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いて、遅延やバンド幅などのネットワークの状態に応じて、最適なネットワークトポロジを構築し、MPI 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いる。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、提案されている様々な手法では、各ノードで送受信出来るポートを 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。ノードに備えている複数の NIC をコレクティブ通信に用いれば、より効率的にコレクティブ通信を行える。そこで本研究では、ノードにある NIC を 2 枚を用いてマルチレーンのツリーを構築して集団通信を行うアルゴリズムを提案する。コレクティブ通信の 1 つである Bcast について、シミュレーター環境上で実験、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 Multirails for Grid Environment

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

The best network topology is constructed on the grid environment according to states of networks of the delay and the bandwidth, etc, and a lot of various optimaization techniques for the MPI collective communication are proposed. Moreover, each node often provides with two or more NIC in the cluster system in recent years. However, the technique for the assumption of the port that can be sent and received by each node and one constructing the topology is most in the proposed various techniques. If two or more NIC with which it provides in the node is used for the collective communication, the collective can communicate more efficiently. Then, in this research, it proposes NIC that exists in the node and it proposes the algorithm to construct the tree of the multirails by using two pieces and to communicate the group. It was confirmed that Bcast that was one of the collective communications was experimented on the simulator environment, evaluated, and the performance had improved more than the techniques so far.

1. はじめに

近年のグリッド技術の発展とクラスタ環境の普及によって、高エネルギー物理学やバイオインフォマティックスなど様々な分野での大規模な科学技術計算を実行させるための環境として、グリッドの利用が高まっている。このような分野における大規模な計算をグリッド上で実行するために、並列プログラミングライブラリである MPI[1] を用いてアプリケーションプログラムを書くことが一般的である。MPI アプリケーションの特性にもよるが、各ノード間で頻繁に通信を繰り返すようなアプリケーションでは、コレクティブ通信の性能が、アプリケーションの性能に大きく関わってくる。グリッド上でこれらの MPI アプリケーションの性能を向上させるためには、単一サイト内だけでなく、複数サイト間をまたがって通信することを想定し、それらのノード間同士でのコレクティブ通信のより一層の性能向上が必要不可欠である。

これまでにも、MagPIe[2], MPICH-G2[3] などグリッド上で動作するように最適化された MPI システムが数多く提案されており、コレクティブ通信の性能向上は、どのシステムにおいても重要な位置づけがなされている。これらのシステムで提案されている最適化されたコレクティブ通信アルゴリズムは、サイト間を結ぶネットワークは高遅延で低バンド幅、各クラスタ内のノードの NIC は 1 枚であるという前提のもとで提案されている。

しかしながら、ネットワークの技術の向上により、最近のクラスタノードでは NIC が複数搭載されてい

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

るものが一般的であり、また、数十 Gbps の処理性能を持つスイッチや、回線が提供されてきている。このような背景から、グリッドのネットワークとして、サイト間を結ぶネットワークがより高バンド幅であるという状況も容易であると考える。そのため、これまでに提案されてきたグリッド環境における MPI コレクティブ通信アルゴリズムでは、このようなネットワーク環境を想定していないため、利用出来るネットワーク性能を十分に引き出すことが出来ない。

そこで本稿では、クラスタノードに搭載されている複数の NIC を有効に用いて、MPI コレクティブ通信のための最適なトポロジ構築アルゴリズムを提案する。今回、コレクティブ通信の 1 つであるブロードキャスト (以下 Bcast) 通信に対して、提案するマルチレーンを用いたツリーを構築し、従来手法でのアルゴリズムによる Bcast との性能の比較を行った。その結果、サイト内、サイト間通信を含むどちらの場合でもメッセージサイズの大きな Bcast 通信において、より高い性能の向上が確認できた。

2. ネットワークモデル

この節では、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とする。

クラスタ内のスイッチは充分な性能を持ち、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成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を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 NIC 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よく動作する。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出来るとする。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伴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[4] 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 アルゴリズムの一つは、single chain によるものである。この手法では、ノード 0 が他の全てのノード $(1 \cdot p - 1)$ 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... と順々に通信を行う。この手法のコストは、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 $(p-1)(l+M/b)$ であり、他の手法より悪い。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良く知られている。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。このときのコストは、パイプライン化に伴うオーバヘッドを無視すれば、$(p-1)l + M/b$ 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。各ノードは二つの子ノードへ、NIC を一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 $\log p(l+2M/b)$ 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。この手法は M が小さいときは良いが、M が充分大きいときは single chain より 2 倍悪い。この理由は、各ノードの持つバンド幅を充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全く行っていない。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。

次節で述べるアルゴリズムは、両者の欠点を解決し、さらに前節で述べたようなグリッド環境で効率よく動作することをねらいとする。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全て活用し、さらにストールの無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

この節では、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。

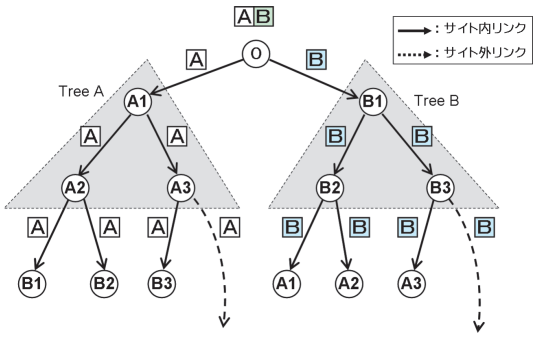

図 1 サイト内のマルチレーンツリー

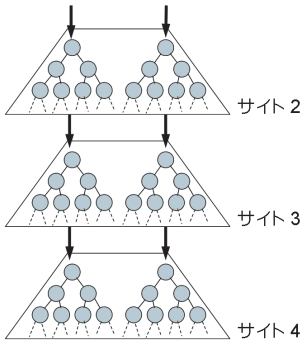

図 2 サイト間のマルチレーンツリー

3.1 サイト内マルチレーンアルゴリズム

$Tree_A$ 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 とリンクが一対一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 $Tree_B$ の二つの同サイズの 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、そうでなくとも良い。$Tree_A$ 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、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通りである。

(1) ノード 0 は Bcast したいメッセージ M を M_A、M_B に二分する。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二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一本ずつ用いる。

(3) (2) の段階では、leaf ノード群は二本の送信リンクを、全ノードが一本の受信リンクを残しているので、それらをもう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 $A_{n+i} (0 \leq i < n)$ は、二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すれば、Bcast は完了である。

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二つの 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 ノードの一つである 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。

なお、本稿では WAN をまたがった一ノード対一ノードの転送が、充分なバンド幅で行えるときに効率よく動作する。これは一本の 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[5] などの WAN 向けの通信ソフトウェアを用い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、今後の課題の一つである。

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通りとなっている。

- 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

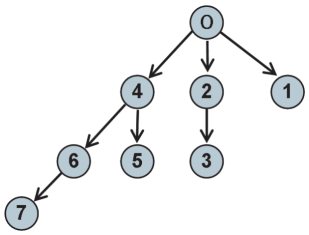

図 3 Binomial Tree

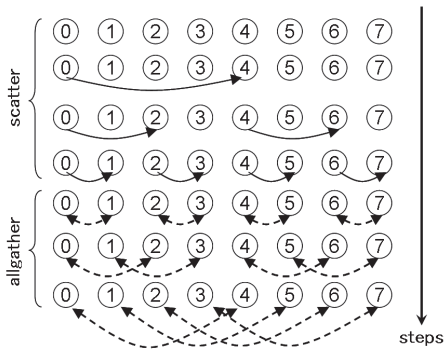

図 4 van de Geijn アルゴリズム

全てのノードが、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出来るので、非常に効率よく Bcast メッセージの転送をすることが可能となる。

4. その他の Bcast アルゴリズム

本節では、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。

4.1 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 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 3 に示す。このツリーは、ノード数が増えるにつれて $\log p$ のオーダー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。

4.2 van de Geijn アルゴリズム

次に、van de Geijn[6] 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通りである。

(1) root ノードは、Bcast したいメッセージを分割し、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 Allgather を行い、分散したデータを集める。

表 1 サイト内におけるコストモデル

multirail (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)	$(p-1) \cdot l + M/b$

これを模式に表したものが図 4 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 512KB 以下のメッセージの Bcast 通信に用いている。[7)8)9)]

4.3 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。multirail、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 $logp$ であるので $\log p(l + M/b)$ 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。

5. 集団通信シミュレータ

本節では、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。

5.1 シミュレータへの要件

提案アルゴリズムを適用した集団通信、既存のアルゴリズムによる集団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、等の設定を行う必要がある。
- 通信機能
  要求に応じて集団通信アルゴリズムを切り替えて

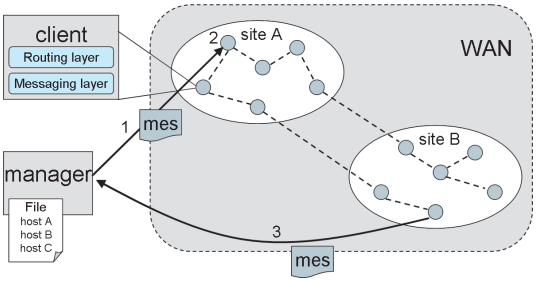

図 5 シミュレータの概要

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。

5.2 シミュレータの実装

上記の要件をもとに、集団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 5 に示す。

集団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ー、ルーティングレイヤーで切り分けられる。ルーティングレイヤーでは、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。

通信レイヤーとして、各ノード間の通信に Overlay Weaver[10] 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、オーバー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成されており、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 (multirail 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、集団通信を開始する。以後全てのノードでは、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。サイト間を越えるリンクでは、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、集団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 $T_{com} = T_{end} - T_{start}$ を計算し集団通信に要した時間として記録する。

表 2 PrestoIII クラスタのスペック

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

6. 評　　価

6.1 評価実験の概要

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。実験では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場合の Bcast 通信を行って比較し、サイト内通信、サイト間をまたぐ通信のどちらにおいても効率よくメッセージ転送が行われることを確認した。実験には本研究室の PretoIII クラスタを用いた。(表 2) マネージャーノードを 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 $n(n = 32, 64, \cdots)$ 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。実験方法を以下に示す。

まず、マネージャーノードから Bcast したいメッセージのサイズ、Bcast で用いるアルゴリズム、現在の時刻のタイムスタンプをまとめたメッセージを rank 0 のノードへ送る。マネージャーノードから rank 0 のノードへ転送したいメッセージサイズと利用するアルゴリズムを送る。その後、クライアントノード同士で指定されたアルゴリズムに従い Bcast を実行し、最後にメッセージを受け取るノードが Bcast が完了したことをマネージャーノードへ通知する。マネージャーノードはメッセージを受け取った時刻と開始時の時刻との差分を計算し、Bcast にかかった時間を記録する。

また、比較対象である single chain、binary Tree アルゴリズムにおける Bcast は以下のようにして実現した。

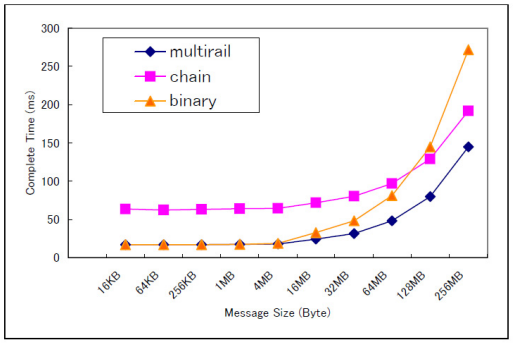

図 6 1 サイトでの Bcast (32 ノード)

Single Chain

全クライアントで仮想的な Ring を構築し、自分の rank が i のときは $i+1$ の rank のノードへメッセージを転送する。このときの通信をパイプライン化するので、そのときの通信コストは、$(p-1)\cdot l + M/b$ となる。サイト間をまたぐ場合のコストはサイト間リンクに比例し、$c\cdot L$ となるので、合計するとそのコストは、$(p-1)\cdot l + M/b + c\cdot L$ である。

Binary Tree

サイト内では、2 本あるリンクを用いて Tree にそってメッセージをパイプライン転送していく。そのときの通信コストは、$\log p\cdot l + 2M/b$ となる。サイト間のコストは、root ノードが始めにメッセージを別サイトでの binary Tree の root ノードへサイトごとに送るので、そのコストは $L + c\cdot M/b$ となる。合計するとそのコストは、$\log p\cdot l + 3M/b + L$ である。

6.2 評価実験結果

6.2.1 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 6 に示す。

メッセージサイズが小さい時の各アルゴリズムでの通信時間はほぼ一定である。提案アルゴリズムでは、メッセージサイズによらず、single chain と比べて常に 40%程度の性能向上が見られた。binary tree と比較すると、メッセージサイズが小さい場合は、binary tree のほうが短い時間で Bcast を完了するが、メッセージサイズが大きくなると通信時間がメッセージサイズに比例して増加していき、十分大きいサイズのメッセージでは、chain の約 2 倍程度の時間がかかっていることが分かる。

この結果から、サイト内における提案アルゴリズムを用いた Bcast 通信が、従来手法よりも高速に行えることを確認した。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。

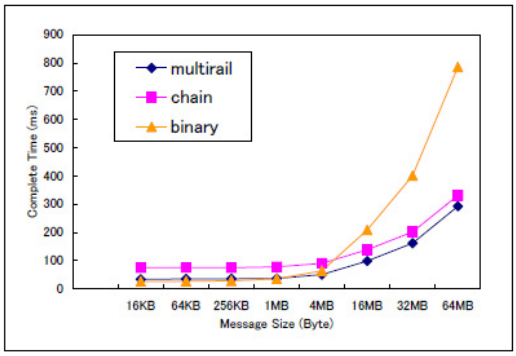

図 7 2 サイトでの Bcast (32 ノード サイト間遅延 10ms)

6.2.2 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。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 $L(=10ms)$ 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 7 に示す。

1 サイトのときとは違い、今回は 2 サイトを利用しているのでサイト間の通信に L の遅延が生じる。multirail、chain においては、サイト間通信が 1 回生じるので、サイト間通信に関しては 1 サイトのときよりも L の分だけ性能が低下している。binary tree においては、rank 0 のノードが他サイトの root ノードに最初に転送し rank 0 のノードはその転送が完了するのを待ってからサイト内への通信を開始するので、rank 0 のノードがあるサイトでは M/b、もう一方のサイトでは遅延 L 分だけ性能が低下している。

binary tree のコストは、$\log p\cdot l + 2M/b + (L + M/B) = \log p\cdot l + 3M/b + L$、multirail のコストは、$\log p\cdot l + M/b + L$ であり、計測されたデータからも十分大きいサイズのメッセージでは multirail と binary tree の性能は、約 3 倍程度の差が生じていることが確認される。

7. 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 や MPICH-G2 が挙げられる。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。

マルチレーンに言及されているものとして SplitStream[4] が挙げられる。SplitStream は peer-to-peer 環境の multicast システムの一つであり、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、分散ハッシュテーブル上に複数の木構造を生成し、各ノードは全ての木に参加する。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の有効利用が可能である。本稿のクラスタ内のアルゴリズムは、SplitStream が目指すトポロジを静的に指定するものとなっている。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、実際に利用されるバンド幅は確率的である。次に、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。

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[11] が開発されている。文献[12] 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[13] 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[12] 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。

このアルゴリズムは、van de Geijn 同様にパイプライン化は困難であると考えられる。提案アルゴリズムは、ノード全体でパイプライン化を行うので、メッセージサイズが大きいとき有利と考えられる。実際の比較は、今後の課題である。

8. おわりに

本稿では、ノードにある NIC を 2 枚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出来る 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集団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。

- 他の集団通信へのマルチレーンの適応
  本稿では Bcast 通信に適応させたが、Allgather、Allreduce などその他の集団通信で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。
- 拡張 van de Geijn アルゴリズムとの比較
  文献[12] 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。
- より現実的な環境での実験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集団通信の実験を行ったが、実環境で起こりうるネットワークの輻輳などが考慮されていない。今後は、サイト間通信にネットワークエミュレータである GtrcNET-1[14] を利用してサイト間通信を行ったり、さらには実際のグリッド環境で動作するシステムの構築も検討する。

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) 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).
3) Karohis, N. T.: MPICH-G2: A Grid-Enabled Implementation of the Message Passing Interface, *Journal of Parallel and Distributed Com-*

puting (JPDC), Vol. 63, No. 5, pp. 551–563 (2003).
4) Castro, M., Druschel, P., Kermarrec, A., Nandi, A., Rowstron, A. and Singh, A.: Splitstream: High-bandwidth multicast in cooperative environments, *In 19th ACM Symposium on Operating Systems Principles* (2003).
5) Kelly, T.: Scalable TCP: Improving Performance in High Speed Wide Area Networks, *First Internation Workshop on Protocols for Fast Long Distance Networks* (2003).
6) 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).
7) 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).
8) Lacour, S.: MPICH-G2:Collective Operations Performance evaluation, optimizations, Technical report, Argonne National Laboratory Mathmatics Computer Science Division (2001).
9) Thakur, R. and Gropp, W.: Improving the Performance of Collective Operations in MPICH, *Euro PVM/MPI* (2003).
10) 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, pp. 358–367 (2006).
11) 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).
12) 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).
13) M.Matsuda, Y.Ishikawa, T.Kudoh, Y.Kodama and R.Takano: Efficient MPI Collective Operations for Clusters in Long-and-Fast Networks, *IEEE Cluster 2006* (2006).
14) GtrcNET-1: `http://projects.gtrc.aist.go.jp/gnet`.