SR8000向き高速メッセージパッシングライブラリFMPLの開発

建部修見[†] 長嶋雲兵[†] 関口智嗣[†]
北林久義[††] 林田義幸[†††]

日立SR8000の通信ハードウェア機構を効率的に利用するための高速メッセージパッシングライブラリFMPLを開発している．FMPLはリモートDMA転送を用いた低コストの汎用1対1通信を実現し，その1対1通信および通信ハードウェア機構を用いブロードキャスト，リダクションおよびバリア同期通信を提供する．1対1通信において日立提供のMPIでは8バイト転送遅延が20μsecであるが，FMPLでは12.8μsecを達成し，低遅延の1対1通信を実現している．本ライブラリはBLACSなどの通信ライブラリを高速実装するためだけではなく，SR8000の通信ハードウェア性能を最大限に引き出す必要のあるアプリケーションなどに利用される．

Design and implementation of FMPL, a fast message-passing library for SR8000

Osamu Tatebe,[†] Umpei Nagashima,[†] Satoshi Sekiguchi,[†]
Hisayoshi Kitabayashi[††] and Yoshiyuki Hayashida[†††]

We are developing a fast message-passing library (FMPL) to maximize capability supported by Hitachi SR8000 communication hardware. FMPL provides low-cost point-to-point communication using remote DMA transfer mechanism and efficient collective communication, such as broadcast, barrier synchronization and reduction, using the low-cost point-to-point communication or the hardware communication support. FMPL achieves a 8-byte latency of 12.8μsec., while MPI achieves 20μsec. FMPL is designed for a lower-layered library of BLACS as well as applications that need maximum performance.

1. はじめに

計算機クラスタを含む分散メモリ型の並列計算機では，通信性能を高めるため，リモートノードのメモリを直接更新，参照する (ハードウェア) 機構を備えているものが増えてきた．このリモートメモリ操作を用いることにより，送信/受信のメッセージパッシングはユーザ空間からユーザ空間へゼロコピーでメッセージを転送することができ，メモリ間コピーのオーバヘッドを削減し，ハードウェアのピーク性能に迫る転送バンド幅を達成することができる [5)~7)]．

リモートメモリ操作を用い，ゼロコピーの送信/受信のメッセージパッシングライブラリを構築する場合，送信/受信のマッチングおよび同期を取ることと，書き込みアドレス (あるいは参照アドレス) をリモートメモリ操作に先立って知る必要がある．送信/受信のマッチングは，受信側が送信側にメッセージヘッダを送ることにより，送信側で処理することができ，その場合，リモートメモリ書き込みにより送信バッファから受信バッファへ直接書き込めるため，ネットワーク的にはもっとも性能がよい [6),7)]．1対1通信をリモートメモリ操作を用いゼロコピーで実装する多くの研究は，メッセージパッシングライブラリの標準であるMPIを実装している．分散メモリ型並列計算機では，多くのアプリケーションはMPIを用い記述されるため，MPIを高速に実装することは非常に重要なことである．

しかしながら，MPIは機能が豊富で，例えば1対1通信でも，標準モード，バッファモード，同期モード，レディーモードの4つのモードがあり，更にそれぞれのモードにブロッキング通信とノンブロッキング通信があり，合わせて8種類存在する．データ型もC言語の構造体より汎用的なユーザ定義派生データ型を作成することができ，また通信ドメインやプロセストポロジを表すコミュニケータがある．MPI(とMPI-2) は従

† 産業技術総合研究所
National Institute of Advanced Industrial Science and Technology
E-mail: {o.tatebe,u.nagashima,s.sekiguchi}@aist.go.jp
†† (株) 日立ビジネスソリューション 基本ソフト事業部
Hitachi Business Solution, Software Development Department
††† (株) 日立製作所 ソフトウェア事業部
Hitachi, Ltd., Software Division

来のメッセージパッシングライブラリのほとんどすべての機能を取り込んだ仕様となっているため，すべての仕様を満たすのは実装が大変である．そこで，多くの実装は ANL の MPICH[3] を用いている．MPICH は層を明確に分け，ADI(仮想デバイスインターフェース) を定義するだけで MPI の全仕様を満たすライブラリを構築することができる．ADI の中でも定義が必須な関数は 13 で，MPI のポーティングが比較的簡単であるため多くの処理系で利用されている [4],[5]．

一方，連立一次方程式解法，最小二乗問題，固有値問題などの行列演算パッケージの ScaLAPACK[1] では，通信層として BLACS(Basic Linear Algebra Communication Subprograms)[2] を用いている．BLACS は行列演算パッケージのための簡単で使いやすいインターフェースを目指して設計された通信インターフェースである．BLACS は主に，さらに汎用的で多くのプラットフォームで利用可能な MPI を用い実装されるが，この場合，BLACS の配列のパック，アンパックなどのコピーのオーバヘッドと，MPI のオーバヘッドと，オーバヘッドが必要以上に大きくなってしまう．そのため，高性能を実現するためには MPI のような機能豊富な通信ライブラリではなく，低コストで汎用的な高速メッセージパッシングライブラリが望ましい．

本研究では，BLACS などのメッセージパッシングライブラリを高効率で実現するための高速メッセージパッシングライブラリ FMPL のデザインおよび実装を行う．FMPL では SR8000 の持つリモート DMA 転送に同期処理を追加するだけの低コストの 1 対 1 転送ライブラリを提供し，その軽い 1 対 1 通信および通信ハードウェア機構を用いた集合通信を提供する．FMPL は高効率の BLACS の実装のためのだけではなく，低オーバヘッドが必要なアプリケーションに対しても利用される．

2. BLACS

BLACS は行列演算パッケージ ScaLAPACK および PBLAS の通信層である．BLAS で行列演算の一般的な基本演算を定義したのと同様に，BLACS では一般的な基本通信が定義されており，行列演算のための簡単で使いやすいインターフェースを目指し設計されている．実装は MPI，PVM の他に IBM MPL，Intel NX ライブラリなどを用いたものがある．

BLACS は以下の特徴を持っている．

- プロセスグリッドとスコープ
- コンテキスト
- 二次元配列ベースの通信
- メッセージタグなしの通信

行列を分散させる場合には，二次元のプロセスグリッドにブロックサイクリックにマッピングする方法が一般的である．この方法は，行列の列ブロック分割，行サイクリック分割などの分割法を含む二次元配列の汎用的な分割法である．BLACS では二次元のプロセスグリッドを仮定している．また，ブロードキャスト，リダクションなどの通信ではプロセスグリッドの同じ行に含まれるプロセス，同じ列に含まれるプロセス，およびプロセス全体といったスコープを指定して通信処理が行われる．

BLACS には MPI のコミュニケータと同様のコンテキストがあり，通信ドメインを分けることができる．BLACS におけるコンテキストは，プロセスグリッドを指定して作成される．

行列演算では二次元 (部分) 配列を利用することが多い．またさらに，主に対称行列などの場合には配列の下三角部分，あるいは上三角部分を利用することが多い．以下は BLACS の送信関数のインターフェースである．

```
vGESD2D( ICONTXT,
         M, N, A, LDA, RDEST, CDEST )
vTRSD2D( ICONTXT, UPLO, DIAG,
         M, N, A, LDA, RDEST, CDEST )
```

ICONTEXT はコンテキストを指定する．M, N, A, LDA で二次元配列を指定する．一次元配列であっても，この指定となる．RDEST，CDEST はコンテキストのプロセスグリッドにおけるプロセス番号である．UPLO は上三角 (台形) あるいは下三角を指定し，DIAG は対角部分は 1 として処理されないか，処理するかを指定する．関数名のはじめのv はI，S，D，C，Z の何れかであり，それぞれデータ型が整数，単精度実数，倍精度実数，単精度複素数，倍精度複素数であることを表している．BLACS の送信/受信関数は，MPI でいうブロッキング通信である．BLACS では 1 対 1 転送の送信関数はこの関数しか用意されず，ノンブロッキング通信のためのインターフェースはない．

BLACS では，メッセージタグにおけるプログラマのミスを防ぐために，メッセージタグはライブラリで付加される．ライブラリが適切なメッセージタグを作成するために，ネットワークの FIFO 性を仮定しており，自プロセスの処理だけでメッセージタグは作成される．

3. SR8000 における通信ハードウェア

SR8000 のネットワークは多次元クロスバネットワークである．8 ノード構成では一次元クロスバ，64 ノード構成では二次元クロスバと，ノード数に応じてクロスバの次元が上がっていく．ノード間の転送速度は最大 1000MB/s である．それら多次元クロスバネットワークを用い，SR8000 ではリモート DMA 転送，ハードウェアバリア機構，ハードウェアブロードキャスト機構を備えている．

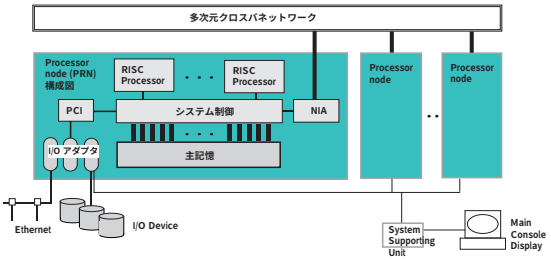

図 1　SR8000 のプロセッサノードの構成

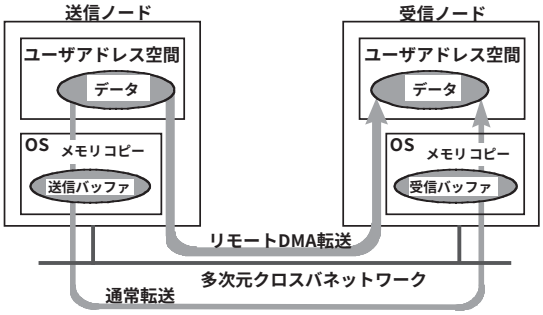

図 2　リモート DMA 転送

図 1 に SR8000 のプロセッサノードの構成を示す．SR8000 のプロセッサノードは大きくプロセッサ，メモリ，NIA(Network Interface Adapter) で構成される．ノード間通信において NIA は送信と受信の二つの処理を実行する．送信処理では，NIA はプロセッサからの送信指示に従ってローカルメモリからデータを取り出して，ノード間通信の単位であるパケットを構成し，ノード間ネットワークへ送り出す．パケットは，ヘッダ部とデータ部から成る．ヘッダ部には，受信ノード番号，送信データサイズ，リモート DMA 識別子などの情報が格納される．受信処理では，NIA はノード間ネットワークからパケットを受信してデータを取り出し，ローカルメモリに書き込む．

3.1　リモート DMA 転送

通常のデータ転送方式では，ユーザプログラムから別のノードのユーザプログラムへ送信するデータを，一度ユーザプログラム空間からカーネル内の送信バッファへコピーした後，パケット単位に編集して相手のノードへ送信する．受信ノードでは，カーネル内の受信バッファに受信したパケット単位のデータから受信データを生成し，ユーザプログラム空間内へコピーすることによって受信データを渡す．このためユーザプログラム空間とカーネル空間の間で少なくとも二回のデータコピーが発生し，データ転送の通信遅延が大きくなる．

これに対しリモート DMA 転送方式では，ユーザプログラム空間内の送信データ領域 (仮想メモリ) とカーネル内のリモート DMA 転送用の送信領域 (物理メモリ) をあらかじめ一対一にマッピングしておく．同様に，受信ノード側でもカーネル内のリモート DMA 転送用の受信領域 (物理メモリ) とユーザプログラム空間内の受信データ領域 (仮想メモリ) をあらかじめ一対一にマッピングしておく．その上で多次元クロスバネットワークを経由した DMA 転送によって，ノード間にまたがったユーザプログラム空間内の送受信領域間でデータを直接転送することができる．

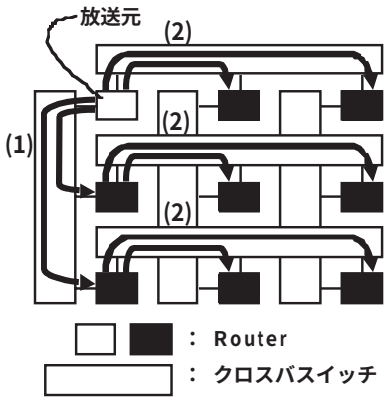

図 3　ハードウェアブロードキャスト機構

メッセージの送信は NIA の制御レジスタに転送制御テーブル (TCW) の先頭アドレスを書き込むことで起動される．TCW には，送信データのアドレス，サイズ，相手先ノード，リモート DMA 識別子，オフセットなど，データ転送に必要な情報が格納される．

このように，カーネルを介さずデータを送受信することができるため，ノード間データ転送の通信遅延を小さくすることができ，高速なノード間通信を実現することができる．

3.2　ハードウェアブロードキャスト機構

図 3 にハードウェアブロードキャスト機構を示す．同時に複数のハードウェアブロードキャストが実行されないように，システム中に一意に「シリアライズクロスバ」が決められる．二次元クロスバの場合，シリアライズクロスバとして決められている y-クロスバから，全 x-クロスバを介してシステム中の全てのノードにデータを転送する．ノードからシリアライズクロスバへのデータ転送は x-クロスバを経由して 1 対 1 転送で行う．二次元クロスバの各クロスバは完全クロスバになっている為，シリアライズクロスバはその完全クロスバ性を利用し，各 x-クロスバへの展開を同時並行動作的に行うことができる．

ブロードキャストの起動は，通常のリモート DMA 転送と同じであるが，リモート DMA 領域の受信フィールドに対する送信権獲得時にブロードキャスト属性を指定する．

ハードウェアブロードキャストを利用のためには，以下のような条件が必要となる．

(1)　プロセスは別々のノードで実行される．

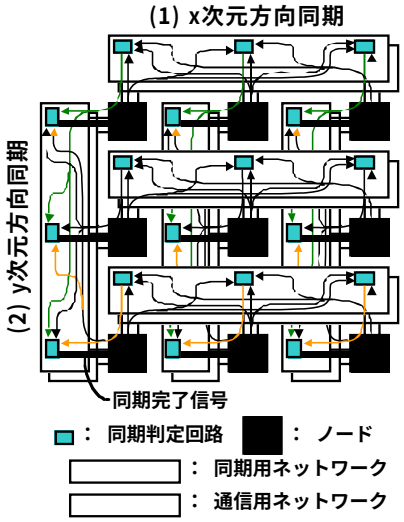

図 4 ハードウェアバリア機構

(2) プロセスを実行するパーティションは，排他属性及びグローバル属性が設定されている．

(3) 実行するパーティションは，共用属性のパーティションと重なりあっていない．

(4) プロセスは，直方体形状に配置されている．直方体形状には，直線及び長方形の形状も含まれる．

さらに，ブロードキャストするアドレスに関しても，送信領域と受信領域のオフセットが各ノード間で同一であるという制限がつく．

3.3 ハードウェアバリア機構

図 4 に二次元クロスバの場合のハードウェアバリア機構を示す．ハードウェアバリア機構は，二次元クロスバの各クロスバが完全クロスバであることを利用し，各 x-クロスバ上で並行して接続されているノードからの同期点到達判定を行い，それを y-クロスバに伝える．このとき特定の y-クロスバに集めるのではなく，全ての y-クロスバに x-クロスバでの同期を伝える．このことにより，各 y-クロスバにおいて，並行動作的に xy 平面の同期を確認し，各 y-クロスバから x-クロスバを介さずに並行して各ノードに同期成立を伝えることができる．

ハードウェアバリア機構は SR8000 のリモート DMA 転送ライブラリの`hmpp_barrier` を用いる．`hmpp_barrier` では同期要因を表すカラーとして 0 から 7 の 8 種類が用意されている．

ハードウェアバリア機構はハードウェアブロードキャストの適用条件のときのみ利用可能となる．`hmpp_barrier` はそれ以外の場合でも使用可能であり，その場合はハードウェアブロードキャスト機構とリモート DMA 機構をソフトウェア的に組み合わせてバリア同期がとられる．

4. FMPL: 高速メッセージパッシングライブラリ

SR8000 では，リモート DMA 転送により高速なメッセージ転送を可能としており，そのための combuf 通信ライブラリが提供されている．しかしながら，リモート DMA 転送では，メッセージ転送側と受信側の同期が取られないため，受信データ領域の上書きが起きる可能性があり，ダブルバッファリングなどのアルゴリズム的な工夫を行うか，メッセージ転送時に同期処理が別途必要となる．リモート DMA 転送は，通信処理と同期処理を分けた通信であるため，1 対 1 通信に比べより柔軟なプログラミングが可能となり，通信オーバヘッドを削減することが可能となるが，その一方で同期操作を別途必要とするためプログラミングおよび正しさの検証が難しくなってしまう．

1 対 1 通信は，通信処理と同期処理を合わせたものであり，送信側は送信命令により送信領域を指定し，受信側は受信命令により受信領域を指定する．送信命令と受信命令は宛先，メッセージタグなどでマッチングが取られ，メッセージの転送が行われる．ここでの同期処理は，送信側が送信を発行するまでは送信領域が転送されることはなく，送信が終了したら送信領域を書き換えることができ，また受信側が受信を発行するまでは，受信バッファが上書きされてしまうことはなく，受信が終了したら受信領域にはデータがそろっているという処理である．

1 対 1 通信は，分散メモリ型並列計算機のプログラミングの基本操作であり，木構造によるブロードキャスト，リダクションなど多くの集合通信も 1 対 1 通信を用いて効率的に実装できる．この基本通信を低コストで実現する高速メッセージパッシングライブラリ FMPL を作成することにより，それらを用い MPI や BLACS などのメッセージパッシングライブラリを効率的に実装することが可能となるだけではなく，ベンチマークプログラムや通信性能をさらに高めたいアプリケーションなどでは，リモート DMA 転送に加え，FMPL を直接利用することにより，より高い性能を得ることが可能となる．

4.1 1 対 1 通信のデザイン

FMPL では 1 対 1 通信にリモートメモリ書き込みを利用し，通信のための一時バッファを介することなく，直接ユーザ空間上のデータ領域にデータ送信可能なゼロコピー通信で実現する．

送信/受信関数は通信バッファの指定以外に，MPI ではタグとコミュニケータ，BLACS ではコンテキストなどといった，送信/受信関数のマッチングのための情報を伴っている．それら送信/受信のヘッダ情報は通常キューで管理され，エンキュー，デキューなどに伴うメモリ管理，キューの中のエントリの検索など

の処理が伴う．これらの処理は基本的にダイナミックな処理であり，重い処理となりうる．

FMPL ではそれらマッチングに伴う検索処理をなくするために，タグなどではなく送信/受信関数でマッチング領域を指定する．送信/受信関数のペアごとにそれぞれ別々にマッチング領域を与えることにより，キュー管理，キュー検索の処理をなくすことができる．

FMPL における 1 対 1 通信の処理の流れは，遅延が短くなるように，受信側が送信側にヘッダ情報を送り，送信側でマッチングを行い，リモートメモリ書き込みによりメッセージ転送を行う．

送信受信関数は以下のインターフェースとなる．

```
fmpl_send(buf, size, dst, mai, ier)
fmpl_recv(buf, size, src, mai, ier)
```

送信/受信関数は通信バッファbuf，バッファサイズsize，送信先プロセスランクdst，受信元プロセスランクsrc の他に，マッチング領域のインデックスmai を指定する．このマッチング領域のインデックスはメッセージタグに相当するものであるが，この領域ではキュー管理は行わないため，送信/受信のペアにおいて生存期間が重なる場合は同じインデックスを用いることはできない．ier はリターンコードであり，正常終了時はFMPL_SUCCESS が返り，エラーが発生した場合はエラーコードがはいる．

受信処理において，送信側のマッチング領域の指定されたインデックスにメッセージのヘッダ情報をリモートメモリ書き込みにより書き込む．送信側は書き込まれたヘッダ情報を確認することにより同期がとられ，リモートメモリ書き込みによりメッセージ転送を行う．

マッチング領域および受信完了を知らせるための受信完了フラグは，FMPL の初期化関数fmpl_init であらかじめ指定する．

```
fmpl_init(ma, fa, count, ier)
```

ma はマッチング領域の先頭アドレス，fa は受信完了フラグの先頭アドレス，count はその要素数である．

fmpl_send，fmpl_recv はブロッキング通信のためのインターフェースであり，これらの関数はそれぞれ送信領域に安全に書き込み可能となったとき，受信領域に全データを受信したときに完了する．ゼロコピーでブロッキング通信を実装する場合，受信関数が発行されない限り送信関数は終了せず MPI でいう同期モードとなってしまう．FMPL では，この同期モードの通信を回避するために，送信側に一時的に送信データを待避させる送信バッファ領域を指定することができる．

```
fmpl_sendbuff_set(buf, size, ipara, ier)
fmpl_sendbuff_check(nsend, nspool, ier)
```

fmpl_sendbuf_set は，送信データ待避領域buf とサイズsize およびタイムアウト時間ipara を指定する．fmpl_send でタイムアウトした場合，送信データは送信バッファにコピーされfmpl_send は終了する．このとき，ユーザは定期的にfmpl_sendbuf_check を発行して，送信バッファにたまっているデータに対し，受信側からヘッダ情報の書き込みがあり送信可能となったデータを送信する必要がある．fmpl_sendbuff_check の発効後，nsend には送信したデータの数，nspool にはまだ送信バッファに残っている送信データの数が返される．

ノンブロッキング通信のためのインターフェースとしてはfmpl_isend，fmpl_irecv およびそれらの完了を待つためのfmpl_wait_isend，fmpl_wait_irecv が提供される．ノンブロッキング通信ではfmpl_isend が完了してもまだ送信バッファを変更することはできず，変更してしまうと送信データが保証されなくなってしまう．同様に，fmpl_irecv が完了してもfmpl_wait_irecv を発行し完了するまでは，受信データは保証されない．FMPL におけるノンブロッキング通信は，特にfmpl_irecv を前もって発行することにより効果的となり，このとき対応するfmpl_isend あるいはfmpl_send は直ちに送信することができるため，送信待ちのオーバヘッドを減らすことができる．

FMPL における 1 対 1 転送の実装デザインでは，受信関数は送信側のマッチング領域にメッセージのヘッダ情報を直接書き込むため，送信プロセスのランクを必ず指定する必要がある．MPI などの受信関数では，送信プロセスのランクに ANY を指定することにより，明示的に送信プロセスのランクを指定しないこともできるが，FMPL ではこの ANY の指定をライブラリレベルではサポートしない．この指定が必須の場合は，ノンブロッキングの受信関数を受信する可能性のあるプロセス分発行し，必要がなくなれば，必要のない受信関数をキャンセルするなどというように，プログラムの書き換えを行うことにより対応することとなる．

4.2 1 対 1 通信の SR8000 における実装

SR8000 のリモート DMA 転送は，TCW の作成と，そのアドレスを NIA の制御レジスタに書き込むことにより起動される．TCW は一度作成すれば，内容の変更がない限り再利用することができ，その場合は，combuf_kick_tcw あるいはcombuf_kick_tcw_fast を用い NIA の起動のみで高速に送信を行うことができる．この送信は TCW 再利用型送信と呼ばれている．送信/受信のマッチングに利用されるマッチング領域への書き込みは，マッチング領域作成時に TCW を作成することにより，この TCW 再利用型送信により送信することができる．

図 5 にゼロコピー通信の処理の流れを示す．ゼロコピー通信ではユーザ空間およびマッチング領域はリモート DMA 転送可能なようにリモート DMA 領域となっている．受信が発行されると，受信領域のヘッダ情報 (オフセット) をマッチング領域に設定し，そのヘッダ情報を送信ノードにcombuf_kick_tcw_fast を用い TCW 再利用型送信により送信する．一方，送信側は，そのオフセットを送信情報にセットし，リモート

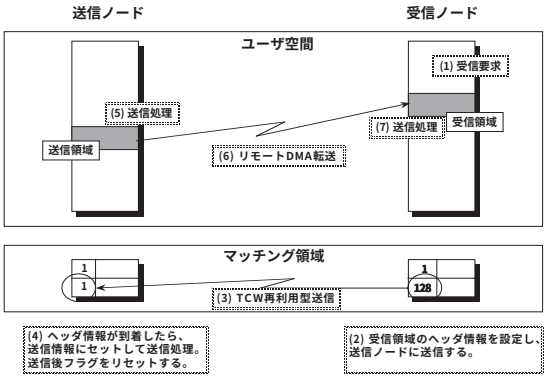

図 5 ゼロコピー通信の処理の流れ

DMA 転送により送信領域を受信領域に転送し，マッチング領域をリセットする．

4.3 集合通信：バリア同期，ブロードキャスト，リダクション

FMPL における集合通信は，プロセスグループを表すコミュニケータを作成し，そのコミュニケータを用い通信する．コミュニケータ作成は以下のインターフェースを用いる．

```
fmpl_comm_create(comm, key, ier)
```

fmpl_comm_create は全プロセスで呼び出し，key によりそのプロセスグループに参加するかどうかを決定する．プロセスグループ番号はcomm に返される．全プロセスが参加するFMPL_COMM_WORLD はあらかじめ定義されている．FMPL で提供される集合通信は，基本的に可能であればハードウェアの機能を利用したものと FMPL の低コストの 1 対 1 通信を利用したもので構成される．

ブロードキャストは，FMPL の低コスト 1 対 1 通信を二分木状に利用したソフトウェアブロードキャストと，SR8000 のハードウェアブロードキャストと二種類用意している．

```
fmpl_bcast(buf, size, root, comm, ier)
fmpl_hbcast(buf, size, root, ier)
```

ハードウェアブロードキャストは 3.2 節の制限がつくが，ソフトウェアブロードキャストは任意のコミュニケータcomm においてブロードキャストすることができる．二種類用意したのは，3.2 節の制限に対し，送信領域，受信領域のオフセットが同一という条件を実行時に判定するコストが大きいためである．

リダクションについては，以下のインターフェースとなっている．

```
fmpl_allreduce(buf, count, func, comm,
                work, ier)
fmpl_reduce(buf, count, func, root, comm,
             work, ier)
```

buf のデータはfunc で指定されるリダクション処理をした結果に更新される．func では，fmpl_vsum，fmpl_vmax，fmpl_vmin があらかじめ提供され，それぞれ総和，絶対値最大，絶対値最小を計算することができる．それぞれの関数のv には BLACS 同様にデータ型を表す文字がはいる．count はbuf の要素数がはいる．work はbuf と同じ大きさの作業領域である．

バリアは以下のインターフェースで与えられる．

```
fmpl_barrier(comm, ierr)
```

comm はバリアをとるコミュニケータである．comm がFMPL_COMM_WORLD の場合 3.3 節の制限が満たされればハードウェアバリア機構のためのライブラリ hmpp_barrier が用いられる．ハードウェア機構が使えない場合，あるはそれ以外のコミュニケータが指定された場合は，FMPL の 1 対 1 通信の同期操作と同様の TCW 再利用型送信を用いた軽い同期操作を用い，二分木状に同期をとることによるバリア同期がとられる．

5. 性 能 評 価

この節では，本研究により作成した FMPL の基本通信の性能評価を行う．評価にあたり，産業技術総合研究所先端情報計算センタ (TACC) の日立 SR8000 64 ノードのうち 16 ノードを使用した．SR8000 のノードは基本的には 8-way SMP であり，プロセッサは 250MHz の PowerPC アーキテクチャを元に疑似ベクトル機構，拡張レジスタ，拡張命令，LTLB 等を追加したプロセッサである．それぞれのプロセッサは協調型マイクロプロセッサ機構と呼ばれる複数のプロセッサを一斉に高速に起動するハードウェア機構がついており，並列ループのループ分割による高速並列実行の支援をしている．ノードあたりのピーク性能は 8000MFLOPS となる．ネットワークは 64 ノードまでは二次元クロスバであり，それぞれのリンクのバンド幅は単方向 1000MB/s である．

5.1 1 対 1 通信

図 6 に 1 対 1 通信のスループットを示す．このスループットは ping-pong 通信の通信遅延を元に求めている．MPI は日立提供のものであり，BLACS は netlib で提供されている MPI ベースのものを日立提供の MPI を用いて実行している．括弧付きの rdma はコンパイル時に-rdma オプションをつけて，ユーザプログラム中の静的領域を静的リモート DMA 領域とし MPI の送受信をゼロコピー通信としたものであり，nordma は送信側，受信側でそれぞれユーザ領域とリモート DMA 領域の間のコピーが発生するバッファ通信としたものである．

FMPL および MPI でゼロコピー通信をしたものはいずれもネットワークの最高バンド幅の 1000MB/s にほぼ近い性能を達成しており，16MB のメッセージサイズでは，FMPL は 999MB/s，MPI(rdma) は 997MB/s を達成した．MPI (nordma) は送信時にユー

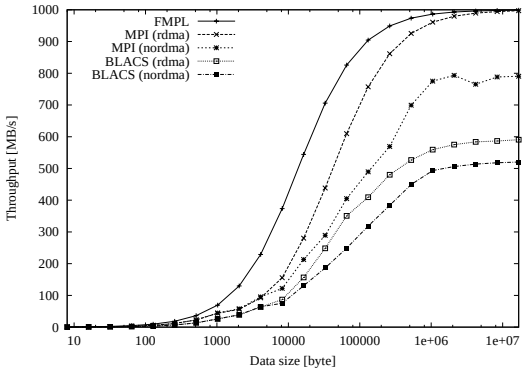

図 6 1 対 1 通信のスループット

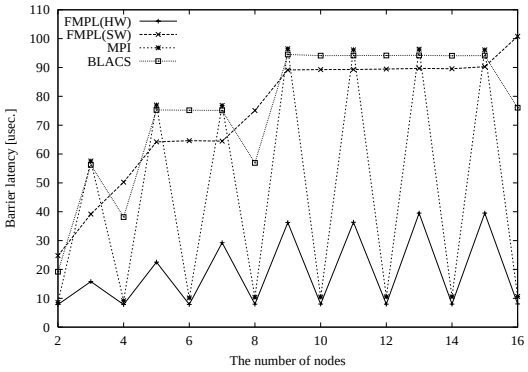

図 7 バリア同期時間

表 1 8 バイト転送時の送信/受信処理の内訳

送信	初期処理	ヘッダ受信	データ送信
	0.5	5.0	8.1
受信	初期処理	ヘッダ送信	データ受信
	0.5	6.1	5.6

[μsec]

ザ空間からリモート DMA 領域へ，受信時にリモート DMA 領域からユーザ領域にコピーが必要であり，16MB で 791MB/s とピークバンド幅の 80%程度しか達成していない．BLACS では送信，受信にデータをパックしてから MPI で転送しているためさらにコピーの回数が増えている．そのため，16MB のときのバンド幅は rdma と nordma でそれぞれ 591MB/s, 522MB/s となっている．

8 バイト転送時の片道の通信遅延は FMPL で 12.8μsec，MPI ではいずれの場合も 20μsec，BLACS ではいずれの場合も 34μsec であった．MPI では，一度目の送受信で受信フィールドの送信権を獲得するための`combuf_get_sendright` を発行し，一度目の送受信が遅くなるが，この通信遅延は二度目以降の遅延である．FMPL では受信から送信のマッチング領域へヘッダ情報の書き込み，送信から受信へのメッセージ転送とメッセージは 1 往復となるが，MPI の場合は，送信から受信へヘッダ情報の送信，受信側でマッチング処理後，送信側へ書き込み先の転送，送信側がメッセージ転送と，メッセージが 1 往復半することとなり，この差がそのまま通信遅延の差となっている．

FMPL における，8 バイト転送時の通信遅延の内訳を**表 1** に示す．この内訳はプロセッサのマシンサイクルカウンタを用い測定した．ハードウェアのネットワーク遅延が 5μsec 程度であるため，低コストで 1 対 1 通信が実現できていることが分かる．ヘッダ送信時間に比べ，データ送信時間が大きいのは，ヘッダ送信は TCW 再利用型送信を行っているのに対し，データ送信は TCW を修正してからリモート DMA 転送を行っているためである．

5.2 バリア同期

図 7 にバリア同期時間を示す．FMPL(HW) はハードウェアバリア機構を利用するための`hmpp_barrier` を用いたものである．TACC の SR8000 は 64 ノード構成であるが，16 ノードのパーティション構成が 2×8 と x-クロスバ方向で分割されているため，ノード数が偶数のときに物理ノード構成が矩形となり，ハードウェアバリア機構が利用できる．FMPL(SW) では二分木で通信しバリア同期をとっているため，バリア同期時間はプロセス数 p に対し $O(\log p)$ となっている．本ソフトウェアバリアの実装では，プロセス数分のバッファ領域を利用しているため，2 の巾乗のプロセス数以外のときは通信段数を一段少なくなっている．

MPI ではノード数が偶数のとき，ハードウェアバリア機構を用いて，非常に高速にバリアをとっている．しかしながらノード数が奇数のときは，このハードウェアバリア機構を用いておらず，この場合は FMPL(SW) のバリア同期より遅くなっている．

5.3 ブロードキャスト

図 8 に長さ 8KB の配列のブロードキャスト時間を示す．ブロードキャストの測定では，ルートプロセスを反復ごとに変えて，反復させて測定した．

FMPL のブロードキャストは`fmpl_bcast` を示している．`fmpl_bcast` は 1 対 1 通信を用い二分木で通信するため，ブロードキャスト時間はプロセス数 p に対し $O(\log p)$ となっている．MPI では`-rdma` オプションの有無に関わらず通信遅延はほとんど変わっていない．MPI では，バリア同期と同様にノード数が偶数のとき，ハードウェアブロードキャスト機構が利用され，非常に高速にブロードキャストを実現しているが，ノード数が奇数のときはハードウェア機構を利用することができず，オーバヘッドが大きくなっている．

5.4 リダクション

図 9 に長さ 8KB の配列要素の総和を求めるリダクション`fmpl_reduce` の経過時間を示す．リダクション時間の計測は，ブロードキャストと同様にルートプロセスを反復ごとに変え，反復させて測定した．

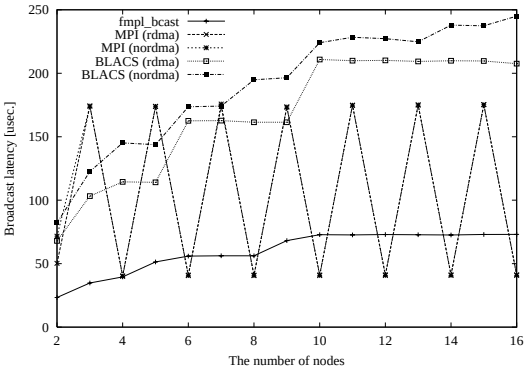

図 8 長さ 8KB の配列のブロードキャスト時間

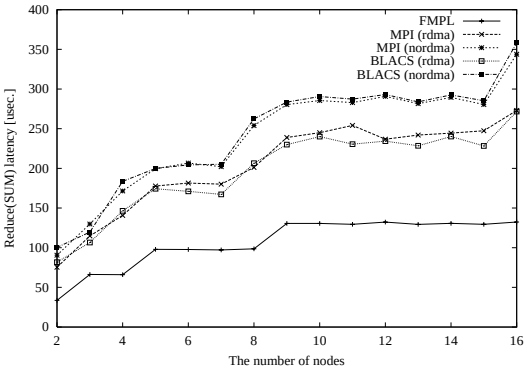

図 9 長さ 8KB の配列要素の総和計算時間

リダクションはいずれのライブラリも二分木で通信して求めるため，リダクション時間はプロセス数 p に対し $O(\log p)$ となっている．8KB 程度の小さい配列であれは，各プロセスの演算処理に比べ，通信遅延が大きく影響し，FMPL のリダクション時間は他の半分程度となっている．

6. まとめと今後の課題

高速メッセージパッシングライブラリ FMPL を開発している．FMPL は SR8000 のリモート DMA 転送に低コストの同期操作を加えた汎用 1 対 1 通信を提供し，その低コストの 1 対 1 通信を利用した集合通信および通信ハードウェア機構を利用した集合通信を提供する．FMPL は BLACS など上位のメッセージパッシングライブラリの構築のためだけではなく，SR8000 の通信ハードウェア性能を最大限に引き出すことが必要なアプリケーションのために設計されている．

1 対 1 通信の同期操作は，受信側からのリモート DMA 転送であり，低コストを実現するために TCW 再利用型送信を用い SR8000 のネットワークハードウェアの起動のみで送信を行っている．この受信側からのリモート DMA 転送で，送信側に受信領域のヘッダ情報が伝えられ，送信側はリモート DMA 転送によりメッセージを書き込む．この方式により，通信遅延は 8 バイトのメッセージの 1 対 1 通信で片道 12.8μsec を達成した．この通信遅延は MPI の 20μsec，BLACS の 34μsec を大幅に改善している．通信スループットもほぼネットワークピーク性能を達成し，メッセージサイズが 16MB のとき 999MB/s を達成した．

ブロードキャストについては，SR8000 のハードウェア機構の利用制限を実行時に調べるコストが高いため，FMPL の 1 対 1 通信で構成したものと，通信ハードウェア機構を用いたものの両方を用意し，ユーザが適宜利用できる形とした．バリア同期については，デフォルトで作成されるコミュニケータ`FMPL_COMM_WORLD` については，ハードウェアバリア機構が利用され，それ以外では TCW 再利用型送信を用いた軽い同期機構で構成される．

現在 FMPL は BLACS に必要なライブラリの実装がほぼ終わり，今後，最適化された BLACS を作成していく予定である．FMPL および最適化された BLACS は，TACC にて配布されることが予定されている．

参 考 文 献

1) Blackford, L. S., Choi, J., Cleary, A., D'Azevedo, E., Demmel, J., Dhillon, I., Dongarra, J., Hammarling, S., Henry, G., Petitet, A., Stanley, K., Walker, D. and Whaley, R. C.: *ScaLAPACK Users' Guide*, SIAM (1997).
2) Dongarra, J. and Whaley, R. C.: A User's Guide to the BLACS v1.1, Technical Report CS-95-281, University of Tennessee (1995).
3) Gropp, W., Lusk, E., Doss, N. and Skjellum, A.: A high-performance, portable implementation of the MPI message passing interface standard, *Parallel Computing*, Vol. 22, pp. 789–828 (1996).
4) Lauria, M. and Chien, A.: MPI-FM: High Performance MPI on Workstation Clusters, *Journal of Parallel and Distributed Computing*, Vol. 40, No. 1, pp. 4–18 (1997).
5) O'Carroll, F., Tezuka, H., Hori, A. and Ishikawa, Y.: The design and implementation of zero copy MPI using commodity hardware with a high performance network, *Proceedings of the 1998 International Conference on Supercomputing (ICS98)*, ACM, pp. 243–250 (1998).
6) 建部 修見, 児玉 祐悦, 関口 智嗣, 山口 喜教: リモートメモリ書き込みを用いた MPI の効率的実装, 情報処理学会論文誌, Vol. 40, No. 5, pp. 2246–2255 (1999).
7) 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 の実装と評価, 情報処理学会論文誌, Vol. 40, No. 5, pp. 2256–2268 (1999).