロセスへのメッセージサイズも同一という前提においては，メッセージの通信デバイスへの割り付け方を変更しても通信デバイスの通信帯域幅を使い切ることができない場合が存在する．

3. 直接網において複数の通信デバイスを有効に使用する隣接通信アルゴリズム

直接網のネットワークにおいて複数の通信デバイスを持

つ並列計算機において実行される隣接通信の高速化のため，隣接プロセス数と通信デバイス数を考慮した通信アルゴリズムの提案を行う．

3.1 提案隣接通信アルゴリズムの実行条件

本隣接通信アルゴリズムの実行の条件としては以下があげられる．

まず，対象となる隣接通信では，隣接プロセスにすべて同じサイズのメッセージを送付するものとする．また，プログラム実行時には，1 計算ノードに 1 プロセスがマッピングされており，さらに隣接プロセスが隣接計算ノードにマッピングされているものとする．隣接プロセスが隣接計算ノードにマッピングされていない場合は，他の通信との通信衝突が発生して所望の性能向上を得ることができない可能性があり，今回は対象としない．また，通信デバイスは，図 1 のようにクロスバスイッチに接続されており，各通信デバイスは，どの方向にも通信を発行できるものとする．最後に通信デバイス数が隣接プロセス数より少なく，通信デバイスとリンクの通信帯域幅は同じとする．これは，ハードウェアコストを考慮すると妥当な仮定であると考えている．

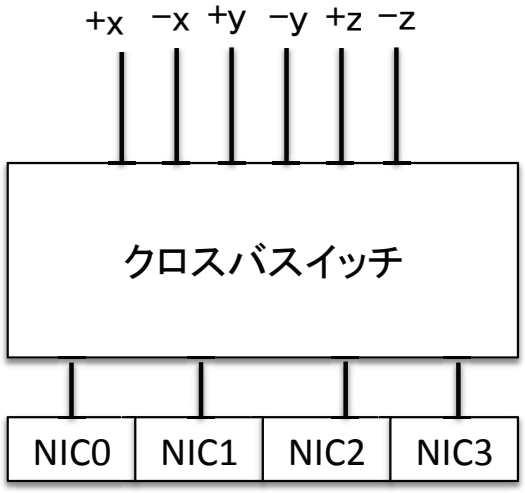

図 1 各通信デバイスと各リンクとの接続関係
Fig. 1 The relation of a connection between each link and each communication device.

3.2 既存実装

Open MPI や MVAPICH での隣接通信の既存実装について述べる．既存実装では，隣接プロセスとの間で非同期送受信を一度に発行し，それらの通信が終わるまで待つという素朴なアルゴリズムで実装されている．ここで，Algorithm 1 に既存の隣接通信アルゴリズムの擬似コードを示す．まず，疑似コード内の変数について述べる．$sendbuf, recvbuf$ は，それぞれ送信および受信バッファの先頭アドレスを示している．また，$inbr$ は，隣接プロセスへのインデックス番号を示す．次に，$snbrid$，$rnbrid$ は，$inbr$ をインデックスとした送信先および受信元プロセス ID が格納されている．また，$size$ は，各プロセスへの通信量，$n\_neighbors$ は，隣接プロセス数を示す．

Algorithm 1 既存の隣接通信アルゴリズム

```
for i = 0 to n_neighbors do
  isend (sendbuf, size, snbrid[inbr])
end for
for i = 0 to n_neighbors do
  irecv (recvbuf + inbr * size, size, rnbrid[inbr])
end for
wait_all
```

ここで，既存の隣接通信アルゴリズムの通信性能について述べる．例えば，図 2 に通信デバイスを 4 基搭載した並列計算機で隣接プロセス数を 6 個とする隣接通信を実行した場合を示す．このとき，一部通信デバイスが使用されていない状態にあることが分かる．このような状態は，すべての通信デバイスの通信帯域幅を有効に使用できているとはいえず，通信性能改善の余地がある．

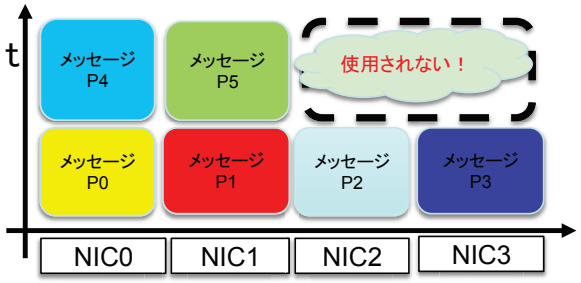

図 2 既存の隣接通信アルゴリズム
Fig. 2 The previous neighboring communication algorithm.

3.3 提案隣接通信アルゴリズム

前小節で示したように，既存の隣接通信アルゴリズムを用いた実装では，すべての通信デバイスの通信帯域幅を有効に使用できない．そこで，本稿では，直接網においてすべての通信デバイスの通信帯域幅を使い切る隣接通信アルゴリズムを提案し，隣接通信の通信性能を向上させることを目指す．

これを実現するためには，まず，すべての通信デバイスに同量のメッセージを割り当てる必要がある．このためには，隣接プロセス数と通信デバイス数を考慮する必要がある．ここで，隣接プロセス数と通信デバイス数をそれぞれ $N, C (N > C \cap C > 1)$ とし，1 プロセスへのメッセージサイズを M とする．すると，各通信デバイスにおいて $\frac{NM}{C}$ のメッセージを割り当てることですべての通信デバイスに同じ通信量を割り当てることができる．ここで，$C > 1$ と

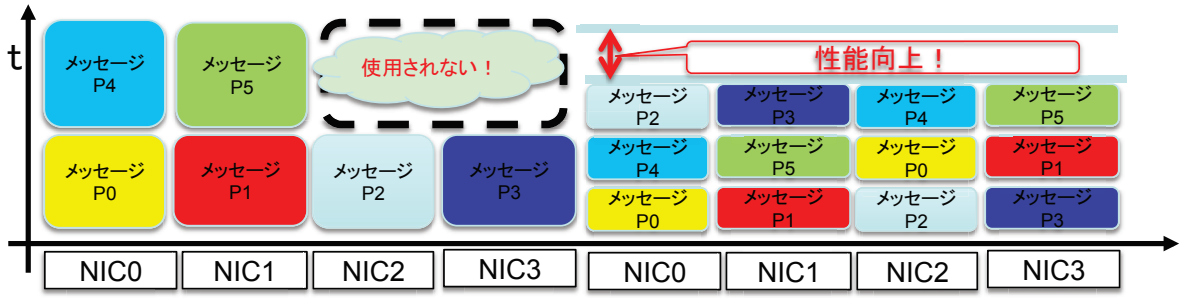

図 3　既存の隣接通信と提案する隣接通信を比較

Fig. 3　To Compare the between proposed neighboring communication and the previous one.

したのは，計算ノードが通信デバイスを 1 基しか搭載していない場合は，既存の隣接通信アルゴリズムで通信帯域幅を使い切ることができるからである．また，$N \leq C$ の条件下は，通信デバイスとリンクの通信帯域幅が等しいという条件により隣接プロセス数の通信デバイスを使用することで隣接プロセスへのリンクの通信帯域幅を使い切ることが出来る．このため，この条件下では，提案する隣接通信アルゴリズムでは通信性能を向上させることはできないので，適用対象外とした．

次に，すべての通信デバイスに同量のメッセージを割り当てるため，メッセージをセグメントに分割する．これは，メッセージをそのまま送付するだけでは隣接プロセスへのメッセージ数と通信デバイス数が一致せず，すべての通信デバイスに同量のメッセージを割り付けることができないからである．

ここで，k を N と C の最大公約数とし，$N = kN'$，$C = kC'$ とおく．このとき，メッセージを $\frac{M}{C'}$ の大きさのセグメントに分割し，N' ステップで送信すると各通信デバイスに同量のメッセージを割り付けることができる．このようにメッセージをセグメントに分割して通信デバイスに割り付けることで，図 3 のように全ステップにおいてすべての通信デバイスが同量のセグメントを常に発行している状態となり，実効通信帯域幅を向上させる．

しかし，このようにセグメントを分割することで通信ステップ数が増加する．このため，提案した隣接通信アルゴリズムは，メッセージサイズが小さい範囲では，通信毎に発生するネットワークレイテンシやソフトウェアオーバヘッドなどのコストが大きくなるため，通信性能が悪化することが考えられる．

また，セグメントの宛先は同一ステップにおいて同一となってはならない．これは，同一ステップにおいて，同一の宛先に通信を発行すると同一リンクを通過するため，通信の競合が発生するからである．このとき，通信デバイスとリンクの通信帯域幅が同じであるという条件から通信性能が悪化する．このため，提案アルゴリズムでは隣接プロセスに ID を設けて，その順序にしたがい各ステップにおいて通信デバイス数分昇順にセグメントを発行する．これにより同一ステップにおいて同一の宛先に通信を発行することを回避している．図 4 に示すように，6 隣接プロセス，4 通信デバイスの場合は 3 ステップで通信が実施される．このように隣接プロセス数が通信デバイス数より大きいという条件があるため，ステップ内で同一の宛先に通信が発行されることは無い．

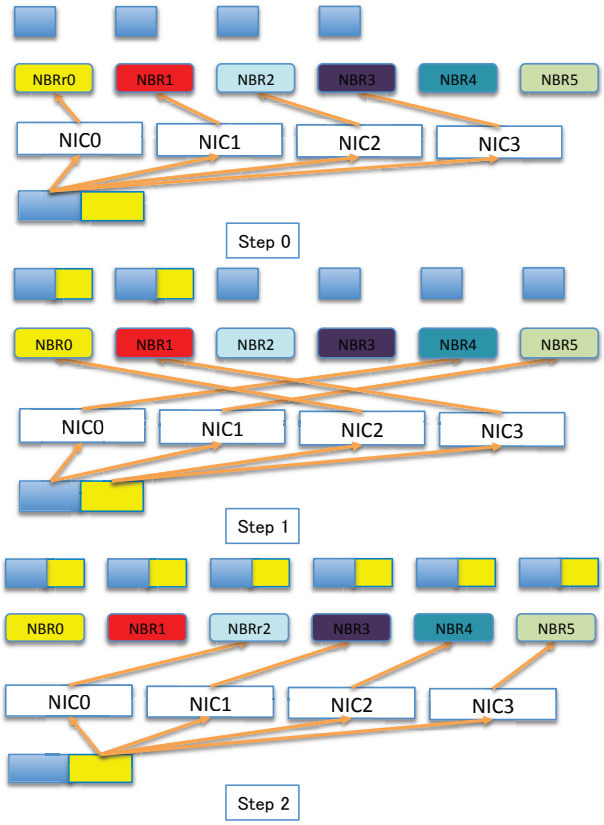

図 4　各セグメントの宛先

Fig. 4　The destination of each segment.

ここで，Algorithm 2 に非同期通信による提案隣接通信アルゴリズムを示す．Algorithm 2 では，MPI 関数の使用を想定したメッセージパッシングによるアルゴリズムを示している．このため，非同期通信関数には通信デバイ

ス ID を指定する引数はおいていない．また，Algorithm 2 における $iseg$ はセグメントのインデックスで seg_size はセグメントのサイズを示す．また，$STEPS$ は，提案隣接通信アルゴリズムの総ステップ数である．それ以外は，Algorithm 1 と同様の変数を用いる．ここでは，各ステップで通信デバイス数分の送受信を発行することを記述するのみとなっている．このため，メッセージの各通信デバイスへの割り付けは，通信ライブラリの実装に依存することとなる．

Algorithm 2 非同期通信による提案隣接通信アルゴリズム

```
Require: N > C ∩ C > 1
  for i = 0 to STEPS do
    for j = 0 to C do
      iseg = (j + i * C)/N
      inbr = (j + i * C) mod N
      isend (sendbuf + iseg * seg_size, seg_size, snbrid[inbr])
    end for
    for j = 0 to C do
      iseg = (j + i * C)/N
      inbr = (j + i * C) mod N
      irecv (recvbuf + inbr * size + iseg * seg_size, seg_size,
      rnbrid[inbr])
    end for
    wait_all
  end for
```

また，提案した隣接通信アルゴリズムの実装として，個別の通信遅延の削減のため，RDMA を直接使用する片側通信関数を用いる．ここで，Algorithm 3 に片側通信関数による提案隣接通信アルゴリズムを示す．Algorithm 3 における 変数 $ncomp$ は，通信が完了した回数を保存するものである．それ以外は，Algorithm 2 と同様の変数を使用している．

Algorithm 3 片側通信による提案隣接通信アルゴリズム

```
Require: N > C ∩ C > 1
  for i = 0 to STEPS do
    for j = 0 to C do
      iseg = (j + i * C)/N
      inbr = (j + i * C) mod N
      put (sendbuf + iseg * seg_size, recvbuf + inbr * size +
      iseg * seg_size, sige_size, snbrid[inbr], j, j)
    end for
    repeat
      for j = 0 to C do
        polling(j)
        if a complete of polling is success then
          increment the number of ncomp
        end if
      end for
    until ncomp == C
  end for
```

ここで，使用されている put 関数は片側通信の発行を行う関数である．この関数の引数はそれぞれ，第 1 引数にローカル側の送信バッファのアドレス，第 2 引数にリモート側の受信バッファのアドレス，第 3 引数にメッセージサイズ，第 4 引数に隣接プロセスの ID，第 5 引数にローカル側の通信デバイス ID，第 6 引数にリモート側の通信デイバス ID を与える．本アルゴリズムでは，片側通信を発行する際，通信デバイス ID を指定するが，ローカル側とリモート側の通信デバイス ID は，同じ値を与える．このように与えると図 5 のように通信デバイスが割り当てられ，異なる通信が同一の通信デバイスにおいて競合することがない．また，polling 関数は引数に通信デバイス ID をとり，ID が指す通信デバイスで発行された通信の完了を確認することが出来る．

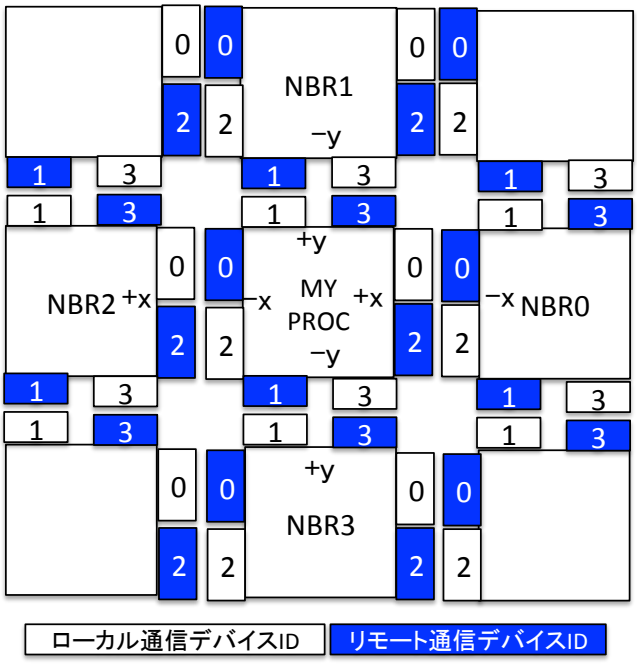

図 5　ローカル通信デバイス ID とリモート通信デバイス ID の関係
Fig. 5　The relation of between the local and the remote communication device ID.

3.4　通信性能モデル式

提案した隣接通信アルゴリズムの効果を示すため，通信性能モデル式を示す．そこで，まず，比較対象とする既存の隣接通信アルゴリズムの通信性能モデル式を示す．このとき，ネットワークレイテンシと通信発行時のソフトウェアオーバヘッドを L，通信デバイスおよび各リンクの通信帯域幅を B とした．

まず，既存の隣接通信アルゴリズムの通信性能モデル式を示す．既存の隣接通信アルゴリズムはメッセージを分割せずに通信を一度に発行するので 1 ステップで通信デバイス数のメッセージを処理できる．隣接プロセス数分のメッセージを処理する必要があるため，隣接通信の終了までは，

$\lceil \frac{N}{C} \rceil$ ステップで実行される．また，1 ステップにおいて発行されるメッセージのサイズは，セグメント分割されないため M となる．これらにより，既存の隣接通信アルゴリズムの通信性能モデル式は式 1 となる．

$$D_{prev} = L\lceil \frac{N}{C} \rceil + \lceil \frac{N}{C} \rceil \frac{M}{B} \quad (1)$$

次に，提案した隣接通信アルゴリズムの通信性能モデル式を示す．提案隣接通信アルゴリズムは前小節で示した通り，N' ステップで実行される．また，1 ステップにおいて，$\frac{M}{C'}$ のサイズに分割したセグメントを発行する．これらより，提案した隣接通信アルゴリズムの通信性能モデル式は式 2 となる．

$$D_{proposed} = LN' + \frac{N'}{C'} \frac{M}{B} \quad (2)$$

これらの通信性能モデル式から，通信性能向上比を求める．本アルゴリズムは，大メッセージサイズにおいての利用を想定している．このため，通信の発行毎に発生するネットワークレイテンシやソフトウェアオーバヘッドの項の値は桁が違うため，無視できるものとする．したがって，通信性能向上比は，通信帯域幅の項の比を取ることで得る．これらより，通信性能向上比は式 3 となる．

$$Ratio_{perf} = \frac{C'}{N'} \lceil \frac{N}{C} \rceil \quad (3)$$

ここで，通信性能モデル式を用いて，6 隣接プロセス，4 通信デバイスでの各隣接通信の性能の比較を行う．このとき，ネットワークレイテンシおよび通信毎に発生するソフトウェアオーバヘッドの合計を 1us とする．また，通信デバイスおよびリンクの通信帯域幅を 5GB/sec とする．ここで，6 隣接プロセス，4 通信デバイスの隣接通信の通信性能モデル式は，式 1 および式 2 から，それぞれ式 4，式 5 のようになる．

$$D'_{prev} = 2L + 2\frac{M}{B} \quad (4)$$

$$D'_{proposed} = 3L + \frac{3}{2}\frac{M}{B} \quad (5)$$

また，通信性能比は，式 3 から 1.33 となる．

ここで，図 6 に式 4，式 5 を用いてそれぞれ各隣接通信の予測通信帯域幅を示す．グラフの縦軸は予測通信帯域幅，横軸はメッセージサイズを示す．まず，メッセージサイズが大きいとき，提案した隣接通信アルゴリズムは，既存の隣接通信アルゴリズムの性能を上回っている．例えば，8MB で提案隣接通信アルゴリズムは 20GB/sec，既存の隣接通信アルゴリズムは，15GB/sec となる．また，メッセージサイズが小さい間は，既存の隣接通信アルゴリズムの方が通信性能が高いが，16KB において提案した隣接通信アルゴリズムが既存の隣接通信アルゴリズムの通信性能を上回った．通信性能比に関してはメッセージサイズが 8MB のときに 1.33 となる．これは，式 3 から求めたものと同様となる．これより，メッセージサイズが大きいところでは，式 3 による通信性能比の見積もりを行う．

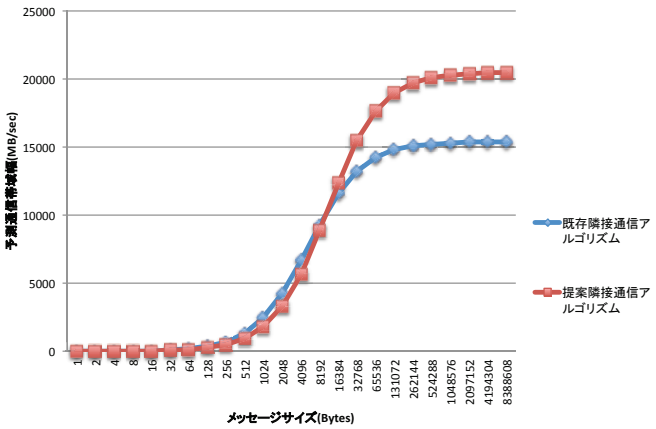

図 6　各隣接通信の予測通信帯域幅

Fig. 6　The estimation of communication bandwidth on each neighboring communication algorithm.

4. 評価実験

本節では，提案した隣接通信アルゴリズムの有効性を示すため，提案した隣接通信アルゴリズムと既存の隣接通信アルゴリズムをそれぞれ実装し，それらの通信性能を比較する実験を行う．今回は，京コンピュータの互換機である Fujitsu PRIMEHPC FX10 を対象とし，6 隣接プロセス，4 通信デバイスにおける各隣接通信の性能評価を行う．

4.1 実験概要

今回は，2 種類の隣接通信アルゴリズムを MPI 関数および富士通拡張 RDMA インターフェース（以下 RDMA インターフェース）を用いてそれぞれ実装した．これら 4 通りの隣接通信をそれぞれ 1000 回実行するプログラムを作成する．これを FX10 の計算ノード 96 個 を用いてそれぞれ実行する．実行時には，割り当てる計算ノードの形状を 4x3x8 として隣接プロセスを隣接計算ノードに割り付けておく．そこで，それぞれの隣接通信の平均実行時間から実効通信帯域幅を得る．

4.2 実験環境

実験環境として東京大学情報基盤センター [8] の Fujitsu PRIMEHPC FX10 を用いた．以下に仕様を述べる．CPU は Fujitsu SPARC64^{TM} IXfx 1.8484GHz (16 コア)，メモリは 32GB，総計算ノード数は 4800 個である．また，OS やコンパイラ，MPI ライブラリは富士通株式会社独自開発のものである．また，ネットワークは，Tofu インターコネクト [9] を用いる．ネットワークトポロジは 6 次元メッシュ/トーラスでユーザは仮想 3 次元トーラスとしてアクセスする．また，各計算ノードでは通信デバイスを 4 基搭載する．

4.3 実験結果

図 7 に各隣接通信の実効通信帯域幅を示す．このグラフの縦軸は実効通信帯域幅，横軸はメッセージサイズをそれぞれ示す．図 7 より，提案隣接通信アルゴリズムが，8MB において MPI 関数による実装では 2 倍，RDMA インターフェースによる実装においては 25% 性能が向上した．また，MPI 関数による実装では，提案隣接通信の通信性能が既存隣接通信の通信性能を 512KB で上回った．一方，RDMA インターフェースによる実装では，128KB で性能が上回った．小メッセージサイズでは，既存隣接通信が高速となり，大メッセージサイズでは，想定通り提案隣接通信が高速となった．この実験結果は，通信性能の入れ替わる点は異なるが通信性能モデル式と同じ傾向である．

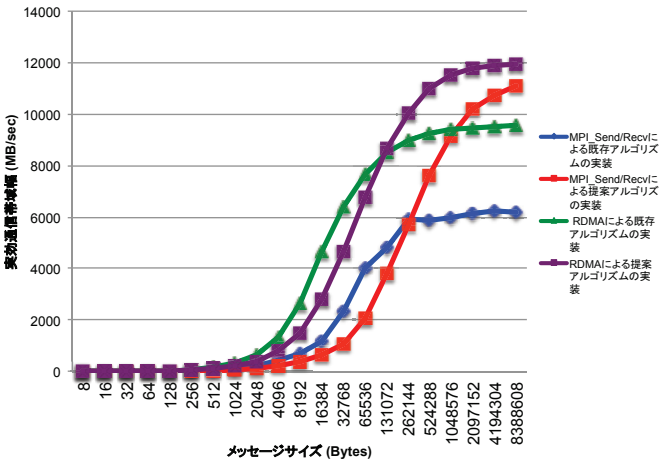

図 7 各隣接通信の実効通信帯域幅

Fig. 7 The effective communication andwidth of each neighboring communication.

5. 考察

実効通信帯域幅は MPI 関数で実装した場合も RDMA インターフェースで実装した場合もともに最大で 約 12GB/sec となった．FX10 の Tofu インターコネクトの通信帯域幅は 5GB/sec であるので，通信性能モデル式で求めたように 20GB/sec となることが期待されたが，これに比べ性能が低くなっている．原因の 1 つとしては，CPU 側のインターフェースにボトルネックがあるためで，これにより通信帯域幅が 15GB/sec 程度に律速される [10]．しかし，これを考慮してもさらに通信性能が低い．このため，まだ他に性能低下要因があると考えられる．この原因の調査は今後の課題である．

次に RDMA インターフェースによる実装と MPI 関数による実装との通信性能差について述べる．ここで，図 8 に各隣接通信の 64KB までの実行時間を示す．このグラフの縦軸は実行時間，横軸はメッセージサイズをそれぞれ示す．図 8 のように小メッセージサイズでは RDMA インターフェースによる隣接通信の方が高速となっている．これは，以下の理由が考えられる．

まず，RDMA インターフェースによる実装では，通信バッファに直接転送が行われ，MPI 関数のようにライブラリが管理する通信バッファからユーザ領域へのコピーは行われない．また，レジストレーションしたメモリ領域を解放せず，繰り返し利用している．メモリレジストレーションのコストは小メッセージサイズの通信を考えるときは比較的大きいもので，これを削減することは全体の性能に寄与する．これらのことから RDMA インターフェースによる実装の方が MPI 関数による実装より全体的に通信性能が向上したものと考えられる．

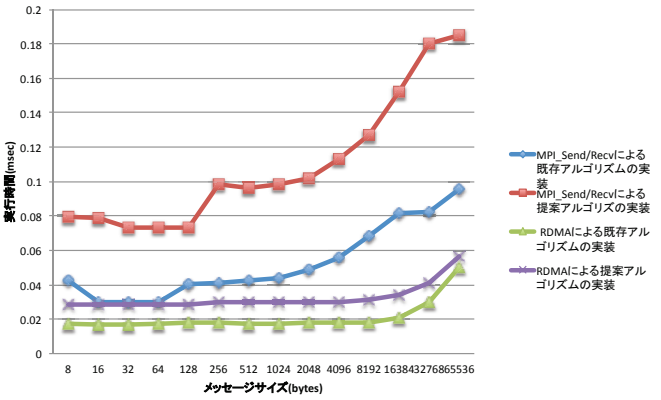

図 8 各隣接通信の実行時間 ($\leq 64KB$)

Fig. 8 The elapsed time of each neighboring communication. ($\leq 64KB$))

ここで，通信性能比の最大値と通信性能が最大となるプロセス数および通信デバイス数の関係について述べる．通信性能比は式 3 で与えられ，この式を最大化することで得られる．式 3 の $\lceil \frac{N}{C} \rceil$ の少数点以下の値を x とおくと，

$$\frac{C'}{N'}\lceil \frac{N}{C} \rceil = \frac{C'}{N'}\lceil \frac{N}{C} - x + x \rceil \tag{6}$$

となる．x を引いた項は常に整数となるので，式 6 は，

$$\frac{C'}{N'}(\frac{N}{C} - x + \lceil x \rceil) \tag{7}$$

となる．この式 7 を展開すると

$$1 - \frac{C'}{N'}x + \frac{C'}{N'}\lceil x \rceil \tag{8}$$

となる．条件より，$N' > C'$ であるので，$0 \leq \frac{C'}{N'} \leq 1$ となる．式 8 の第 2 項目を最小，第 3 項目を最大とすれば，式が最大値をとる．すなわち，$x \to 0$ の時，最大となり，その値は 2 となる．x が 0 に近づくのは，N と C の値が近づくときである．つまり，提案した隣接通信アルゴリズムの性能向上が最大となるのは，隣接プロセス数が通信デバイス数に限りなく近づくが一致しないときで，通信性能比は 最大でほぼ 2 倍となる．

6. おわりに

本稿では，ネットワークアーキテクチャを考慮した N 隣接プロセス，C 通信デバイスにおける隣接通信アルゴリズムの提案を行った．また，既存実装と比較する性能評価実験を行い，提案した実装は既存の実装に対して MPI 関数による実装において最大で 2 倍，RDMA インターフェースによる実装において最大で 25% の通信性能向上を実現することを示した．また，この実験結果に対する考察を行った．

今後の課題としては，以下が挙げられる．まず，今回実験した 6 隣接プロセス，4 通信デバイス以外の条件で性能評価実験を行う．また，大メッセージサイズにおける提案隣接通信の実効通信帯域幅の低下原因を調査する．また，提案隣接通信と既存隣接通信の通信性能が入れ替わるメッセージサイズをより小さくするためチューニングを行う予定である．

謝辞　本研究は，科学技術振興機構戦略的創造研究推進事業 (CREST)「ポストペタスケール高性能計算に資するシステムソフトウェア技術の創出」研究領域，「省メモリ技術と動的最適化技術によるスケーラブル通信ライブラリの開発」 の一部として実施された．また，主に東京大学情報基盤センターの Fujitsu PRIMEHPC FX10 System (Oakleaf-FX) を利用した．ここに記して謝意を表します．

参考文献

[1] Message Passing Interface Forum, :A Message Passing Interface Standard, Version 3.0, Technical report, 2012.
[2] Open MPI: Open Source High Performance Computing, 入手先 ⟨http://www.open-mpi.org/⟩.
[3] MVAPICH: MPI over InfiniBand, 10GigE/iWARP and RoCE, 入手先 ⟨http://mvapich.cse.ohio-state.edu/⟩.
[4] T. Hoefler, and J. L. Traff, :Sparse collective operations for MPI, in Proceedings of the 23rd IEEE International Parallel and Distributed Processing Symposium (IPDPS), HIPS Workshop, 2009.
[5] Y. Morie, and T. Nanri, :Task Allocation Optimization for Neighboring Communication on Fat–Tree, in Proceedings of The Fifth International Symposium on Advances of High Performance Computing and Networking, 2012.
[6] Y. Morie, and T. Nanri, :A neighbor communication algorithm with making an effective use of NICs on multidimensional-mesh/torus, International Conference on Simulation Technology (JSST2013), Sep. 2013.
[7] 畑中 正行，堀 敦司，石川 裕，:RDMA スケジューリングによる MPI 通信の高速化, 情報処理学会研究報告, 2013–HPC–140(17), pp.1–6, 2013.
[8] 東京大学情報基盤センタースーパーコンピューティング部門, 入手先 ⟨http://www.cc.u-tokyo.ac.jp/system/fx10/⟩
[9] Y. Ajima, Y. Takagi, T. Inoue, S. Hiramoto and T. Shimizu, :The tofu interconnect, in Proceedings of the 19th IEEE Annual Symposium High Performanace Interconnects, pp.87–94, 2011.
[10] N. Shida, S. Sumimoto, and A. Uno, :MPI Library and Low-Level Communication on the K computer, FUJITSU Scientific &Technical Journal, pp.324-330, 2012.