グリッド環境におけるマルチレーンを用いたMPIコレクティブ通信アルゴリズム

千葉立寛[†] 遠藤敏夫[†] 松岡聡[†,††]

グリッド環境上における MPI コレクティブ通信の性能は，ネットワークトポロジに強く依存しており，これまでにも最適なネットワークトポロジを構築してコレクティブ通信を高速化させるための様々な手法が数多く提案されてきた．また，近年のクラスタシステムでは，各ノードが複数の NIC を備えていることが多い．しかしながら，これまでに提案されている手法は，各ノードの送受信が実行できるポートを 1 つと仮定してトポロジを構築する手法がほとんどである．そこで我々は，各ノードにある 2 枚の NIC のバンド幅を最大限利用するマルチレーンブロードキャストツリー構築アルゴリズムを提案する．このアルゴリズムでは，ブロードキャストするメッセージを 2 つに分割し，2 枚の NIC を用いて 2 つの独立したバイナリツリーを構築して，それに沿って分割したメッセージのパイプライン転送を行う．また，提案アルゴリズムは，クラスタ，グリッド両方のシステムで効果的に実行でき，NIC を 1 枚だけ備えるノードに対しても複数のソケットを用意することで動作可能である．本稿では，ブロードキャスト通信に対してシミュレータ環境上で実験，評価を行い，従来手法よりも性能が向上したことを確認した．

MPI Collective Operations Algorithm by Using Multi-lane for Grid Environment

Tatsuhiro Chiba,[†] Toshio Endo[†] and Satoshi Matsuoka[†,††]

The performance of MPI collective operations, such as broadcast and reduction, is heavily affected by network topologies, especially in grid environments. Many techniques to construct efficient broadcast trees have been proposed for grids. On the other hand, recent high performance computing nodes are often equipped with multi-lane network interface cards (NICs), most previous collective communication methods fail to harness effectively. Our new broadcast algorithm for grid environments harnesses almost all downward and upward bandwidths of multi-lane NICs; a message to be broadcast is split into two pieces, which are broadcast along two independent binary trees in a pipelined fashion, and swapped between both trees. The salient feature of our algorithm is generality; it works effectively on both large clusters and grid environments. It can be also applied to nodes with a single NIC, by making multiple sockets share the NIC. Experimentations on a emulated network environment show that we achieve higher performance than traditional methods, regardless of network topologies or the message sizes.

1. はじめに

近年のグリッド技術の発展にともない，これまでは単独のスーパコンピュータやクラスタで行われてきた大規模な科学技術計算を，グリッドで高性能に実行することが現実的になりつつある．また，マルチフィジクスシミュレーションのために複数アプリケーションをグリッド上で連携させる試みも多くなされている．このようなアプリケーションはメッセージパッシングインタフェースである MPI[1] を用いて記述されるのが一般的であり，GridMPI[2] や MPICH-G2[3] などのグリッド向けの MPI 実装も提案されている．

MPI アプリケーションの性能に大きく影響する事項の 1 つはコレクティブ通信である．グリッド向け MPI においてもコレクティブ通信の性能向上は重要な位置づけがなされており，MagPIe[4] などの様々なアルゴリズムが提案されてきた．ただし既存アルゴリズムの多くは，サイト間を結ぶネットワークのバンド幅はサイト内ネットワークよりも低いことを前提に設計されている．また，各ノードが複数のネットワークインタ

† 東京工業大学
Tokyo Institute of Technology
†† 国立情報学研究所
National Institute of Infomatics

フェースカード（NIC）を持つ場合でもそれらを特に考慮していない．

しかしながら，TeraGrid [5] や SuperSINET などのインフラの整備にともない，サイト間ネットワーク上でユーザが数 Gbps の性能を利用することが現実的になってきている．そのためサイト間ネットワークが各ノードのバンド幅以上の性能を持つ場合が増加してきている．また，最近のクラスタノードには NIC が複数搭載されていることが多く，既存のアルゴリズムでは，そのようなネットワークの性能を十分に引き出すことが困難である．

そこで我々では，上記のようなネットワーク環境を有効に利用する MPI コレクティブ通信アルゴリズムの研究を行っている．本稿ではコレクティブ通信の 1 つであるブロードキャストを行う MPI_Bcast（以下 Bcast）通信についてのアルゴリズムを提案する．アルゴリズムの特徴は，複数の木構造からなるマルチレーンツリーと呼ばれるトポロジを用いることと，アルゴリズム全体でストールのないパイプライン転送を行うことである．マルチレーンツリーは複数の NIC のバンド幅を効率的に利用することができ，かつサイト間の通信量を削減するように設計されている．シミュレータ上にグリッド環境を構築し，提案アルゴリズムと従来アルゴリズムの性能の比較を行う．その評価により，単一サイトでもグリッド環境でも提案手法が高性能であることを示す．

2. ネットワークモデル

本章では，今回想定するネットワーク環境について述べ，また単純なアルゴリズムの説明を通じて本稿で注目する課題点について述べる．

2.1 ネットワーク環境

MPI アプリケーションを実行するネットワーク環境として，複数のクラスタが高バンド幅の WAN により接続されていることを想定する．全ノードは，それぞれ 2 枚の均一性能の NIC を持ち，同時に 2 台のノードと独立に通信を行えるものとする．これらの NIC では，それぞれが全二重通信を行えるものとする．

クラスタ内のスイッチは十分な性能を持ち，クラスタ内の全ノードが同時に通信を行っても性能低下は起こらないものとする（クラスタ内でスイッチが階層構造をなす場合にこの仮定があてはまらないことがあるが，各エッジスイッチに直接接続されるノード群をクラスタと見なして本稿のアルゴリズムを適用することができる）．また，各ノードのバンド幅を NIC1 枚あたり $b/2$，合計 b としたとき，任意のクラスタ間の WAN バンド幅が b 以上であるときに，本稿のアルゴリズムは効率良く動作する．この想定は近年の WAN 性能の向上により現実的になりつつある．

なお，各ノードは 2 枚の NIC を 1 本のリンクにアグリゲートして用いることもできるとする．本稿の議論では簡単のため，アグリゲーション処理や後述のパイプライン処理にともなうコストは考慮しない．

本稿のアルゴリズムは，各ノードが 1 枚のみ NIC を持つ環境であっても，ノードの各ペア間で複数ソケットを張ることによりそのまま適用できる．そのようなアプローチは peer-to-peer multicast の一手法である SplitStream [6] などでも採用されている．

2.2 単純なアルゴリズムと課題

最も単純な Bcast アルゴリズムの 1 つは，single chain によるものである．この手法では，ノード 0 が他のすべてのノード（$1 \cdots p-1$）にサイズ M のデータを送信する場合に，ノード 0 がノード 1 へ送信し，その後ノード 1 がノード 2 へ送信 $\cdots$ と順々に通信を行う．この手法の通信コストは，通信遅延 l，バンド幅 b（2 枚の NIC をアグリゲートすることを仮定）としたときに $(p-1)\cdot(l+M/b)$ であり，後述するこれまでに提案されてきた Bcast アルゴリズムと比べてもその性能は悪く，コストはノード数 p に比例して増大する．しかし，この手法は通信のパイプライン化により改善できることはよく知られている．つまり，各ノードはデータを一部でも受け取れば，すぐにそれを次のノードへ送信するようにすることができる．このときのコストは，パイプライン化にともなうオーバヘッドを無視すれば，$(p-1)\cdot l + M/b$ と改善される．それでも依然 p に比例する項があるため，ノード数 p が大きい場合には問題である．

コストが p に比例しない手法として binary tree（二分木）を用いるものが考えられる．各ノードは 2 つの子ノードへ，NIC を 1 枚ずつ用いてデータを送信することができる．NIC あたりのバンド幅を $b/2$ とすると，コストは $\log p \cdot (l + 2M/b)$ となる．この手法は容易にパイプライン化でき，そのときのコストは $\log p \cdot l + 2M/b$ である．しかしながら，この手法は M が小さいときはよいが，M が十分大きいときは single chain と比べて 2 倍のコストがかかる．この理由は，各ノードの持つバンド幅を十分に活用できていないためである．各ノードは受信のために 1 枚の NIC しか使っておらず，また，二分木の葉ノードは送信をまったく行っていない．一方 single chain では，最後のノードを除き，各ノードは NIC を 2 枚とも送受信に活用している．

次章で述べる提案アルゴリズムは，両者の欠点を解決し，さらに前章で述べたようなグリッド環境で効率良く動作することを狙いとする．そのために，以下のような特徴を持っている．

- 二分木に基づくことにより，コストはノード数 p ではなく，$\log p$ に比例する項を持つ．
- 各ノードのバンド幅をほぼすべて活用し，さらにストールのないパイプライン化が可能である．これにより M/b にかかる係数は 1 となり，M が大きいときに single chain と同等の低コストである．
- クラスタをまたぐ通信量は小さく抑えられており，グリッド環境でも効率的に動作する．

3. 提案アルゴリズム

本章では，本稿で提案するマルチレーンツリートポロジによる Bcast アルゴリズムを紹介する．まずサイト内のアルゴリズムを述べ，次に複数サイト向けにアルゴリズムを拡張する．その後，アルゴリズムの特徴を述べる．

3.1 サイト内マルチレーンアルゴリズム

図 1 にサイト内のアルゴリズムが作成するマルチレーントポロジの構造を示す．アルゴリズムの説明をするにあたって，簡単のため次のような前提条件を設ける．n を整数としたとき，サイト内には $4n-1$ 台のノードがあり，それぞれ 2 枚の NIC を持つとする．以下では NIC とリンクが 1 対 1 対応するとして議論する．

ノードのうち Bcast の基点であるものをノード 0 と呼ぶ．残りのノード群を $2n-1$ 台ずつに二分し，図 1 のように $Tree_A$ と $Tree_B$ の 2 つの同サイズの binary tree を作成する．図は leaf の高さがそろっている木を示すが，そうでなくともよい．$Tree_A$ に含まれるノードに $A_1, A_2 ... A_{2n-1}$ という名前をつけ，A_i の子ノードは A_{2i}, A_{2i+1} であるとする．このとき，

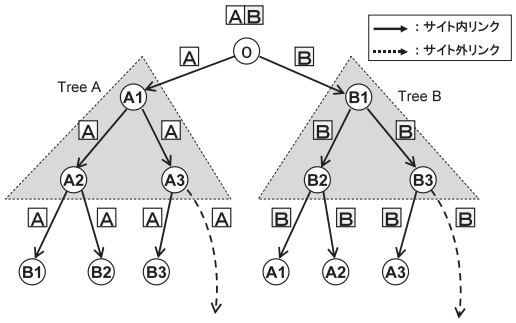

図 1 サイト内のマルチレーンツリー
Fig. 1 Multi-lane tree in a single site.

binary tree の leaf ノードは $A_n ... A_{2n-1}$ の n 個となる．$Tree_B$ についても同様に構築する．

このトポロジを用いた Bcast 通信の概要は以下のとおりである．

(1) ノード 0 は Bcast するメッセージ M を M_A，M_B の 2 つに分割する．そして M_A を $Tree_A$ の根ノードである A_1 へ，M_B を B_1 へ，リンクを 1 本ずつ用いて送信する．

(2) 各ノードは，受け取ったメッセージをそれぞれの木に沿って 2 つの子ノードへパイプライン転送する．送信の際には，各子ノードのためにリンクを 1 本ずつ用いる．

(3) (2) の段階では，leaf ノード群は 2 本の送信リンクを，全ノードが 1 本の受信リンクを残しているので，それらをもう一方の木への送信のために用いる．具体的には，$Tree_A$ 中の leaf ノードである A_{n+i}（$0 \le i < n$）は，2 本の送信リンクを用いて B_{2i+1}，B_{2i+2} へ送信を行う（図では簡単のために，$Tree_A$ の leaf ノードの下に $Tree_B$ のノードを記述している）．ここで，$Tree_A$ の leaf ノード数は n，$Tree_B$ のノード数は $2n-1$ なので，不足なく木をまたがった転送が可能である．$Tree_B$ の leaf から $Tree_A$ への送信も同様に行う．また，(2) と (3) にわたって，ストールのないパイプライン転送が可能である．

(4) 全ノードが M_A と M_B の両方を受け取り，それを結合して Bcast を完了する．

3.2 サイト間マルチレーンアルゴリズム

前述のアルゴリズムをノードが複数のサイトにまたがる場合に拡張する．Bcast の基点ノードを含むサイトをサイト 1 とし，それ以外をサイト 2，3... とする．各サイトが含むノード数は異なってよい．複数サイトにおけるマルチレーントポロジの構造は，**図 2** に示すようにサイトを chain 状につないだものとなる．そして各サイト内でそれぞれ 2 つの binary tree $Tree_A$ と $Tree_B$ を構築する．

各サイトでサイト内アルゴリズムを動作させ，サイト間のメッセージ転送について以下のように行う．サイト c の $Tree_A$ と $Tree_B$ の leaf ノードの 1 つである A_{2n-1}，B_{2n-1} は，サイト $c+1$ の $Tree_A$ と $Tree_B$ の根ノードにそれぞれ M_A と M_B を送信する．この転送もまた，各リンクが同等の転送速度である限り，ストールすることなくパイプライン転送が可能である．結局，本アルゴリズムでは全サイトにまたがったパイプライン転送を行うことになる．

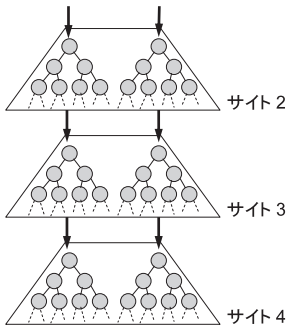

図 2 サイト間のマルチレーンツリー
Fig. 2 Multi-lane tree in multiple sites.

なお，本稿では WAN をまたがった 1 ノード対 1 ノードの転送が，十分なバンド幅で行えるときに効率良く動作する．これは 1 本の TCP 接続を用いた場合には非常に困難である．実際には，ノード間で複数の TCP 接続を張ったり，Scalable TCP [7] などの WAN 向けの通信ソフトウェアを用いたりすることが考えられる．それらの技法を用いた実環境での本アルゴリズムの評価は，今後の課題の 1 つである．

3.3 アルゴリズムの特徴

このアルゴリズムの特徴は以下のとおりとなっている．

- Bcast するメッセージを 2 つに分割.
- それぞれの分割したメッセージを転送するための独立な binary tree を 2 枚の NIC を用いて構築.
- それぞれのメッセージを独立な Tree に沿ってパイプライン転送.

すべてのノードが，必ず 2 本のリンクを用いて送受信を行い，それ以上のリンクを用意する必要がない．また，2 本のリンクで作られる Tree は独立である．そのため，2 枚ある NIC と 2 本のリンクが対応し，NIC のバンド幅をほぼ使いきることができる．そして，リンク張替えや受信待ちのストールが発生せずにパイプライン転送できるので，非常に効率良く Bcast メッセージの転送をすることが可能となる．

4. 実装に関する議論

本章では，提案手法を実環境に構築する際に問題となる点を議論する．提案アルゴリズムを MPICH-p4 などの下層で TCP を利用している MPI に組み込むことを想定して，サイト間通信やパイプライン転送処理の TCP を用いた実装について議論する．

4.1 サイト間通信の実装と予備実験

3.2 節で述べたとおり，本稿のアルゴリズムではサイト間通信において十分なバンド幅を利用できるときに効率良く動作する．この仮定の現実性を探るため，エミュレートによって仮想的な WAN 環境を構築し，以下のような予備実験を行った．ギガビットイーサネットで接続された 2 台のノード間にネットワークエミュレータ GtrcNET-1 [8] を接続した．そして様々な遅延を発生させ，ノード間で複数の TCP コネクションを張り，同時に通信を行った場合の合計バンド幅を測定した．

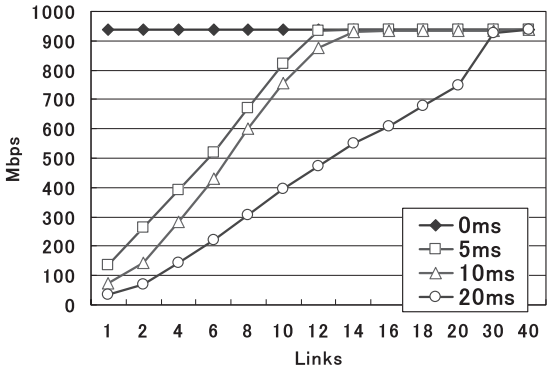

図 3 複数の TCP コネクションを用いたときの合計バンド幅（遅延：0〜20 ms）
Fig. 3 Total bandwidth by using multiple TCP links for delay of 0 to 20 ms.

0 ms から 20 ms の遅延がある状態でコネクション数を変えたときの合計バンド幅を**図 3** に示す．1 本のコネクションのみ用いたときのバンド幅は，遅延が大きくなるにつれて性能が低下している．たとえば 20 ms の遅延がある場合，その性能は 40 Mbps 程度であり，バンド幅の 4%程度の性能しか引き出すことができない．しかしながら，複数のコネクションを張れば，WAN をまたぐノード間のトータルバンド幅の性能は線形に向上していくことが確認できる．そして，コネクション数を十分に増やすことで，理論値の 90%以上の性能のバンド幅を WAN をまたぐノード間で引き出すことが可能となる．この結果から，WAN をまたぐときのバンド幅に対する我々の仮定は，現実的であることが確認できる．

4.2 パイプライン転送の実装

サイト内でのパイプライン

パイプライン転送を行うよく知られた手法として，メッセージを固定サイズ（たとえば 4 KB）のチャンクに分割して順次転送する手法があげられる．これはファイル転送ツールである Dolly+ [9] などでも採用されており，本稿でもこれを基本方針として採用する．受信と送信を並行に行うためにスレッドもしくは select/poll システムコールを用いることが考えられるが，簡単のためスレッドの場合を説明する．本稿のア

ルゴリズムでは2枚のNICを用いてそれぞれ送受信を行うので，各NICにつき送信スレッドと受信スレッドを起動する．ノードあたりの通信スレッドは合計で4つとなる．二分されたメッセージ M_A，M_B は，さらにチャンク単位に分割されて転送される．受信スレッドはチャンクの受信を行ったらそれをメモリに書き込み，すぐに次の受信を開始する．送信スレッドはメモリ上のチャンクに気がついたらそれをNICへ送信する．二分木の内部ノードでは，双方の送信スレッドが同一のチャンクを送ることになる．

サイト間でのパイプライン

サイト間のパイプライン転送には，4.1節で述べたとおり複数のTCPコネクションを用いた転送を行う．サイト間においてチャンク単位でメッセージを転送するために以下の処理を行う．他サイトに送信を行うノードは，受信完了したチャンクを順々にラウンドロビン方式で，用意しておいた複数のコネクションに割り当てていく．他サイトから受信を行うノードは，複数のコネクションからチャンク群を受け取り，正しい順序に戻して木構造上の子ノードへ順次送信する（out-of-orderに送信する手法も今後検討予定である）．このとき，各々のコネクションに対してスレッドを1つずつ割り当てたとすると，サイト間コネクションが多いときにコンテキストスイッチのコストが増大すると考えられる．たとえば，4.1節で述べたように，ノード間の遅延が20 msある場合，十分なバンド幅を得るためには約30本のコネクションが必要になる．このような場合にはスレッドではなくselectまたはpollシステムコールの利用が望ましいと考えられる．

以上のようにTCPを用いたパイプライン転送を実現することができるが，WAN上のリンクの性能を十分に引き出すためには，TCP/IPの特性をふまえ様々なパラメータの調整をしていく必要がある．たとえば，ネットワークの帯域遅延積から得られる最適なソケットバッファサイズ，TCPウィンドウサイズ，メッセージチャンクのサイズ，TCPコネクション数などである．グリッドにおける大規模データ転送のためのプロトコルであるGridFTP[10]は，このようなパラメータを最適化する機構を備えており，また文献11)ではこれらを動的に最適な値に設定するGridFTP-APTという機構を提案している．これらを参考にしてWAN上の通信のスループットを最大限引き出す手法を，今後検討していく必要がある．

5. その他のBcastアルゴリズム

本章では，提案アルゴリズムと前述した単純なアルゴリズム，そしてより広く使われているアルゴリズムの比較を行う．

5.1 Binomial Tree

Binomial Tree（2項木）は，これまでのMPI実装では，サイト内のノードに対するBcast通信で用いられてきた．MagPIe[4]におけるサイト内Bcast通信では，Binomial Treeが用いられている．Binomial Treeの構造を**図4**に示す．このツリーは，ノード数が増えるにつれて $\log p$ のオーダで木の深さが増加していくので，主にメッセージサイズが小さいBcast通信に対して用いられる．

5.2 van de Geijnアルゴリズム

次に，van de Geijn[12]らによって提案されたBcastのアルゴリズムについて述べる．このアルゴリズムの流れは以下のとおりである．

(1) rootノードは，Bcastしたいメッセージを分割し，MPI_Scatter（以下，Scatter）のように他の各ノードに対して送信を行う．
(2) 各ノードでは，分割されたメッセージをそれぞれのノードがMPI_Allgather（以下，Allgather）を行い，分散したデータを集める．

これを模式に表したものが**図5**である．これは，まずrootノードから全体に対してScatterを行いデータを分散させ，その後，Allgatherを実行して各ノードでその分散したデータを集めてBcastを行うという

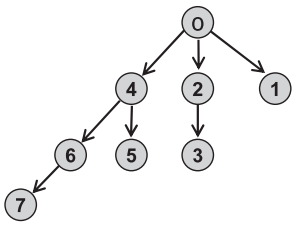

図4 Binomial Tree
Fig. 4 Binomial Tree.

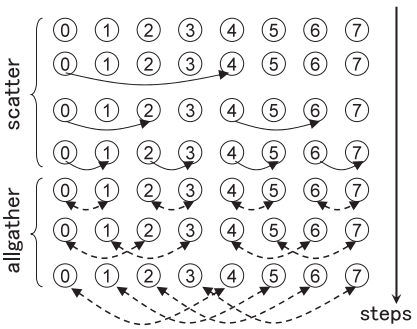

図5 van de Geijnアルゴリズム
Fig. 5 van de Geijn algorithm.

表 1 サイト内におけるコストモデル
Table 1 Estimated costs of Bcast algorithms in a single site.

multi-lane (pipeline)	$\log p \cdot l + M/b$
Van de Geijn	$2\log p \cdot l + 2p/(p-1) \cdot M/b$
binominal	$\log p(l + M/b)$
binary (pipeline)	$\log p \cdot l + 2M/b$
chain (pipeline)	$(p-1) \cdot l + M/b$

アルゴリズムになっている．

MPICH-G2 ではこのアルゴリズムを 512 KB 以下のメッセージの Bcast 通信に用いている[13)~15)]．

5.3 コストモデルによる各アルゴリズムの比較

これまでに述べてきた Bcast 用アルゴリズムのコストモデルをまとめたものを**表 1** に示す．multi-lane, binary tree, single chain に関しては前節までで紹介した．また，binominal tree, van de Geijn アルゴリズムでは，パイプライン化が困難と考えられるため，その影響を考慮しない．

binomial tree は，木の深さが $\log p$ であるので $\log p \cdot (l + M/b)$ となる．van de Gein モデルは，Scatter フェーズでは binomial tree を用いて各ノードにメッセージを送信し，$(l + M/2b) + (l + M/4b) + (l + M/8b) + \cdots = \log p \cdot l + p/(p-1) \cdot M/b$ となる．Allgather フェーズでは recursive doubling を用いてメッセージを root に送信し，Scatter フェーズと同じコストになる．よって，2 つのフェーズを合計し $2 \cdot \log p + 2p/(p-1) \cdot M/b$ がコストモデルとなる．

6. 集団通信シミュレータ

本章では，実験に必要なシミュレータの機能とそれを実現するシミュレータ実装について述べる．

6.1 シミュレータへの要件

提案アルゴリズムを適用したコレクティブ通信，既存のアルゴリズムによるコレクティブ通信をシミュレーションするために，シミュレータでは下記のような機能が必要とされる．

- ネットワークのシミュレーション
  実際にはメッセージが届いていてもサイト間の遅延分だけメッセージが到着するのが遅れているというシミュレートをする必要がある．また，何サイトあるのか，各サイトに何ノードあるのか，などの設定を行う必要がある．
- 通信機能
  要求に応じてコレクティブ通信アルゴリズムを切り替えてプロセス間で通信ができる機構が必要である．
- 複数 NIC のシミュレーション
  複数の NIC があり，それを別々に利用したりアグリゲートして利用しているということをシミュレートする機構が必要である．

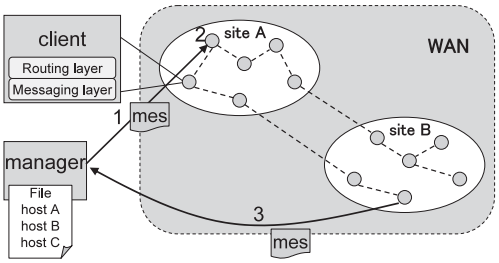

図 6 シミュレータの概要
Fig. 6 Overview of our simulator.

6.2 シミュレータの実装

上記の要件をもとに，コレクティブ通信シミュレータの実装を行った．シミュレータの概要を**図 6** に示す．

コレクティブ通信をシミュレートする Client は，その機能として通信レイヤ，ルーティングレイヤで切り分けられる．ルーティングレイヤでは，マネージャからの指示により設定ファイルを読み込み，ネットワーク設定に応じて提案手法やその他の方法によるトポロジを設定できる．

通信レイヤとして，各ノード間の通信に Overlay Weaver[16)] で提供されているメッセージングサービスを用いた．Overlay Weaver は，オーバレイネットワークを構築するためのツールキットで，ルーティングアルゴリズム，メッセージングサービスの切り分けがなされており，新しいアルゴリズムの実装が容易となっている．シミュレータでは，Overlay Weaver で提供されているメッセージングサービスの送受信のスレッドを 2 つ作成し，仮想的にノードに NIC が 2 枚ある環境を設定した．

ネットワークのシミュレーションは，送信元，受信先のノードが設定したネットワークの中でどこに属しているかを判断し，各 Client ごとでそれに応じてスリープ時間を計算してシミュレートする．

シミュレータは以下の流れで集団通信を行う．

(1) 各プロセス起動時に設定ファイルを読み込み，初期化処理を実行し，サイト間の遅延，バンド幅などのネットワーク状況，サイトごとのノード数，自身のランク，binary tree などのトポロジを設定する．初期化終了後，マネージャノードは，ランク 0 のノードに対して利用する通信アルゴリズム（multi-lane など），メッセージサイズ，マネージャの現在の時刻 T_{start} をまとめたメッセージを送信する．

(2) ランク 0 のノードは，要求されたアルゴリズムから使用するトポロジを選び，メッセージを送信する相手を決定し，コレクティブ通信を開始する．以後すべてのノードでは，メッセージを受信した後，設定されたトポロジを参照して，メッセージを転送する．サイト間を超えるリンクでは，メッセージの受信側では，サイト間遅延の分だけスリープして遅延をシミュレートする．パイプライン化されない転送の場合，送信側では，M/b だけスリープして転送が終了するのをシミュレートする．

(3) 最後にメッセージを受信したノードは，M/b だけスリープしてメッセージ転送コストをシミュレートする．その後，コレクティブ通信終了のメッセージをマネージャノードに対して送る．マネージャノードでは，受け取った時刻を T_{end} として $T_{com} = T_{end} - T_{start}$ を計算しコレクティブ通信に要した時間として記録する．

7. 評　　価

本稿で提案するマルチレーンアルゴリズムの有効性を示すために，シミュレータを用いて Bcast 通信を実行，評価した．提案アルゴリズムとともに，従来手法として通信のパイプライン化が可能な single chain，binary tree アルゴリズムを用いた Bcast 通信の実験を行い，通信完了時間を比較した．実験には本研究室の PrestoIII クラスタを用いた（**表 2**）．マネージャノードを 1 台，メッセージパッシングを行うクライアントノードを 32 台用意し，シミュレータで仮想的に複数サイト環境を構築して実験を行った．

2 サイトの実験における single chain，binary Tree アルゴリズムでの Bcast 通信は，以下のようにして実現した．ここで l はサイト内の遅延，L はサイト間の遅延とし，また，各サイトは p のノード，2 サイト合計で p' のノードを持っているとする．

Single Chain

まず，1 つ目のサイトで single chain を作成し，その chain の最後のノードは，2 つ目のサイトの先頭ノードに接続する．以下サイトの増加に対し同様に繰り返し，全ノードで chain を構築する．このときのコストの合計は，$2(p'-1)\cdot l + L + M/b$ となる．

Binary Tree

まず，1 つ目のサイトのルートノードが 2 つ目のサイトの代表ノードへメッセージを送信する．2 つ目のサイトでは，メッセージを受け取り，2 つ目のサイト内で代表ノードを基点として binary tree を構築し，メッセージの転送を行う．また，1 つ目のサイトでは，2 つ目のサイトの代表ノードのメッセージの送信が終わった後，サイト内で binary tree を構築し，メッセージの転送を行う．このときのコストの合計は，$\log p \cdot l + L + 3M/b$ となる．

7.1 1 サイトでの Bcast の性能

1 サイト環境において，メッセージサイズを変化させたときの Bcast 通信が完了するまでの平均時間を各アルゴリズムごとに計測した．ノード数は 32 ノードである．その結果を**図 7** に示す．

メッセージサイズが小さい場合，multi-lane と binary tree は single chain と比べてつねに良い性能を示し，わずかながら binary tree のほうが multi-lane よりも性能が良い．しかし，メッセージサイズが大きくなると，binary tree はどんどん性能が低下し，メッセージサイズが 128 MB 以上になると single chain よりも時間がかかる結果となった．十分大きいメッセージサイズでは，multi-lane と比べて約 2 倍の実行時間がかかっている．また，single chain は，メッセージサイズによらず multi-lane に対してつねに 50 ms 程度を加算した実行時間で完了している．

この結果から，サイト内における multi-lane を用いた Bcast 通信が，メッセージサイズの大小にかかわらず従来手法よりも高速に実行できることを確認した．メッセージサイズが小さい場合での binary tree

表 2 PrestoIII クラスタのスペック
Table 2 PrestoIII cluster.

OS	Debian/Linux (kernel 2.6.16)
CPU	Opteron242 (1.6 GHz) * 2
Memory	2 GB DDR (PC2100)
NIC	1000Base-T

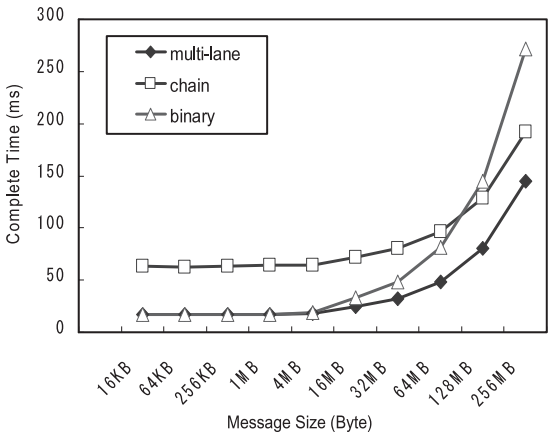

図 7 1 サイトでの Bcast（32 ノード）
Fig. 7 Results of Bcast on one site (32 nodes).

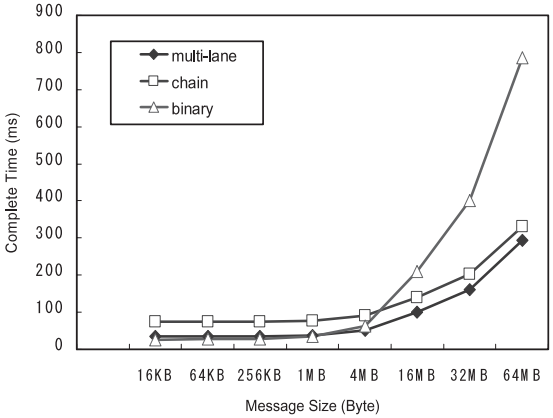

図 8 2 サイトでの Bcast（16 ノードずつ 2 サイト合計 32 ノード，サイト間遅延 10 ms）
Fig. 8 Results of Bcast on two sites (32 nodes in total,inter-site latency is 10 ms).

と multi-lane との差は，非常に軽微である．また，提案アルゴリズムは，メッセージサイズが大きい場合に特に有効であることが示されている．

7.2 2 サイトでの Bcast の性能

2 サイト環境において，メッセージサイズを変化させたときの Bcast 通信完了までの平均時間を計測した．1 サイト 16 ノード，2 サイトの合計が 32 ノードで，サイト間の遅延を L（= 10 ms）と設定した．またサイト間のバンド幅は十分に大きいとする．このときの結果を**図 8** に示す．

2 サイトを用いる場合，1 サイトのときと比べて，どのアルゴリズムに対してもサイト間通信遅延 L の影響が生じる．binary tree は，1 サイトのときと同様にメッセージサイズが小さい場合は multi-lane と比べるとサイト間遅延 L だけ性能が良く，また single-chain と比べると 2 倍程度高速に Bcast 通信を行える．しかしながら，メッセージサイズが大きくなるにつれて性能が低下し，16 MB 以上になると single chain よりも遅くなり，十分大きなメッセージサイズでは，single chain，multi-lane と比べて約 3 倍の実行時間がかかっている．これは，binary tree がサイト間通信が完了するまでサイト内の通信の実行をサスペンドしているためと考えられる．また single chain は，1 サイトのときと同様に，メッセージサイズによらず multi-lane に対してつねに 50 ms 程度を加算した実行時間で完了している．

この結果から，2 サイトの環境においても，multi-lane を用いた Bcast 通信が，メッセージサイズによらずに従来手法よりも高速に実行できることを確認した．

8. 関 連 研 究

従来からあるグリッド上の MPI システムにおけるコレクティブ通信アルゴリズムの多くは，ツリーを基にしたトポロジをネットワークの状況やその他のリソースの状況から判断して構築するが，その多くがシングルレーンで構築されたツリーであり，マルチレーンの利用に言及しているものは少ない．マルチレーンを利用しない MPI システムとして，MagPIe[4] や MPICH-G2[3] があげられる．これらのシステムにおけるコレクティブ通信トポロジは，どちらもサイト間とサイト内という WAN と LAN の階層構造を意識して，通信遅延が大きい WAN 間通信では flat tree を利用し，通信遅延が小さい LAN 内通信では binomial tree や recursive doubling，ring などのトポロジをメッセージサイズに応じて使い分けている．

複数レーンまたは複数コネクションを利用するコレクティブ通信アルゴリズムとして，以下があげられる．Johnsson ら[17] は hypercube network のためのコレクティブ通信トポロジを提案している．Bcast 通信に対しては，Spanning Binomial Tree（SBT）トポロジを拡張した n Edge-disjoint Spanning Binomial Tree（nESBT）トポロジを提案している．これは n 次元の hypercube 上に，各ノードが持つ n 個のポートを利用して n 個の SBT をエッジの重なりなしに構築するものである．メッセージを n 個に分割し，それぞれを各木構造に沿ってパイプライン転送するアルゴリズムであり，本研究はそれと同様の手法を採用する．一方，本研究では hypercube ではなく，スイッチベースのクラスタを WAN で接続したグリッド環境を対象とし，ノードの距離を考慮したアルゴリズムである点が異なる．

SplitStream[6] は peer-to-peer 環境の multicast システムの 1 つであり，主な対象は大規模な動画やファイルの転送である．このシステムでは，各ノードが作成する複数の TCP コネクションを利用して，分散ハッシュテーブル上に複数の木構造を生成する．そしてストリームを複数に分割し，それぞれを別の木構造を通して流すことにより，各ノードのバンド幅を有効に利用する．SplitStream が主な対象とするのは，参加ノードの増減が激しい peer-to-peer 環境であるため，本研究とは以下の点が異なる．まず，各ノードの入出力ストリーム数の調整は動的に行われ，実際の各ノードの通信量は変動しうる．一方，本研究の手法ではノード集合が静的であるという想定のもとで，通信量は安定している．次に，SplitStream の木構造は広

域にちらばった全ノードにより構成されるため，WANを通過する通信が多くなると考えられる．一方，本研究ではクラスタ内とクラスタ間を区別し，クラスタをまたぐ通信量を小さく抑えている．

文献 18) では，複数枚の NIC を用いて WAN 間の通信に対して高いバンド幅を得る Balanced Multicasting を提案している．このシステムでは，モニタリングシステムから得たバンド幅の情報をもとに動的にマルチキャストツリーを構築することに焦点を当てているのでこの点で本研究とは異なる．

また本研究と同様に，高遅延，高バンド幅なグリッド環境に最適化する MPI システムとして GridMPI[2] が開発されている．文献 19) の中で，クラスタ間通信で複数のノードによってコネクションを張りつつ，WAN の輻輳を避けるためにコネクション数を制限するという手法を提案している．また文献 20) では，van de Geijn アルゴリズムをグリッド用に改良し，それを用いた Bcast 通信を提案している．van de Geijn アルゴリズムの前半部分の Scatter をサイト内で行い，その後，Scatter されたデータをクラスタ間でコピーする．このとき，文献 19) で提案されている複数コネクションを張ることによって，WAN のバンド幅を有効に用いてクラスタ間でのコピーを実現する．その後，各クラスタ内で Allgather を実行して Bcast を行うというアルゴリズムになっている．

9. おわりに

本稿では，ノードにある 2 枚の NIC を用いたマルチレーンツリーを提案した．また，そのツリーを利用して 2 枚の NIC のバンド幅を最大限有効に利用できる Bcast 通信用アルゴリズムを提案し，既存の Bcast 通信で用いられる binary tree，single chain との性能の比較，評価を行った．その結果，サイト内における Bcast 通信では，特にメッセージサイズが大きい場合，binary tree を用いた場合と比べて 60%程度の時間で通信を行えることが確認された．サイト間をまたぐ Bcast 通信でも，サイト内と同様に低コストで通信が行えることが確認された．メッセージサイズが大きい場合，binary tree を用いた場合と比べて 3 倍程度高速に通信が行えることを確認した．

提案アルゴリズムは，ノード数 p が増えても binary tree に基づいたトポロジなので各ノードでの遅延が $\log p$ で抑えられ，また，各ノードのバンド幅をほぼ使い切り，single chain のようにストールせずにメッセージ転送をパイプライン化することで，メッセージ転送によるコストがほぼ 1 となるという特徴を持ったアルゴリズムである．以上から，今後のグリッド環境において提案アルゴリズムにおける Bcast 通信の手法が有効になると考えている．

また今後の課題としては，以下を考えている．まず，Bcast 以外のコレクティブ通信，MPI_Allgather，MPI_Allreduce などに対してもマルチレーン化による性能向上が期待できるので，それらについても今後検討していく．文献 19) で提案されていたサイト間を複数コネクションで結ぶ手法と拡張 van de Geijn アルゴリズムと提案アルゴリズムを比較する必要がある．また，本稿ではシミュレータを用いて仮想的に複数サイトを構築し，コストモデルをもとに遅延を計算してコレクティブ通信の実験を行った．また，すべてのノードの NIC の枚数や性能が均一であることを仮定し，ネットワークの輻輳や WAN 間通信で起こりうる性能低下の影響を無視していた．しかしながら，より現実的なグリッド環境に対応させるため，よりヘテロな環境に対してもアルゴリズムの拡張を行っていく必要がある．そして，提案アルゴリズムの実装を行い実環境での実験を検討していく．

謝辞 本研究の一部は，文部科学省科学研究費補助金（特定領域研究 18049028，若手研究（B）17700050）の支援によって行われた．

参考文献

1) Gropp, W., Lusk, E., Doss, N. and Skjellum, A.: High-performance, portable implementation of the MPI: Message Passing Interface Standard, *Parallel Computing*, Vol.22, No.6, pp.789–828 (1996).
2) 松田元彦，石川　裕，鐘尾宜隆，枝元真彦，岡崎史裕，鯉江英隆，高野了成，工藤知宏，児玉祐悦：GridMPI Version 1.0 の概要，*Summer United Workshops on Parallel, Distributed and Cooperative Processing* (*SWoPP*) (2005).
3) Karonis, N.T.: MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, *Journal of Parallel and Distributed Computing* (*JPDC*), Vol.63, No.5, pp.551–563 (2003).
4) Kielmann, T., Hofman, R.F.H., Bal, H.E., Plaat, A. and Bhoedjang, R.A.F.: MagPIe: MPI's collective communication operations for clustered wide area systems, *ACM SIGPLAN Notices*, Vol.34, No.8, pp.131–140 (1999).
5) TeraGrid. http://www.teragrid.org/
6) Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A. and Singh, A.: Splitstream: High-bandwidth multicast in cooper-

ative environments, *19th ACM Symposium on Operating Systems Principles* (2003).
7) Kelly, T.: Scalable TCP: Improving Performance in High Speed Wide Area Networks, *1st International Workshop on Protocols for Fast Long Distance Networks* (2003).
8) Kodama, Y., Kudoh, T., Takano, R., Sato, H., Tatebe, O. and Sekiguchi, S.: GNET-1: Gigabit Ethernet Network Testbed, *IEEE International Conference on Cluster Computing*, pp.185–192 (2004).
9) Dolly+. http://corvus.kek.jp/manabe/pcf/dolly/
10) Mandrichenko, I., Allcock, W. and Perelmutov, T.: GridFTP v2 Protocol Description, *GGF Document Series GFD.47* (2005).
11) 伊藤建志，大崎博之，今瀬　眞：データ転送プロトコル GridFTP のための並列 TCP コネクション数調整機構の性能評価，技術報告，電子情報通信学会 (2006).
12) Barnett, M., Shuler, L., Gupta, S., Payne, D.G., van de Geijn, R.A. and Watts, J.: Building a high-performance collective communication library, *Supercomputing*, pp.107–116 (1994).
13) Pjesivac-Grbovic, J., Angskun, T., Bosilca, G., Fagg, G.E., Gabriel, E. and Dongarra, J.J.: Performance Analysis of MPI Collective Operations, *19th International Parallel and Distributed Processing*, IEEE Computer Society Press (2005).
14) Lacour, S.: MPICH-G2: Collective Operations Performance evaluation, optimizations, Technical report, Argonne National Laboratory Mathematics Computer Science Division (2001).
15) Thakur, R., Rabenseifner, R. and Gropp, W.: Optimization of Collective Communication Operations in MPICH, *International Journal of High Performance Computer Applications*, Vol.19, No.1, pp.49–66 (2005).
16) 首藤一幸，田中良夫，関口智嗣：オーバレイ構築ツールキット Overlay Weaver, 情報処理学会論文誌：コンピューティングシステム，Vol.47, No.SIG12 (ACS 15), pp.358–367 (2006).
17) Johnsson, S.L. and Ho, C.-T.: Optimum Broadcasting and Personalized Communication in Hypercubes, *IEEE Trans. Comput.*, Vol.38, No.9, pp.1249–1268 (1989).
18) den Burger, M., Kielmann, T. and Bal, H.E.: Balanced Multicasting: High-throughput Communication for Grid Applications, *ACM/IEEE Conference on Supercomputing* (2005).
19) 松田元彦，石川　裕，工藤知宏，児玉祐悦，高野了成：グリッド上のコレクティブ通信アルゴリズム，*Summer United Workshops on Parallel, Distributed and Cooperative Processing* (*SWoPP*) (2006).
20) Matsuda, M., Ishikawa, Y., Kudoh, T., Kodama, Y. and Takano, R.: Efficient MPI Collective Operations for Clusters in Long-and-Fast Networks, *IEEE Cluster 2006* (2006).

(平成 18 年 10 月 10 日受付)
(平成 19 年 2 月 6 日採録)

千葉　立寛（学生会員）

1983 年生．2006 年東京工業大学理学部情報科学科卒業．現在，同大学大学院情報理工学研究科数理・計算科学専攻修士課程在学中．グリッド環境における MPI 実行，集団通信アルゴリズムに興味を持つ．

遠藤　敏夫（正会員）

1974 年生．2001 年東京大学大学院理学系研究科情報科学専攻博士課程修了．博士（理学）．科学技術振興機構研究員，東京大学大学院情報理工学系研究科特任助手等を経て，2006 年より東京工業大学学術国際情報センター特任講師．主に分散・並列処理の研究に従事．日本ソフトウェア科学会，ACM，IEEE-CS 各会員．

松岡　　聡（正会員）

1963 生．1986 年東京大学理学部情報科学科卒業．1989 年同大学大学院博士課程から，学情報化学科助手に採用，同大学情報工学専攻講師を経て，1996 年に東京工業大学情報理工学研究科数理・計算科学専攻助教授．2001 年 4 月に東京工業大学学術国際情報センター教授，2002 年より国立情報学研究所の客員教授を併任．博士（理学）．