2003－HPC－95　(19)
2003／8／5

地球シミュレータ用高速ノード間通信ライブラリの構築と評価

板倉憲一[†]　宇野篤也[†]

地球シミュレータの通信機構は，MPI2 の単方向通信に非常に適合した通信プリミティブを持っている．これによって，単方向通信に特化した API を作成し，より低レイテンシで高速な通信ライブラリをユーザに提供することができる．また，既存 MPI の実装では，演算と通信のオーバラップができない．地球シミュレータのハードウェアの構成では，CPU での演算と通信機構の通信処理は同時に行うことができ，メインメモリのスループットもそれをサポートできる．さらに，地球シミュレータの通信機構は，一定のストライドを持った非連続なメモリ領域のデータを対象としたデータ転送プリミティブを持っており，これをユーザが利用するための方法が提供されていない．

これらの点から，本研究ではハードウェアの性能を引き出しやすく，アプリケーションのチューニングの方法を広げるようにより高速な通信ライブラリを構築し，その性能評価を行った．評価の結果から，短メッセージにおいては，従来より 46% レイテンシを短縮することが可能となった．また，演算時間で隠蔽できる通信処理のパターンにおいて，高速化が可能なることを示した．しかし，ストライド転送おいては，メモリバンクのコンフクトなどが原因となりスループットは高くない結果となった．

Development and performance evaluation of a message passing library on the Earth Simulator

Ken'ichi ITAKURA[†] and Atsuya UNO[†]

In the Earth Simulator, all nodes have the the Remote Access Control Unit which is responsible for the communication with other nodes. This unit has a primitive functions which are suitable for the one side communication in MPI2. If API's are supported to use hardware functions directly, they will be lower latency and higher throughput.

The installed MPI library is not supporting to overlap with CPU execution and data transfer. However, the hardware is supported. Furthermore, this library is not supporting the special function of the Remote Access Control Unit. The RCU has the function which carries out communication for a discontinuous memory domain with fixed stride.

In this study, we develop new message passing library which provide API's to use directory hardware functions. From the evaluation, the latency of the short message is 46% shorter than the installed library. Moreover, the results show that CPU execution can overlap with the data transfer. However, stride data transfer don't show the performance, because of the memory bank conflicting.

1. はじめに

地球シミュレータ[1] は Linpack 性能 が 35TFlop/s の世界最高速並列計算機である．全体で 640 個の計算ノードから構成される分散メモリ型並列計算機であり，ノード間の並列化には MPI を用いるのが一般的である．現在の地球シミュレータで使える MPI の実装は，ハードウェアの機能を十分に活かしきれていない[2]．例えば，地球シミュレータの通信機構は，MPI2[3] の単方向通信と非常に相性の良い実装となっており，さらに短いレイテンイシでユーザに提供することが可能である．また，地球シミュレータのノード間通信用ハードウェアはブロックストライド転送に対応している．このため，地球シミュレータ専用の MPI の実装で非連続領域の扱いをそのままハードウェアの転送命令に翻訳することにより，ブロックストライド転送による，袖領域のデータ交換や転置転送が素直に実行できる．MPI で標準的に定義されているデータタイプでは連続領域しか扱えない．派生データタイプを用いると，ユーザインターフェイス上では非連続領域を扱うことができるが，ほとんどの MPI の実装ではそれはソフトウェア的に処理がなされいる．これは，一般的な通信用ハードウェアでは非連続領域を効率良く扱う方法が実装されておらず，ソフトウェアで処理した方が柔軟性が高まるためである．

本研究では，地球シミュレータのハードウェアの機能を有効に利用できるようにするために，既存の MPI ライブラリの上で利用できる高速な通信ライブラリを

† 海洋科学技術センター，地球シミュレータセンター
Earth Simulator Center, Japan Marine Science and Technology Center

構築し，その評価を行なった．これらは，標準の通信関数と引数の型や個数を極力揃えるようにし，ユーザが移行しやすいことを考慮した．

2. 地球シミュレータの通信機構

2.1 ノード間通信制御ハードウェア

地球シミュレータのノードには 8 個の計算用ベクトルプロセッサ (AP) とリモートアクセス制御装置 (RCU: Remote Access Control Unit) があり，16GB の主記憶を共有している．RCU は 12.3 GB/sec の双方向リンクで単段のクロスバネットワーク (IN: Interconnection Network) と繋っており，ノード間の通信を処理する．この RCU 内にはアドレス変換テーブル情報を登録できるので，ユーザーの論理アドレス空間の番地によって通信の送受信のデータを指定できる．

RCU を制御するノード間アクセス命令は，２ノード間の主記憶データ転送機能であり，その実行形態として，同期型と非同期型がある．同期型は，命令終了時にリプライを CPU に返すタイプであり，非同期型は，命令終了時に主記憶上に終了ステータスを書き込む．非同期型ノード間アクセス命令は，命令実行時にデータ転送リクエストをシステム上のリングバッファにキューイングし，直に CPU を解放する．

リングバッファは非同期型ノード間アクセス命令の転送パラメータ (128B) を格納したバッファであり，論理空間上の連続した領域に 2MB 境界で配置される．これは 4 セット用意されており，OS 用，MPI ライブラリの制御用，MPI のユーザ通信データ用，予備に分れている．本研究での通信ライブラリは，既存の MPI に付加する形で実装し，リングバッファの取扱いは MPI に依頼して行う．これにより，通常の MPI の通信と共存している．

2.2 グローバルメモリ

RCU によるノード間データ転送命令は，ユーザレベルでは論理アドレス空間によって，対象となる番地が指定されるが，この時は通常の CPU の使うアドレス空間ではなく，グローバルアドレス空間での論理アドレスを指定しなくてはならない．これは，RCU が通信を行うときに，指定された論理アドレスが必ず物理メモリアドレス空間上に対応している必要があるためである．このように，常に物理メモリアドレスにピンダウンしたメモリをグローバルメモリと呼び，640 ノードで共通のグローバルアドレス空間上のアドレスを持つ．

図 1 に グローバルアドレス空間とユーザの仮想アドレス空間の関係を示す．このピンダウンされたグローバルアドレス空間のメモリ領域は，通常の CPU の使うアドレス空間にもマップされているため，通信で送受信するデータ領域をそのまま CPU が扱うこともできる．

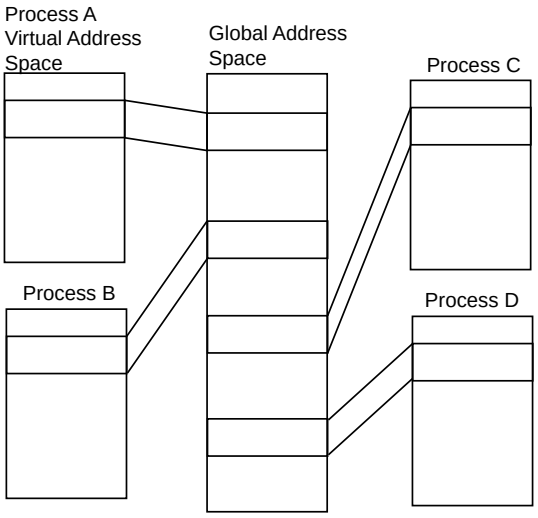

図 1 グローバルアドレス

グローバルアドレスは，コンパイル時に -gmalloc オプションを付け，Fortran90 の alloc 文もしくは MPI_Alloc_mem によりユーザが確保できる．地球シミュレータでは，CPU の通常の演算命令に対してはグローバルアドレスのメモリと通常のメモリは差異はなく，ユーザアプリケーションで確保できる約 15GB のメモリを全てグローバルアドレスとして確保することも可能である．

2.3 非同期型ノード間アクセス命令

非同期型ノード間アクセス命令の通信形態は Block 転送，3 ディスタンス転送，リスト転送の 3 種類がサポートされている．それぞれ，read (remote 側から local 側への転送), write (remote 側から local 側への転送) がある．ここでは，3 ディスタンス転送のついて，その詳細を述べる．地球シミュレータのハードウェアがサポートするストライド転送は 3 つの距離をパラメータとして規則的な様々なデータパターンに基づいてメモリをアクセスし，データの送受信を行う．アクセスパターンのパラメータには以下の６つがある．

- 第一ディスタンス
- 第一要素数
- 第二ディスタンス
- 第二要素数
- 第三ディスタンス
- 通信量

さらに，３つのディスタンスは，local 側と remote 側で独立して決めることができる．図 2 に 3 ディスタンスの転送イメージを示す．

２次元のデータを X-Y 方向でプロセスに分割し，その境界面においてデータを相互に送り合うプログラムでは，各プロセスの担当しているデータ領域にデータ転送のための袖領域を余分に確保し，計算の手順を単純化するのが一般的である．このとき，各プロセスの

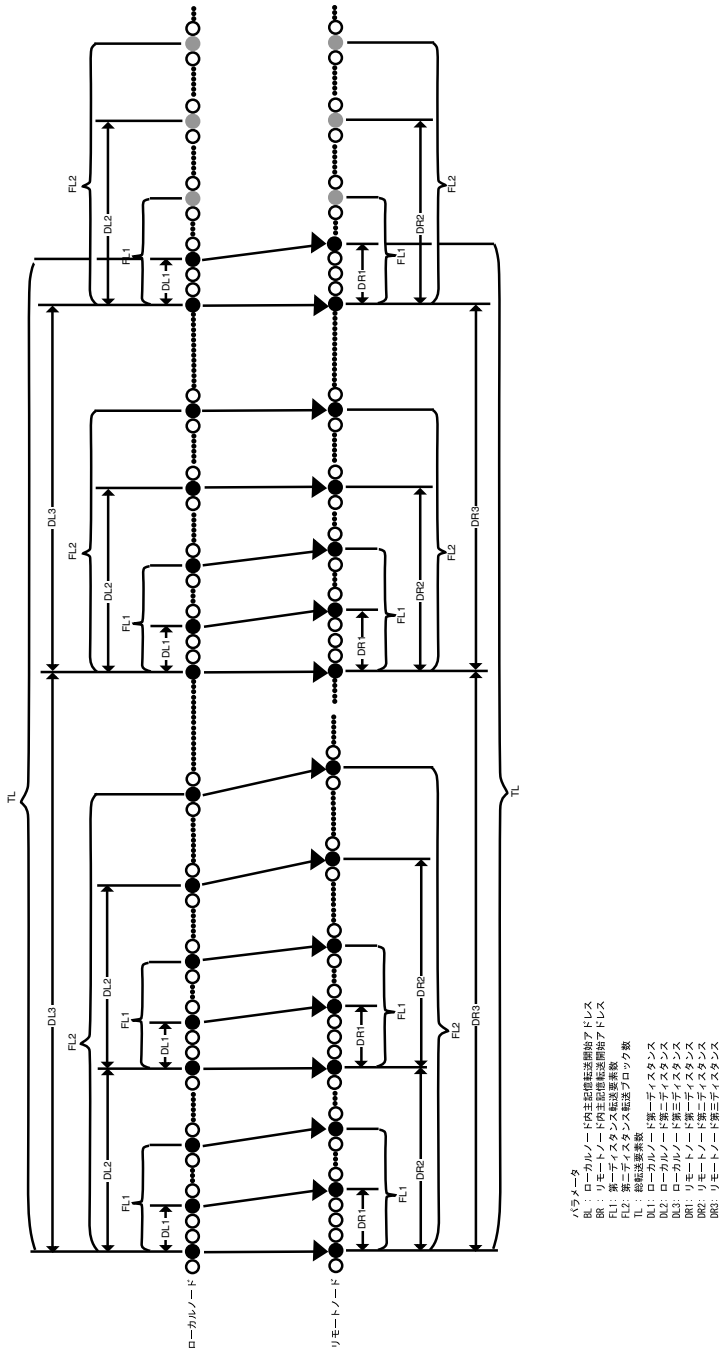

図 2　3 ディスタンス転送イメージ (地球シミュレータ設計書より)

持っている２次元データのうち，一方向は連続メモリアクセスになるが，もう一つの方向は非連続で一定の間隔を置いたストライドアクセスとなる．

さらに，３次元以上でデータをプロセスに分割した場合に，その境界面のデータはあるブロックを一定の間隔で繰り返すブロックストライドのアクセスとなる．このような場合に，３ディスタンス転送によって単一の命令により袖領域のデータを転送することができる．

また，３次元 FFT 等ででよく用いられる転置転送は，ローカル側とリモート側でディスタンスやストライドのパラメータが異る例である．

表 1　既存の MPI 実装と高速通信版 (RNA 版) の対応

機能	MPI 版	RNA 版
通信領域設定	MPI_Win_Create	RNAMPI_Win_Create
	MPI_Alloc_mem	RNAMPI_Alloc_mem
片側通信	MPI_Put	RNAMPI_Put
	MPI_Get	RNAMPI_Get
３ストライド通信	-	RNAMPI_Put3
	-	RNAMPI_Get3
通信制御	MPI_Win_fence	RNAMPI_Win_fence

3. 高速ノード間通信ライブラリの実装方法

3.1 MPI に準拠したインターフェイス

MPI2 では，片側通信機能が用意された．これによって，リモート側での操作を不要とした read 処理 (MPI_Get), write 処理 (MPI_Put) が行える．現時点で，MPICH や LAM といった一般的なライブラリでも，このインターフェイスが実装されており，片側通信機能を使ったアプリケーションも増加していくと考えられる．また，地球シミュレータでは，グローバルメモリにデータを置いて MPI_Put を用いるのが最も高速であるため，これらを使ったプログラム作成を推進している．

我々は，地球シミュレータの通信ハードウェアの機能に直接合う MPI_Put, MPI_Get について，高速通信版 (以下 RNA 版) として RNAMPI_Put, RNAMPI_Get を作成した．表 1 に対応する関数を示す．対応する既存関数があるものは，同じ引数を取るように設計した．３ストライド転送は，通常 MPI_Datatype の引数の部分を配列に変更し，転送要素データタイプ，第一ディスタンス，第一ディスタンス要素数，第二ディスタンス，第二デシスタンス転送ブロック数，第三ディスタンスの情報を渡す．

4. 実　　験

4.1 1 対 1 転送の性能

最も基本的な通信性能評価として，２プロセスを２ノードに配置しノード間のピンポン転送を MPI_Put もしくは RNAMPI_Put によって行い，片道の通信に要した時間からそのスループットを求めた．時間測定には，MPI_Win_fence の時間を含む．結果のグラフを図 3 に示す．

図中の MPI, RNA はそれぞれ，既存の MPI 通信ライブラリ，本研究で実装した高速通信ライブラリを示す．また，MPI_with_assertion は既存の MPI 通信ライブラリを使用しているが，MPI_Win_fence に高速化のヒントを与えている．いくつかの MPI 関数では，その内部処理の最適化に有効な情報を MPI 処理系に与えることができるように，assertion 引数が定義されている．地球シミュレータ向け MPI では，MPI_Win_fence について，(1) RMA 通信で MPI_Put または MPI_Get

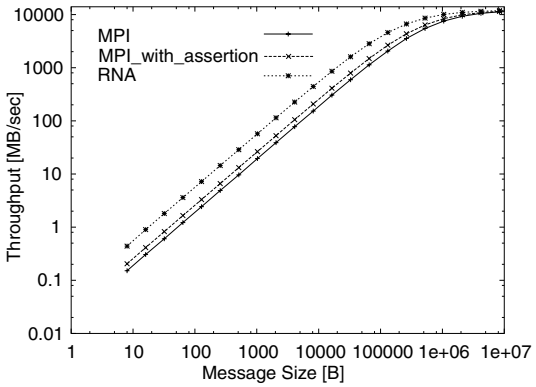

図 3　既存の MPI 実装と高速通信ライブラリのメッセージ長とスループット

だけを利用している, (2) ローカル側のメモリ領域とリモートのウィンドウで参照されるメモリ領域がグローバルメモリ上に割り当てられている，といった情報を MPI 処理系に与えるための assertion 仕様が実装されている．おの assertion を MPI_Win_fence に引数として渡すことで，その内部処理は最適化され，時間の短縮が図られる．

結果から，MPI_with_assertion は MPI よりも高速であるが，RNA 版は更に高いスループットを示す．最終的なスループットは同程度であるが，立ちあがりのオーバヘッドが短い．表 2 に 8B と 64MB のメッセージ長に対する測定結果を示す．

表 2　既存の MPI 実装と高速通信ライブラリの性能測定結果

(8B メッセージ)	片道の時間 [micro sec]	スループット [MB/sec]
MPI	50.4	0.151
MPI_with_assertion	37.3	0.204
RNA	17.3	0.440

(64MB メッセージ)	片道の時間 [micro sec]	スループット [MB/sec]
MPI	5338.5	1198.8
MPI_with_assertion	5325.3	1201.7
RNA	5306.1	1206.1

多くのアプリケーションプログラムでは，MB 単位の通信よりも KB 単位の通信の方が多い．例えば地球シミュレータでターゲットとしている大気シミュレーションのコードにおいても数 KB の通信が使われており，この通信のオーバヘッドを低減することは実アプリケーションの高速化に高く貢献できるものと考えられる．

4.2 通信と演算のオーバラップ処理

次に，通信と演算のオーバラップ処理の性能評価を行う．一般に，MPI_Isend や MPI_Irecv は通信処理と他の演算処理をオーバラップさせるために使われる．但し，

```
subroutine work(r,n)
integer n,i
real*8 r(n),a1,a2
do i=1,n
  a1=r(i)*r(i)+r(i)
  a2=a1*r(i)+r(i)
  r(i)=a2*r(i)+a1
enddo
return
end
```

(a) 演算のルーチン

```
subroutine work2(p,w,n,to)
integer w,n,err
real*8 p(n)
call mpi_win_fence(0,w,err)
call mpi_put(p,n,MPI_REAL8,to,
             0,n,MPI_REAL8,w,err)
call mpi_win_fence(0,w,err)
call mpi_win_fence(0,w,err)
call mpi_put(p,n,MPI_REAL8,to,
             0,n,MPI_REAL8,w,err)
call mpi_win_fence(0,w,err)
return
end
```

(b) 通信のルーチン

```
subroutine work3(p,q,w,n,to)
integer w,n,err
real*8 p(n),q(n/2)
call mpi_win_fence(0,w,err)
call mpi_put(p,n,MPI_REAL8,to,
             0,n,MPI_REAL8,w,err)
call work(q,n/2)
call mpi_win_fence(0,w,err)
call mpi_win_fence(0,w,err)
call mpi_put(p,n,MPI_REAL8,to,
             0,n,MPI_REAL8,w,err)
call work(q,n/2)
call mpi_win_fence(0,w,err)
return
end
```

(c) 演算と通信を同時実行するルーチン

図 4 通信と演算のオーバラップ処理

表 3 通信と演算のオーバラップ処理の時間測定結果

通信ライブラリ	計測ルーチン	処理時間 [sec]
MPI	(a) 演算ルーチン	0.4698
	(b) 通信ルーチン	0.08489
	(c) 同時実行ルーチン	0.5547
RNA	(a) 演算ルーチン	0.4698
	(b) 通信ルーチン	0.08474
	(c) 同時実行ルーチン	0.4700

本当にオーバラップできるかどうかは，ハードウェアの実装および MPI ライブラリの実装に依存する．また，MPI_Put, MPI_Get もその通信完了は MPI_Win_fence で確認するため，他の演算とオーバラップをする実装をすることがユーザからは期待される．

ここでは，図 4 に示す a) 演算のみの場合，b) 通信のみの場合，c) 両者をオーバラップさせた場合の 3 種類のサブルーチンに対して，既存の MPI 実装を用いた場合と表 1 に対応する関数を高速通信ライブラリに置き換えた場合で時間測定を行った．測定結果を表 3 に示す．

結果から明らかなように，既存 MPI の MPI_Put は演算とオーバラップできていない．単純な通信ルーチンにおいて，関数コストを詳細に計測した結果，MPI_Put ではほとんと時間を消費しておらず，MPI_Win_fence の部分で通信時間を消費している．また，演算と通信の同時実行ルーチンにおいても，演算部分 (call work()) で時間消費された分は考慮されず，MPI_Win_fence の部分で同じ通信時間を消費している．この結果，既存 MPI の実装は MPI_Put のおいては，パラメータの確保だけを行い，MPI_Win_fence で実際の通信処理が行われているように推測される．このような実装は MPI_Get も同じであるだけでなく，MPI_Isend と MPI_Wait の関係においても観測され，通信と演算のオーバラップができない．

これに対して，我々が実装した高速通信ライブラリでは RNAMPI_Put において通信ハードウェアである RCU を起動している．RCU は CPU の処理とオーバラップして通信処理を行うため，高速通信ライブラリの同時実行ルーチンの処理時間はほぼ演算ルーチンの時間だけで終了しており，通信時間は演算によって隠蔽できたことを示している．

演算と通信がオーバラップできるかどうかはアプリケーションに強く依存するところであるが，少なくともハードウェアはその能力を備えているので，通信ライブラリでユーザに機能を提供するべきだと考える．

4.3 ストライド転送性能

ストライド転送の実効性能を評価するために，2 ノード間での片側通信の測定を以下のようなルーチンで行った．

```
rnampi_win_fence(...)
t0=mpi_wtime()
rnampi_put3(...)
rnampi_win_fence(...)
t1=mpi_wtime()
```

ここでは，ストライド転送のパターンとして，データの要素サイズは 8B で固定，第 1 ディスタンスを 8B, 16B, 24B ,,, 128B とし，16MB の領域内の有効なデータを送る．表 4 に測定結果を示す．

結果から，連続アクセスであるディスタンスが 8B の場合に比べて，半分のメモリバンクしか使用しないディスタンスが 16B の時にはスループットが 63% に低下する．さらに 8B*偶数の時には，バンクコンフリクトの影響で性能が低下しディスタンスが 128B の時には 7.8% にまで低下してしまう．このバンクコンフリコトの影響は予想しえたことであるが，ストライド幅を 8B*奇数にした場合にも同様に性能が低下している原因は不明である．8B*奇数の場合にはバンクを順番にではないが，均等に使用するので性能が低下する原因は別のところにあると考えられる．この点は早急に解明したい．

表 4 ストライド転送測定結果

第一ディスタンス [B]	転送要素数	時間 [msec]	スループット [MB/sec]
8	2097152	2030	7881
16	1048576	1612	4962
24	699050	1779	2997
32	524288	1607	2488
40	419430	1778	1800
48	349525	1623	1642
56	299593	1778	1285
64	262144	1608	1243
72	233016	1784	996
80	209715	1623	985
88	190650	1780	816
96	174762	1624	820
104	161319	1777	692
112	149796	1618	706
120	139810	1780	599
128	131072	1624	615

5. ま と め

本研究では，地球シミュレータのノード間高速通信ライブラリを構築し，その評価を行った．地球シミュレータのノード間通信のハードウェアに適合した通信パターンに対しては，できるだけ単純化した API によってハンドリングすることにより，低レイテンシの通信を実現することが可能となった．また，本来ハードウェアが持っている計算と通信のオーバラップ機能をユーザに提供することが可能となり，アプリケーションによっては処理時間の短縮を行うことができる．通信ハードウェアの持っている ストライドデータ転送機能は，アクセスするパターンがバンクコンフリクトを起す場合に，トータルの通信性能が低下する結果となった．これに対して，バンクコンフリクトが起きないような奇数飛びのアクセスを行うなどのチューニングによって性能向上ができるかどうかを引き続き調査していく．また，いくつかのアプリケーションを用いて，連続アクセスでの転送，ストライドデータの転送を既存の MPI の実装と比較し，より高速な通信ライブラリを開発したいと考えている．

謝辞

本研究を行うにあたり，地球シミュレータを利用する機会を与えて頂いた地球シミュレータセンター 佐藤哲也センター長に感謝します．本研究は地球シミュレータ共同プロジェクト「地球シミュレータのマルチノード用並列計算ライブラリの構築」の一環として実施されたものである．

参 考 文 献

1) Habata, S., Yokokawa, M. and Kitawaki, S., “The Earth Simulator System,” NEC Research & Development, Vol.44, No.1, pp.21–26, Jan. 2003.
2) 上原 均，田村 正典，板倉 憲一, 横川 三津夫,「地球シミュレータの MPI 性能評価」, 情報処理学会論文誌：ハイパフォーマンスコンピューティングシステム, Vol44, No.SIG 1 (HPS 6), pp.24–34, Jan. 2003.
3) MPI Forum: MPI-2: Extensions to the Message-Passing Interface (1997). http://ww.mpi-forum.org