社団法人　情報処理学会　研究報告
IPSJ SIG Technical Report

2005－ARC－162 （31）
2005－HPC－101 （31）
2005／3／9

IBM pSeriesにおけるGridMPIの実装と性能評価

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

GridMPI はクラスタ計算機用 MPI 実装の一つである YAMPII をベースに、IMPI プロトコルを実装したグリッド環境用 MPI 通信ライブラリである。GridMPI の適用範囲を拡大するため、ベンダ MPI を GridMPI のネットワーク通信層に追加した。ネットワーク通信層はベンダ MPI と IMPI の 2 系統で構成されることになり、単一スレッドで実行する場合には片方でブロッキングするとデッドロックを引き起こしてしまう。そこでネットワーク通信層をスレッドで処理する方式とポーリングで処理する方式を GridMPI に実装した。GridMPI の性能を並列ベンチマークで評価した結果、IBM pSeries 単体性能ではポーリング方式で IBM のベンダ MPI の 90%以上の性能を得た。

Performance Evaluation of the GridMPI for IBM pSeries

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

GridMPI is an MPI library for the Grid environment, in which the IMPI protocol is implementend based on the YAMPII MPI implementation for the cluster environment. In order to expand the scope of GridMPI, a Vendor MPI layer is added to the network communication layer of GridMPI. Since the communication layer now consists of two protocols of Vendor MPI and IMPI, a deadlock is caused by using simple blocking-wait in them. To prevent it, we have implemented two methods, multithreading and polling. Performance evaluation using NPB (NAS Parallel Benchmarks) shows that the polling method performed at more than 90% of the performance of IBM MPI on IBM pSeries.

1. はじめに

グリッドコンピューティングの応用の１つとして、地理的に離れた複数の計算資源を連携した大規模なアプリケーション実行がある。我々はグリッド環境で MPI を用いた高性能計算を行うことを目指して GridMPI[1)~3)] を開発している。

GridMPI はクラスタ計算機用 MPI 実装の１つである YAMPII[4),5)] をベースに、IMPI[6)] プロトコルを実装したグリッド環境用 MPI 通信ライブラリである。クラスタ内のメッセージ交換には YAMPII を使用し、クラスタ間のメッセージ交換には IMPI プロトコルを使用する。YAMPII では P2P 層（ネットワーク通信層）として TCP/IP を使用するソケット P2P と、SCore[7)] クラスタが提供するネットワーク通信層を使用する SCore P2P を実装している。

グリッド環境には PC クラスタだけでなくベンダ製の専用通信装置を持つ並列計算機が存在する。そのような計算機は並列アプリケーションを高性能に実行する環境が整備され、独自に最適化された MPI も実装されている。これらベンダ製計算機を含めたグリッド環境を GridMPI で構築する場合、ソケット P2P を用いたのではベンダ独自の通信装置を利用できないため、高い通信性能を得ることができない。この通信装置を使用するには通信装置ごとの通信インタフェースに対応する必要があり、GridMPI でこれに対応した開発や保守を継続することは困難である。

ベンダから提供される MPI 実装 (ベンダ MPI) はベンダ独自の通信装置を使用しているが、その MPI のインタフェースは規格化されている。我々はこの点に着目し、ベンダ MPI インタフェースをバイト列の転送に利用するベンダ MPI P2P の開発を行うことにした。本論文では、IBM pSeries 690(p690) 用ベンダ MPI P2P の GridMPI 実装を報告する。

以下、ベンダ MPI の実装について説明した後、GridMPI の性能評価を行い、最後にまとめる。なお、文中では並列計算機を総称してクラスタと呼んでいる。

2. ベンダ MPI の実装

図 1 に GridMPI のソフトウエアアーキテクチャを

† 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
††† 株式会社アックス
AXE,Inc.

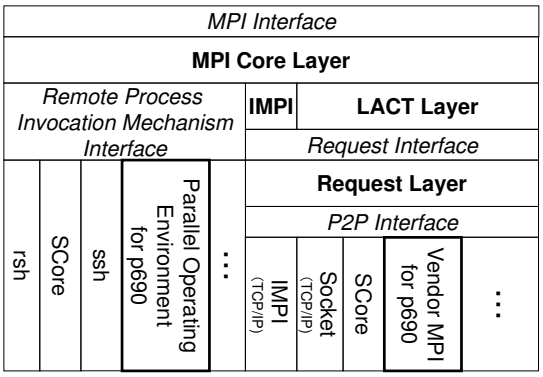

図 1 GridMPI のソフトウエアアーキテクチャ

示す。今回の開発では、P2P インタフェースにベンダ MPI P2P を、RPIM(Remote Process Invocation Mechanism) インタフェースに p690 の POE(Parallel Operating Environment)[11] を追加実装した。

P2P 層は 1 対 1 通信機能を提供し、送信したデータが確実に受信側で受信されることを保障する。P2P 層は、クラスタ内やクラスタ間のメッセージが到着したことや送信が可能になったことなどの通信イベントを検出して、送受信処理を行う。

GridMPI ではクラスタ内通信にソケット P2P を、クラスタ間通信に IMPI P2P を実装している。IMPI P2P は IMPI プロトコルに規定されたデータ送受信とデータ変換やフロー制御などを行う。IMPI P2P はソケット P2P と同じ TCP/IP プロトコルで実装されているので、`select` 関数を用いて 2 つの P2P 層の通信イベントを一元的に検出可能である。従って、GridMPI の実装では、`select` 関数を用いて通信イベントをブロッキングしてウエイトする実装となっていた。

一方、クラスタ内通信にベンダ MPI P2P を、クラスタ間通信に IMPI P2P を用いた場合、通信イベントを `MPI_Probe`、`MPI_Test` などの関数と `select` 関数で検出する必要がある。GridMPI ではベンダ MPI P2P とソケット P2P のどちらからメッセージが到着するか分からない。このため、片方の P2P 層の通信イベント待ちでブロッキングすると、他方の通信イベントの処理が行えずに GridMPI がデッドロックする可能性がある。

これを解決するには次の 2 つ実装方式が考えられる。

1) スレッド方式
   各 P2P 層を処理するスレッドを生成して、各々の通信イベントをブロッキングして検出し、処理する方式
2) ポーリング方式
   各 P2P 層の通信イベントをブロッキングせずに交互にポーリングして検出し、処理する方式

この 2 方式を実装する。

以下、ベンダ MPI P2P の通信方式、スレッド方式の実装、ポーリング方式の実装と RPIM の実装について説明する。

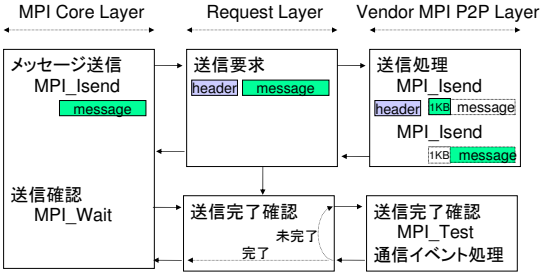

図 2 ベンダ MPI P2P の送信処理

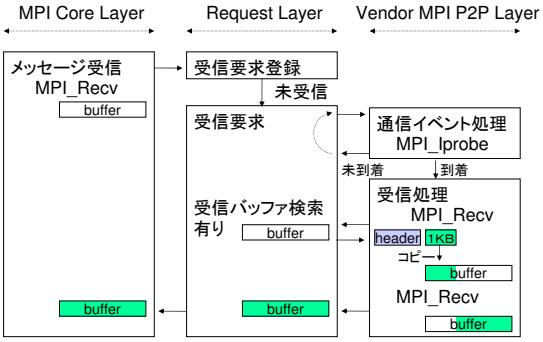

図 3 ベンダ MPI P2P の受信処理 (1)

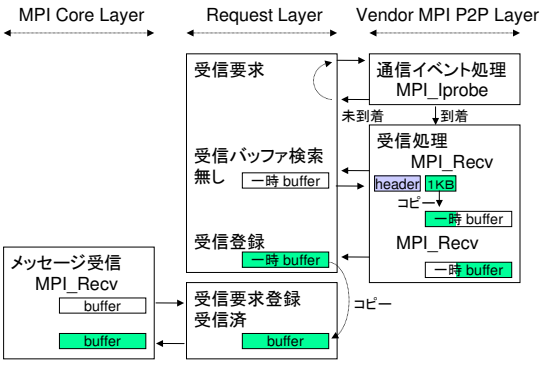

図 4 ベンダ MPI P2P の受信処理 (2)

2.1 ベンダ MPI P2P の通信方式

ベンダ MPI P2P の通信方式はベンダ MPI の `MPI_Isend`、`MPI_Test` 関数と `MPI_Recv` 関数をバイト列の転送手段として使用する。メッセージ長はヘッダ情報で管理している。メッセージ長が短い場合 (ショートメッセージ) には、ヘッダ情報と合わせてメッセージが転送される。メッセージ長が長い場合 (ロングメッセージ) には、メッセージは 2 回に分けて転送される。最初にヘッダ情報とショートメッセージ分のデータを転送し、続けて残りのメッセージを転送する。現在の実装ではショートメッセージは 1000 bytes 以下、ロングメッセージはこれを超えるものとしている。

送信処理を図 2 に示す。送信はリクエスト層からの送信要求時に `MPI_Isend` 関数でメッセージを送信し、リクエスト層からの送信完了確認時に `MPI_Test`

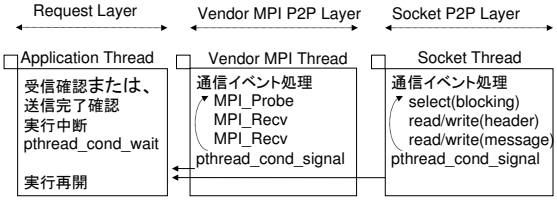

図 5 スレッド方式の処理

関数でベンダ MPI の送信の完了を確認して終了する。送信が完了していない間は、他の通信イベントの処理を行う。

受信処理を図 3、図 4 に示す。図 3 はメッセージ到着前に受信要求が行われた場合、図 4 はメッセージ到着後に受信要求が行われた場合である。リクエスト層からの受信要求で `MPI_Iprobe` 関数で通信イベントを待つ。メッセージが到着している場合には、`MPI_Recv` 関数でヘッダ情報とメッセージを受信する。図 3 に示すように、MPI Core 層がメッセージ受信を要求したメッセージであれば、指定されたバッファが存在する。この領域に受信したメッセージをコピーする。ロングメッセージである場合には続けて `MPI_Recv` 関数を呼び出す。図 4 に示すように、MPI Core 層がメッセージ受信を要求していないメッセージであれば、指定されたバッファが存在しない。そこで、`malloc` 関数により一時的なバッファ(一時バッファ) を割当て、この領域に受信したメッセージをコピーする。ロングメッセージである場合には続けて `MPI_Recv` 関数を呼び出す。MPI Core 層がメッセージ受信を要求した時に、一時バッファから指定されたバッファへコピーする。

2.2 スレッド方式

スレッド方式の処理を図 5 に示す。スレッド方式は、ソケット P2P を処理するスレッド (ソケットスレッド) とベンダ MPI P2P を処理するスレッド (ベンダ MPI スレッド) を生成する。ベンダ MPI スレッドは `MPI_Probe` 関数で、ソケットスレッドは `select` 関数で通信イベントの発生をブロッキングして待ち、その処理を行う。

リクエスト層から受信確認や送信完了確認など P2P 層の通信イベントを処理する呼び出しが行われた時は、アプリケーションの実行をブロッキングして通信処理待ちの状態となる。ベンダ MPI スレッドかソケットスレッドが通信イベントを処理した時に、アプリケーションの実行が再開される。

スレッド方式のリクエスト層は、ソケットスレッド、ベンダ MPI スレッドとアプリケーションのスレッドそれぞれから呼び出される。このため、スレッド方式のリクエスト層はスレッドセーフである必要がある。今回の実装では、アプリケーションのスレッドが使う MPI インタフェースとソケットスレッド、ベンダ MPI スレッドが使う P2P インタフェースを排他制御することで、同時にリクエスト層を呼び出せるスレッドを 1 つに制限する実装としている。

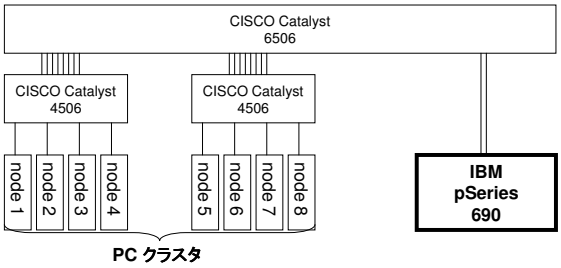

図 6 性能測定環境の構成

2.3 ポーリング方式

ポーリング方式は、第 2.1 節の送信処理、受信処理にソケット P2P の通信イベント処理を追加した形となる。ポーリング方式は、リクエスト層の受信確認や送信完了確認時に、ベンダ MPI P2P の通信イベント処理の後、ソケット P2P の通信イベント処理を行う。ベンダ MPI P2P で `MPI_Iprobe` 関数や `MPI_Test` 関数を呼び出してノンブロッキングで通信イベントを検出して処理した後、ソケット P2P で `select` 関数をノンブロッキングで呼び出すことで通信イベントを検出して処理する。通信イベントがない場合には、通信イベントを 1 つ処理するまで交互にこれらを繰返す。

2.4 プロセス起動

GridMPI では、各ノードにプロセスを起動するのに rsh や ssh を用いている。p690 でベンダ MPI を使用するには、POE 環境の `poe` コマンドで起動する必要がある。GridMPI の RPIM にこの機能を追加した。

また、GridMPI では、アプリケーションを実行するプロセスとは別に、これらを制御するプロセス (制御プロセス) を生成する。この制御プロセスのコードは GridMPI ライブラリをリンクしたアプリケーションプログラムに含まれている。アプリケーションプログラムは POE 環境で起動しなければならないので、制御プロセスもベンダ MPI の 1 プロセスとなり、アプリケーションが必要とするプロセス数より 1 多いプロセス数で起動しなくてはならない。

これを回避するため、制御プロセスを `mpirun` コマンドに含めるように実装している。なお、本実装はポーリング方式にのみ対応している。

3. GridMPI の性能評価

3.1 性能評価環境

図 6 に今回使用した性能評価環境を示す。p690 と 6506 スイッチは 1Gbps リンク 2 本の計 2Gbps で、PC クラスタの各ノードは 1Gbps で 4506 スイッチに接続した構成である。6505、4506 スイッチ間は 8 本の 1Gbps リンクをトランクして 8Gbps で接続している。

p690 の諸元を表 1 に示す。p690 は 32Way の共有メモリ型並列計算機である。PC クラスタの諸元を表 2 に示す。PC クラスタは 2 種種の PC で構成されているが、その性能はほとんど変わらない。

並列ベンチマークとして NAS Parallel Benchmarks(NPB) バージョン 2.4 を-O3 オプションでコンパ

表 1　IBM pSeries 690 の諸元

共有メモリ型演算サーバ	
CPU	POWER4+ 1.3GHz 32way
メモリ	64GB
NIC	1Gbps PCI card 2 個
OS	AIX 5L 5.2

表 2　PC クラスタの諸元

PC クラスタ (8 台)		
	HPC LinuxNetworx node 1〜4 (4 台)	NEC Express 5800 node 5〜8 (4 台)
CPU	Intel Xeon 2.80GHz 2way	
メモリ	1GB	
NIC	Intel 82546EB	
OS	Fedora Core release 1(2.4.28 SMP)	
NIC ドライバ	Intel(R) PRO/1000 5.4.11-k1-NAPI	

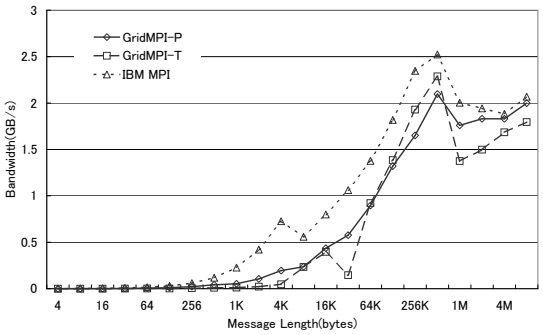

図 7　一対一通信のバンド幅

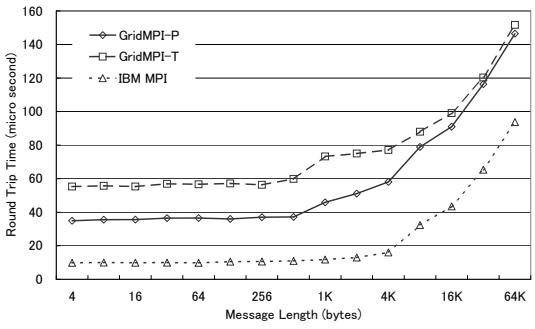

図 8　一対一通信のラウンドトリップ時間

イルしたものを使用した。

グラフ中で、GridMPI-T はスレッド方式で実行、GridMPI-P はポーリング方式で実行した結果を指す。また、(8) などの括弧内の数字は実行したプロセス数を示す。但し、NPB の BT、SP では (8) は 9 プロセス、(32) は 25 プロセスを指す。

3.2　p690 における GridMPI 基本性能

GridMPI の基本性能を確認するため、p690 上の通信性能の評価を行った。`MPI_INT` 型配列をメッセージとする一対一のバンド幅、ラウンドトリップ時間と `MPI_Alltoall` を使用した全体全のバンド幅を計測した。

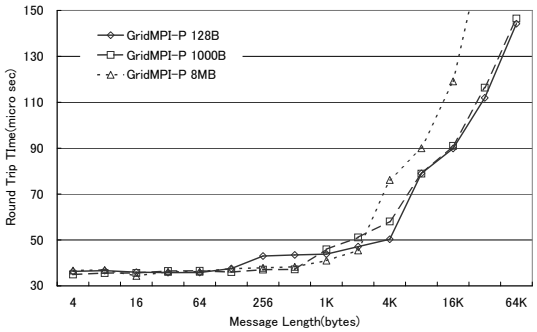

図 9　ショートメッセージのサイズを変更したラウンドトリップ時間

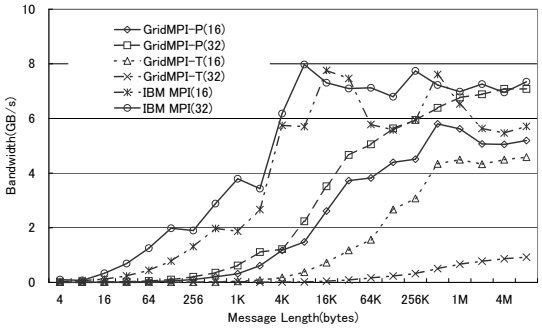

図 10　16, 32 プロセスによる全対全バンド幅

図 7 に一対一通信のバンド幅を、図 8 にラウンドトリップ時間を示す。IBM MPI のバンド幅は最大 2.5GB/s であり、メッセージ長が長くなると、スレッド方式はこのバンド幅を使い切ることができるが、ポーリング方式は少し低い性能となっている。スレッド方式はポーリング方式と比べてメッセージ長が小さい時にバンド幅やラウンドトリップ時間の性能が低い。これはスレッドの切替によるオーバヘッドが主な原因と考えられる。

なお、結果で IBM MPI が 4KB のメッセージ長を境に性能が変化するのは IBM MPI のメッセージ交換が 4KB 単位で行われているからである。

次にショートメッセージのサイズを評価する。ショートメッセージは第 2.1 節で説明したように、受信後にアプリケーションが指定したメモリ領域に常にコピーする必要がある。一方、ベンダ MPI の送受信は 1 回の操作で終了する。図 9 にショートメッセージのサイズを 128B、1000B と 8MB に変更したポーリング方式によるラウンドトリップ時間を示す。メッセージサイズが 2KB までは、ショートメッセージのサイズによる性能差はほとんどない。8MB のショートメッセージサイズでは、4KB のメッセージで IBM MPI のパケットサイズを超えてしまうので性能が低下する。また、4KB を超えるメッセージは、データコピーのオーバヘッドが大きくなり、ロングメッセージで送信する方が有利である。ショートメッセージのサイズは、2KB から 4064B の間が適切と考える。ショートメッセー

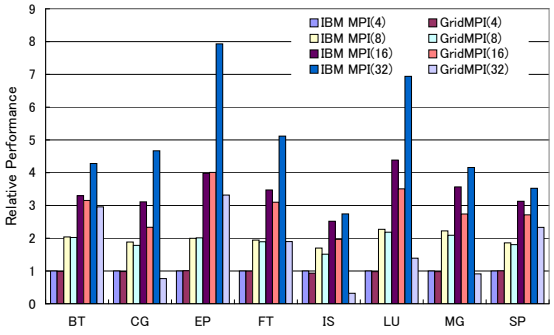

図 11 スレッド方式による p690 単体での GridMPI の NPB 性能

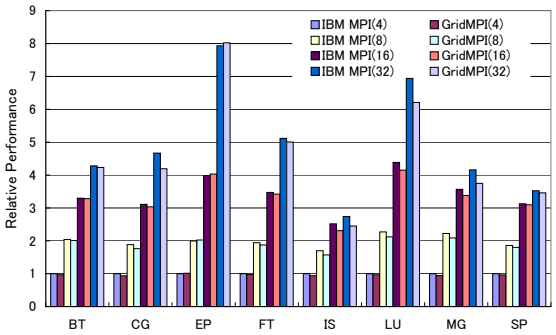

図 12 ポーリング方式による p690 単体での GridMPI の NPB 性能

ジサイズの 4064B は、これと GridMPI のヘッダサイズ 32B の合計が IBM MPI のパケットサイズ 4KB を超えない最大値である。

図 10 に `MPI_Alltoall` による全対全通信のバンド幅を示す。バンド幅は全プロセスが送信するバイト数を転送時間で割った値である。p690 内部の共有メモリによって得られるバンド幅は最大 8.0GB/s である。

スレッド方式の性能はポーリング方式より低下している。これは、IBM MPI がビジーウエイトしており、実行可能な状態にあるスレッド数が CPU 数を超えるためである。例えば 16 プロセスの並列実行では、アプリケーションのスレッドが 16 個、ベンダ MPI スレッドが 16 個、制御プロセスのベンダ MPI スレッドが 1 個の計 33 個のスレッドが実行可能な状態になり、CPU 数 32 を上回ることによって性能が落ちている。32 プロセスでは、さらに多くのスレッドが実行可能な状態になることで一層の性能低下が見られる。また、ポーリング方式でもメッセージ長が 32KB より小さいと性能が大きく低下している。

3.3 p690 単体における GridMPI の NPB 性能

図 11 にスレッド方式の、図 12 にポーリング方式の GridMPI の p690 単体における NPB 性能を計測した結果を示す。図は IBM MPI の 4 プロセスの実行結果で各ベンチマーク毎に正規化している。

スレッド方式では、16 プロセス以上になると性能が上がらなくなる。これは、前節で説明した、IBM MPI がビジーウエイトすることによる。一方、ポーリング方式では、このようなことは起こらない。8 プロセス以下では 2 つの方式に大きな性能差は見られない。32

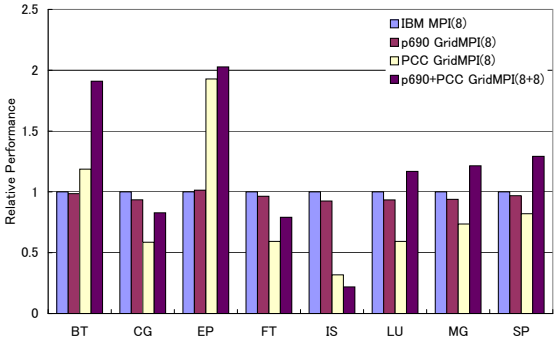

図 13 ヘテロ環境における GridMPI の NPB 性能

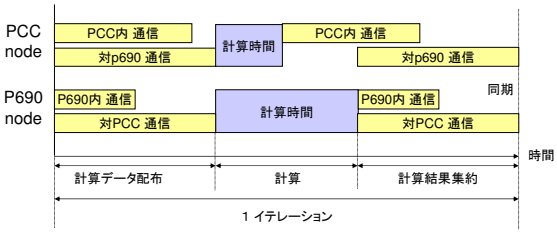

図 14 NPB のモデル

プロセスの実行において、ポーリング方式の GridMPI は IBM MPI の 90%以上の性能が得られる。

3.4 ヘテロ環境における GridMPI 性能

図 13 に p690 と PC クラスタで構成されたヘテロ環境 (演算性能や通信性能が不均一な計算機が混在した環境) における GridMPI の NPB 性能を計測した結果を示す。図は IBM MPI の 8 プロセスの実行結果で各ベンチマーク毎に正規化し、p690 における GridMPI はポーリング方式である。図の p690+PCC GridMPI(8+8) は p690 の 8 プロセスと PC クラスタの 8 プロセスの計 16 プロセスで実行した結果を指す。BT、EP、LU、MG と SP で p690 と PC クラスタの各々の単体性能を超えた結果が得られた。

NPB は各プロセスが同じ計算量となるように領域分割して各プロセスで計算を行う。このため、図 14 に示すように、データの配布、計算、集約を 1 イテレーションとして繰返すプログラムとしてモデル化される。PC クラスタの CPU 性能は p690 と比べて 2 倍の速度であるが、内部ネットワーク性能は p690 と比べて極端に悪いクラスタである。性能が悪い CPU 上のプロセスの計算時間は長くなり、また、性能が悪いネットワーク上でのデータの配布や集約 (データ交換) にかかる通信時間は長くなる。結果として同期ポイントで、性能の悪い CPU および性能の悪いネットワークに合わせることになり、NPB の性能は悪くなる。

データ交換をほとんどしない EP では、イテレーション毎にとる同期で遅いほうの p690 の CPU 実行時間に合わせることになる。ヘテロ環境での EP 性能は CPU 性能の悪い p690 の 16 プロセスで実行したのと同程度の性能にしかならない。

計算量に比べてデータ交換の比率が高い IS では、

PC クラスタ単体の性能よりヘテロ環境の性能が低下している。本評価環境では p690 と PC クラスタ間の接続に転送速度のボトルネックが存在する。データ交換するプロセス数が増加することで、一度に大量のパケットがボトルネックに集中した結果、大量のパケットロスが発生したため、性能が低下したと考える。これはヘテロ環境に限らずに同じ構成の PC クラスタ同士をボトルネックリンクで接続した構成でも、IS の性能は低下することが知られている。

CG では、倍精度浮動小数点演算の精度の違いにより計算が収束しなかった。この精度を合わせることで正しい結果が得られるようになった。詳細は付録 A.1 を参照されたい。

4. 関連研究

いくつかの MPI 実装がベンダ MPI のサポートを表明している。例えば、MPICH-G2[8]、Stampi[9] や PACX-MPI[10] などである。

MPICH-G2[8] は、ベンダ MPI をバイト列転送の P2P として利用する。クラスタ間の通信には、Globus-io と呼ばれる独自のプロトコルを使用している。ベンダ MPI と Globus-io は交互にポーリングしている。

Stampi[9] は、ベンダ MPI を P2P として利用するのではなく、MPI として利用している。これは一対一通信でも集団通信でもクラスタ内部で終結するメッセージ交換は、そのままベンダ MPI の呼び出しを実行している。

5. まとめ

GridMPI のベンダ MPI 実装においてスレッド方式とポーリング方式を実装して評価した。ポーリング方式で IBM MPI の 90%以上の NPB 性能を確認した。しかし、スレッド方式では p690 の IBM MPI がビジーウエイトするので、実行可能なスレッド数が CPU 数を超える場合に性能が出なくなった。p690 においては、スレッド方式よりもポーリング方式が有効である。

ヘテロ環境の NPB 性能では BT、EP、LU、MG と SP で単体性能を超えることが確認された。CPU 性能やネットワーク性能に大きな差があるクラスタで構成されたヘテロ環境では、各プロセスに均一な計算量を割当てる NPB の特性上、その性能がでにくいと言える。

今後は、p690 と PC クラスタ間の接続にネットワーク遅延を入れることにより、グリッド環境に近い環境で評価すると共に、他のベンダ MPI を実装することを考えている。

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) 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, 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, 2004.
3) GridMPI. http://www.gridmpi.org/
4) 石川. YAMPII もう一つの MPI 実装. 情報処理学会, SWoPP'04, 2004.
5) YAMPII, Yet Another MPI Implementation. http://www.ilab.is.s.utokyo.ac.jp/yampii/
6) 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.
7) PC Cluster Consortium. http://www.pccluster.org/
8) MPICH. http://www-unix.mcs.anl.gov/mpi/mpich/
9) 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.
10) PACX-MPI. http://www.hlrs.de/organization/pds/projects/pacx-mpi/
11) IBM Parallel Environment for AIX 5L Operation and Use, Volume 1,2 Using the Parallel Operating Environment Version 4 Release 1, 2000. (SA22-7948-00, SA22-7949-00)

付録

A.1 ヘテロ環境での浮動小数点演算

倍精度浮動小数点演算の精度が p690 の CPU POWER4+と PC クラスタの CPU Intel x86 ではデフォルトで異なるものになっている。POWER4+は乗算と加算をオーバーラップさせることで 1 クロックサイクルで 2 つの倍精度浮動小数点演算を行うことが出来る。この乗加算命令を使用した時に IEEE 標準の倍精度と異なる演算精度になる。また、Intel x86 では拡張倍精度 (80bits) を用いるので、これも IEEE 標準の倍精度と異なる演算精度になる。

今回の計測では、NPB の CG で正しい結果が得られなかった。このため、p690 のコンパイル時に `-qfloat=nomaf:rndsngl` オプションを、PC クラスタのコンパイル時に `-msse2 -mfpmath=sse` オプションを指定して、IEEE の倍精度に統一することで正しい結果が得られるようになった。なお、これらのコンパイルオプションは演算速度にほとんど影響しない。