ハイパフォーマンス
コンピューティング 87−12
（２００１．７．２６）

# MPI-2用ベンチマークプログラムライブラリMBL2の構築と評価

上原 均　津田 義典　横川 三津夫
日本原子力研究所 地球シミュレータ開発特別チーム

代表的なメッセージ通信 API である MPI には，基本仕様としての MPI-1 と，拡張仕様としての MPI-2 がある．MPI-1 で定義される関数のベンチマークは幾つか提案されているものの，MPI-2 で定義される関数のベンチマークプログラムは少なく，その計測項目も限られている．そこで我々は MPI-2 の，特に並列 I/O（MPI-I/O）と片側通信（RMA）の性能を詳細に測定する MBL2(MPI benchmark program library for MPI-2) を構築した．本稿では MBL2 の概要と幾つかの計算機の MPI-2 性能データを報告する．

# MBL2: MPI benchmark program library for MPI-2

Hitoshi Uehara　Yoshinori Tsuda　Mitsuo Yokokawa
Japan Atomic Energy Research Institute, Earth Simulator Development Team

MPI is one of major message communication interfaces for application programs. The MPI consists of an MPI-1 as a basic specification, and an MPI-2 as extensions. Some benchmark programs for MPI-1 have been proposed already. However benchmark programs for MPI-2 are a little and their measurements are limited. We have developed an MPI benchmark program library for MPI-2 (MBL2) which measures the detail performance of MPI-I/O and RMA functions of MPI-2. In this report, we describe the MBL2 and performance data of MPI-2 on VPP5000 and SX-5, which we measured using MBL2.

## 1　はじめに

科学技術計算分野では，大規模数値シミュレーションや実行の高速化の為に，従来から MPI[1, 2] や PVM[3] 等のメッセージ通信ライブラリを用いた分散メモリ型並列プログラムが開発されてきた．MPI ライブラリは，大気大循環シミュレーションなどの地球変動研究の為の超大規模計算向けプラットフォームとして現在開発中である地球シミュレータ [4] でも提供される事になっている．

メッセージ通信を用いた分散メモリ型並列プログラムを高速化するには，並列アルゴリズムの効率化や負荷均等化も大切だが，メッセージ通信部分の高速化も重要となる．その手法としては通信パターンの効率化や非同期通信による通信時間隠蔽等が考えられ，これらの高速化を行うには計算環境での通信性能値を把握する事が必要である．こうした性能値を考慮したソフトウェア開発は，特に地球シミュレータのような大規模計算機システムを効率的に運用する上で重要となる．

このような性能把握の必要性から，基本的な 1 対 1 通信や集合通信を定めた MPI-1 での関数群の性能を測定するベンチマークは幾つか提案されており，PMB[5] や beff[6] 等がある．しかし，並列I/O(MPI-I/O) や片側通信 (Remote Memory Access, RMA) 等の拡張的 API を定めた MPI-2 での関数群の性能を測定するベンチマークはあまり提案されていない．数少ないベンチマークの一つである PMB においても，その検証項目は少ない．その為，MPI-2 関数の性能を十分に検討・分析したり，MPI-2 関数を用いたユーザアプリケーションの効率化に有用なデータを得る事が難しいのが現状である．

そこで我々は，MPI-2 の API，特に MPI-I/O と RMA の性能を詳細に測定する MBL2(MPI benchmark program library for MPI-2)[7] の開発を試みた．この MBL2 では MPI-I/O と RMA の諸関数に関して多角的に測定でき，更にユーザによる拡張まで考慮した構造となっている為，MBL2 によって PMB 等で計測できるデータよりも詳細で有用なデータが提供できる．本稿では，この MBL2 の概要と，その MBL2 で計測した富士通 VPP5000 等での性能値について報告する．

## 2　MBL2の構築

### 2.1　構築方針

MBL2 の目的は，MPI-2（MPI-I/O, RMA）関数に関する詳細な性能データの取得である．それには単純な状況下での計測だけではなく，より実際的で複雑な通信状況下での計測や同等の機能を異なる

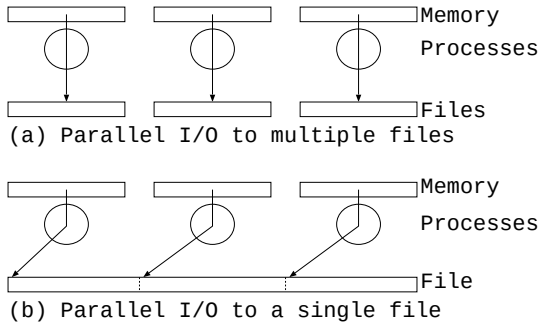


図 1: 並列 I/O のイメージ図

実装で実現した場合についての計測といった多角的な計測が必要不可欠である．半面，そのような多角的な計測は測定項目の爆発的な増大を招き，仮に単体プログラムとした場合にはコードが巨大化して，保守性や拡張性が低下してしまうデメリットが大きい．

既存の PMB や beff を見てみると，これらでは基本的に実行時オプションで測定項目を指定する方式を取っている．しかし MBL2 で想定される爆発的に増大した測定項目を，実行時オプションで制御するのはあまりに煩雑である．

そこで我々は，1) 測定は設定ファイルに基づいて行う, 2) 完全に分離できる MPI-I/O と RMA の測定を別プログラムで行う，という方針で MBL2 を開発した．また PMB や beff は C 言語で記述されているが，Fortran ユーザが MPI を利用する事を考慮し，よりユーザの利用環境に近い状態で測定すべく MBL2 も Fortran でコーディングした．

## 2.2 MPI-I/O の測定詳細

MPI-2 では一般的なファイル I/O 関数以外に，複数のプロセスが一つのファイルにアクセスする並列 I/O が提供される．従来の I/O でも図 1(a) のような複数プロセスから複数ファイルへの I/O は可能であったが，MPI-2 で提供される並列 I/O は図 1(b) のような 1 ファイルに対する I/O であり，集合的処理も可能である．これら MPI-I/O での Read/Write 関数群は，以下の 3 点から分類できる．

- ポジショニング（ファイルポインタ種別）: 明示指定／プロセス毎独立／共有
- 同期性 : ブロッキング／非ブロッキング
- 処理 : 非集合／集合的

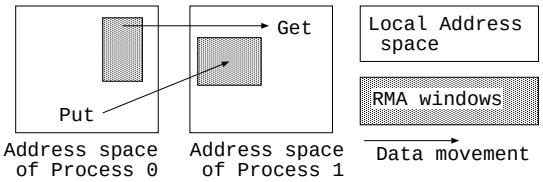


図 2: RMA 通信のイメージ図

この分類は直交しているので，Read/Write 関数は各々$3 \times 2 \times 2 = 12$ 種類定義されている．よって MPI-I/O での処理を効率化するには，各関数の性能特性を把握した上で適切に選択しなければならない．そこで MBL2 では，全 I/O 関数の性能測定を行えるようにし，更に Sync 処理を含む場合やシーク処理の性能測定も行えるようにした．

## 2.3 片側通信 (RMA) の測定詳細

MPI-2 で定義された RMA とは，図 2 に示すような，あるプロセスで定義したウィンドウと呼ばれる特定メモリ領域を他のプロセスから直接的にアクセスできる（ように見える）通信手段である．MPI-2 の RMA では，`MPI_Get`，`MPI_Put`，`MPI_Accumulate` の三関数によって，データの受信／送信／操作が可能である．`MPI_Accumulate` での操作では，MPI-1 の`MPI_Reduce` 関数と同様の操作（例えば総和）が指定できる．RMA は，送信側のみ又は受信側のみに通信を記述すれば良いのでコーディング量が少なくすみ，通信記述が複雑な場合で有用である．

よって，RMA の計測では各関数単体での性能測定だけでなく，むしろ複雑な，特に実際のアプリケーションに見られるような通信パターンでの性能評価が重要になる．また，RMA は非同期通信の一種なので，MPI-1 での非同期通信（`MPI_Isend`/`MPI_Irecv`）との性能比較も，RMA 利用そのものを検討する観点から重要である．

そこで MBL2 では，前述の各関数単体での性能以外に，実際のアプリケーション中で良く見られる shift 通信や全プロセスが自分以外のプロセス全てにデータを分配する exchange 通信等での性能を測定できるようにした．またウィンドウ作成の所要時間や fence 処理時間を測定し，RMA 処理全般の性能も評価できるようにした．

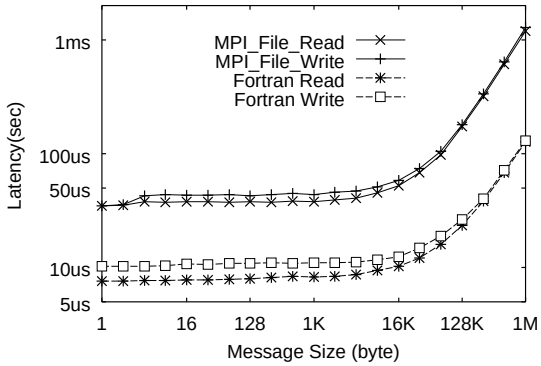


図 3: VPP5000 での Read/Write 所要時間

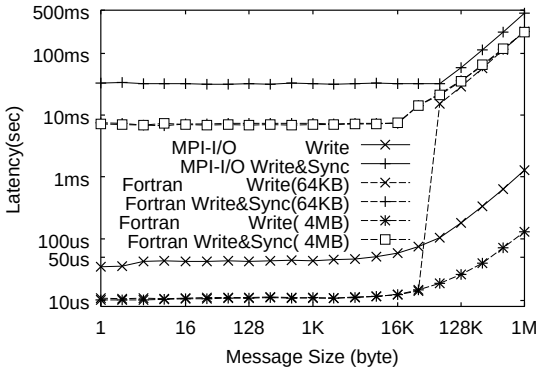


図 4: VPP5000 での Write 所要時間 (Sync 有無)

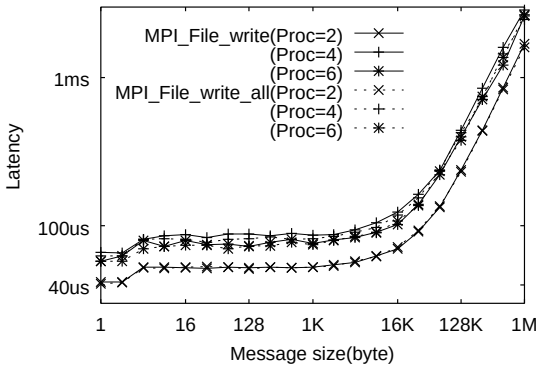


図 5: VPP5000 での並列書き込みの所要時間

### 2.4 フレームワーク技術の利用

MPI-2 を実際に利用するユーザにとって最も有用なデータとは，自身のアプリケーション中にある MPI-2 関数の性能値やそれを用いた処理での性能値であろう．これは自身のアプリケーションの効率化に反映させやすい為であるが，反面，各個人毎に異なる要求である為，全ユーザの要求を満たす測定の記述を事前に行うのは事実上不可能である．

そこで MBL2 では，ユーザ独自の測定パターンが必要な場合はユーザ自身で MBL2 を拡張して所望の性能データを得られるように，拡張・変更が容易なフレームワーク技術 [8] を導入した．具体的には，共通利用できるメイン関数を変更の無い部分（フローズンスポット）に，各計測パターン毎に記述が異なる計測ルーチンを変更の多い部分（ホットスポット）としてプログラムを構造化し，更に予めユーザが拡張できる計測ルーチンも組み込んだ．ユーザは拡張用ルーチン内部を記述するだけで独自の計測パターンを追加できる．また既存の計測パターンの変更も，ごく限定された範囲の修正のみで可能である．

## 3 MBL2による性能測定例

### 3.1 VPP5000 での計測とその評価

ここでは日本原子力研究所に導入された富士通 VPP5000 (9.6Gflops/PE, 64PE, クロスバ 1.6GB/s ×2 /PE) 上の MPI-2(MPLIB-sr2.3.1, Patchlevel: 2.0.17) について，MBL2 を用いて性能測定した結果を示す．MBL2 は VPP5000 フロント（Solaris）上の F90 クロスコンパイラでコンパイルした（主なオプション：`-O3`, `-novdopt`）．

まず MPI-I/O に関する計測として，1 プロセスが 1 ファイルを読み書きした場合の所要時間を計測した[1]．各 I/O データ量について各 100 回試行し，ランク 0 で所要時間を計測した．また比較のため，Fortran での read 関数/write 関数を用いた場合も同様に計測した．これらの計測結果を図 3 に示す．この結果から MPI-I/O では Read で最大 0.88GB/s, Write で最大 0.82GB/s のスループットが確認された（Read/Write 共に 1MB 長時）．また 1byte 時の所要時間は Read で 35.3$\mu$sec, Write で 35.4$\mu$sec であった．

表 1: MPI-I/O 諸関数の所要時間

| 測定項目 | 所要時間 |
|---|---|
| `MPI_File_Seek` 関数 | 1.05 $\mu$sec |
| `MPI_File_Seek_Shared` 関数 | 3.61 msec |
| `MPI_File_Sync` 関数（2 ノード） | 5.27 msec |
| （4 ノード） | 10.46 msec |
| （6 ノード） | 13.69 msec |

[1] 当該環境では NFS, FPFS, DPFS の三種のファイルシステムを利用できるが，ここでは NFS での性能値を示す．

さらにファイルへ変更を書き込むSync処理を含めたWrite性能も測定した．この結果を図4に示す．Sync処理を含む場合の処理時間はMPI，Fortran共に非常に増大している事から，VPP5000上でのSync処理は相当に高コストと言える．また16KB長以下では所要時間があまり変化していない事から，Sync処理ではスタートアップコストが比率的に大きく，アプリケーションの最適化ではSync回数の低減が有効と思われる．

また図4では，Fortran I/Oバッファサイズ4MB時に対して64KB時は性能が大きく低下する事も確認できる．MPI-I/Oでは通信バッファサイズを変更したが，性能変化は見られなかった．これにより，測定した環境でのMPI-I/Oでは，Fortran I/Oバッファや MPI通信バッファと異なるバッファを用いていると推測される．

ここまでのMPI-I/OとFortran I/Oの性能を比較すると，総じてFortranのI/O性能の方が優れているといえる．これは，その性能差からI/Oバッファ処理の差（例：コピー処理回数など）に因るものではないかと推測する．

複数プロセスからの1ファイルへの並行書き込みについては，`MPI_File_write`関数を用いた実装と`MPI_File_Write_all`関数を用いた実装の両方を測定した．この結果を図5に示す．この計測からVPP5000では，差は小さいが同数ノードでの性能は`MPI_File_Write_all`関数の方がやや優れる事が確認された．

ファイルシークや書き込み量0byte時のSync操作の性能測定結果を表1に示す．この表から，各プロセス間で共有される共有ファイルポインタでのシーク操作（`MPI_File_Seek_Shared`関数）のコストの高さが明白である．またSync操作を行う`MPI_File_Sync`関数では，ファイルにアクセスするプロセス間で同期を取る為にプロセス数増大に併せたコストの増大が顕著である．

次にRMAの計測結果例として，`MPI_Get/MPI_Put/MPI_Accumulate`関数のレイテンシとスループットを各々図6，図7に示す．`MPI_Accumulate`関数でのオペレーションにはバイナリ・アンド（`MPI_BAND`）を指定した．この計測での最大スループットおよび1byte時の所要時間を表2に示す．ここで最大スループットはハードウェア性能にごく近い，良好な値が得られている．

次に自分以外の全プロセスにデータを転送する

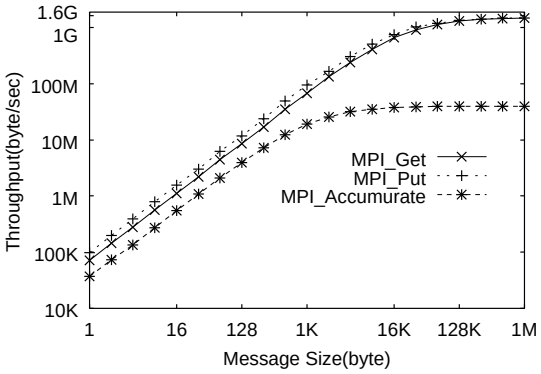


図 6: VPP5000でのRMA関数のスループット

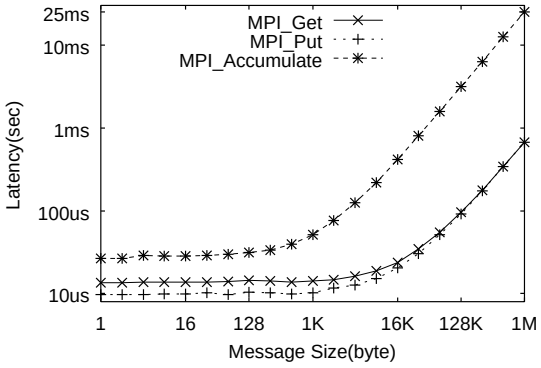


図 7: VPP5000でのRMA関数の所要時間

Exchange通信パターンについて，1メッセージ長最大1MBまで，プロセス数2,4,6の場合について，ランク0での平均処理時間（fence処理含む）をレイテンシとして測定した．ここでは，`MPI_Put`関数と`MPI_Win_fence`関数を用いた実装と，MPI-1で定義される`MPI_Isend`関数/`MPI_Irecv`関数による実装について測定した．その結果を図8に示す．

ここでRMA通信が比較的短いメッセージ長時では`MPI_Isend/MPI_Irecv`実装よりも性能的に優れているが，1メッセージが16KB長を越える付近で逆転している．これらの事から，単にRMA通信が

表 2: VPP500上の最大スループットと所要時間

| 関数名 | 最大スループット |
|---|---|
| `MPI_Get` | 1.56GB/s |
| `MPI_Put` | 1.57GB/s |
| `MPI_Accumulate` | 0.042GB/s |
| 関数名 | 1byte指定時の所要時間 |
| `MPI_Get` | 13.49$\mu$sec |
| `MPI_Put` | 9.81$\mu$sec |
| `MPI_Accumulate` | 26.44$\mu$sec |

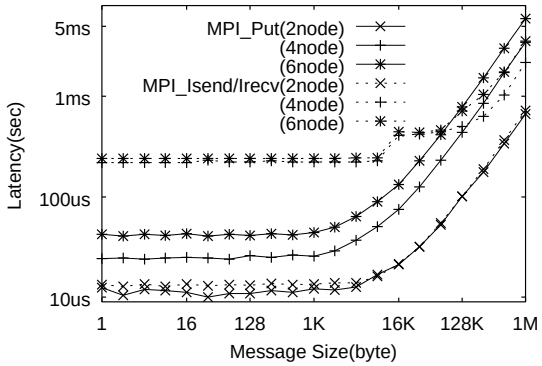


図 8: VPP5000 での Exchange 通信の所要時間

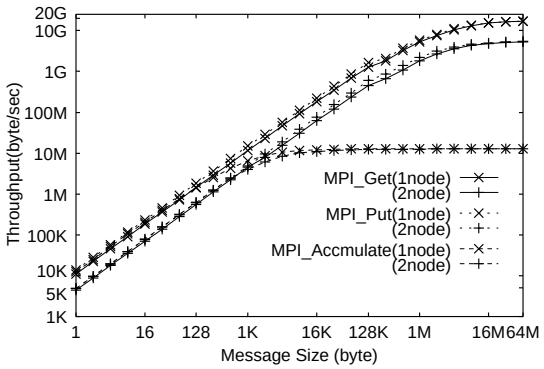


図 9: SX-5 上での RMA（Get/Put）スループット

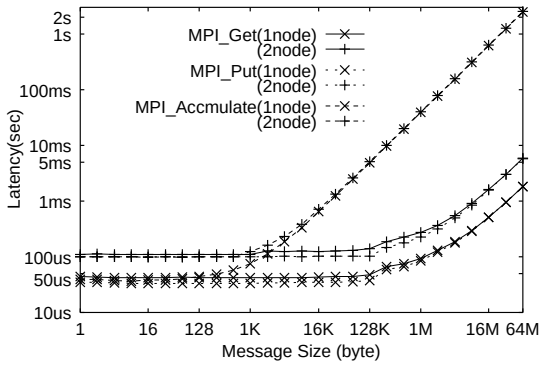


図 10: SX-5 上での RMA（Get/Put）処理時間

従来の 1 対 1 通信より優れている訳ではなく，通信パターンやメッセージ長で選択すべきである事が判る．なお VPP5000 では PE がクロスバ結合である為に，通信経路衝突等による性能低下は特に見られなかった．

この Exchange 通信では，プロセス数 2, 4, 6 での送受信データ総量の比が 1 : 6 : 15 となる．この比は，`MPI_Put` 関数を用いた実装でのメッセージ長 8KB 以上のレイテンシ実測値での比とほぼ合致した．逆に 8KB 未満が合致しない原因としては，バッファリングの影響と fence 処理時間が実通信時間に対して相対的に大きい事が考えられる．

更に RMA 通信を行わない状態での fence のみのコストは 2 ノードで 5.79$\mu$sec，4 ノードで 11.26$\mu$sec，6 ノードで 25.26$\mu$sec であった．`MPI_Win_fence` 関数による同期処理ではプロセス数増大に比したコスト増大は予測されたが，この結果から更に多数のプロセス数では fence コストが相当な負荷となる事が予測される．

### 3.2 SX-5 での計測とその評価

ここでは NEC SX-5(8GFlops/PE, 8cpu/1node + 16cpu/1node, 8GB/s クロスバ接続) での MPI-2 での RMA 性能を示す．この計測環境ではメモリファイルシステムを用いた構成であった為，非常に高い MPI-I/O 性能値が計測された．しかし，その性能値は実際の運用環境での性能値と大きく異なると考えられる．

従って SX-5 上の計測では，1 ノードに 2 プロセス配置時と 2 ノード 2 プロセス（1 ノード 1 プロセス）配置時で，RMA の三関数について計測した．そのスループットとレイテンシを各々図 9, 図 10 に示す．この計測では，16MB 以上のメッセージ長でスループット向上率が小さくなったので，64MB 長時のスループットを最大スループットとして表 3 に示す．またデータサイズ 1byte 指定時の所要時間は，表 4 に示す通りである．ここで 1 ノード時が 2 ノード時よりも格段に高性能な理由は 1 ノード時はメモリ転送の RMA 関数が実装されているためと思われる．

## 4 PMB との比較と考察

MPI-1 に関するベンチマークは多々存在するが，MPI-2 に関するベンチマークは我々が調査した範囲では，PMB しか詳しいドキュメントやコード，計測データを入手できなかった．ここでは MBL2 と PMB[5] を比較検討する．

この PMB(Pallas MPI Benchmarks) とは Pallas 社が公開している MPI 用のベンチマークソフトで，MPI-1 用と MPI-2 用がある．この PMB の Web サイト [5] ではコードとドキュメント，更には NEC や富士通，日立などの各社のマシン上での計測データも公開されている．

まず測定項目数で比較すると，PMB(part.MPI-

表 3: SX5 での RMA のスループット

| | | |
|---|---|---|
| 1 ノード | Get | 18.26GB/s |
| | Put | 18.35GB/s |
| | Accumulate | 0.014GB/s |
| 2 ノード | Get | 5.71GB/s |
| | Put | 5.77GB/s |
| | Accumulate | 0.013GB/s |

表 4: SX5 での RMA のレイテンシ

| | | |
|---|---|---|
| 1 ノード | Get | 44.64$\mu$sec |
| | Put | 34.75$\mu$sec |
| | Accumulate | 39.26$\mu$sec |
| 2 ノード | Get | 110.95$\mu$sec |
| | Put | 100.79$\mu$sec |
| | Accumulate | 100.81$\mu$sec |

2) では MPI-I/O については 37 項目，RMA については僅か 6 項目である．それに対して MBL2 では MPI-I/O については 58 項目，RMA については 25 項目であり，より詳細なデータが得られる．

次に測定内容で比較すると，PMB では MPI-I/O では Read/Write, RMA では Get/Put/Accumulate と一通りは計測するが，ファイル Sync や RMA での fence の性能測定が行われず，MPI-I/O での Read/Write 以外の測定は僅か 1 項目（open/close）しかない．MBL2 では，これら PMB で計測しない点についても計測している．多数のプロセスによる大規模並列処理ソフトウェアを効率化する上で，同期処理が含まれるファイル Sync や RMA での fence のコストは重要な検討項目である事を考えると，MBL2 による計測結果は有用である．

また I/O 性能や RMA の各関数の性能測定についても，PMB では一つずつしか測定パターンが用意されていない．しかし MBL2 では異なるファイルポインタを用いた場合や Sync 処理を含む場合等，計測状況を変えた複数個の計測パターンを用意しており，より詳細な性能測定を可能としている．

測定結果については，PMB では複数回（10〜1000 回程度）の試行結果の平均値を出力するが，特に I/O 性能やノード間通信時性能は OS 等の影響を受け易く大きく変動しやすいので，単なる平均値のみでは正確な性能評価（例：処理時間の変動）が難しい．MBL2 では平均値の算出方法にも選択肢を用意している他，更に詳細出力モードをコンパイル時オプションとして用意しており，それを用いる事で平均値だけで無く，各試行結果を出力でき，詳細な性能評価を行う事が出来る．

以上の点から，MBL2 は MPI-I/O や RMA の諸関数に関する性能詳細を評価する上で既存の PMB よりも有用であると考えられる．

## 5 まとめ

本稿では，我々が開発している MBL2 の概要と，それによる性能測定結果を述べた．MBL2 では，MPI-2 の各関数に関して多角的に性能データを取得する．これらの性能データはマシン毎に異なる為，そのマシン用の応用ソフトウェアを開発／効率化する上で重要と思われる．また現在開発中である地球シミュレータの性能評価にも有効と考える．

また MBL2 は，MBL[9] 構築でも用いたフレームワーク技術を用いて開発しているので，種々の計測パターンを容易に追加／変更できる．その特長を活かすと，より多くのアプリケーションの開発／効率化に必要な性能データが速やかに得られる．今後は種々のマシンで MPI-2 の性能を測定し，有用な情報を得られるライブラリにする予定である．

## 参考文献


[1] MPI Forum. MPI: A Message-Passing Interface Standard, 1995.

[2] Message Passing Interface Forum. MPI-2: Extensions to the Message-Passing Interface. Technical report, http://www.mpi-forum.org., 1997.

[3] Al Geist and other. *PVM: Virtual Machine*. MIT Press, 1994.

[4] 地球シミュレータ研究開発センターホームページ. `http://www.gaia.jaeri.go.jp`.

[5] Parras MPI Benchmarks. `http://www.pallas.de/pages/pmbd.html`.

[6] Effective Bandwidth Benchmark. `http://www.hlrs.de/mpi/b_eff/`.

[7] 上原均, 津田義典, 横川三津夫. MPI-2 用ベンチマークプログラムライブラリ MBL2 の開発. 並列シンポジウム JSPP2001 論文集, pp. 91–92, 2001.

[8] E. Gamma, R. Helm, R. Johnson, and J. Vlissides. *Design Pattern Elements of Reusable Object-Oriented Software*. Addison Wesley Longman, 1995.

[9] 上原均. MPI ベンチマークプログラムライブラリの開発. Technical report, 日本原子力研究所, 2000. JAERI-Data/Code 2001-010.