GridMPI™ Version 1.0の概要

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 は、グリッド環境上において、i) インターオペラビリティ、ii) 高性能、ii) 耐故障性を提供する MPI 通信ライブラリ実装である。並列コンピュータ間通信プロトコルの一つとして、異なる MPI 実装間で通信するための標準規格である IMPI (Interoperable MPI) プロトコルが実装されている。広域網を介した並列コンピュータ間における高性能通信を実現するために、ネットワークトポロジと通信パターンに応じて通信アルゴリズム／送信バンド幅を調整する機能を有する。耐故障性として、チェックポイントリスタート機能を実現している。アプリケーションプログラムを変更することなく GridMPI ライブラリをリンクするだけで、チェックポイントリスタートが可能となる。

Overview of the GridMPI™ Version 1.0

MOTOHIKO MATSUDA ,[†] YUTAKA ISHIKAWA ,[†,††] YOSHITAKA KANEO ,[†]
MASAHIKO EDAMOTO ,[†] FUMIHIRO OKAZAKI ,[†] HIDETAKA KOIE ,[†]
RYOUSEI TAKANO ,[†] TOMOHIRO KUDOH [†] and YUETSU KODAMA[†]

GridMPI is an MPI implementation that supports i) interoperability, ii) high performance, and ii) fault tolerance 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 communication between parallel computers. In order to realize high performance communication using the Internet, GridMPI implements several communication algorithms and a transmission bandwidth control mechanism for the network topology and communication patterns. The checkpoint/restart function is implemented as a fault tolerance capability. An application linked with the GridMPI library is able to save a checkpoint file without any application program modifications.

1. はじめに

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[1)~4)] の開発を行なっている。GridMPI 開発に当たっては、グリッド環境の特徴である、異機種計算環境、通信遅延およびバンド幅に制限がある広域網上での通信、サイト障害に伴う耐故障を考慮にしている。

本稿では、GridMPI version 1.0 の機能について述べる。GridMPI version 1.0 は MPI-2 機能全てを実現している。GridMPI version 1.0 のソフトウェアアーキテクチャならびに特徴的機能について述べた後、現在の GridMPI version 1.0 の実現方法を紹介する。GridMPI version 1.0 の性能を比較するために、WAN 環境を模擬する実験装置において、NAS Parallel Benchmarks[14)] プログラムを用いて、MPICH[5)]、LAM/MPI[6)] と比較する。FT, IS では、GridMPI は、通信遅延の大きさに関わらず、MPICH や LAM/MPI に比べて 1.2 倍以上の性能を得ている。他の BT, CG, LU, MG, SP では、MPICH や LAM/MPI とほぼ同等の性能が得られている。

なお、GridMPI が対象とする並列コンピュータは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。

† 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

2. GridMPI の概要

2.1 ソフトウェアアーキテクチャ

GridMPI のソフトウェアアーキテクチャを図 1 に示す。GridMPI は、東大で開発されている YAMPII[7)] をベースとし、プロセス生成、通信機構のそれぞれに対して、様々な既存ライブラリが利用可能となるようなモジュラ構造となっている。MPI Core 部は YAMPII で提供される部分であり、グループ、コミュニケータ、トポロジなどの機能が実現される。MPI Core 部の下位レイヤにリモートプロセス生成機能と通信機能が定義

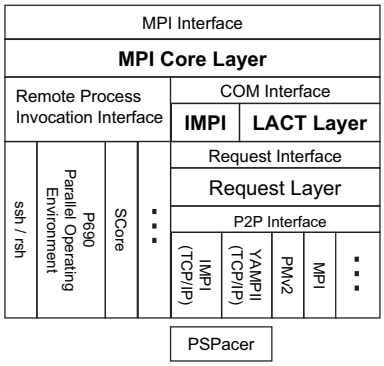

図 1 GridMPI のソフトウエアアーキテクチャ

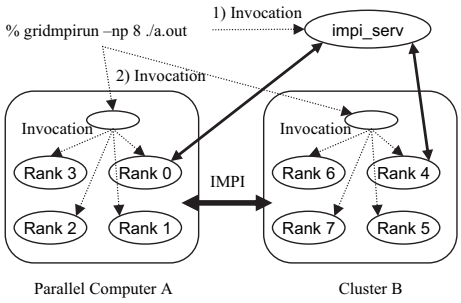

図 2 GridMPI 実行概要

される。通信機能は、IMPI (Interoperable MPI)[8] が定義している集団通信プロトコルおよび LACT (Latency-Awawre Communication Toplogy) 通信層の 2 つが定義される。IMPI 集団通信プロトコルおよび LACT は、下位通信レイヤである P2P 層とのインターフェイスである Request 層を用いて実現される。

GridMPI version 1.0では、クラスタ内通信は YAMPII で実装されている通信プロトコルが使用できる。また、商用並列コンピュータ上では、ベンダが提供する MPI 通信ライブラリが使用される。クラスタ間あるいは並列コンピュータとの通信では、IMPI 通信プロトコルが使用される。

GridMPI の実行概要を図 2 に示す。

(1) gridmpirun コマンドにより、IMPI サーバが起動される。IMPI サーバは IMPI 規格で定義されているサーバであり、各並列コンピュータ上で起動される MPI プロセス間の情報交換のために使われる。

(2) ユーザが必要とするプロセス数分のコンピュータを探し、各並列コンピュータ上で GridMPI プロセスを生成する。gridmpirun コマンドで実現されている計算資源割り当ては、configuration ファイルで規定される計算資源リストを使った単純なものとなっている。GridMPI version 1.0 では、資源割り当ては NAREGI プロジェクト [9] で開発されているワークフローや Super Scheduler との連携を想定しているからである。

(3) 各並列コンピュータ上の GridMPI プロセスは、IMPI サーバを介してアドレス情報を交換し、他の並列コンピュータ上の GridMPI プロセスとの通信路を確立する。

2.2 通信パターンに基づくメッセージ送出量調整

広域網を介して相互接続される並列コンピュータ間では一つの通信路が共有される。MPI 通信におけるバリア同期や集合通信のような全体通信に対して、ノード間での通信遅延およびバンド幅を基準にしたネットワークトポロジに応じて最適な通信を実現する必要がある。

例えば、2 つの並列コンピュータのノード間で 1 対 1 通信している時の広域網通信路の使用バンド幅と、2 つの並列コンピュータ間で全対全通信している時の広域網通信路の使用バンド幅は異なる。広域網において 1 対 1 通信した後に全対全通信する場合、TCP/IP は通信パターンが変更されても 1 対 1 通信時と同じバンド幅でデータ送信を行なうために、広域網上の通信路が輻輳してしまう。

GridMPI では、TCP/IP のコネクション単位で精密に送出量を調整できるモジュール PSPacer[10] と連携することで、ネットワークトポロジと通信パターンに応じたバンド幅制御を提供する。PSPacer は、ギャップパケットと呼ぶ任意サイズの PAUSE フレームを送信パケット間に挿入することで、精密なバンド幅制御とバーストトラヒックの平滑化を実現する。また、Linux のトラヒック制御機構である iproute2 を利用して実装されており、クラスタ内通信とクラスタ間通信のように、ネットワークトポロジに応じて、コネクション群をクラス単位に分割し、各クラスに対してバンド幅を制御できる。

2.3 チェックポイントリスタート

耐故障性として、GridMPI では、チェックポイントリスタート機能を実現している。アプリケーションプログラムを変更することなく GridMPI ライブラリをリンクするだけで、チェックポイントリスタートが可能となる。gridmpirun プロセスに対して QUIT シグナルを送るとカレントディレクトリにチェックポイントファイルが作成される。リスタートするには gridmpirun に-restart オプションをつける。

3. 実　　装

3.1 IMPI

IMPI プロトコルはトランスポート層に TCP/IP を想定している。以下特徴を述べる。

3.1.1 IMPI サーバ

前節で述べた通り、クラスタをまたがる MPI プロセスの情報 (IP アドレス、ポート番号等) 交換を行なうため、IMPI サーバと呼ばれるプロセスが存在する。

MPI_Init 時に IMPI サーバを介して各クラスタで動作する MPI プロセス情報を交換し、全対全で接続を行なう。MPI_Finalize では MPI プロセス間で同期を行なった後、終了処理を完了する。

3.2 1 対 1 通信方式

P2P 通信はデータをパケットに分割して送受信する。メッセージのサイズにより通信アルゴリズムを切り替えており、短いメッセージでは eager プロトコル、長いメッセージでは rendezvous プロトコルとなる。

IMPI ではフロー制御についても規定されている。MPI の 1 対 1 通信において指定する送信、受信ランクに対応したカウンタを持ち、1 パケット送信、受信する度にカウンタをアップする。送信側はカウンタがある閾値に達すると送信を止め受信側からの制御パケットの到着を待つ。受信側はカウンタがある閾値に達すると送信側へ制御パケットを送る。送信側カウンタの閾値は受信側カウンタの閾値以上である。

3.2.1 異機種間でのデータ形式

IMPI では異機種間での MPI の実行を想定しており、異機種間で流れるメッセージの形式を規定している。MPI-2 規定の external32 形式を採用している。

3.3 集団通信アルゴリズム

集団通信、コミュニケータ操作は IMPI では擬似コードとして定義されている。集団通信はメッセージのサイズ、実行するクラスタ数によりアルゴリズムが切り替わる。

3.4 ベンダ MPI

GridMPI では、ベンダが提供する MPI 通信ライブラリの内、MPI_Isend と MPI_Recv 関数をバイト列の転送手段として使用することで、ベンダ MPI の P2P 層を実現している。

クラスタ内通信にベンダ MPI を使用した場合には、クラスタ間通信の通信イベントは select 関数で、クラスタ内通信の通信イベントは MPI_Probe 関数で検出する。シングルスレッドプログラムでは、通信イベントを検出するために、一方の関数でブロッキングして待つことはデッドロックを引き起こすため許されない。また、イベント待ち用にスレッドを使用することは、IBM の並列実行環境では性能に著しい劣化が見られた。そこで、GridMPI では select と MPI_Probe を交互にポーリングする実装としている [3]。

ベンダ MPI P2P 層の実装により、GridMPI のクラスタ内通信の性能は IBM MPI の 90%以上の性能が得られた。

3.5 通信パターンに基づくメッセージ送出量調整

GridMPI と PSPacer[10] の関係を**図 3** に示す。PSPacer は、Linux カーネルモジュールとして実装されており、パケットがインタフェースキューからネットワークインタフェースへ送出される時点で、バンド幅制御を行っている。本連携機構は、PSPacer に対するパラメータの取得、設定を中継するデーモン (pspd) と、MPI ライブラリが pspd と通信を行うためのライブラリ (libpsp) から構成される。

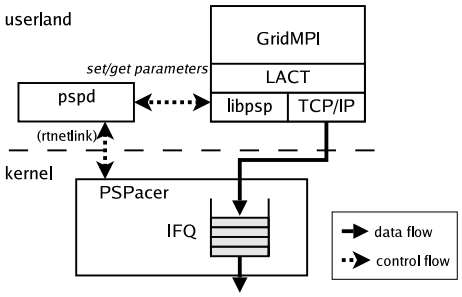

図 3 GridMPI ライブラリと PSPacer モジュール

libpsp は、MPI 通信パターンが切り替わる際に、利用可能なネットワークバンド幅、コミュニケータ、プロセスが所属するクラスタの情報からターゲットレートを計算し、その変更を pspd に要求する。現在の実装では、MPI_Alltoall と MPI_Alltoallv 関数の実行中に、ターゲットレートを「利用可能ネットワークバンド幅 / クラスタ内ノード数」に制限する。例えば、それぞれ 16 ノードからなる二つのクラスタが、1 ギガビットリンクで接続されている場合、コミュニケータ MPI_COMM_WORLD に対する MPI_Alltoall 関数実行時には、各ノードの送信バンド幅を $62.5Mbps(=1Gbps/16)$ に制限する。

pspd が要求を受信した時、MPI 関数の実行は完了しても、オペレーティングシステム内部のソケットバッファやインタフェースキューにパケットがバッファリングされている可能性がある。これらのパケットは変更要求前のターゲットレートで送信する必要があるので、インタフェースキューが空になるのを待ってから、PSPacer に対してターゲットレート変更を要求する。

現在の実装では、ネットワークトポロジ、および各ネットワークバンド幅を、MPI プログラム実行前に設定する必要があり、実行時に動的に変更することはできない。今後は、LACT 層におけるメッセージのルーティングと連携した通信制御を実現する予定である。

3.6 チェックポイントリスタート

MPI プログラムのチェックポイントでは、全ての MPI プロセスの状態と通信の状態を一貫した状態で二次記憶に退避する必要がある。チェックポイントに関する様々な研究 [11),12)] がなされているが、GridMPI では、無駄なチェックポイントが避けられることと開発の容易性から Coordinated Checkpointing(チェックポイント時に各プロセスが同期をとる方式) を採用している。チェックポイント時に、ソケット層から物理層までの間にデータが滞留していないことを確認し、各プロセス間の状態認識を一致させるための同期機構を定義する。このために、YAMPII と IMPI それぞれのプロトコルを拡張している。

プロセス単体のチェックポイントリスタート機能は

表 1 チェックポイントリスタート動作環境

OS/arch	static link	dynamic link
Linux-2.4/i686	yes	yes
Linux-2.6/i686	yes	yes
Linux-2.6/x86_64	yes	no
FreeBSD-5/i386	yes	yes

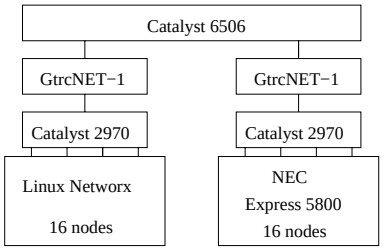

図 4 実験環境の構成

GridMPI から切り離して任意のプログラムから利用できるチェックポイントライブラリ (libckpt と呼ぶ) として実現している。チェックポイントリスタートライブラリ libckpt は移植性と実装の容易さからユーザ空間で実装している。

保守性の観点からシステムコールラッパはつかわずプロセスの状態を得るのに/proc ファイルシステムを併用する。対応 OS は Linux, FreeBSD で、動作確認を行った環境を**表 1** に示す。

チェックポイントではメモリ、レジスタなどの他にファイルディスクリプタ、ソケット、シグナルも保存される。ソケットは TCP と UNIX をサポートし、リスタートではチェックポイント時と同じアソシエーションを復元する。チェックポイントリスタートをプロセスマイグレーションの目的で行う場合はエンドポイント書き換えルールをファイルに記述しリスタート時に環境変数で指示する。

リスタートはプログラムから直接リスタート API を呼ぶことで行う。動的リンクの場合はリスタート専用コマンド ckpt_redo でもリスタートできる。各プロセスはチェックポイント時とリスタート時とで同じ OS、CPU に割り当て場合にのみリスタート可能であり、ヘテロ環境に対応している GridMPI ではリスタート時のリソース指定に注意が必要である。

差分チェックポイントは実装していない。Coordinated Checkpointing ではチェックポイント時の同期待ち時間のオーバヘッドがある。このためチェックポイントの頻度が上げられず差分が大きくなりあまり有効でないと判断した。スレッドプログラムはサポートしていないが Linux NPTL 版スレッドサポートを検討中である。

4. 評　　価

4.1 実 験 環 境

図 4 に実験環境を示す。また、使用した PC クラスタの諸元を**表 2** に示す。グリッド環境を模擬する測定環境を作るため、それぞれ 16 台の PC から構成されるクラスタをギガビットイーサネットで接続し、クラスタ間は、ハードウェアネットワークテストベット GtrcNET-1[13] を用いて、片道 0ms〜10ms の遅延を挿入した。クラスタ内はそれぞれ単一のノンブロッキングスイッチ (Catalyst 2970) で接続されている。

表 2 PC クラスタ諸元

	NEC Express 5800	Linux Networx
CPU	Intel Xeon (2.8 GHz)	Intel Xeon (2.8 GHz)
チップセット	ServerWorks GC-LE	Intel E7501
メモリ	1GB	1GB
NIC	Intel 82546EB	Intel 82546EB
OS	FedoraCore 3 (kernel 2.6.11-1.27_FC3)	
NIC ドライバ	e1000 5.6.10.1-k2-NAPI	
ソケットバッファ	2 MB	

表 3 MPI 通信ライブラリバージョン

MPI 通信ライブラリ	バージョン
GridMPI	1.0 α
MPICH	1.2.7
LAM/MPI	7.1.1

評価には、MPI の代表的なベンチマークである NPB (NAS Parallel Benchmarks)[14] バージョン 3.2 を使用した。評価に使用した MPI 通信ライブラリのバージョンを**表 3** に示す。すべての MPI 実装に対して、ソケットバッファサイズは 2MB に設定した。

4.2 NPB

図 5、**図 6** に NPB ベンチマークの結果を示す。GridMPI、MPICH、LAM/MPI について、クラスタ間ネットワークの遅延時間を変化させながらベンチマークを行った。グラフは遅延時間 0 における GridMPI の性能を 1 とする相対性能である。

図 5 に BT、CG、LU、MG、SP の結果を示す。遅延に従って性能が低下するが、実装間でほぼ同様の傾向を示していることが分る。GridMPI の場合、クラスタ間通信にはデータ表現のバイトオーダ変換が行われておりそのオーバーヘッドが入っている。一方、MPICH、LAM/MPI はクラスタ設定でありバイトオーダ変換は行われていない。

図 6 にペーシングを行なう場合を含めた FT、IS の結果を示す。ペーシングは全対全通信である MPI_Alltoall と MPI_Alltoallv 関数に対して適用したので、これらの関数を利用する FT、IS に効果がある。GridMPI/PSP がペーシングを行った場合の結果である。FT、IS の結果の結果から、遅延のあるなしに関らずペーシングにより大きく性能向上することが確認できる。

4.3 通信パターンに基づくメッセージ送出量調整

PSPacer を用いた場合、IS では、1.6 から 2.0 倍と遅延の増加に比例して、性能向上が得られた。また、

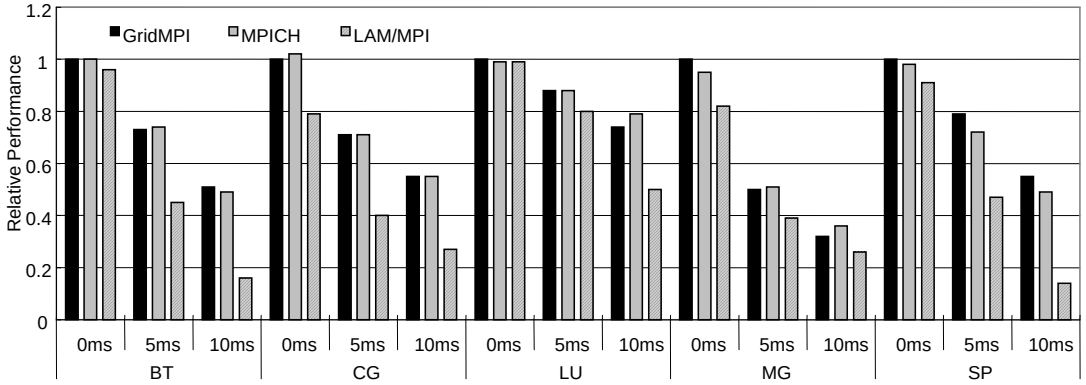

図 5 NPB ベンチマーク、BT、CG、LU、MG、SP の結果 (CLASS=B,NPROCS=32)

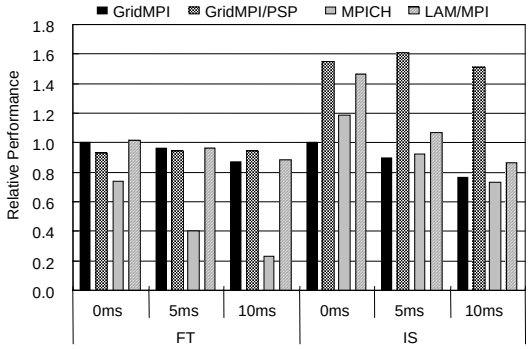

図 6 NPB ベンチマーク、FT、IS の結果 (CLASS=B,NPROCS=32)

FT では、遅延が小さいときには効果がないが、遅延 10ms のときには 1.1 倍の性能向上が得られた。

遅延 10ms で、IS を実行したときの通信バンド幅を**図 7** に示す。なお、測定は、GtrcNET-1 を用いて 1ms 間隔で行った。PSPacer なしの場合、MPI_Alltoallv 関数により、全ホストによる通信がボトルネックリンクに一斉に集中するため、パケットロスが発生している。さらに、パケットロスの有無により、コネクション間のバンド幅利用に不公平性が生じ、プロセス間の同期に時間がかかる。一方、PSPacer ありの場合、パケットロスは発生せず、高いバンド幅利用率を維持している。

ペーシングはパケットの送信間隔を平滑化するため、遅延の増加をもたらす。しかし、グリッド環境においては、この遅延による性能低下より、パケットロスによる TCP/IP 輻輳制御によるオーバヘッドの方が大きいため、ペーシングは有効な手法であると考える。

4.4 チェックポイントリスタート

チェックポイントリスタート有効化による速度低下は見られなかった。注意すべき点として、チェックポイントリスタート有効時は簡単のため静的リンクになっており、Linux では若干の速度低下がある。

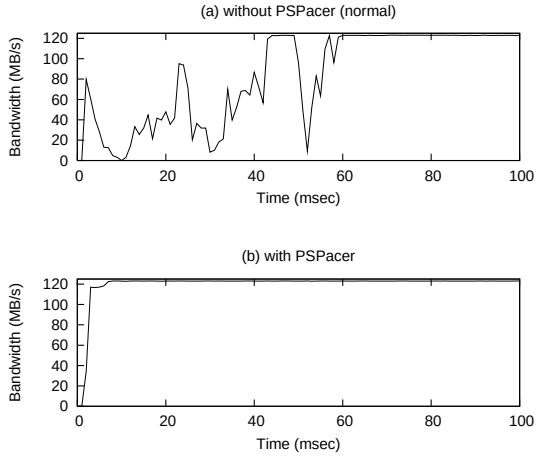

図 7 IS におけるクラスタ間の通信帯域 (遅延 10ms)

5. 関連研究

IMPI は、HP-MPI[15]、LAM/MPI[6]、MPI/Pro[16] 等で実装されている。LAM/MPI は P2P 通信と一部の集団通信のみ IMPI を実装している。HP-MPI は異機種間での実行をサポートしていない。

MPICH-G2[5] は IMPI プロトコルを使わず、Globus Toolkit の通信コンポーネントである Globus I/O を利用し、独自プロトコルにより MPI 実装間の相互運用を実現している。いくつかの広域 MPI はベンダ MPI への拡張として実装されている。例えば、Stampi[17] や PACX-MPI[18] などである。これらはクラスタ内の MPI 操作をベンダ MPI 関数を呼び出す実装になっている。GridMPI では 1 対 1 通信機能としてのみ利用している点が異なる。

TCP/IP の断続通信時に対するペーシングの効果は、WEB トラフィックへの対応として提案されたが [19]、

公平性に対する問題 [20] などが指摘されており、広く利用されるまで至っていない。一方、MPI 通信への適用に関して、論文 21) では、通信再開時に、スロースタートの代わりに高精度タイマにより生成したクロックによるペーシングを用いる効果について述べている。本論文では通信再開時に限定せず、定常状態に対してもペーシングが有効であることを示した。また、MPICH-GQ[22] は、MPI プログラムを広域網で効率的に動作させるために、プロセッサやネットワークなどの資源に対して QoS 機構を導入している。そして、トークンバケット方式を利用したバンド幅制御では、動的で適切なトークンバケットサイズの設定が必要であり、現実的に実装することは困難であると結論付けている。本論文では、MPI 通信ライブラリに適用できる精密なバンド幅調整機能を提案した。

SCore[23] はクラスタのための並列計算システムソフトウェアでユーザ空間で実装された Linux の静的リンクプログラムのチェックポインタをもつ。Victor C. Zandy の ckpt[11] はユーザ空間で実装された Linux 動的リンクプログラムのチェックポインタである。Berkeley Lab Checkpoint/Restart(BLCR)[12] はカーネル空間で実装された Linux 動的リンクプログラムのチェックポインタであり、LAM/MPI と結合して MPI プログラムのチェックポイントが可能である。

6. ま と め

GridMPI は、MPI-2 機能を有し、異機種計算機環境かつ通信遅延と通信バンド幅が制限される Grid 環境で高性能な MPI 通信を提供すべく、開発を進めてきている。本稿では、GridMPI version 1.0 の概要を紹介し、グリッド環境を模擬した環境における NPB ベンチマークの結果を MPICH, LAM/MPI で比較した。

NPB ベンチマークの FT、IS の結果の結果から、遅延のあるなしに関らず GridMPI の性能は、MPICH、LAM/MPI に優れている。また、GridMPI は異機種間通信のためデータ表現変換を行なっており、そのオーバーヘッドを含んでいるが、MPICH、LAM/MPI とほぼ同等の性能が得られている。

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) 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.
2) 松田, 石川, 鐘尾, 枝元, 岡崎, 工藤, 児玉, 手塚. GridMPI の性能評価. 情報処理学会, SWoPP'04, 2002-HPC-99-22, 2004.
3) 岡崎, 石川, 松田, 鐘尾, 枝元, 高野, 工藤. IBM pSeries における GridMPI の実装と性能評価. 情報処理学会, HOKKE'05, 2002-HPC-101-31, 2005.
4) GridMPI. http://www.gridmpi.org
5) MPICH. http://www-unix.mcs.anl.gov/mpi/mpich/
6) LAM/MPI. http://www.lam-mpi.org
7) 石川. YAMPII もう一つの MPI 実装. 情報処理学会, SWoPP'04, 2004-HPC-099, 2004.
8) 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.
9) 超高速コンピュータ網形成プロジェクト NAREGI. http://www.naregi.org
10) R. Takano, T. Kudoh, Y. Kodama, M. Matsuda, H. Tezuka, and Y. Ishikawa. Design and Evaluation of Precise Software Pacing Mechanisms for Fast Long-Distance Networks. PFLDnet05, 2005.
11) Ckpt - A process checkpoint library. http://www.cs.wisc.edu/ zandy/ckpt/
12) Berkeley Linux Checkpoint/Restart (BLCR). http://ftg.lbl.gov/checkpoint
13) Y. Kodama, T. Kudoh, R. Takano, H. Sato, O. Tatebe and S. Sekiguchi. GNET-1: Gigabit Ethernet Network Testbed. Proc. of IEEE Cluster2004, pp.185-192, 2004.
14) 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
15) Hewlett-Packard. HP MPI User's Guide, Eighth Edition. 2003.
16) MPI Software Technology, Inc. http://www.mpi-softtech.com
17) 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.
18) PACX-MPI. http://www.hlrs.de/organization/pds/projects/pacx-mpi
19) M. Aron and P. Durschel. TCP: Improving Startup Dynamics by Adaptive Timers and Congestion Control. Tech. Rep. TR98-318, Rice Univ., 1998.
20) A. Aggarwal, S. Savage, and T. Anderson. Understanding the performance of TCP pacing. IEEE INFOCOM2000, 2000.
21) 松田, 高野, 石川, 工藤, 児玉, 岡崎, 手塚. MPI ライブラリと協調する TCP 通信の実現. 情報処理学会, SACSIS'05, 2005.
22) A. Roy, I. Foster, W. Gropp, N. Karonis, V. Sander, and B. Toonen. MPICH-GQ: Quality-of-Service for Message Passing Programs. IEEE SC2000, 2000.
23) PC Cluster Consortium. http://www.pccluster.org