GridMPI™ の性能評価

松田元彦[†] 石川 裕[†,††] 鐘尾宜隆[†]
枝元真彦[†] 岡崎史裕[†] 工藤知宏[†]
児玉祐悦[†] 手塚宏史[†]

GridMPI はグリッド環境での高性能計算用 MPI 実装である。グリッド上の並列コンピュータ間通信プロトコルの一つとして IMPI(Interoperable MPI) プロトコルを実装している。IMPI は異なる MPI 実装間で通信するための標準規格として提案されているものである。GridMPI の IMPI プロトコル実装における、一対一通信、全対全通信、ブロードキャスト通信、リダクション通信の性能評価を行なう。クラスタ用 MPI 実装と比較することで、IMPI プロトコルの定義するフロー制御、ランデブ通信、コレクティブ通信アルゴリズムの問題を明らかにする。さらに、クラスタ用 MPI 実装に対しても、ボトルネック・リンクが存在するネットワークでのコレクティブ通信アルゴリズムについて示唆する。

Performance Evaluation of the GridMPI™

MOTOHIKO MATSUDA ,[†] YUTAKA ISHIKAWA ,[†,††] YOSHITAKA KANEO ,[†]
MASAHIKO EDAMOTO ,[†] FUMIHIRO OKAZAKI ,[†] TOMOHIRO KUDOH ,[†]
YUETSU KODAMA [†] and HIROSHI TAZUKA[†]

GridMPI is a high-performance MPI implementation for the Grid environment. The IMPI protocol, which is an existing standard for interconnecting different MPI implementations, is implemented as one of the protocols for inter-cluster communication in GridMPI. Evaluation of the IMPI protocol is performed on operations including point-to-point, all-to-all, broadcast, and reduction using the GridMPI implementation. Comparison between the IMPI protocol and a cluster MPI protocol reveals that the IMPI's flow-control, rendezvous communication, and algorithms for collective communication have non-optimal performance. On the opposite, it also reveals that a cluster MPI protocol has non-optimal performance in collective communication in a network with a bottle-neck link.

1. はじめに

グリッドの重要な応用に、インターネット上の複数の計算資源を駆使した大規模並列アプリケーションの実行がある。ネットワークの性能向上により、特にメトロポリタン・ネットワーク内においては、従来非現実的とされてきた、分散した計算資源上での並列アプリケーションの実行が現実性を帯びてきた。そこでわれわれはグリッド環境で MPI を用いた高性能計算を行うことを目指して GridMPI[3),7)] の開発を行なっている。

我々は論文 12) において、クラスタ 2 台を人工的に通信遅延を与えるルータにより接続した環境で、NAS 並列ベンチマーク [1)] の性能評価を行なった。その結果、4 ミリ秒程度までの遅延であれば、CG や LU といったベンチマークではクラスタを 2 台接続することによって 1 台の場合の 1.2〜2 倍に性能が向上することを明らかにした。この結果を元に、GridMPI では、遅延が数ミリ秒であるメトロポリタンネットワークで接続されたクラスタ計算機群を主対象として開発を行っている。

クラスタ環境と、複数のクラスタが相互接続されたグリッド環境では、通信に関わるパラメータが大きく異なる。クラスタ内での通信遅延は数マイクロ秒であるのに対し、クラスタ間では数ミリ秒の遅延が想定される。これまで、MPI 通信ライブラリはクラスタ計算機などの遅延の小さなネットワークを念頭に設計されてきた。遅延に 1000 倍の差異がある場合、通信プロトコル、集合通信アルゴリズム等に異なるものが必要になってくる [8),9)]。例えば典型的な MPI の実装では、メッセージサイズが大きい場合にランデブ通信が用いられる。これは、通信データの転送に先だってハンドシェークを行ない、コピーするメモリ領域を特定してからデータを転送するもので、バッファ使用量を制限するとともにメモリコピーを削減する。しかし、あらかじめ行うハンドシェークには最低でも往復の遅延時

† 産業技術総合研究所 グリッド研究センター
National Institute of Advanced Industrial Science and Technology
†† 東京大学 大学院情報理工学系研究科
University of Tokyo

間がかかるため、グリッド環境での通信には適さない。

GridMPI では GridMPI 独自のプロトコルを含めて複数のプロトコルをサポートしていく予定であるが、まず最初の実装として、IMPI(Interoperable MPI)[2] プロトコルを実装した。GridMPI では、クラスタ用 MPI 実装である YAMPII[6),16)] をベースに、IMPI プロトコルを追加実装した。IMP は MPI 実装間を接続するためのプロトコルを規定しており、第 0.0 版が 2000 年に発行された。現在これが最新版である。IMPI は、MPI 実装間のネットワーク接続には TCP/IP を前提としており、起動時のコネクション確立方法、パケット・フォーマット、ヘテロ環境対応のためのバイトオーダー等のデータ型表現、パケットフロー制御、コレクティブ通信アルゴリズムというように MPI 実装に関するプロトコル全般を定義している。

最近いくつかの MPI 実装が IMPI のサポートを表明している。例えば、HP MPI[4]、LAM/MPI[11]、MPI/Pro[13]、PACX MPI[17] などが IMPI プロトコルを実装している (部分的な実装も含む)。他に Stampi[5] なども IMPI のプロトコルを一部で利用している。IMPI をサポートすれば、単に GridMPI 内だけでなく複数の MPI 実装と相互接続できるメリットがある。ただし、IMPI 規格自体はグリッドに向けて設計されたものではなく、IMPI プロトコルには遅延の小さな LAN などの通信環境を想定していると考えられる部分がある。しかし、MPI 間を相互接続するための基本機能が IMPI には定義されているので、それをベースに GridMPI のプロトコル設計を進めるのが早道だと判断した。

本稿は、IMPI プロトコルの GridMPI への実装を通して、GridMPI に適した通信プロトコルを検討し、今後の GridMPI プロトコルの設計に役立てるのが目的である。以下では、GridMPI の実装方法、GridMPI の性能評価について述べた後、性能上の問題点について議論し、最後にまとめを述べる。

なお、GridMPI は対象をクラスタ計算機群に限定してはいないが、以下簡単のため「クラスタ内」「クラスタ間」という言葉を用いる。

2. GridMPI の実装

GridMPI は現在、クラスタ計算機向けの MPI 実装の一つである YAMPII[6),16)] をベースに、IMPI プロトコルの処理を追加する形で実装されている。

YAMPII はソフトウェア階層のそれぞれで拡張可能なように設計されている。P2P 層 (ネットワーク通信) では、例えば Unix 等のソケット・インタフェイスに相当する部分を他の実装に置き換え可能である。YAMPII の P2P 層には現在 TCP/IP を利用するソケット P2P 実装と、SCore[14)] クラスタシステムの提供するネットワーク層を利用する SCore P2P 実装がある。また、MPI Core 層では、コミュニケータ毎に MPI オペレーションを置き換え可能になっている。例えばコミュニケータ毎に異なるコレクティブ通信アルゴリズムを定義することなどが可能である。

GridMPI の IMPI 実装では、P2P 層と MPI オペレーションを置き換えている。P2P 層では、IMPI プロトコルのパケット生成、送受信、IMPI のパケットフロー制御を実装している。MPI オペレーションでは、コレクティブ通信のアルゴリズムとして IMPI 準拠のものを実装している。**MPI_Send** や **MPI_Recv** といった基本通信プリミティブに関しては、P2P 層で違いが吸収されるので無変更のまま使用している。

IMPI 実装と YAMPII オリジナルのソケット P2P 実装はともに Unix のソケットを利用しており、P2P 層の構成として大きく異なる部分はない。ただし、通信性能の面では、IMPI の実装には YAMPII にはない、いくつかのオーバーヘッドが存在する。IMPI 実装では IMPI 準拠のパケットフォーマットを用いており YAMPII に比べてヘッダサイズが大きい。特に IMPI ではヘッダ内の整数値として 64bit 表現が多用されている。また、IMPI ではメッセージデータのバイトオーダーやパック形式が規定されているため、データの変換を行なう必要がある。また、IMPI では最大パケットサイズが規定されており、大きなパケットを分割する必要と、フロー制御のために制御パケットの送受信を行なう必要がある。

IMPI ではコレクティブ通信のクラスタ間のアルゴリズムは規定しているが、クラスタ内については自由度がある。例えばブロードキャストでは、クラスタ間でデータを通信した後、そのデータをクラスタ内のプロセス間で通信するが、クラスタ内の部分の実装は自由である。GridMPI では、クラスタ内の通信に YAMPII のコレクティブ通信を使用している。

IMPI 規格では、MPI プロセスと「ホスト」を区別している。ここでホストとは、IP アドレスを持ち外部へ接続しているノードを指す。一つの構成として、MPI プロセスのそれぞれが IP アドレスを持ち外部と通信できるホストとなるものがある。クラスタ計算機などはこの構成にあたる。その対極の構成として、単一ノードのみがホストとして外部と通信するものがある。専用 MPP 計算機などでは、この構成をとる場合がある。ただし、クラスタ計算機であっても単一ノードのみがホストとなる構成も可能である。単一ノードがホストになる場合、すべてのメッセージを一旦ホストノードでフォワードするなどの処理が必要になる。IMPI 規格上は、MPI プロセスとホストの関係は自由で、クラスタに対して任意の数のホストを指定することも可能である。

GridMPI の開発では、複数のコレクティブ通信アルゴリズムを実験することを念頭に置いている。そのため、各 MPI プロセスがホストとなる構成を採っている。MPI プロセスのそれぞれが直接クラスタ外部と通

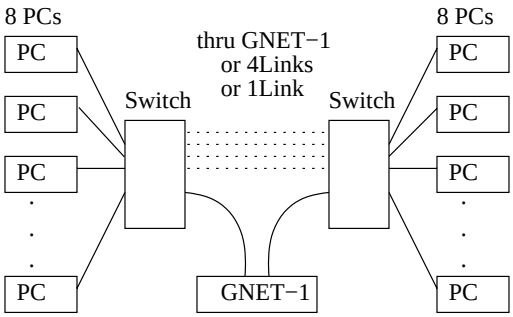

図 1 実験環境の構成

表 1 PC クラスタ諸元

ノード PC (16 台)	
CPU	Pentium4 2.8C（2.8 GHz）
マザーボード	Intel D865GLC
メモリ	1GB DDR400
NIC	Intel 82547EI（オンボード CSA 接続）
OS	RedHat 9（Linux-2.4.20）
NIC ドライバ	Intel e1000 5.2.16
ネットワークスイッチ	
	CISCO Catalyst 3750G-24T

信を行えるのでアルゴリズムの記述に制約がない。一方、単一ノードがホストとなる場合、常にホストノードを経由するのでコレクティブ通信の方法が限定されてしまう。

GridMPI では、当面の設計として、セキュリティを考慮しないこととした。現状では、セキュリティを確保した通信は暗号化などに大きなオーバヘッドがあるため、高性能計算の制約になる。一方、GridMPI が対象とするのは、高速ネットワークで相互接続された計算機センターにおかれた計算機群を用いて大規模な科学技術計算を実行する環境である。そこでは、専用回線が利用できるなど、セキュリティが確保されていることが期待できる。また、近い将来ネットワークのバンド幅や遅延に大きな影響を与えない高性能な暗号化ハードウェアが利用できるようになることを期待している。また、IMPI 規格はセキュリティについて規定していない。

3. 評　　価

3.1 実験環境

図 1 に実験環境を図示する。また、使用した PC の諸元を**表 1** に示す。グリッド環境に近い測定環境を作るため、16 台の PC から構成されるクラスタを 2 つに分割しそれぞれをスイッチで接続した。クラスタ間の接続状態により性能がどのように変わるかを評価するために、スイッチ間のリンクについて、4 リンク (アグリゲート) による 4Gbps の接続、1 リンクでの 1Gbps の接続、1 リンク (1Gbps) でかつ間に片道 1ms の遅延を挿入した接続、の 3 種の接続を用いた。ネットワークの遅延挿入には、ワイヤレートで正確に遅延を挿入できるハードウェア・ネットワーク・エミュレータである GNET-1[10] を用いた。

ここで示す実験環境のようにスイッチ間リンクにボトルネック・リンクがあると、コレクティブ通信の性能にアルゴリズムなどによる違いが生じる。また、スイッチ間を 4 リンクで結ぶ構成を使用することで、クラスタ間接続のバンド幅が大きい場合にそのバンド幅を有効に活用できるかどうかが確認できる。これは、LAN よりバンド幅が大きい 10Gbps といった広域網を利用する場合に重要になってくる。

グラフ中に用いる凡例は「IMPI(d)」、「YAMPII(4)」などであるが、IMPI は IMPI プロトコルを、YAMPII はベースになっているクラスタ用 MPI を指す。カッコ内の文字はリンクの状態を示すが、次のような意味である。

(4)	スイッチ間を 4 本のリンク (4 アグリゲート) で接続
(1)	スイッチ間を 1 本のリンクで接続
(d)	スイッチ間を 1 本のリンクで接続、片道 1ms の遅延を挿入

また、バンド幅のグラフではデータサイズ (X 軸) は対数で表示している。

3.2 NAS 並列ベンチマーク

まず、MPI 実装のベンチマークとして広く利用されている NAS 並列ベンチマーク [1] の性能をみる。

図 2 に NAS 並列ベンチマークの結果を示す。グラフは YAMPII で 1ms の遅延がある場合 (YAMPII(d)) を 1 とした相対性能である。性能は「Mop/s total」値を元にしている。IS で 4 リンクを使用する場合グラフの範囲を超えているが、YAMPII(4) 時 2.8 倍、IMPI(4) 時 2.7 倍であった。

IMPI は、YAMPII と大変良く似た性能を示した。CG と FT では、わずかであるが IMPI の方が YAMPII より性能が良い。プロトコル上 IMPI は YAMPII に比べてオーバーヘッドが大きいことを考えると興味深い。以降、基本通信の性能をそれぞれ評価していくことにする。

3.3 一対一バンド幅

バンド幅の実験では整数型のデータを通信している。このため、IMPI プロトコルを使用する場合バイトオーダーの変換が行なわれている。

図 3 に一対一のバンド幅を示す。2 台のノード間通信なので 1Gbps を超えることは無く、スイッチ間のリンク数の影響を受けないので、遅延がある場合とない場合のみを示した。実験では一方的に指定サイズのメッセージを連続して繰り返し送信している (**MPI_Send**)。連続して送信しているので、理想的にはバンド幅は遅延の影響を受けない。YAMPII では、遅延の影響を受

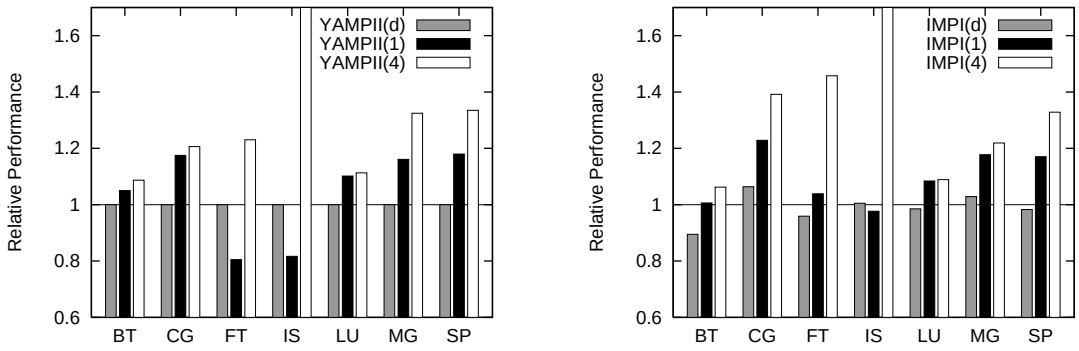

図 2 NAS 並列ベンチマーク結果

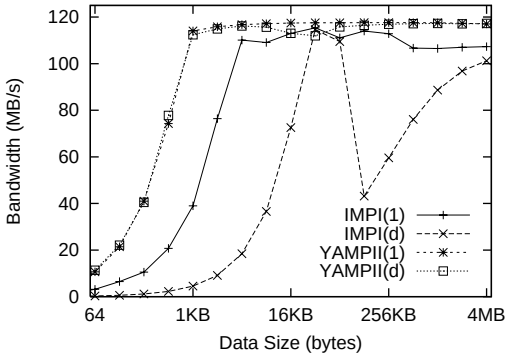

図 3 一対一バンド幅

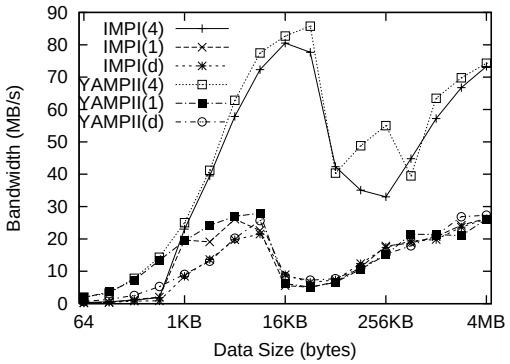

図 4 全対全バンド幅

けていないことが分かる。一方、IMPI では立上りが遅いこと、64KB で性能が低下していること、遅延の影響を受けていることが分かる。

IMPI では、データのバイトオーダーの変換を行なっている。また、パケットのヘッダサイズが大きいため、オーバーヘッドが大きい。これが立上りが遅い原因だと考えられる。

IMPI ではパケットサイズに上限を設けている。この上限値自体は実装依存であり、複数の実装が混在する環境では一番小さい値に合わせる。現在、GridMPI の IMPI 実装では上限を 64KB に設定している。IMPI では、メッセージが複数パケットに分割される場合にランデブ通信を行なう。メッセージの最初のパケットは受信側の状態によらず送信するが、続くパケットは受信側からの応答を待ってから送信する。メッセージサイズが 64KB のときに性能が大きく低下しているのはこのためである。

また、IMPI では、通信リンクを共有するプロセス間で通信リンクの利用を公平にする目的で、パケットの流量制限が導入されている。これは、パケット数のカウンタを使用する単純なクレジットベースのもので、受信側からの応答なしに送信できるパケット数を制限している。このパケット数は実装依存であるが、現

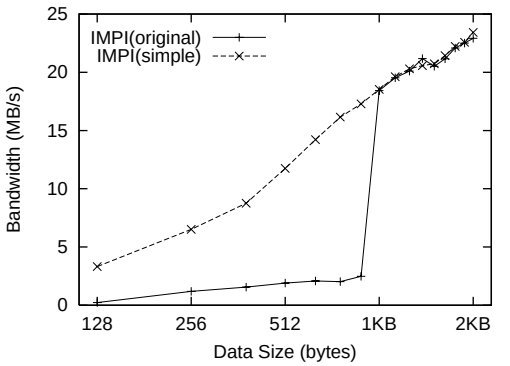

図 5 全対全バンド幅 (アルゴリズム変更)

在は 20 パケットに設定している。通信リンクの公平な利用のためには小さい値で制限するべきであるが、パケットサイズが小さい場合には、帯域遅延積に対して十分なパケットが送信されず、バンド幅が低下している。

3.4 全対全バンド幅

図 4 に全対全通信 (**MPI_Alltoall**) のバンド幅を示す。バンド幅は、1 つのプロセスからの全送信バイト数を通信時間で割って求めた。IMPI と YAMPII はほ

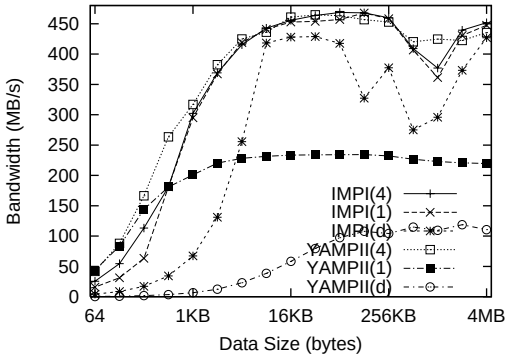

図 6 ブロードキャスト・バンド幅

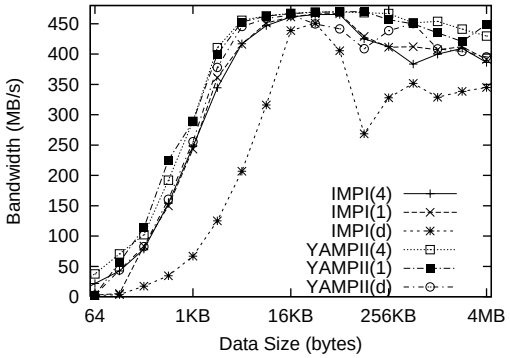

図 7 リダクション・バンド幅

ぼ同じような性能を示している。全対全通信では毎回受信により同期するため、一対一の場合のような一方的な転送による IMPI と YAMPII の差が出ていない。16KB と 64KB の点での性能の落ち込みは、スイッチにおけるパケットロスに起因する TCP/IP の輻輳制御が原因だと考えられる [15]。

グラフを良く見ると、1KB 以下のメッセージサイズでは YAMPII に比べて IMPI の性能が低くなっているのが分かる。IMPI では、全対全、ギャザー、スキャッターなどでメッセージサイズが小さい場合に、クラスタ内での一つのノードに一旦メッセージを集約し、クラスタ間では一つのメッセージとして通信を行なっている。既定値では 1KB 未満のサイズの場合にこのアルゴリズムが選択される。これはクラスタ内の通信オーバーヘッドと通信遅延が小さい場合には有効であるかもしれないが、今回のケースでは集約するオーバヘッドのほうが大きく性能が低下していると考えられる。

1KB 以下での性能低下の原因がこのアルゴリズムにあることを明らかにするため、IMPI 上で YAMPII と同じアルゴリズムを使用した場合との比較を行なった。**図 5** にその結果を示す。「original」が IMPI のアルゴリズムを用いた場合、「simple」が単純な YAMPII のアルゴリズムを用いた場合である。図から IMPI のアルゴリズムによって性能が低下していることが分かる。今回評価に用いたような環境では、ネットワークの輻輳などを考慮しなければ、集約するよりも直接送信する方が速いといえる。

3.5 ブロードキャスト・バンド幅

図 6 にブロードキャスト通信 (**MPI_Bcast**) のバンド幅を示す。バンド幅の値は、送信バイト数とプロセス数の積 (必要最小の通信量) を時間で割った値を使用した。

IMPI は、遅延がある場合に少し性能が落ちるが、ほぼ一定の性能を示した。リンク数が 1 と 4 の場合で違いがない。IMPI のブロードキャスト通信のアルゴリズムは、最初のステップでクラスタ間の通信を行ない、その後はクラスタ間の通信を行なわない。そのため、リンク数による違いが発生しない。

一方、YAMPII の性能は、クラスタ間のリンクの状態に大きく影響を受ける。YAMPII では、binominal-tree アルゴリズムを使用している。このアルゴリズムはネットワークの構成を考慮しておらず、最後のステップでは 8 プロセスが一斉にクラスタ間のリンクを経由する通信を行なう。このため、リンクの状態に大きく影響を受ける結果となっている。

3.6 リダクション・バンド幅

図 7 にリダクション通信 (**MPI_Reduce**) のバンド幅を示す。バンド幅の値は、ブロードキャスト同様、送信バイト数とプロセス数の積 (必要最小の通信量) を時間で割った値を使用した。

遅延がある場合に IMPI の性能が低いことを除けば、ほぼすべてが同様の性能を示している。YAMPII の用いるアルゴリズムはバイナリ・ツリーに従って通信を行なうもである。アルゴリズムはブロードキャストと似ているが結果は異なっている。実際には、i 番目のステップでプロセス番号が 2^i 違うもの同士でリダクションを行なっている。そのため、クラスタ間をまたがるのは最後のステップだけであり、これは IMPI のアルゴリズムと一致している。

4. 議　　論

GridMPI のプロトコルとして IMPI プロトコルを実装し、その性能評価として一対一通信バンド幅、コレクティブ通信バンド幅、アプリケーション・ベンチマークの計測を行なった。IMPI はそのプロトコル設計から、MPP 計算機等を LAN で接続する状況を想定していると考えられる。そのようなプロトコルをグリッド環境で用いるには、いくつかの明らかに改良すべき点がある。しかし逆に、オーバーヘッドが大きいはずの IMPI プロトコルがクラスタ用の MPI の性能を上回る場合もあった。このことから、クラスタ用 MPI である YAMPII にも改善点が考えられる。

- IMPI プロトコルは遅延があるネットワークでは、

大きいメッセージサイズ時に行なうランデブ同期、ネットワークリンクを公平に共有するためのフロー制御に性能上の問題がある。

- グリッド環境では広域網として 10Gbps といったバンド幅が大きいネットワークも利用可能であるが、IMPI のコレクティブ通信アルゴリズムではクラスタ間に大きなバンド幅があっても利用できない場合がある。
- 全対全通信等でクラスタ内でメッセージを集約する通信は、条件によっては性能低下をまねく。クラスタ計算機における通信オーバーヘッドでは、メッセージの集約が一般に性能上の問題になる。
- IMPI のコレクティブ通信はアルゴリズムが定義済で、新たなアルゴリズムを導入することができない。できるのはアルゴリズムの切替えポイントを決める、簡単なパラメータの調整だけである。アルゴリズムを追加選択するための機構が必要である。
- クラスタ計算機であってもネットワーク構成によってはボトルネック・リンクが存在する。その場合、IMPI の定義するように、クラスタを複数クラスタに分かれているものとして扱うコレクティブ通信アルゴリズムの方が性能が良い。

5. ま と め

GridMPI のグリッド向けプロトコルとして IMPI プロトコルを実装し、その性能評価を通して問題点を議論した。今後は、コレクティブ通信のアルゴリズム群を豊富にサポートすること、TCP/IP の輻輳制御等を考慮に入れた通信制御が課題である。実験では TCP/IP の輻輳制御により性能が大きく左右され、リダクションやアプリケーション・ベンチマークの IS や FT で性能が大きく低下していることが観測された。今後は、コレクティブ通信と TCP/IP の動作を協調させ双方の挙動に整合性をとることでグリッド向け高性能 MPI 実装として性能向上をさせていくことを考えている。

謝　　辞

なお、本研究の一部は文部科学省「経済活性化のための重点技術開発プロジェクト」の一環として実施している超高速コンピュータ網形成プロジェクト（NAREGI：National Research Grid Initiative）による。

参 考 文 献

1) D. Bailey, T. Harris, W. Saphir, R. van der Wijngaart, A. Woo, and M. Yarrow. The NAS Parallel Benchmarks 2.0. International Journal of Supercomputer Applications, 1995. http://www.nas.nasa.gov/Software/NPB
2) W. L. George, J. G. Hagedorn, and J. E. Devaney. IMPI: Making MPI Interoperable. Journal of Research of the National Institute of Standards and Technology, Vol.105, N.3, May 2000.
3) GridMPI. http://www.gridmpi.org
4) Hewlett-Packard. *HP MPI User's Guide, Eighth Edition*. 2003.
5) T. Imamura, Y. Tsujita, H. Koide, and H. Takemiya. An Architecture of Stampi: MPI Library on a Cluster of Parallel Computers. Proc. of EuroPVM/MPI 2000, LNCS1908, pp.200–207, 2000.
6) 石川. YAMPII もう一つの MPI 実装. 情報処理学会, SWoPP'04, 2004.
7) 石川, 松田, 工藤, 手塚, 関口. GridMPI – 通信遅延を考慮した MPI 通信ライブラリの設計. 情報処理学会, SWoPP'03, 2002-HPC-95-17, 2003.
8) N. T. Karonis, B. R. de Supinski, I. Foster, W. Gropp, E. Lusk, and J. Bresnahan. Exploiting Hierarchy in Parallel Computer Networks to Optimize Collective Operation Performance. Proc. of the 14th International Parallel Distributed Processing Symposium (IPDPS '00), 2000.
9) T. Kielmann, R. F. H. Hofman, H. E. Bal, A. Plaat, and R. A. F. Bhoedjang. MagPIe: MPI's Collective Communication Operations for Clustered Wide Area Systems. Proc. of the 7th ACM SIGPLAN Symp. on Principles and Practice of Parallel Programming (PPoPP'99), 1999.
10) 児玉, 工藤, 佐藤, 関口. ハードウェアネットワークエミュレータを用いた TCP/IP 通信の評価. 情報処理学会, SWoPP'03, 2003.
11) LAM/MPI. http://www.lam-mpi.org
12) M. Matsuda, Y. Ishikawa, and T. Kudoh. Evaluation of MPI Implementations on Grid-connected Clusters using an Emulated WAN Environment. CCGrid2003, 2003.
13) MPI Software Technology, Inc. http://www.mpi-softtech.com
14) PC Cluster Consortium. http://www.pccluster.org
15) 高野, 工藤, 児玉, 松田, 手塚, 石川. GridMPI のための TCP/IP 輻輳制御実装方式の検討. 情報処理学会, SWoPP'04, 2004.
16) YAMPII, Yet Another MPI Implementation. http://www.ilab.is.s.u-tokyo.ac.jp/yampii
17) PACX MPI. http://www.hlrs.de/organization/pds/projects/pacx-mpi