情報処理学会論文誌　コンピューティングシステム　Vol. 2　No. 1　133–145 (Mar. 2009)

プライベートアドレスを有するクラスタ群のための高性能MPI通信リレー機構

高野了成[†1]　松田元彦[†2]　工藤知宏[†1]
児玉祐悦[†1]　岡崎史裕[†1]　石川裕[†1,†2]

本論文では，プライベートアドレスを有する複数のクラスタ群を接続するための，高性能 MPI 通信リレー機構を提案する．本機構の目的は，(1) プライベートアドレスクラスタ上の MPI プロセスが IMPI プロトコルを介して外部のクラスタと通信可能になること，および (2) 複数組のフロントエンドノード間の接続を束ねることで，1 台のフロントエンドノードのインタフェース速度に制約されない，クラスタ間通信を実現することである．提案機構を実装し，GridMPI と LAM/MPI という異なる IMPI 実装間における相互接続を確認した．さらに 10 Gbps の物理帯域を有する広域ネットワークを模擬した実験環境を用いて評価を行った．その結果，大半の NAS 並列ベンチマークはトランク数の増加に対して性能が向上すること，伝送遅延の増加に従って，リレー機構のオーバヘッドが相対的に減少することを確認した．

High Performance MPI Communication Relay Mechanism on Multiple Private Address Clusters

Ryousei Takano,[†1] Motohiko Matsuda,[†2]
Tomohiro Kudoh,[†1] Yuetsu Kodama,[†1]
Fumihiro Okazaki[†1] and Yutaka Ishikawa[†1,†2]

In this paper, we propose a high performance message relay mechanism to support multiple private address clusters. The aims are (1) MPI processes running on private address clusters can communicate with any IMPI standard-based MPI implementation on external clusters, (2) multiple pairs of front-end nodes simultaneously communicate between clusters to achieve inter-cluster communication without being limited to the interface speed of a front-end node. We present the proposed mechanism achieved the interoperability of two IMPI implementations: GridMPI and LAM/MPI. We also confirmed the effectiveness of the proposed mechanism by experiments using a 10 Gbps emulated WAN environment. The performance of most of the NAS Parallel Benchmarks improved as the number of trunks increases. The overhead of the relay mechanism relatively decreases as the propagation delay increases.

1. はじめに

近年の広域ネットワーク技術の急速な発展により，地域的に離れたコンピューティング資源を利用した大規模並列アプリケーションの実行が実用になりつつある．このような用途に適した MPI（Message Passing Interface）プログラム実行環境を提供することを目指して，MPICH-G2[1)]，PACX-MPI[2)]，StaMPI[3)]，MC-MPI[4)] に代表される Grid-enabled MPI システムが提案されている．

従来のクラスタ向け MPI 実装では，すべての計算ノードが相互に直接通信できることを前提としている．しかし，グリッド環境では，ジョブ実行ごとに複数の組織をまたがる仮想組織（VO: Virtual Organization）が構築される．一部のクラスタはプライベートアドレスやファイアウォール下で運用される可能性があるので，上記の前提を満たせるとは限らない．アカデミアに限らず，ビジネス用途に対するグリッド利用が進展すれば，セキュリティや運用性の面からプライベートアドレスの問題は避けて通れない．このような状況に鑑みると，プライベートアドレスを有するクラスタをグリッド環境へ結合し，従来利用できなかった資源をも利用可能にすることは，Grid-enabled MPI システムに対する大きな課題の 1 つである．その際にはさまざまなネットワーク環境下での実行，およびクラスタ間帯域の効率的な利用を考慮する必要がある．

プライベートアドレスを有する計算機間の通信を実現する一般的な手法として，ユーザレベルプロキシ，NAT トラバーサル（Network Address Translation traversal），VPN（Virtual Private Network）などが提案されている．プライベートアドレスに対応した MPI システムはいくつか存在するが[2)–7)]，これらの手法の少なくとも 1 つを用いている．各手法には次に述べる利点と欠点がある．ユーザレベルプロキシは移植性が高く，実装も容易である．さらに広域ネットワークに対するパラメータ（たとえばソケットバッファサイズ，パケット

†1 産業技術総合研究所情報技術研究部門
Information Technology Research Institute, National Institute of Advanced Industrial Science and Technology (AIST)
†2 東京大学大学院情報理工学系研究科
Graduate School of Information Science and Technology, The University of Tokyo

© 2009 Information Processing Society of Japan

フィルタリングのルール）の設定がフロントエンドノードに集約できるという利点もある．しかし，プロキシプロセスとカーネル間でパケットをコピーする必要があるので，他の手法と比較して通信性能は低い．TCP NAT トラバーサルの性能は高いが，仕様が標準化されていないので，NAT 機器のアルゴリズムによっては利用できない．VPN は，プライベートネットワーク間をトンネリングによって接続する技術であるが，通信に参加するすべてのノードに対して重複しないプライベートアドレスを設定する必要があるという制限がある．

ここで国内学術系機関における代表的な大型クラスタと対外接続回線を取り巻く現状を考察する．T2K オープンスパコン[8] の共通仕様では，クラスタ内のノード間帯域は 40 Gbps 以上と規定されている．このノード間ネットワークは，MPI 通信あるいはファイルサーバへのアクセスに利用されるが，TCP/IP を用いた対外接続用には別途ギガビットイーサネットを有している．一方で，SINET3 や JGN2plus といったネットワークは 10 Gbps での接続環境を提供している．そこで本論文では，クラスタ間帯域（たとえば 10 Gbps）は，計算ノードにおける対外接続用インタフェース帯域（たとえば 1 Gbps）よりも広く，かつ，全計算ノードの対外接続帯域の合計は，クラスタ間帯域よりも広いと仮定する．この仮定下で，1 台のフロントエンドノードがクラスタ間の通信を中継する場合，クラスタ間帯域はフロントエンドノードのインタフェース速度によって制限される．したがって，フロントエンドノードが性能のボトルネックとなり，上記の例ではクラスタ間帯域の利用率は 1/10 に制限される．クラスタ間帯域を効率良く利用するには，クラスタごとに複数のフロントエンドノードが並列に動作する必要がある．

我々は，グリッド環境で MPI アプリケーションを高性能に実行するため，通信遅延を考慮した通信および異機種間における高い相互運用性を実現する GridMPI 通信ライブラリ[9] を開発している．GridMPI は MPI-1.2 規格および MPI-2.0 規格に準拠したグリッド環境向け MPI 通信ライブラリである．GridMPI は東京大学で開発されたクラスタ向け MPI システムである YAMPI[10] をベースシステムとし，クラスタ間通信に IMPI（Interoperable MPI）プロトコル[11] を採用している．IMPI は異なるアーキテクチャやベンダ提供の MPI 通信ライブラリとの相互接続性を実現するための標準仕様であるが，プライベートアドレスへの対応については言及されていない．

本論文では，プライベートアドレスを有するクラスタ群におけるクラスタ間通信を中継するユーザレベルプロキシである IMPI リレー[12] について述べる．IMPI リレーは，IMPI 規格におけるホストとプロセスの対応管理の仕組みを活用することで，IMPI 規格に準拠したメッセージ中継を実現する．また，NAT 機器の挙動やプライベートアドレスの重複に制限されないので，幅広い実行環境で利用できる．さらに，ユーザレベルプロキシにおける性能問題を解決するために，複数組の IMPI リレーの接続を束ねて集約することで，クラスタ間帯域の利用効率の向上を図るトランキング機能を提供する．

以下，2 章で IMPI リレーの概要とトランキング機能の設計について述べる．3 章で広域ネットワークを模擬した実験環境上で提案機構の効果を評価する．続いて，得られた実験結果に対する考察を 4 章で行う．5 章で関連研究について簡単に言及する．最後に 6 章でまとめを行う．

2. 設　　計

2.1 用語の定義

本論文で使用する用語を次に定義する．これらの定義は IMPI 規格[11] に基づいている．

- **プロセス**は，MPI プロセスと等価であり，`Host ID` と `Proc ID` の組によって識別される．プロセスは必ず 1 つの**ホスト**に属し，ホストは 1 つもしくは複数のプロセスを有する．プロセスが動作する計算機は，ホストと物理的に同じ計算機である必要はない．
- **クライアント**は，クラスタと等価であり，`Client ID` により識別される．1 つのアプリケーションは複数のクライアントで実行される可能性がある．また，クライアントは 1 つ以上のホストを有する．
- **サーバ**は，すべてのクライアントから接続可能な計算機上で動作し，クライアント間で実行に参加するホストやプロセスの数やアドレス，フロー制御などの各種パラメータなどの情報を交換するための接続点を提供する．アプリケーションごとに必ず 1 つ存在する．
- メッセージは**パケット**に分割され送信される．すべてのパケットには送受信先のプロセスやデータ長などを含むヘッダを有する．

以下，`Host ID` が h で `Proc ID` が p のプロセス識別子を $h{:}p$ と記す．さらに，すべてのホストがプライベートアドレスだけを有するクラスタを，プライベートアドレスクラスタあるいは PC と記す．また，すべてのホストがグローバルアドレスを有するクラスタを，グローバルアドレスクラスタあるいは GC と記す．

2.2 基 本 設 計

IMPI リレーは，クラスタのフロントエンドノード上で動作するユーザレベルプロキシである．なお，フロントエンドノードは，グローバルアドレスおよびプライベートアドレスを有する．

© 2009 Information Processing Society of Japan

IMPI リレーは，IMPI 規格に準拠しつつ，プライベートアドレスクラスタとその外部のクラスタ間で授受されるメッセージを中継するために，ゲートウェイホストとして動作する．ゲートウェイホストは，クライアント内部から，仮想的に外部のクライアント内のすべてのプロセスを有するホストとして見える．これをプライベートゲートウェイホストと呼ぶ．プライベートゲートウェイホストは外部のクライアントと同数存在する．一方，外部のクライアントからは，自らの属するクライアント内のすべてのプロセスを有するホストとして見える．これをグローバルゲートウェイホストと呼ぶ．各ゲートウェイホストに送信されたメッセージは，本来の宛先であるホストに転送される．

上記の仕組みを実現するには，(1) メッセージの中継に用いる経路テーブルの作成，(2) 経路テーブルに基づいたパケット転送処理が必要となる．我々は IMPI プロトコルの枠組みをそのまま活用することで，MPI プロセス間のメッセージを中継できることに着目した．(1) については，初期化時に IMPI サーバとクライアント間で授受される MPI プロセス情報を取得し，ゲートウェイホストとプロセスのマッピングに合わせて書き換えることで，経路テーブルを作成する．このテーブルを Host-Proc ID テーブルと呼ぶ．(2) については，IMPI パケットには送受信先の `Host ID` と `Proc ID` が含まれており，IMPI リレーは Host-Proc ID テーブルを基に IMPI パケットを適切な宛先に転送する．

2.3 IMPI リレーの実行概要

IMPI リレーの実行概要を**図 1** に示す．クライアント 0 はプライベートアドレスクラスタ，クライアント 1 はグローバルアドレスクラスタである．各クライアントは 2 台のホストからなり，各ホストはそれぞれ 1 つのプロセスを有している．各プロセスには MPI レベルの識別子であるランク番号のほかに，IMPI レベルの識別子（*h:p*）が割り当てられている．IMPI リレー *R0* が提供するグローバルゲートウェイホストの `Host ID` を *R0G*，プライベートゲートウェイホストの `Host ID` を *R0P* と記す．

ホスト A0，A1 とホスト B0，B1 は直接通信できないが，ホスト A0，A1 と *R0P*，または，ホスト B0，B1 と *R0G* は相互に直接通信できる．IMPI リレーは，ホスト A0 と A1 に対して，ランク 2 を *R0P:0*，ランク 3 を *R0P:1* と対応付ける．すなわち，ランク 0，1 は，`MPI_COMM_WORLD` が 4 つのプロセス（*A0:0*，*A1:0*，*R0P:0*，*R0P:1*）から構成されると認識する．ランク 0 からランク 2 に対してパケットを送信する場合，ランク 0 はランク 2 の ID を *R0P:0* と認識しているので，*R0P* にパケットを送信する．IMPI リレーは，*R0P:0* 宛のパケットを受信すると，本来の宛先であるホスト *B0* に転送する．

一方，クライアント 1 に対しては，ランク 0，1 を *R0G* が有するプロセスに対応付け，

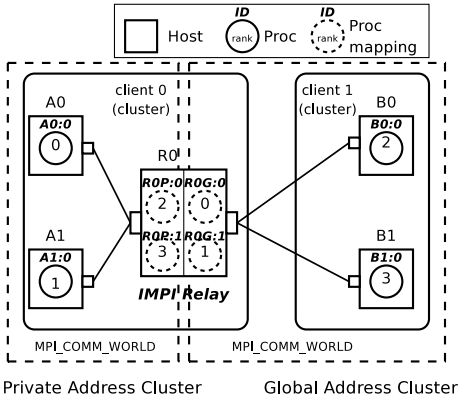


図 1　IMPI リレーの実行概要
Fig. 1　Overview of IMPI relay.

`MPI_COMM_WORLD` を（*R0G:0*，*R0G:1*，*B0:0*，*B1:0*）と認識させる．

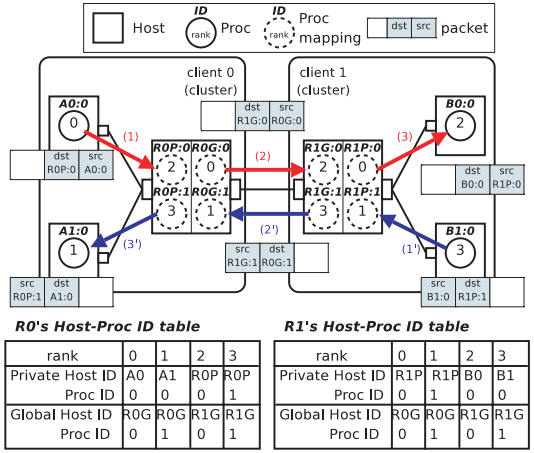
2.4 Host-Proc ID 変換とパケット転送

IMPI 規格では，初期化時に，すべてのホスト間で全対全接続することを規定している．そのために，IMPI サーバを介して，クライアントを構成するホスト，プロセスの数やアドレスなどの情報を収集し，各クライアント，さらに各ホストに分配する．プライベートアドレスクラスタでは，クライアントと IMPI サーバ間の通信もまた IMPI リレーによって中継される．この際，IMPI リレーは授受される情報に含まれる `Host ID`，`Proc ID` から，Host-Proc ID テーブルを生成し，それに基づいて IMPI リレーやクラスタ内のホストに対して中継する情報の内容を書き換える．

IMPI サーバが保持する Host-Proc ID テーブルは，クライアント間の通信に利用するグローバルマッピングと，クライアント内の通信に利用するプライベートマッピングの 2 つからなる．IMPI リレーは，自らが属するクライアント情報としてグローバルマッピングの内容を IMPI サーバに送信する．したがって，すべてのクライアントは，同じ内容のグローバルマッピングを保持する．一方，クラスタ内のホストには，プライベートマッピングが分配される．プライベートマッピングの内容はクライアントごとに異なる．これは直接通信が可能な計算ノードのアドレスや数がクライアントごとに異なるからである．

初期化処理後，IMPI リレーによるパケット転送の流れを**図 2** に示す．ここでは，クライアント 0，1 ともに，プライベートアドレスクラスタである．クライアント 0 側の IMPI リ

© 2009 Information Processing Society of Japan



rank	0	1	2	3
Private Host ID	A0	A1	R0P	R0P
Proc ID	0	0	0	1
Global Host ID	R0G	R0G	R1G	R1G
Proc ID	0	1	0	1

rank	0	1	2	3
Private Host ID	R1P	R1P	B0	B1
Proc ID	0	1	0	0
Global Host ID	R0G	R0G	R1G	R1G
Proc ID	0	1	0	1

図 2　パケット転送と Host-Proc ID 変換（矢印はランク 0 から 2 または 3 から 1 の通信を示す）
Fig. 2　Packet forwarding and Host-Proc ID translation. Ranks 0 and 3 send a packet to ranks 2 and 1, respectively.

レー *R0* は，クライアント 0 に対して，ランク 2 を *R0P:0*，ランク 3 を *R0P:1* に対応付ける．一方，クライアント 1 に対して，ランク 0 を *R0G:0*，ランク 1 を *R0G:1* に対応付ける．クライアント 1 側の IMPI リレー *R1* の動作も同様である．

ここでランク 0 がランク 2 宛にパケットを送信する場合を考える．(1) ランク 0 はパケットの送信元を *A0:0*，送信先を *R0P:0* として *R0P* に送信する．(2) *R0* はプライベートマッピングをグローバルマッピングに変換するために，Host-Proc ID テーブルを参照し，送信元を *R0G:0*，送信先を *R1G:0* と書き換えて *R1* に転送する．続いて，(3) *R1* はグローバルマッピングをプライベートマッピングに変換するために，Host-Proc ID テーブルを参照し，送信元を *R1P:0*，送信先を *B0:0* と書き換えて *B0* に転送する．同様に，ランク 3 からランク 1 にパケットを送信する場合は，*B1* → *R1* → *R0* → *A1* の経路でパケットが転送される．

2.5　トランキング機能

2.5.1　概　　要

ここまでの機能では，その性能がフロントエンドノードのインタフェース速度に制限されるので，複数組の IMPI リレー間接続を集約し，クラスタ間通信性能を向上させるトランキング機能を実現する．この際，各クラスタごとに複数のフロントエンドノード上で IMPI

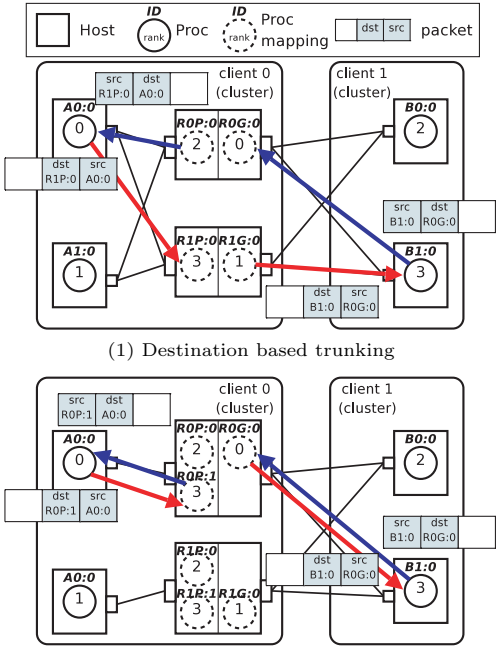


図 3　トランキング使用時のパケット転送方式（矢印はランク 0 と 3 間の通信を示す）
Fig. 3　Packet forwarding schemes with trunking. Ranks 0 and 3 communicate with each other.

リレーが実行され，それらが協調して動作する．なお，一部の IMPI リレーが異常終了やネットワーク障害によって処理を継続できない場合に，他の IMPI リレーが処理を代行して縮退動作するようなフォールトトレラント機能は議論の対象外とする．

トランキング機能は Host-Proc ID 変換の単純な拡張として実現できる．本論文では，**図 3** に示す 2 種類の方式について検討する．プライベートアドレスクラスタであるクライアント 0 には，2 台のフロントエンドノードが存在し，それぞれで IMPI リレーが動作している．クライアントごとに起動される IMPI リレーの数をトランク数と呼ぶ．トランク数はクライアントごとに異なる可能性がある．各 IMPI リレーに均等に通信が発生する場合には，トランク数に比例してクラスタ間の利用可能帯域が増加する．たとえば，図 3 において，ランク 0 と 2 および 1 と 3 の組が同時に通信を行った場合，クラスタ間帯域はトラン

© 2009 Information Processing Society of Japan

ク数 1 の 2 倍になる．

両方式は，グローバルゲートウェイホストのマッピングに関しては違いがない．プライベートゲートウェイホストのマッピング，すなわち，クラスタ内部から IMPI リレーを介してクラスタ外部と通信する観点から見ると，送信先ベース・トランキング方式では，メッセージの送信先を基に転送先の IMPI リレーが決まる．一方，送信元ベース・トランキング方式では，送信元ホストを基に転送先の IMPI リレーが決まる．

送信先ベース・トランキング方式は，クラスタ内およびクラスタ間通信はともに IMPI 規格に準拠している．しかし，送信元ベース・トランキング方式では，クラスタ間通信は IMPI 規格に準拠するが，クラスタ内通信においてプロトコルの拡張が必要となる．現在は，プロトコルの拡張が不要であり，かつ実装が容易であるという理由から送信先ベース・トランキング方式を採用している．

2.5.2　送信先ベース・トランキング方式

送信先ベース・トランキング方式（図 3 上）では，プライベートゲートウェイホスト（*R0P*, *R1P*）とクラスタ内部の 2 ホスト（*A0*, *A1*）が相互に接続を確立する．そして，クライアント 0 に対しては，ランク 2 を *R0P:0*，ランク 3 を *R1P:0* に対応付ける．クライアント 1 に対しては，ランク 0 を *R0G:0*，ランク 1 を *R1G:0* に対応付ける．しかし，この方式には往路と復路の経路が非対称になるという問題ある．たとえば，図中の矢印で示すようにランク 0 と 3 の往復経路は，$A0 \rightarrow R1 \rightarrow B1 \rightarrow R0 \rightarrow A0$ となる．これは直感的でないほかにも，次に述べる問題がある．ランク 0 はランク 3 を *R1P:0* だと認識しているが，送信元 ID が *R1P:0* のパケットを，ホスト *R0P* から受信することになる．これは送信元 ID が同一であったとしても，複数の TCP コネクションからパケットを受信する可能性があることを意味する．IMPI の規格には反しないが，実装方法によっては正しく動作しない可能性があるので，IMPI 通信ライブラリはこの点を考慮して実装する必要がある．

2.5.3　送信元ベース・トランキング方式

送信元ベース・トランキング方式（図 3 下）の往復経路は対称である．さらに，プライベートゲートウェイホストとクラスタ内部のホスト間の接続数を送信先ベース・トランキング方式よりも削減できる．グローバルゲートウェイホストのマッピングは方法 1 と同じである．しかし，プライベートゲートウェイホストのマッピングが異なる．本方式の場合は，各 IMPI リレーがすべてのリモートプロセス（ランク 2，3）への通信を中継する可能性があるので，ホスト A0，A1 ごとに，異なるプライベートマッピングを提供する必要がある．つまり，ホスト A0 に対しては *R0P* が，ホスト A1 に対しては *R1P* が，すべてのリモートプロセスを有するマッピングを提供する．しかし，IMPI 規格では，クライアント内は同じマッピングを共有することを仮定しており，ホストごとに異なるマッピングを提供できない．したがって，IMPI リレーごとに異なるマッピングを提供するためには，クラスタ内通信に関して，プロトコルの大幅な拡張を必要とする．

3. 実　　験

3.1 実験環境

使用した実験環境を**図 4** に示す．**表 1** にクラスタを構成するノード PC とスイッチの諸元を示す．各クラスタは 16 台の計算ノードと 8 台のフロントエンドノードから構成される．計算ノードはギガビットイーサネットと Myrinet-2000 インタフェースを持つ．ギガビットイーサネットはクラスタ間通信に，Myrinet-2000（MX 1.6）はクラスタ内通信に使用した．

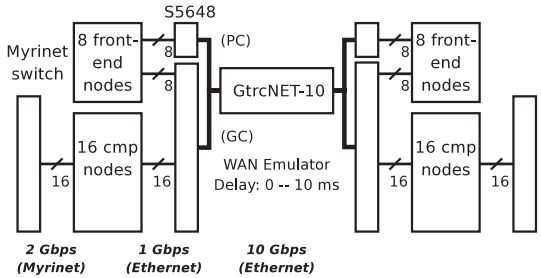


図 4　実験環境
Fig. 4　Experimental setting.

表 1　PC クラスタの諸元
Table 1　PC cluster specifications.

Node PC	
CPU	Opteron/2.0 GHz dual
Memory	6 GB DDR333
Ethernet	Broadcom BCM5704
Myrinet	Myricom M3F-PCIXD-2
OS	SuSE Enterprise Server 9
Compiler	Intel Compiler 9.1
Switch	
Huawei-3Com S5648 + optional 10 Gbps port	
Myricom M3-SW16-8F + M3-SPINE-8F	

© 2009 Information Processing Society of Japan

フロントエンドノードは 2 つのギガビットイーサネットインタフェースを持ち，片方にプライベートアドレスを，もう片方にグローバルアドレスを割り当てた．各クラスタに 1 台のイーサネットスイッチ（Huawei-3Com S5648）を配置し，VLAN 機能によってプライベートアドレス用とグローバルアドレス用の 2 つのネットワークに分割した．

2 クラスタを GtrcNET-10[13] を介して接続し，10 Gbps の物理帯域を持つ広域ネットワークを模擬した．GtrcNET-10 は，10 ギガビットイーサネット XENPAK と 1 ギガバイト DDR-SDRAM を各 3 ポートずつ，大規模 FPGA（Xilinx XC2VP100）に接続した構成になっている．3.4 節の実験では，クラスタ間の片道伝送遅延を 0 ミリ秒から 10 ミリ秒に模擬した．往復遅延はその 2 倍になる．

実験は，2 クラスタがグローバルアドレスクラスタ（GC-GC），片方だけがプライベートアドレスクラスタ（PC-GC），2 クラスタがプライベートアドレスクラスタ（PC-PC）という，3 パターンの設定で行った．PC 設定時はフロントエンドノード上の IMPI リレーを経由し（図 4 中の経路 PC），GC 設定時はフロントエンドノードを経由せずに，対向クラスタと接続する（図 4 中の経路 GC）．

Linux カーネル 2.6.23 を使用し，いくつかの TCP 関連パラメータを**表 2** に示すように変更した．GridMPI および IMPI リレーのソケットバッファを 2.5 MB に拡大した．これはソケットバッファサイズなどのパラメータがデフォルト値では，帯域遅延積の大きなネットワークで十分な性能が得られないからである．さらに，実行ごとの性能のばらつきを抑えるために，tcp_no_metrics_save を有効にし，同一ルートでの通信メトリックスの再利用を禁止した．

3.2　相互運用性の検証

IMPI リレーにおける相互運用性を確認するために，GridMPI 以外の IMPI 実装である LAM/MPI 7.1.4 と相互接続が可能であるか実験した．クライアント 0 はプライベートアドレスクラスタとし，IMPI リレーを介して GridMPI を実行した．クライアント 1 はグローバルクラスタとし，LAM/MPI を実行した．その結果，1 対 1 通信および集団通信の `MPI_Bcast` が正常動作することを確認した．しかし，`MPI_Reduce` では，LAM/MPI のプロセスが異常終了し，実行に失敗した．LAM/MPI の IMPI 実装[14] はほとんどの集団通信をサポートしていないなど不完全な実装であり，十分な試験を実施できなかったが，集団通信は 1 対 1 通信の組合せで実装されるので，最低限の相互運用性は確認できたと考える．

表 2　ネットワークパラメータ
Table 2　Network parameters.

sysctl parameters	
net.core.wmem_max	2500000
net.core.rmem_max	2500000
net.ipv4.tcp_rmem	2500000 2500000 2500000
net.ipv4.tcp_wmem	2500000 2500000 2500000
net.ipv4.tcp_no_metrics_save	1
Environment variables for GridMPI	
IMPI_SOCBUF	2500000
IMPI_RELAY_SOCBUF	2500000

3.3　マイクロベンチマーク

3.3.1　通信遅延時間

IMPI リレーを経由することで増加する片道遅延時間を測定した．実験では，ペイロードサイズが 0 バイトのメッセージを送受する *ping-pong* プログラムを用いて，2 クラスタ間の往復遅延時間を測定し，その結果を 1/2 することで求めた．クラスタ設定が GC-GC で 29.8 マイクロ秒，PC-GC で 54.5 マイクロ秒，PC-PC で 78.9 マイクロ秒という結果を得た．これより，IMPI リレーを経由するごとに片道遅延時間が約 25 マイクロ秒増加することが分かる．

3.3.2　1 対 1 通信性能

2 クラスタ間の 1 対 1 通信帯域の測定結果を**図 5** に示す．メッセージサイズが小さい場合は，IMPI リレーのオーバヘッドによる通信遅延の増加が原因で，GC-GC，PC-GC，PC-PC の順に帯域が低下する．メッセージサイズが 4 KB 以上になると，オーバヘッドは

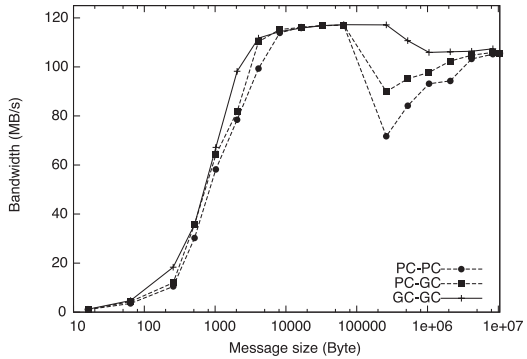


図 5　2 クラスタ間の 1 対 1 通信帯域
Fig. 5　Point-to-point communication bandwidth between two clusters.

© 2009 Information Processing Society of Japan

相対的に小さくなるので，すべての結果はほぼ同じになる．メッセージサイズが 64 KB を超える 256 KB の時点から帯域の落ち込みが見られる．これはメッセージサイズが IMPI パケットの最大サイズ（IMPI_C_DATALEN）を超えると，通信プロトコルが *eager* プロトコルから *rendezvous* プロトコルへ切り替わるからである．この閾値のデフォルト値は 64 KB である．*rendezvous* プロトコルでは，メッセージごとに通信を同期するので，通信遅延が隠蔽できなくなり，性能が低下する．この影響はより遅延の大きい IMPI リレー使用時の方が大きい．また，メッセージサイズの増加に従い，IMPI リレーのオーバヘッドは相対的に減少する．得られた結果については，4.3 節でより詳細に議論する．

3.3.3　全対全通信性能

2 クラスタ間における全対全通信の実行時間を，GC-GC 設定，すなわち IMPI リレー未使用時，および PC-PC 設定でトランク数 1，2，4，8 の各条件で測定した．各クラスタごとに 16 プロセス，合計 32 プロセスを使用した．全対全通信は全プロセスから全プロセスへデータを交換する集団通信であり，その性能は 2 分割帯域（bisection bandwidth）に大きく依存する．

結果を**図 6** に示す．メッセージサイズが 1024 バイトより小さい場合は，トランキング機能による性能向上は見られない．これは IMPI 規格で，メッセージサイズが小さい場合の集団通信には，*short* アルゴリズムを用いるよう定義されているからである．*short* アルゴリズムは，まず送信側のクラスタ内で各ノードのデータをマスタノードに集めて，対向クラスタのマスタノードに送信する．続いて受信側のクラスタではマスタノードからクラスタ内の各ノードにデータが分配される．このようにクラスタ間通信性能は，マスタノード間の通信性能に制限されるので，トランク数を増やしても効果は得られない．

メッセージサイズが 1024 バイト以上の場合では，トランク数が 2 倍になるごとに，実行時間が 2 倍高速になる．しかし，メッセージサイズが 4096 バイトから 512 KB の範囲において，直感的ではない現象を観測した．特にメッセージサイズが 4096 バイトから 32 KB の間では，IMPI リレー未使用の性能が最も悪い結果となった．この原因は経路途中の輻輳と考えられる．Linux カーネルのネットワーク統計情報を調査したところ，パケットロス数の増加は，メッセージサイズの増加に比例している．一方，TCP 再送タイムアウト（RTO: Retransmission TimeOut）の発生数は，メッセージサイズが 4096 バイトのときに 21049 と，1024 バイト，512 KB のときの値 195，2303 と比べて大きく増加していることが分かった．つまり，RTO の発生が，性能劣化に大きな影響を及ぼしている．問題の詳細については 4.2 節で議論する．

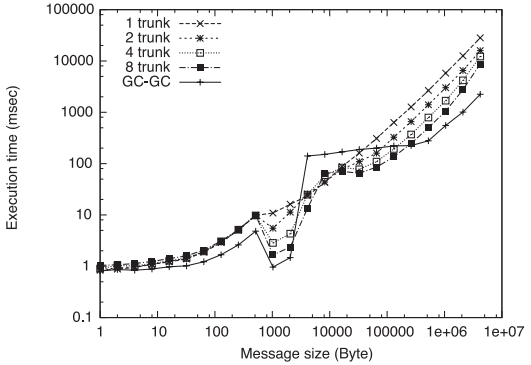


図 6　2 クラスタ（32 プロセス）間の全対全通信実行時間
Fig. 6　*All-to-all* communication execution time with 32 processes (16 processes for each cluster).

3.4　NAS 並列ベンチマーク

IMPI リレーのトランキング機能の効果を NAS 並列ベンチマーク 3.2 を用いて確認した．問題サイズはクラス B を使用した．プロセス数は CG，EP，FT，IS，LU，MG ベンチマークで 32 プロセス，BT，SP ベンチマークで 25 プロセスとした．後者の場合，計算ノード数を 13 と 12 に分けて各クラスタに配置した．各ベンチマークを片道伝送遅延を 0 ミリ秒から 10 ミリ秒まで 2 ミリ秒刻みで変えながら，各々 3 回実行し，最も良い結果を用いた．また，高遅延環境下における，*rendezvous* プロトコルによる性能低下を避けるために，IMPI_C_DATALEN を十分に大きな値（21474836 = 0x7fffffff）に設定し，プロトコルの切替えを回避した．

0 ミリ秒遅延時における結果を**表 3** に示す．トランク数 1，2，4，8，IMPI リレー未使用（GC-GC）の各条件で測定した．単位は Mop/s（Million operations per second）であり，括弧の中はトランク数 1 の結果に正規化した相対性能である．また，クラスタを複数化したときのスケーラビリティを評価するために，単独クラスタ（GC）での結果を併記する．トランク数 1 の場合には，2 つのクラスタを結合することで GC の性能を超えたのは，通信量が少ない EP と LU だけであった．しかし，トランキング機能を用いることで，IS と MG を除くと，性能が 50%から 100%向上した．IS はクラスタ間接続がボトルネックとなるために大きく性能を落とし，8 トランクの場合と比較しても，性能が 38%低下した．一方，MG では L2 キャッシュミスの影響から，160%と super linear な性能向上を示した．この結果は，一般的には複数のクラスタの結合によって性能が向上することを示している．ま

© 2009 Information Processing Society of Japan

表 3　NPB ベンチマークの性能（Mop/s total）
Table 3　Results of NPB benchmarks with 0 millisecond of delay (Mop/s total).

	CG	EP	FT	IS	LU	MG
1 trunk	1400.4 (1.0)	619.2 (1.0)	1493.2 (1.0)	46.2 (1.0)	18035.2 (1.0)	5755.7 (1.0)
2 trunk	2287.4 (1.6)	616.4 (1.0)	2860.0 (1.9)	82.9 (1.8)	18792.8 (1.0)	8776.1 (1.5)
4 trunk	2660.3 (1.9)	612.6 (1.0)	4575.3 (3.1)	120.4 (2.6)	19177.5 (1.1)	10781.2 (1.9)
8 trunk	2978.8 (2.1)	616.6 (1.0)	6585.6 (4.4)	159.1 (3.4)	19657.3 (1.1)	11884.4 (2.1)
GC-GC	3998.2 (2.9)	615.3 (1.0)	8205.0 (5.5)	73.8 (1.6)	20317.6 (1.1)	14523.6 (2.5)
GC	2273.6 (1.6)	307.7 (0.5)	4937.7 (3.3)	256.6 (6.1)	12081.6 (0.7)	5148.3 (1.2)

	BT	SP
1 trunk	12498.9 (1.0)	4180.3 (1.0)
2 trunk	13658.7 (1.1)	5080.2 (1.2)
4 trunk	15917.7 (1.3)	6116.7 (1.5)
8 trunk	16873.5 (1.4)	6950.1 (1.7)
GC-GC	18468.4 (1.5)	7733.5 (1.8)
GC	12160.2 (1.0)	5148.3 (1.2)

た，利用可能なメモリ量が増加するので，より大きな問題サイズも実行可能となる．

表 3 の大半の結果では，トランク数の増加に対して性能が向上している．トランク数を 1 から 8 へ増やした場合の性能向上率に注目すると，FT と IS はそれぞれ 4.4 倍，3.4 倍高速化した．これらのベンチマークは他のベンチマークよりも広帯域を必要とするので，トランキング機能によって著しい性能向上が得らたからである．CG，MG，BT，SP はそれぞれ 2.1，2.1，1.4，1.7 倍高速化した．EP と LU はほとんど効果が見られなかった．これらのベンチマークが非常に小さいトラフィックしか生成せず，クラスタ間通信が性能のボトルネックではないからである．

さらに特筆すべきなのは，IS ベンチマークではトランク数が 2 以上の場合に，IMPI リレー未使用時の性能を上回っている点である．トランク数が 2，4，8 の場合，IMPI リレー未使用時と比較して，1.1 倍，1.6 倍，2.2 倍高速化している．この際の RTO 数は，59，170，493 であったのに対して，IMPI リレー未使用時では，1457 と明らかに多かった．IS ベンチマークは `MPI_Alltoallv` 関数を繰り返し実行するベンチマークであり，本実験では，各プロセスが持つ約 128 KB のデータを相互に交換する．`MPI_Alltoall` 関数と `MPI_Alltoallv` 関数の違いはあるものの，図 6 に示した結果と同様の傾向を示している．つまり，IMPI リレーを使用した場合，クラスタ間接続を介してメッセージを送信するノード数がトランク数に制限されるので，深刻な輻輳を回避したと考える．

しかし，トランク数の増加に従い，その効果が逓減している．たとえば，CG では，トランク数が 2 倍になるごとに，性能は 1.6 倍，1.2 倍，1.1 倍に増加している．これは各 IMPI リレーの転送トラフィック量に偏りが生じるので，性能向上が単純にトランク数に比例しないからである．

なお，遅延 0 ミリ秒の実験はイントラネット環境を模擬していると考えることができる．グリッド環境以外のイントラネット環境においても，複数のプライベートアドレスクラスタを結合する目的に対して，提案機構が有用であることを示している．

伝送遅延を 0 ミリ秒から 10 ミリ秒に設定したときの各ベンチマーク結果について，トランク数が 1 でかつ遅延 0 ミリ秒の結果に正規化した相対性能を**図 7** に示す．FT と IS は遅延よりも帯域の影響が大きく，遅延が増加してもトランキング機能の効果は高い．IS ではトランク数 8 の結果がつねに IMPI リレー未使用時を上回っている．一方，その他のベンチマークは，帯域よりも遅延の影響が大きいので，遅延の増加に従い，トランキング機能の効果も縮小する．

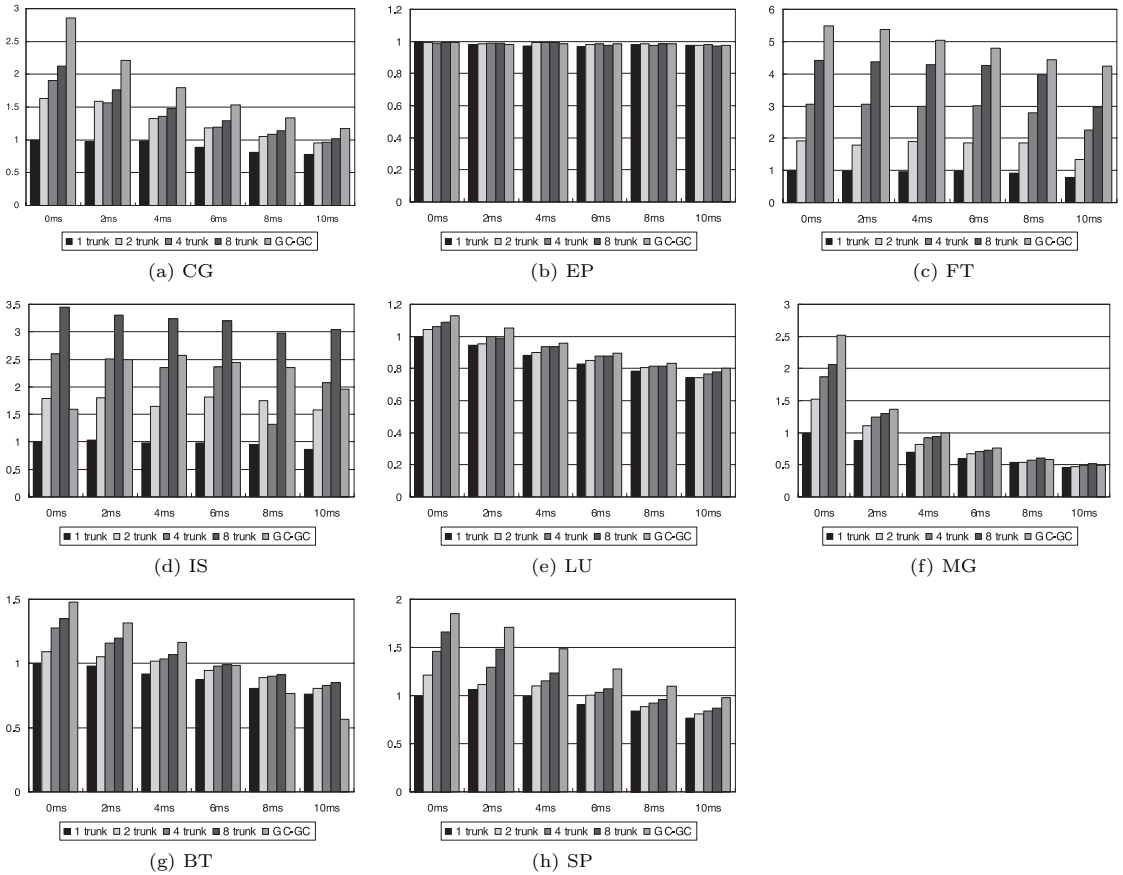
また，EP，LU，SP 以外のベンチマークでは，伝送遅延の増加に従い，IMPI リレーの有無に対する性能差が縮小する．なかでも BT と MG は，IMPI リレー使用時の方が性能が良くなり，結果が逆転している．IMPI リレー未使用時に対するトランク数 8 の性能比を，遅延 0 ミリ秒と 10 ミリ秒に着目して比較すると，CG で 66%から 87%，MG で 78%から 105%，BT で 91%から 151%となる．

IMPI リレーの相対的な性能向上の要因として 2 つの理由が考えられる．1 つは，そのオーバヘッドが伝送遅延に対して相対的に減少するからである．もう 1 つは，TCP ストリームが IMPI リレーにより終端されていることによる効果である．これは PEP（Performance Enhancing Proxy）[15] で用いられる手法と同等の効果をもたらす．つまり MPI プロセス間の接続は IMPI リレーを境に 2 つに分割され，プライベートアドレスクラスタ内のプロセスに対する TCP 輻輳制御は，IMIP リレーとの間の遅延に影響される．これは実際のプロセス間の遅延と比較して小さいので，輻輳発生後に輻輳ウィンドウを素早く回復できる．

3.5　IMPI プロトコルのオーバヘッド

IMPI プロトコル処理のオーバヘッドの影響を評価するために，GC-GC 設定における GridMPI と YAMPI の性能を NPB を用いて比較した．両者は，クラスタ間通信に対して，IMPI と TCP のどちらを用いるかだけが異なる．問題サイズおよびプロセス数は前節の実験と同様である．結果を**表 4** に示す．両者の結果には有意な差はなく，特に 1 対 1 通信を用いた場合における実アプリケーションへの影響は無視できると考える．なお，クラスタ内

© 2009 Information Processing Society of Japan



(a) CG　(b) EP　(c) FT

(d) IS　(e) LU　(f) MG

(g) BT　(h) SP

図 7　NPB ベンチマークの相対性能（トランク数 1 かつ遅延 0 ミリ秒の結果に正規化）
Fig. 7　Relative performance of NPB benchmarks normalized to the '1 trunk' with 0 milliseccond of the delay cases.

表 4　GridMPI と YAMPI の NPB 性能比較
Table 4　NPB performance comparison between GridMPI and YAMPI.

	CG	FT	IS	LU	MG	BT	SP
YAMPI	3178.0 (1.0)	6970.5 (1.0)	99.9 (1.0)	18892.3 (1.0)	13359.7 (1.0)	17946.2 (1.0)	6836.6 (1.0)
GridMPI	3491.3 (1.1)	6675.1 (1.0)	89.6 (0.9)	19258.2 (1.0)	13050.3 (1.0)	16921.1 (0.9)	6046.9 (0.9)

© 2009 Information Processing Society of Japan

通信には MX ではなくギガビットイーサネットを用いたので，表 3 における GC-GC と比較して，全体的に性能が低い．

4. 議　　論

4.1 IMPI 規格への準拠

IMPI リレーは IMPI 規格に準拠しつつ，プライベートアドレスクラスタ間におけるパケット転送とトランキング機能を実現した．IMPI 規格に準拠するために生じた問題点を次にまとめる．

本論文では，クライアント数が 2 の場合を例に，IMPI リレーの動作を述べたが，クライアント数の上限は IMPI 仕様の制限から 32 である．IMPI リレーは，クライアント内部に対して，1 つのクライアントを 1 つのプライベートゲートウェイホストに対応付ける．つまり，クライアント数が n の場合には，IMPI リレーは，自らのクライアントを除く $n-1$ 台のプライベートゲートウェイホストとして動作する．この方式では，各プライベートゲートウェイホストの `Host ID` が重複するので，`Proc ID` は全プライベートゲートウェイホストに対して一意性を保証する必要がある．代替方法として，1 つのプライベートゲートウェイホストがすべてのリモートプロセスを有するマッピングも考えうるが，プライベートアドレスクラスタ内では実際のクライアント数に関係なく，クライアント数は 2 と認識されてしまう．これはクライアント間のトポロジを意識した通信アルゴリズムを使用する場合に問題となる．この問題を避けるために現在の方法をとっている．

トランキング機能の実現方式として，2.5 節に示した 2 種類を提案し，IMPI リレーでは，送信先ベース・トランキング方式を採用した．クラスタ内通信に関して，送信元ベース・トランキング方式の実現にはプロトコルの拡張が必要だからである．ただし，両方式ともにクラスタ間通信は IMPI に準拠する．これは外部クラスタの MPI 実装が IMPI に準拠するならば，トランキング機能を利用できることを意味する．今後，IMPI 規格に対して MPI-2 機能を拡張すると同時に，対外接続が限定された計算機への対応としてリレー機構やトランキング機能を標準化することが有用であると考える．

4.2 トランキング機能による通信性能向上

最適なトランク数はノード数，クラスタ間帯域，遅延時間，スイッチのバッファサイズ，実行する MPI プログラムの通信パターンなど複数の要因に依存する．計算よりも通信が主体で，大量のトラフィックを生成するプログラムに対しては，トランキング機能は性能に著しい効果を及ぼす．そのような場合は，トランク数は集約した通信帯域がクラスタ間帯域を超過しない限り大きくすべきである．一方，通信よりも計算が主体で，広帯域を必要としないプログラムに対しては，トランキング機能の効果は比較的小さい．

計算ノードから集約されたトラフィックがクラスタ間接続の物理帯域を超過する場合，クラスタ内のスイッチがボトルネックとなり，バッファあふれによりパケットが破棄される．この影響が顕著に現れたのが，全対全通信性能（3.3.3 項）や NPB IS ベンチマーク（3.4 節）であり，RTO の発生が大幅な性能劣化の要因となっていることを観測した．また，トランク数をクラスタ間帯域を超えない程度に設定することで，RTO の発生を抑え，性能が向上することを確認した．

RTO の発生は次のように説明できる．`MPI_Alltoall` 関数などの集団通信は複数の 1 対 1 通信，すなわち IMPI の場合は TCP ストリームの組合せで実現される．対向する 2 つの TCP ストリームの双方において，メッセージの末尾のパケットが破棄された場合，重複 ACK もしくは選択的 ACK を返信できないので高速再送に失敗し，相互にパケットを再送できないデッドロック状態に陥る[16]．RTO 時間は $RTT + 200$ ミリ秒であり，その期間は次のイテレーションに進むことができず，プログラム全体の実行が停止する．この挙動は TCP としては正常であるが，メッセージパッシングプログラムにおけるトランスポートプロトコルとして TCP を利用する場合には，性能劣化の原因となる，注意を要する点である．

4.3 通信時間増加の影響

現在の IMPI リレー実装は，受信した IMPI パケット全体をいったんバッファに格納してから ID を書き換えて転送するストア・アンド・フォワード方式で動作する．これは複数ストリームから読み込んだパケットを，1 つのストリームに集約するときに，パケット単位で排他的に書き込むためである．したがって，通信時間はパケットのサイズと数，さらに IMPI リレーの中継数に比例して増加する．

IMPI リレーの中継数を N_R，メッセージ中のパケット数を N_P，1 パケットあたりの送信時間を T とすると，片道通信時間は $(N_P + N_R)T$ と表せる．さらに，*rendezvous* プロトコル時の 1 対 1 通信性能は，リレー未使用時の通信時間 $N_P T$ とリレー使用時の通信時間 $(N_P + N_R)T$ の比から，$N_P/(N_P + N_R)$ に低下すると見積もることができる．たとえば，メッセージサイズが 256 KB（$N_p = 4$）かつ PC-PC（$N_R = 2$）の場合，帯域は 2/3 倍に低下する．図 5 の実験結果はこの見積りを裏付けている．

5. 関 連 研 究

過去に MPICH-G2[1]，PACX-MPI[2]，StaMPI[3]，MC-MPI[4] など，いくつものマル

© 2009 Information Processing Society of Japan

チクラスタおよびグリッド環境向け MPI システムが提案されている．MPICH-G2 はプライベートアドレスクラスタをサポートしておらず，すべての参加ノード間での完全な接続性を前提としている．その他の実装はプライベートアドレスクラスタをサポートしているが，トランキング機能はサポートしていない．

MC-MPI は，プロファイリング実行によって得られた情報に基づいて，プロセス間の局所性を考慮した接続管理機構を実現する．直接通信できないプロセス間では，自動的に経路を選択してメッセージが中継される．一方，IMPI リレーはグローバルアドレスを持ち，すべてのホストからの接続性が保証されていることを前提とするので，静的かつ手動による設定で十分と考える．実際の運用では，あらかじめ IMPI リレーのアドレスとポート番号をクラスタ固有の環境変数に設定しておくことで，`mpirun` コマンド実行時に IMPI リレーへの接続を制御している．また，IMPI リレーはトランキング機能によって高いクラスタ間通信性能を提供できる利点がある．

MPICH-GP[5] は，カーネルレベルの NAT サービスとユーザレベルプロキシを組み合わせた方式を用いて，プライベートアドレスクラスタに対応している．MPICH-GP は MPICH-G2 を拡張したシステムであり，システム全体でノードを一意に識別するために，GP_GUID 識別子を追加導入している．一方，IMPI リレーは専用プロトコルを定義する代わりに，標準化された IMPI プロトコルに準拠した．MPICH-GP の後継プロジェクトである MPICH-GX[6] は，TCP ホールパンチング（TCP NAT トラバーサル）を採用しているので，その性能はユーザレベルプロキシ方式を上回る．しかし，TCP ホールパンチングの可用性は NAT 機器が使用するアルゴリズムに依存するので，あらゆる環境で利用できるとは限らない．Guha ら[17] は，TCP ホールパンチングを使って TCP 接続を確立できる割合は平均で 88%であると報告している．

GridMPI では，各 MPI プロセスが直接 IMPI プロトコルを用いてクラスタ間通信を行う．一方，LAM/MPI の IMPI 実装[14] では，クラスタごとに 1 つ存在する impid プロキシプロセスが，MPI プロセスとして動作し，クラスタ内部通信から IMPI へのプロトコル変換を行う．impid をフロントエンドノード上で実行するように制御できれば，プライベートアドレスクラスタ間の通信は可能と思われる．MPI/PRO[7] は NAT と IMPI サーバを組み合わせてプライベートアドレスクラスタの使用を可能にしているが，詳細な情報を得ることはできなかった．

IPv6 による通信はもう 1 つの解決策であり，GridMPI を含めていくつかの MPI 実装は IPv6 をサポートしている．しかし，現状では大半のサイトでは IPv6 の対外接続性を提供していない．ハイパフォーマンスコンピューティング分野における IPv6 の普及は残された課題の 1 つである．

6. ま と め

本論文では，プライベートアドレスクラスタ群を含むグリッド環境において，高性能な MPI 通信を実現するためのリレー機構について述べた．IMPI 規格に準拠したクラスタ間の通信を中継し，さらに，フロントエンドノードのインタフェース速度に制約されない，高い帯域利用効率を実現するために，各クラスタごとに複数のフロントエンドノードを用いて通信路を集約するトランキング機能を提案した．提案手法を IMPI リレーとして実装し，GridMPI と LAM/MPI という 2 種類の IMPI 実装が，相互接続できることを確認した．また，10 Gbps の物理帯域と 0 から 10 ミリ秒の伝送遅延を持つ広域ネットワークを模擬した実験環境を用いて，提案手法の効果を確認した．大半の NPB ベンチマークは，トランク数に比例して性能向上が得られた．特に，通信が主体である FT と IS ベンチマークはトランキング機能によって 4.4 倍および 3.4 倍の高速化を達成した．また，遅延の増加に従って，リレー機構のオーバヘッドは相対的に減少した．遅延 10 ミリ秒下でトランク数を 8 にした場合，グローバルアドレスクラスタ間の性能と比較して，FT を除いて 90%程度以上の結果が得られた．

本論文では，ユーザレベルプロキシ上にトランキング機能を実装したが，他のプライベートアドレスクラスタ対応手法に対しても適用可能かつ有効な機能であると考える．今後は，GridMPI の IMPI 実装を Open MPI[18] に移植し，複数の IMPI 実装間に対する相互運用性をより詳細に検証する予定である．

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．また，本研究の遂行にあたり貴重な意見をいただいた東京農工大学吉澤康文教授に感謝する．

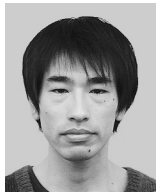
参 考 文 献

1) Karonis, N.T., Toonen, B. and Foster, I.: MPICH-G2: A Grid-Enabled implementation of the Message Passing Interface, ANL/MCS-P942-0402 (2002).
2) Gabriel, E., Resch, M. and Rühle, R.: Implementing MPI with Optimized Algorithms for Metacomputing, *The 3rd MPI Developer's and User's Conference* (1999).
3) Imamura, T., Tsujita, Y., Koide, H. and Takemiya, H.: An Architecture of Stampi:

© 2009 Information Processing Society of Japan

MPI Library on a Cluster of Parallel Computers, *EuroPVM/MPI 2000*, pp.200–207 (2000).
4) Saito, H. and Taura, K.: Locality-aware Connection Management and Rank Assignment for Wide-area MPI, *CCGrid 2007*, IEEE (2007).
5) Park, K., Park, S., Kwon, O. and Park, H.: MPICH-GP: A Private-IP-enabled MPI over Grid Environments, *The 2nd International Symposium on Parallel and Distributed Processing and Applications* (*ISPA 2004*) (2004).
6) MPICH-GX. http://supercom.yonsei.ac.kr/project/mpi/mpi.html
7) Raman, P., Skjellum, A., Dimitrov, R., Rajaram, K. and Banglore, P.: High Productivity MPI-Grid, Multi-Cluster, and Embedded System Extensions, *HPEC 2004* (2004).
8) T2K Open Supercomputer Alliance. http://www.open-supercomputer.org/
9) GridMPI. http://www.gridmpi.org/
10) YAMPI. http://www.il.is.s.u-tokyo.ac.jp/yampi/
11) George, W.L., Hagedorn, J.G. and Devaney, J.E.: IMPI: Making MPI Interoperable, *Journal of Research of the National Institute of Standards and Technology*, Vol.105, No.3, pp.343–428 (2000).
12) Takano, R., Matsuda, M., Kudoh, T., Kodama, Y., Okazaki, F., Ishikawa, Y. and Yoshizawa, Y.: High Performance Relay Mechanism for MPI Communication Libraries Run on Multiple Private IP Address Clusters, *CCGrid 2008*, IEEE (2008).
13) GtrcNET-10. http://projects.gtrc.aist.go.jp/gnet/
14) Squyres, J.M., Lumsdaine, A., George, W.L., Hagedorn, J.G. and Devaney, J.E.: The interoperable message passing interface (IMPI) extensions to LAM/MPI, *MPI Developer's conference* (*MPIDC*) (2000).
15) Border, J., Kojo, M., Griner, J., Montenegro, G. and Shelby, Z.: Performance Enhancing Proxies Intended to Mitigate Link-Related Degradations, RFC 3135 (2001).
16) 高野了成，工藤知宏，児玉祐悦，松田元彦，手塚宏史，石川　裕：GridMPI のための TCP/IP 輻輳制御実装方式の検討，*SWoPP 2004*，情報処理学会 (2004).
17) Guha, S. and Francis, P.: Characterization and Measurement of TCP Traversal through NATs and Firewalls, *Internet Measurement Conference 2005* (2005).
18) Open MPI. http://www.open-mpi.org/

(平成 20 年 7 月 23 日受付)
(平成 20 年 12 月 15 日採録)



高野　了成（正会員）

2005 年東京農工大学大学院電子情報工学専攻博士後期課程単位取得満期退学．2003 年（株）アックス入社．2008 年より独立行政法人産業技術総合研究所情報技術研究部門研究員．博士（工学）．グリッド環境における並列通信ライブラリ，高性能通信機構の研究開発に従事．オペレーティングシステム，広域分散システムにおける資源管理に興味を持つ．



松田　元彦（正会員）

1988 年京都大学理学部卒業．同年住友金属工業（株）入社．1995 年から 1999 年まで技術研究組合新情報処理開発機構に出向．2003 年から 2007 年まで独立行政法人産業技術総合研究所．現在，東京大学大学院情報理工学系研究科特任准教授．工学博士．並列計算システム，クラスタシステムおよびグリッド環境での高性能計算に関する研究に従事．



工藤　知宏（正会員）

1991 年慶應義塾大学大学院理工学研究科博士課程単位取得退学．東京工科大学助手，講師，助教授を経て，1997 年より新情報処理開発機構並列分散システムアーキテクチャつくば研究室長，2002 年より産業技術総合研究所グリッド研究センタークラスタ技術チーム長．2008 年より同所情報技術研究部門インフラウェア研究グループ長．博士（工学）．並列処理，通信アーキテクチャに関する研究に従事．IEEE CS 会員．



児玉　祐悦（正会員）

1988 年東京大学大学院情報工学専門課程修士課程修了．同年通産省電子技術総合研究所入所．2001 年独立行政法人産業技術総合研究所に改組．現在，同研究所情報技術研究部門主任研究員．データ駆動やマルチスレッド等の並列計算機システムの研究に従事．特にプロセッサアーキテクチャ，FPGA，ネットワークエミュレータ等に興味あり．博士（工学）．情報処理学会奨励賞，情報処理学会論文賞（1990 年度），市村学術賞（1995 年）等受賞．電子情報通信学会，IEEE CS 各会員．

© 2009 Information Processing Society of Japan



岡崎　史裕（正会員）

1987 年京都大学大学院工学研究科電気 2 専攻修士課程修了．同年住友金属工業（株）入社．1998 年より 4 年間技術研究組合新情報処理開発機構の並列処理の研究に従事．2003 年より独立行政法人産業技術総合研究所グリッド研究センターに派遣．2008 年より独立行政法人産業技術総合研究所情報技術研究部門に派遣．ネットワーク関連の技術開発に従事．



石川　　裕（正会員）

1987 年慶應義塾大学大学院工学研究科電気工学専攻博士課程修了．工学博士．同年電子技術総合研究所入所．1993 年技術研究組合新情報処理開発機構出向．2002 年より東京大学大学院情報理工学系研究科コンピュータ科学専攻教授．クラスタ・グリッドシステムソフトウェア，高信頼システムソフトウェア開発技術，実時間分散システム，次世代高性能コンピュータシステム等に興味を持つ．

© 2009 Information Processing Society of Japan