並列アプリケーションによる MPI/MBCF の評価

森本 健司　　松本 尚　　平木 敬
{morimoto, tm, hiraki}@is.s.u-tokyo.ac.jp
東京大学大学院理学系研究科情報科学専攻

概要

汎用超並列オペレーティングシステム SSS–CORE 上のメモリベース通信 MBCF を用いて実装された MPI ライブラリ MPI/MBCF の性能を、NAS Parallel Benchmarks の実行により評価した。MPI/MBCF は MBCF のメモリベース FIFO によりライブラリが提供すべきメッセージのバッファリングを、遠隔メモリ書き込みによりメッセージのバッファリングを必要としない通信を実現する。この実装の性能をワークステーションクラスタ上で並列アプリケーションにより評価し、共有メモリ通信機能であるメモリベース通信を用いてメッセージパッシングライブラリを実現することの有効性を検証する。

Performance Evaluation of MPI/MBCF with Parallel Applications

Kenji Morimoto, Takashi Matsumoto, and Kei Hiraki
Department of Information Science, Faculty of Science, University of Tokyo

Abstract

We evaluated the performance of the MPI/MBCF by executing the NAS Parallel Benchmarks. The MPI/MBCF is an MPI library implemented with the Memory-Based Communication Facilities (MBCF) on the SSS–CORE, a general-purpose massively-parallel operating system. To implement the MPI/MBCF, *Memory-Based FIFO* of the MBCF is used for message buffering provided by the MPI library, and *Remote Write* for communication without message buffering. This paper shows performance evaluation of the MPI/MBCF on a cluster of workstations with parallel applications, and verifies whether it is effective to construct a message passing library with the MBCF which are based on the shared memory model.

1　はじめに

分散メモリ型並列計算機環境での通信モデルとして、メッセージパッシングモデルと共有メモリモデルとが挙げられる。メッセージパッシングモデルではタスクの間に通信路を設け、この通信路に対してメッセージを send, receive することで通信を行う。このモデルはデータの転送媒体であるプロセッサ間通信ネットワークに着目し、仮想化したものである。一方、共有メモリモデルでは、タスク間で共有するアドレス空間を設け、このアドレス空間に対する load, store[1]により通信を行う。このモデルはプロセッサの操作の対象であるメモリ空間に着目して仮想化したものであり、「メモリベース」のモデルと言える。

これらのモデルを通信モデルとして見た場合、一方により他方をエミュレート可能であり、2 つのモデルは同等の表現力を持つ。従来はこの 2 つのうち、通信機能としてメッセージパッシングモデルの方が効率的であるという主張がなされ、多くの通信ライブラリがメッセージパッシングモデルに基づいて実装された。しかし、通信の最適化や効率の観点からは、プロセッサや MMU といったメモリを対象としたアーキテクチャによるサポートを受けることのできる共有メモリモデルに基づく通信機能の方が、原理的に優っている [9]。

我々はこれまでの研究で、メッセージパッシング

[1] これらの操作は必ずしもマシン語命令の load, store による細粒度メモリアクセスを意味するわけではない。

モデルでのプログラミングのための標準的なライブラリである MPI (Message Passing Interface Standard) [3, 4] を、共有メモリ通信機能であるメモリベース通信機能 (MBCF: Memory-Based Communication Facilities) [7] を用いて MPI/MBCF としてワークステーションクラスタ上に実装してきた [10]。この実装の基本性能の評価により、共有メモリ通信機能を用いてメッセージパッシング通信ライブラリを実装することの優位性を示している。

本稿では、並列計算機のためのベンチマークとして広く用いられている NAS Parallel Benchmarks (NPB) [1, 2] 中のアプリケーションプログラムを MPI/MBCF 上で実行し、実アプリケーションでの MPI/MBCF の性能評価を行った。

本稿の構成は以下の通り。2 章で MPI/MBCF およびそのベースに用いた MBCF に関して説明し、それらの基本性能を示す。3 章で NPB を用いた性能評価の結果を示し、4 章で主張をまとめる。

2 MPI/MBCF

2.1 メモリベース通信 MBCF

MBCF はソフトウェアによる遠隔メモリアクセス機能で、現在当研究室で開発を進めている汎用超並列オペレーティングシステム SSS–CORE [6] 上で提供されている。MBCF の特徴を以下に挙げる。

1. 通信先メモリへの直接的な操作が可能である
2. 保護・仮想化をページ管理機構・アドレス変換機構を用いて実現する
3. 汎用の通信ハードウェアを用いる
4. チャネルレスである
5. 通信先での操作が高機能である
6. 通信の到着保証、順序保証がなされている

MBCF の機能のうち、MPI/MBCF の実装には遠隔メモリ書き込みおよびメモリベース FIFO を用いている。遠隔メモリ書き込みはデータをリモートタスクの仮想アドレス空間に直接書き込む機能である。メモリベース FIFO は受信側ユーザが自分のアドレス空間に確保した領域を FIFO (リングバッファ) としてシステムに通知し、リモートからの書き込みを可能とするものである。

ワークステーションを 100Base-TX ネットワークにより接続した環境での MBCF の性能は以下の通りである。計測に使用した機器は、ノードワークステーションとして Axil 320 model 8.1.1 (Sun SS20 互換機、85 MHz SuperSPARC CPU × 1)、ネットワークインタフェイスとして各ノードワークステーションに Sun Microsystems Fast Ethernet SBus Adapter 2.0 、ネットワークとして SMC TigerStack 100 5324TX (HUB) および Bay Networks BayStack 350T (スイッチング HUB) である。OS としてワークステーションクラスタ版 SSS–CORE Ver. 1.1 を使用した。2 ノード間での遠隔メモリ書き込みの one-way latency および peak bandwidth を計測し、データサイズ 4 byte 時の one-way latency **24.5 μs**、データサイズ 1408 byte 時の peak bandwidth **11.48 MByte/s** (半二重ネットワーク) および **11.93 MByte/s** (全二重ネットワーク) という結果を得た。これらの値は 100Base-TX のハードウェア性能を効率良く引き出した値であり、専用の内部相互結合網を持つ MPP とほぼ同等の性能を持つことが示される [8]。

2.2 MPI/MBCF の実装方式

メッセージパッシング送信を行う場合、通信対象として指定されるのはタスク間に設けられた通信路のみであり、対象アドレス空間内での受信位置は指定されない。このためメッセージパッシング通信システムでは、送信側はデータを一旦送信側ないしは受信側に設けた固定バッファに転送し、対応する受信が発行後に改めてこのバッファから受信用領域にデータをコピーする。しかし、受信が送信に先行する場合には、受信側はあらかじめ受信位置を送信側に通知することが可能である。この情報を利用することで、共有メモリ通信システムでは送信側はデータを受信用領域へ直接書き込むことが可能となる。

MPI/MBCF の実装では、MBCF の遠隔メモリ書き込みにより受信側からの通知 (送信要求) がある場合の送信操作を、メモリベース FIFO により送信要求がない場合の (固定バッファへの) 送信操作を実現することにより、高速なメッセージパッシング通信ライブラリを提供する。具体的には、送信起動関数は以下のように実装される。

1. 受信側からの送信要求を調べる。送信と対応するものがあれば遠隔メモリ書き込みにより受信バッファへデータを転送し、終了。対応するものがなければ 2 へ。
2. 受信側のメモリベース FIFO に対しメッセージ (ヘッダおよびデータ) を転送し、終了。

受信起動関数は以下のように実装される。

1. (先行する受信の 2 のステップにおいて) メモリベース FIFO から既に取り込まれたメッセー

ジのリストを調べる。受信と対応するものがあればリストから受信バッファへデータをコピーし、終了。対応するものがなければ 2 へ。

2. メモリベース FIFO からヘッダを取り込み、受信との対応を調べる。対応するならばデータを FIFO から受信バッファへ取り込み、終了。対応しないならば (ヘッダを付加しつつ) データをリストに取り込み、 2 へ。 FIFO が空ならば 3 へ。

3. 送信側に送信要求を送り、終了。

送信要求の管理にはメモリベース FIFO を用いる。メッセージと送信要求との混在を防ぐため、これらは別々の FIFO に格納される[2]。さらに、異なる MPI プロセスからのメッセージ・送信要求の混在を防ぐために、 MPI プロセス数 n に対して $2n$ 個のメモリベース FIFO を用いる。

メッセージと送信要求にはそれぞれ通し番号を付与し、送信要求が最新のメッセージ到着状況を反映しているかどうかを送信側において調べる。これにより、対応するメッセージと送信要求とが同時に発行された場合には、送信側において送信要求を棄却し、送信と受信とを正しく対応させることを可能とする[3]。

2.3 MPI/MBCF の基本性能

2.1 節と同様のワークステーションクラスタ環境において MPI/MBCF の基本性能 round-trip time および peak bandwidth を測定した。

round-trip time は 2 つの (異なるノード上の) MPI プロセスの間をメッセージが往復するのに要する時間である。 2 つのプロセスは各々前もってノンブロッキング受信 `MPI_Irecv()` を起動し、一方は `MPI_Send()` (ブロッキング送信), `MPI_Wait()` (受信完了待ち) を、他方は `MPI_Wait()`, `MPI_Send()` を実行する。表 1 に、前者のプロセスにおける送信の開始から受信の完了までを round-trip time として測定した結果を示す。

peak bandwidth は 2 つの (異なるノード上の) MPI プロセスの間で、一方から他方へ連続的にメッセージを転送した際の転送バンド幅である。半二重ネットワークおよび全二重ネットワーク上で測定した peak bandwidth を表 2 に示す。

本節に示した MPI/MBCF の性能は 2.1 節に示した MBCF の性能に近い値と言え、 MPI を実現するためのオーバヘッドが小さいことが分かる。特に、 MPI/MBCF による通信は、低レンテンシで、かつ、メッセージサイズに対する bandwidth の立ち上がりが鋭いため、細粒度通信が必要なアプリケーションや通信の統合が充分になされていないアプリケーションにおいても通信効率を保つことが可能であることが示される。

3 NAS Parallel Benchmarks による性能評価

3.1 NAS Parallel Benchmarks

NAS Parallel Benchmarks (NPB) は、 NASA Ames Research Center において開発された航空力学数値シミュレーションプログラムを基にした、並列計算機向けのベンチマークである。問題とそれを解くアルゴリズム、問題サイズのクラス、プログラミングモデルを定めた NPB 1.0 [1] と、 MPI を用いた実際のプログラムを提供する NPB 2.x [2] とがある。以下の 5 つのカーネルプログラムおよび 3 つの計算流体力学 (CFD) アプリケーションからなっている。

- カーネル
  - **EP** 乗算合同法による正規乱数の生成
  - **MG** 簡略化されたマルチグリッド法による 3 次元ポアソン方程式の解法
  - **CG** 共役勾配法による正値対称疎行列の最小固有値問題の解法
  - **FT** 高速フーリエ変換による 3 次元偏微分方程式の解法
  - **IS** 大規模整数ソート
- CFD
  - **LU** Symmetric SOR による LU 分解
  - **SP** 非優位対角なスカラ五重対角方程式の解法
  - **BT** 非優位対角な 5×5 ブロックサイズの三重対角方程式の解法

NPB 2.x では IS のみ C + MPI で、 IS 以外は Fortran90 + MPI で記述されている。 8 つの問題それぞれに関して問題サイズが小さい方から順に class S (サンプル)、 class W (小規模ワークステーションク

[2] メモリベース FIFO はユーザのアドレス空間に複数個設けることが可能である。

[3] 送信要求は棄却される可能性があるため、送信要求を出した受信の全てが必ず遠隔メモリ書き込みによりデータを受け取るとは限らない。よって、受信操作の 2 においてメモリベース FIFO から取り込まれたヘッダはまず、以前に発行されたペンディングな受信のリストに対して比較される必要がある。

表 1: 100Base-TX における MPI/MBCF の round-trip time

message size (byte)	0	4	16	64	256	1024	4096
round-trip time (μs)	**71**	85	85	106	168	438	1026

表 2: 100Base-TX における MPI/MBCF の peak bandwidth

message size (byte)	4	16	64	256	1024	4096	16384	65536	262144	1048576
半二重 (Mbyte/s)	0.14	0.53	1.82	4.72	8.08	9.72	**10.15**	9.78	9.96	10.00
全二重 (Mbyte/s)	0.14	0.57	1.90	5.33	10.22	11.68	11.77	11.85	11.85	**11.86**

ラスタ向け)、 class A (中規模ワークステーションクラスタ向け)、 class B (中規模並列計算機向け)、 class C (大規模並列計算機向け) というクラス分けがなされている。

3.2 評価環境

2.1 節および 2.3 節と同様のワークステーションクラスタ環境で、 SSS–CORE 上の MPI/MBCF を用いて NPB Rev. 2.3 を実行し、ワークステーションの台数を変化させて実行時間を測定した。ネットワークにはスイッチング HUB を用いた。比較のため、同一の機器の上で SunOS 4.1.4 上の MPICH Ver. 1.1 [5] を用いた場合の実行時間を測定した。 MPICH は Argonne National Laboratory および Mississippi State University において開発された MPI の実装で、ワークステーションクラスタに対する実装では TCP ソケットによる通信を行っており、メッセージパッシング通信機能を用いた実装となっている。

コンパイラとして gcc-2.7.2.3 および g77-0.5.21 を用いた。 8 つの問題のうち FT のプログラムは g77 ではコンパイルできないため今回の評価の対象から省いた。

問題サイズとして class W を用いた。これは、 class A のプログラムが、 SSS–CORE 上では動作したものの SunOS 4.1.4 上ではメモリ不足により実行できなかったためである。

3.3 評価結果

まず表 3 に、 7 つのベンチマークプログラムの特性を示す。これらは MPI/MBCF のソース中に計数コードを挿入し、ノード数 8 (SP, BT は 9) の条件の下でプログラムを実行し、測定したものである。表の項目のうち、通信データレートは MPI 関数によって送信されたデータのバイト数 (ヘッダ部分は除く) を全ノードに関して合計し、実行時間で割ったものである。通信メッセージレートは送信されたメッセージの個数を全ノードに関して合計し、実行時間で割ったものである。遠隔メモリ書き込み利用率は、全ノードから送信されたデータのうち、対応する受信が先行していたために MBCF の遠隔メモリ書き込みにより転送されたもののデータ量 (バイト数) の割合である。

表 4 に EP の実行結果を示す。 2^{26} 個の乱数を求めている。 EP では通信は実行結果の最終的な収集においてのみ発生し、実行時間のほとんどは浮動小数点数の演算である。このため、この結果は通信性能ではなくワークステーションの浮動小数点数演算性能を表していると言える。

表 4: NPB EP の実行時間 (単位 : sec)

ノード数	1	2	4	8
MPI/MBCF	121.04	60.53	30.26	15.14
(speed-up)	(1.00)	(2.00)	(4.00)	(7.99)
MPICH	125.56	60.61	32.13	16.25
(speed-up)	(1.00)	(2.07)	(3.91)	(7.73)

表 5 に MG の実行結果を示す。問題サイズは $64 \times 64 \times 64$、イテレーション数は 40 である。分割境界でのデータのやりとりのため、数百 byte から数 Kbyte 程度のメッセージの 1 対 1 通信が非常に頻繁に実行される。このため、メッセージサイズが小さい場合にも効率良く通信を行うことのできる MPI/MBCF が MPICH に大きく優っている。ほぼ全ての受信関数において送信元としてワイルドカード `MPI_ANY_SOURCE` が指定されているため、 MPI/MBCF において送信要求を発行することができず、遠隔メモリ書き込みの利用が妨げられて

表 3: NPB ベンチマークプログラムの特性

プログラム	EP	MG	CG	IS	LU	SP	BT
通信データレート (MByte/s)	0.00	9.68	12.69	13.58	1.89	7.83	5.32
通信メッセージレート (個 /s)	4	4670	2138	466	1199	421	488
遠隔メモリ書き込み利用率 (%)	51.10	0.01	53.33	99.22	13.37	49.01	47.24

いる。送信元を静的に指定するようにプログラムを書き換えることにより MPI/MBCF ではさらなる性能向上が見込まれる。

表 5: NPB MG の実行時間 (単位 : sec)

ノード数	1	2	4	8
MPI/MBCF	39.16	22.61	14.36	7.48
(speed-up)	(1.00)	(1.73)	(2.73)	(5.24)
MPICH	38.81	31.30	21.01	13.72
(speed-up)	(1.00)	(1.24)	(1.85)	(2.83)

表 6 に CG の実行結果を示す。問題サイズは 7000、イテレーション数は 15 である。リダクション操作のための集団通信を 1 対 1 通信関数で記述したメッセージサイズ数十 Kbyte 程度の通信、および同じくメッセージサイズ数十 Kbyte 程度の通常の 1 対 1 通信が実行される。通信の頻度は MG ほど高くはなく、また、メッセージのサイズが MG より大きいため、 MPICH の性能は MG でのものより向上している。さらに MPI/MBCF では、遠隔メモリ書き込みを全データの半分以上に対して適用することで通信の効率を上げていることが分かる。

表 6: NPB CG の実行時間 (単位 : sec)

ノード数	1	2	4	8
MPI/MBCF	69.05	35.99	20.40	11.02
(speed-up)	(1.00)	(1.92)	(3.38)	(6.27)
MPICH	68.75	40.01	27.79	14.59
(speed-up)	(1.00)	(1.72)	(2.47)	(4.71)

表 7 に IS の実行結果を示す。 2^{20} 個の整数のソートを 10 イテレーション行っている。各イテレーションで 1 Mbyte 程度のメッセージを全対全の集団通信関数により交換している。各ノードでの計算時間が短いため、台数が増えるにつれこの集団通信の時間が支配的となるが、 MPI/MBCF では集団通信関数を受信起動が先行するように実装することにより効率的に通信を行っていることが示される。

表 7: NPB IS の実行時間 (単位 : sec)

ノード数	1	2	4	8
MPI/MBCF	10.05	6.33	4.39	3.02
(speed-up)	(1.00)	(1.59)	(2.29)	(3.33)
MPICH	10.25	7.09	5.61	4.81
(speed-up)	(1.00)	(1.45)	(1.83)	(2.13)

表 8 に LU の実行結果を示す。問題サイズは $33 \times 33 \times 33$、イテレーション数は 300 である。メッセージサイズ数百 byte 程度の 1 対 1 通信でデータをやりとりしている。メッセージサイズは小さいものの通信頻度が高くないため、 MPI/MBCF, MPICH ともに台数効果が出ている。 LU においても受信関数の送信元指定に `MPI_ANY_SOURCE` が用いられており、 MPI/MBCF での遠隔メモリ書き込みの利用の機会を減らしている。

表 8: NPB LU の実行時間 (単位 : sec)

ノード数	1	2	4	8
MPI/MBCF	1004.24	527.69	286.45	160.36
(speed-up)	(1.00)	(1.90)	(3.51)	(6.26)
MPICH	1081.51	611.92	320.70	185.04
(speed-up)	(1.00)	(1.77)	(3.37)	(5.84)

表 9 に SP の実行結果を示す。問題サイズは $36 \times 36 \times 36$、イテレーション数は 400 である。メッセージサイズ数十 Kbyte 程度の 1 対 1 通信が呼ばれる。通信の頻度は低く、通信パターンが受信先行となっているため、特に MPI/MBCF で大きな台数効果を得ている。

表 10 に BT の実行結果を示す。問題サイズは $24 \times 24 \times 24$、イテレーション数は 200 である。数 Kbyte から数十 Kbyte 程度のメッセージを 1 対 1 通信でやりとりしている。 SP 同様通信の頻度が低く、通信パターンが受信先行となっているため、 MPI/MBCF で大きな台数効果を得ている。

全プログラムを通じて、 MPI/MBCF による実行

表 9: NPB SP の実行時間 (単位 : sec)

ノード数	1	4	9
MPI/MBCF	1256.45	342.84	154.91
(speed-up)	(1.00)	(3.66)	(8.11)
MPICH	1391.16	475.27	231.66
(speed-up)	(1.00)	(2.93)	(6.01)

表 10: NPB BT の実行時間 (単位 : sec)

ノード数	1	4	9
MPI/MBCF	616.73	155.17	67.30
(speed-up)	(1.00)	(3.97)	(9.16)
MPICH	627.29	214.14	96.02
(speed-up)	(1.00)	(2.93)	(6.53)

効率が MPICH の実行効率を上回っている。特に、小さなメッセージを頻繁に通信し合う場合 (MG) には MPICH がオーバヘッドの大きさから性能を大きく落とすため、通信パターンが受信先行となる場合 (CG, IS, SP, BT) には MPI/MBCF が遠隔メモリ書き込みを用いて効率良く通信を行うため、MPI/MBCF と MPICH との差が大きくなっている。

4 まとめ

低コスト・高機能な共有メモリ通信機能であるメモリベース通信を用いて実装された MPI ライブラリ MPI/MBCF の性能を、100Base-TX で接続されたワークステーションクラスタ上で並列アプリケーション NAS Parallel Benchmarks の実行により評価した。メッセージパッシング通信機能を用いた実装との比較から MPI/MBCF の効率の良さが示された。特に、通信メッセージサイズが小さなアプリケーションや、受信の発行が対応する送信の発行に先行する通信パターンのアプリケーションにおいて MPI/MBCF が効率良く通信を行えることが実証された。以上の結果から、共有メモリ通信機能であるメモリベース通信をメッセージパッシング通信ライブラリのベースとして用いることの有効性が示された。

今回の実験での比較は、汎用ハードウェアを用いた全く同一の機器の上で行われたものであり、オペレーティングシステムおよび通信ライブラリを効率の良いものとすることで実アプリケーションに変更を加えることなくその実行効率を改善することが可能であることを示している。このことはさらに、専用の通信ハードウェアを導入することなくワークステーションクラスタを構築することの有効性を示すものである。

謝辞

本研究の一部は情報処理振興事業会 (IPA) が実施している独創的情報技術育成事業、および新情報処理開発機構 (RWCP) の支援を受けた。

参考文献

[1] D. Bailey and et al. The NAS parallel benchmarks. Technical Report RNR-94-007, NASA Ames Research Center, March 1994. `http://www.nas.nasa.gov/NAS/NPB/`.

[2] D. Bailey and et al. The NAS parallel benchmarks 2.0. Technical Report NAS-95-020, NASA Ames Research Center, December 1995. `http://www.nas.nasa.gov/NAS/NPB/`.

[3] Message Passing Interface Forum. MPI: A Message-Passing Interface Standard. `http://www.mcs.anl.gov/mpi/`, May 1995.

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

[5] W. Gropp, E. Lusk, N. Doss, and A. Skjellum. A High-Performance, Portable Implementation of the MPI Message-Passing Interface Standard. *Parallel Computing*, Vol. 22, No. 6, pp. 789–828, September 1996.

[6] 松本尚, 古荘進一, 平木敬. 汎用並列オペレーティングシステム SSS–CORE の資源管理方式. 日本ソフトウェア科学会第 11 回大会論文集, pp. 13–16, October 1994.

[7] 松本尚, 駒嵐丈人, 渦原茂, 竹岡尚三, 平木敬. 汎用超並列オペレーティングシステム : SSS–CORE — ワークステーションクラスタにおける実現 —. 情報処理学会研究報告 96-OS-73, Vol. 96, No. 79, pp. 115–120, August 1996.

[8] 松本尚, 平木敬. 100BASE-TX によるメモリベース通信の性能評価. コンピュータシステムシンポジウム論文集, pp. 101–108, November 1997.

[9] 松本尚, 平木敬. 共有メモリ vs. メッシージパッシング. 情報処理学会研究報告 97-ARC-126, Vol. 97, No. 102, pp. 85–90, October 1997.

[10] 森本健司, 松本尚, 平木敬. メモリベース通信を用いた高速 MPI の実装方式. 並列処理シンポジウム JSPP'98 論文集, pp. 191–198, June 1998.