メッセージ通信の分散メモリ型並列計算機性能への影響

——通信と演算のオーバラップと直接メッセージ受信の効果——

堀江健志† 小柳洋一† 今村信貴†
林憲一† 清水俊幸† 石畑宏明†

本論文では，メッセージハンドリングのオーバヘッドを削減する三種類の方法，(1)送信と演算とのオーバラップ，(2)直接メッセージ受信，(3)受信と演算とのオーバラップを適用した場合の効果を定量的に評価する．評価にはわれわれが開発したメッセージレベルシミュレータを用い，メッセージパッシングで記述された10種類の応用問題を対象とする．シミュレーションの結果，通信と演算とのオーバラップと直接メッセージ受信により通信のオーバヘッドが減少すること，特に，受信と演算とのオーバラップにより通信オーバヘッドとともにアイドル時間も減少し性能が大幅に向上することを示す．

Effect of Message Communication on Distributed-Memory Parallel Computer Performance

——Overlapping Communication with Computation and Direct Message Receiving——

Takeshi Horie,† Youichi Koyanagi,† Nobutaka Imamura,†
Kenichi Hayashi,† Toshiyuki Shimizu† and Hiroaki Ishihata†

The performance of message-passing applications depends on cpu speed, communication, and message handling overhead. In this paper we investigate the effect of applying techniques to reduce message handling overhead on the execution efficiency of ten different applications. Using a message level simulator set up for the architecture of the AP 1000, we showed that overlapping communication with computation and transferring received messages directly to a user area improves performance. In particular the speedup of overlapping computation with message reception is almost double because this reduced message handling overhead and idle time.

1. はじめに

分散メモリ型並列計算機の設計では，メッセージハンドリングのオーバヘッドの削減と高スループットで低レイテンシな相互結合網（ネットワーク）の構築が重要な課題である．本論文では，メッセージハンドリングのオーバヘッドを削減する手法を適用したときの効果について，シミュレーションによる10種類の応用問題の定量的な評価を報告する．

メッセージハンドリングには，メッセージのフォーマッティング，ネットワークへのメッセージ送信，メッセージバッファからのメッセージサーチ，メッセージ受信待ち機構，ユーザデータ領域へのコピーが含まれる．メッセージハンドリングは二種類のオーバヘッド時間に分けることができる．一つは，メッセージの組み立てと通信機能の設定などの一定の時間を要するもの，もう一つは，メッセージサイズに比例する時間を要するものである．メッセージハンドリングを削減するためのいくつかの方法が提案され[1)~3)]，前者のオーバヘッド時間は，メッセージフォーマッタやメッセージを探すハードウェアなどにより削減することができる．

それに対して，メッセージ長に比例したオーバヘッドを削減するためには，一つの方法として演算と通信とをオーバラップさせる方法が考えられる．メッセージをネットワークへ送信するとき，送信完了までの間に演算により送信領域がアクセス（書き込み）されないならば送信と演算をオーバラップすることができる．同様に，メッセージを受信するとき，メッセージ受信領域がすぐにアクセスされない限りメッセージヘッダの到着直後にプロセッサは演算を開始すること

† (株)富士通研究所並列処理研究センター
Parallel Computing Research Center, Fujitsu Laboratories Ltd.

ができる．このときメッセージがユーザデータ領域へ転送されている間は，演算と受信がオーバラップすることになる．

受信時のメッセージハンドリングのオーバヘッドを削減するための他の方法として，受信したメッセージを直接ユーザ領域へ転送する方法がある．これはメッセージバッファからユーザ領域へのメッセージのコピーのオーバヘッドを取り除くことができる．

本論文では，分散メモリ型並列計算機における，通信と演算のオーバラップと直接メッセージ受信の効果を定量的に評価する．本論文で得られた結果は，10種類の応用問題を対象にメッセージレベルシミュレーションにより得られたものである．われわれは，評価のためにメッセージレベルシミュレータを開発した．本シミュレータは，メッセージパッシングアーキテクチャを対象にしており，応用問題に対する通信性能や演算性能の影響などを容易に調査することが可能である．また，対象とした応用問題は，並列化コンパイラが出力したコードを含め数値計算の分野のプログラムである．

評価の結果，演算と通信のオーバラップならびにメッセージを直接ユーザ領域へ転送することにより，通信のオーバヘッドは削減されることがわかった．特に，メッセージ受信におけるオーバラップにより，性能が2倍になる場合もあった．これは，受信のオーバラップによりメッセージハンドリングとともにアイドル時間も減少するからである．

本論文では，評価のために使用したシミュレータと応用問題などについて述べた後，通信オーバヘッドを削減する三種類の手法を適用した場合の効果についてシミュレーションによる結果を示す．最後に，関連研究について簡単に述べ，得られた結果についてまとめる．

2. モデル，シミュレーション，応用問題

本章では，アーキテクチャモデル，シミュレータ，応用問題について述べる．

2.1 アーキテクチャモデル

本論文では，メッセージパッシングのアーキテクチャを持つ AP 1000[4)~6)]をシミュレーションの基本モデルとした．プロセッサは1対1通信とバリア同期専用のネットワークで結合されている．この1対1通信ネットワークはすべてのプロセッサ，行方向あるいは列方向プロセッサへの放送通信機能を持っている．なお，本論文の評価では，AP 1000 が持つ放送ネットワークは使用しなかった．

AP 1000 は，基本的なプログラミングモデルとしてブロッキングしないメッセージパッシングをサポートしている．メッセージの送信は，宛先プロセッサの ID を指定して，相互結合網に直接メッセージを転送し，転送完了を待つ．送信したメッセージは，宛先プロセッサのメッセージバッファに自動的に（プロセッサの介入の必要なく）転送される．このメッセージバッファはリングバッファになっており受信プロセッサは，メッセージバッファを直接アクセスすることができ，必要があればユーザのデータ領域にコピーする．

図1に通信モデルを示す．通信遅延時間は，次のようになる．

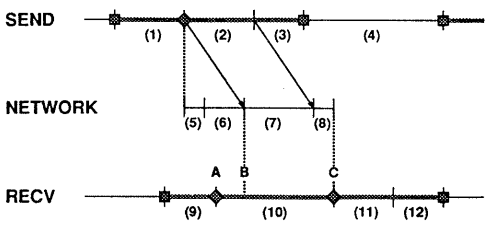
送信 *send_prolog_time*＋*send_msg_time*
×*msg_size*＋*send_epilog_time*

受信（メッセージコピーなし） *recv_prolog_time*
＋*recv_wait_time*＋*recv_epilog_time*

受信（メッセージコピーあり） *recv_prolog_time*
＋*recv_wait_time*＋*recv_msg_time*
×*msg_size*＋*recv_epilog_time*

2.2 シミュレータ

通信の性能への影響を評価するためにわれわれはメッセージレベルシミュレータ（MLSim）を開発した．MLSim は，AP 1000 上に開発されており，メッセージパッシングマシンをシミュレーションするト



(1) *send_prolog_time*
(2) *send_msg_time* × msg_size
(3) *send_epilog_time*
(4) *computation_time_factor* × computation_time
(5) *network_prolog_time*
(6) *network_delay_time* × distance
(7) *network_msg_time* × msg_size
(8) *network_epilog_time*
(9) *recv_prolog_time*
(10) *recv_wait_time*
(11) *recv_msg_time* × msg_size
(12) *recv_epilog_time*
A *msg_wait_time*
B *msg_header_recv_time*
C *msg_recv_time*

図1 通信モデル
Fig. 1 Communication model.

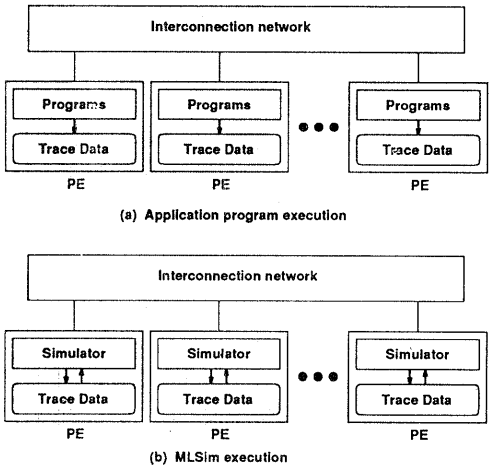


図 2 メッセージレベルシミュレータ
Fig. 2 Message Level Simulator.

レースドリブンシミュレータである[7].

MLSim は，AP 1000 で動作する実際の応用問題から得られた実行トレースを入力として，トレース情報の時刻を変更しながら通信の振舞いをシミュレーションする．**図 2** にシミュレータの実行を示す．実行トレースは AP 1000 のプロセッサのメモリ上に蓄積され，実行終了後トレース情報をメモリに残したまま MLSim を起動する．これは，実行トレースの大きさがプロセッサ台数に比例して膨大になり，ホストプロセッサにローディングできないこと，さらに，MLSim そのものが並列に実行でき高速にシミュレーションできることが理由である．

パラメータとして *send_prolog_time* などの送信遅延時間などが MLSim に与えられ，メッセージ通信の順序関係を保ちながら，通信遅延やアイドル時間を求めていく．

シミュレーションは，トレース情報をもとに各プロセッサでメッセージ通信とバリア同期に必要な時間を計算しながら実行する．メッセージの送信イベントに対しては，実際にメッセージを送信する．このメッセージには，送信時刻とメッセージが記されている．メッセージ受信イベントに対しては，送信プロセッサから該当するメッセージを受信する．受信したメッセージから受信時刻を求め，メッセージ受信に必要な時間を求める．純粋なユーザプログラムの実行時間は，ライブラリ終了と次のライブラリ開始の間の時間として求めることができる．この時間も計算性能をパラメータとして変更することができる．

```
#
#                    model
#
# ---- computation ----          # ---- send ----
computation_factor  0.25         send_prolog_time     0.0
#                                 send_msg_time        0.049
# ---- network ----              send_epilog_time     0.0
network_prolog_time 0.16         # ---- recv ----
network_msg_time    0.049        recv_prolog_time     0.0
network_delay_time  0.16         recv_msg_time        0.136
network_epilog_time 0.0          recv_epilog_time     0.0
```

図 3 本評価に用いたパラメータ
Fig. 3 Parameters for MLSim.

本論文のシミュレーションでは，プロセッサの性能は AP 1000 のプロセッサの 4 倍で，メッセージハンドリングにかかる一定時間のオーバヘッドは 0 とし，ネットワークの性能は AP 1000 と同じものとする．これは，将来の並列計算機を考えた場合，相互結合網の性能に比べてプロセッサの性能の向上の方が著しいこと，メッセージのハンドリングは送信におけるメッセージのフォーマッティングや受信におけるメッセージのサーチがハードウェア化され現在の AP 1000 に比べてかなり小さくなると考えたからである．**図 3** に本評価で用いたパラメータを示す．

ネットワークにおける輻輳による影響を評価するために，MLSim は物理的な転送単位のレベルでのネットワークシミュレータとともにシミュレーションを行うことができる．しかし，本論文では，対象としてい

表 1 応用問題の内容
Table 1 Applications.

応用問題	内容
LINPACK	密行列 (1000×1000) をブロック LU 分解で解くプログラム[13].
SCG	二次元ポアソン方程式を SCG 法で解く[14]. 行列は，200×200.
MD	領域分割を用いた液体原子の分子動力学シミュレーション[15]. 粒子数は 21600.
QCD	QCD のモンテカルロシミュレーション[16]. 格子は，16×16×16×16.
SHALLOW	並列言語 Dataparallel-C で記述した二次元グリッド *shallow-water* 方程式[17].
OCEAN	並列言語 Dataparallel-C で記述した海流シミュレーション[17].
ORTHES	OXYGEN でコンパイルされた Householder 変換プログラム[18].
TSDE	OXYGEN でコンパイルされた SOR プログラム[18].
AMBER	粒子分割を用いた水とタンパク分子の分子動力学シミュレーション[19]. 原子数は 2000.
SLALOM	直方体内部の光の分布をラジオシティで解く[20]. パッチ数は 500.

るアプリケーションがすべて規則的な通信パターンを使用していること，フリットレベルのネットワークシミュレータがかなりの時間を必要とすることから，ネットワークの輻輳の影響は考慮しなかった．

2.3 応用問題

表1に10種類の応用問題の内容について示す．なお，各アプリケーションの主要な計算部分だけを測定の対象とし，ホストとプロセッサ間のデータ転送の時間は含めなかった．各アプリケーションのメッセージ長などの詳細な性質については文献7)に示す．

3. 通信と演算のオーバラップと直接メッセージ受信

3.1 評価する通信方法

以下の三種類の方法によりメッセージ通信のオーバヘッドを削減する．

- 送信と演算のオーバラップ

送信と演算のオーバラップでは，プロセッサはメッセージ送信命令を発行した直後，送信メッセージ領域にデータを書き込まない限り演算を開始することができる（**図4**参照）．

送信の演算のオーバラップの方法として，メッセージをバッファに一旦コピーする方法も考えられるが，コピーのオーバヘッドが生じ，通信遅延が大きくなってしまう．ここでは，プロセッサが直接メッセージをネットワークに送信する場合を考える．

- 直接メッセージ受信

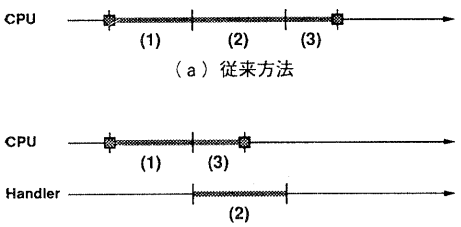
循環メッセージバッファ（リングバッファ）は，非同期にメッセージを受信する場合には有効な機能であり，J-Machine[8]あるいはAP1000[5]はこの受信機構を採用している．しかし，この方法ではメッセージをメッセージバッファからユーザ領域にコピーする必要があり，最近の研究ではそのオーバヘッドが指摘されている[9]．循環メッセージバッファでは，応用プログラムから直接このバッファ領域をアクセスすることもできるが，われわれの調査では，AP1000で動作する多くの応用問題でメッセージバッファからユーザ領域へメッセージをコピーしている．

受信するメッセージを直接ユーザ領域へ転送することにより，コピーのオーバヘッドを削減することができる．メッセージの到着よりも早くプロセッサがメッセージ受信要求を発行した場合，メッセージを直接指定した領域へ転送することができる．そうでない場合には，メッセージは一旦メッセージバッファに転送され，プロセッサがそのメッセージの受信要求を発行したとき，メッセージがユーザ領域へコピーされる．この方法を**直接メッセージ受信**と呼ぶ（**図5**参照）．

メッセージを直接ユーザ領域へ転送する他の方法として，メッセージヘッダに受信アドレスを入れておく方法がある．しかし，受信メッセージをユーザ領域へ転送可能かどうか調べる同期機構とともに送信側で受信メッセージのアドレスを知る必要がある．ここでは，直接メッセージ受信を評価する．

- 受信と演算とのオーバラップ

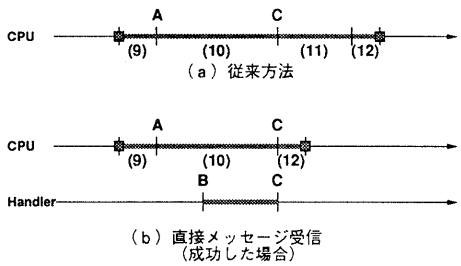
直接メッセージ受信により通信オーバヘッドを減らすことができるが，この方法では受信と演算はオーバラップしていない．メッセージのヘッダが到着するとすぐプロセッサが実行を開始できるとすると，



（a）従来方法

（b）送信と演算のオーバラップ

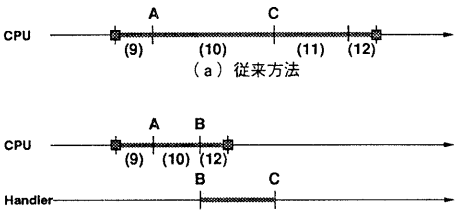
図4 送信と演算のオーバラップ
Fig. 4 Computation and message sending overlap.



（a）従来方法

（b）直接メッセージ受信
（成功した場合）

図5 直接メッセージ受信
Fig. 5 Direct message receiving.



（a）従来方法

（b）受信と演算のオーバラップ

図6 受信と演算のオーバラップ
Fig. 6 Computation and message receiving overlap.

メッセージ受信と演算をオーバラップすることができる．この考えはメッセージのコピーにも適用することができる．メッセージの受信要求がメッセージの到着よりも遅いとき，一旦メッセージをメッセージバッファに転送し，その後，メッセージの受信要求が発行されたときメッセージバッファからユーザ領域へのコピーとプロセッサの実行とをオーバラップさせることができる．この場合，メッセージ受信におけるオーバヘッドはアイドル時間だけになる（**図6**参照）．

より積極的な方法として，プロセッサはメッセージ受信要求を発行した後，メッセージの到着を待たないで実行を開始する方法が考えられる．しかし，メッセージによる同期が成立しなくなり，このままでは適用することができない．ここでは，メッセージ受信要求を発行したらメッセージヘッダが到着するまで待つものとする．

3.2 送信と演算のオーバラップ

プロセッサが送信命令（宛先プロセッサ ID，メッセージ ID，データ領域，サイズ）を発行すると，プロセッサは演算を開始し，DMA コントローラがネットワークへメッセージを転送する．プロセッサが送信領域に書き込みを行うと，メモリフォールトが発生し，プロセッサはその領域が利用可能になるまでそのアクセスを繰り返す．

ハードウェアとして以下の機能が必要になる．

- アクセス範囲を示すレジスタ（トップアドレスとボトムアドレス）とアクセスチェックのための比較器
- メッセージ転送のための DMA コントローラ

アドレス範囲を示すレジスタは，転送開始を示すトップアドレスと転送終了を示すボトムアドレスからなり，DMA コントローラの転送のたびにトップアドレスが更新される．プロセッサがトップアドレスとボトムアドレスの間をアクセスするとアクセス例外を発生させる．

メッセージを連続して転送すると，前の送信処理が終了していない場合が生じる．このときも送信と演算をオーバラップさせるためには，レジスタと比較器が複数必要になる．送信コマンドはキューイングされるだけで，相互結合網へのメッセージ転送は，順次 DMA コントローラにより行われるものとする．ここでは，レジスタと比較器が一つの場合と必要な数だけのレジスタと比較器がある場合をそれぞれシミュレーションした．

表2と**図7**に送信と演算のオーバラップの効果について示す．表2では，レジスタと比較器が一つの場合に，前の送信コマンドが終了しておらず送信コマンドの発行が待たされる割合と，レジスタと比較器が複数ある場合の必要なレジスタと比較器の数も示す．な

表 2 送信オーバラップの効果
Table 2 Effect of sending overlap.

応用問題	台数	ノーマル (ms)	オーバラップ (ms)	フルオーバラップ (ms)	待つ割合 (%)	キューの深さ
LINPACK	64	1193	1193(0.0)	1193(0.0)	1.3	3
SCG	64	1178	1142(3.0)	1142(3.1)	16.3	2
MD	64	991	987(0.4)	987(0.4)	0.4	2
QCD	64	1341	1337(0.3)	1337(0.3)	0.0	1
OCEAN	64	2189	2171(0.8)	2171(0.8)	0.0	1
SHALLOW	64	1081	1002(7.3)	1002(7.3)	0.0	1
ORTHES	64	303	303(0.0)	303(0.0)	0.0	1
TSDE	64	1353	1352(0.1)	1352(0.1)	0.9	23
AMBER	64	285	282(0.9)	280(1.6)	47.8	30
SLALOM	64	696	692(0.7)	691(0.7)	57.1	13
LINPACK	256	500	500(0.0)	500(0.0)	0.3	3
SCG	256	437	409(6.6)	408(6.7)	24.9	3
MD	512	141	139(1.5)	139(1.5)	0.4	2
ORTHES	256	191	191(0.0)	191(0.0)	0.0	1
SLALOM	256	307	301(2.2)	300(2.3)	56.7	26

- （ ）内は，ノーマルに対する性能向上の割合（%）．
- 『オーバラップ』は，一つの送信コマンドだけが演算とオーバラップ可能な場合の実行時間．
- 『フルオーバラップ』は，複数の送信コマンドが演算とオーバラップ可能な場合の実行時間．
- 『待つ割合』は，一つの送信コマンドだけが演算とオーバラップ可能な場合，次のコマンドが前のコマンドの終了を待つ割合（%）．
- 『キューの深さ』は，複数の送信コマンドが演算とオーバラップ可能な場合に必要になったキューの深さ．

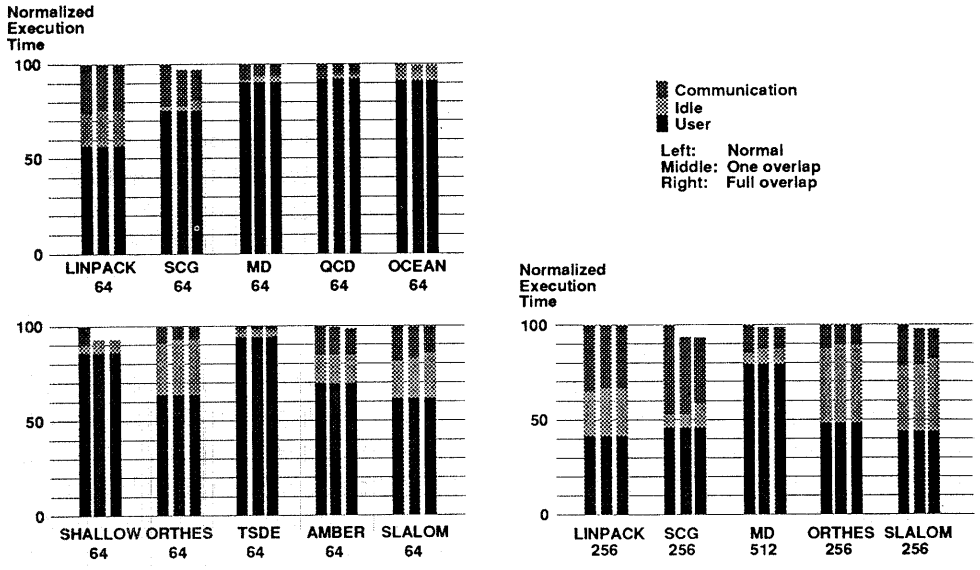


図 7 送信オーバラップの効果
Fig. 7 Effect of sending overlap.

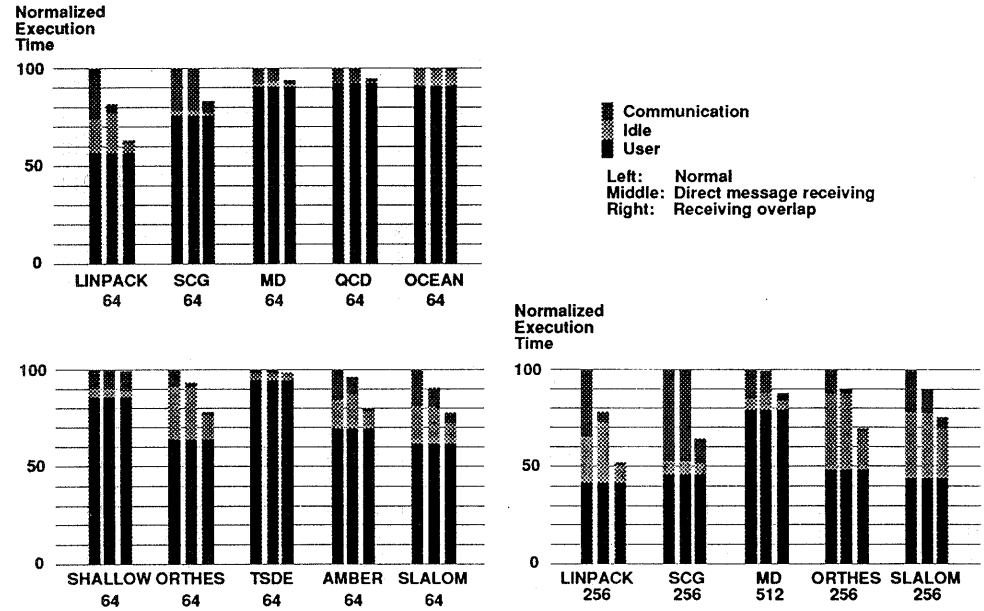


図 8 直接メッセージ受信と受信オーバラップの効果
Fig. 8 Effect of receiving messages directly and overlapping receiving with computation.

お，図7と図8の実行時間は，ノーマルの場合の実行時間で正規化されており，応用問題の下の数字はプロセッサ台数を表す．

レジスタと比較器が一つの場合と可能な限りある場合ではほとんど性能に差がなく，また，性能向上はいずれにしろ大きくない．例えば，プロセッサ256台で実行した SLALOM では，26 個のレジスタと比較器が必要だが性能向上はわずかに 2.3% であった．

理由の一つは送信オーバヘッド時間の割合が小さいからである．もう一つは次のように考えられる．多く

表 3 受信オーバラップと直接メッセージ受信の効果
Table 3 Effect of receiving messages directly and overlapping receiving with computation.

応用問題	台数	ノーマル (ms)	直接受信 (ms)	オーバラップ (ms)	成功率 (%)	キューの深さ
LINPACK	64	1193	977(18.1)	753(36.9)	87.7	7
SCG	64	1178	1178(0.0)	979(16.8)	8.2	3
MD	64	991	990(0.1)	930(6.2)	25.7	3
QCD	64	1341	1341(0.0)	1265(5.6)	2.6	3
OCEAN	64	2189	2189(0.0)	2183(0.3)	32.4	3
SHALLOW	64	1081	1081(0.0)	1071(0.9)	20.1	3
ORTHES	64	303	283(6.7)	237(21.9)	93.2	4
TSDE	64	1353	1353(0.0)	1333(1.5)	1.1	64
AMBER	64	285	274(3.9)	227(20.1)	34.5	64
SLALOM	64	696	630(9.5)	540(22.5)	63.6	6
LINPACK	256	500	390(22.0)	260(48.0)	88.8	—
SCG	256	437	437(0.1)	280(35.8)	5.3	—
MD	512	141	140(0.7)	124(12.3)	34.9	—
ORTHES	256	191	171(10.5)	132(30.7)	93.5	—
SLALOM	256	307	275(10.5)	232(24.6)	57.9	—

- ( ) 内は，ノーマルに対する性能向上の割合 (%).
- 『直接受信』は，直接メッセージ受信の実行時間.
- 『オーバラップ』は，受信と演算がオーバラップしたときの実行時間.
- 『成功率』は，直接メッセージ受信において，メッセージが直接ユーザ領域に転送された割合 (%).
- 『キューの深さ』は，直接メッセージ受信において，プロセッサがメッセージ要求したときにすでにメッセージバッファに到着しているメッセージの数（最大値）．64プロセッサ台数のときのみ評価した.

の場合，プロセッサAがプロセッサBにメッセージを送信した後，プロセッサAはその後プロセッサBが送信するメッセージを待つ．このとき，送信オーバヘッドが減っても，次に受信するプロセッサBからのメッセージの到着時刻は早くはならないため，アイドル時間が増え性能が向上しない．この現象は，SCG あるいは ORTHES のアイドル時間の増加として現れている．

3.3 直接メッセージ受信

直接メッセージ受信では，メッセージの到着よりも早くプロセッサがメッセージ受信要求（送信元プロセッサ ID，メッセージ ID，アドレス，サイズを指定）を発行した場合，メッセージを直接指定した領域へ転送することができる．直接メッセージ受信を実現するためには以下のハードウェア機能が必要である．

- 要求したメッセージと入ってくるメッセージとのメッセージ ID を比較する比較器
- ネットワークからメッセージバッファあるいはメッセージバッファからユーザ領域へ転送する DMA コントローラ
- すでに到着したメッセージのメッセージ ID をキューイングするメッセージ ID バッファ
- 要求したメッセージとすでに到着したメッセージ（メッセージ ID バッファにある）とのメッセージ ID を比較する比較器．

表3と図8にシミュレーションの結果を示す．ここでは送信と演算とのオーバラップはさせていない．

多くの応用問題で直接メッセージ受信によりコピーのオーバヘッドは減っているが，成功率（メッセージが直接ユーザ領域に転送された割合）は応用問題の性質に依存している．例えば，LINPACK あるいは ORTHES では高い成功率でコピーのオーバヘッドが減っているのに対して，QCD あるいは TSDE では成功率は低く性能は向上していない．

受信メッセージバッファのキューの深さは，メッセージ ID バッファのハードウェア量に影響する．TSDE と AMBER 以外は，メッセージバッファにすでに到着しているメッセージ数は多くない．TSDE と AMBER の場合は，プロセッサ台数と同じ 64 であり，これは，各プロセッサがすべてのプロセッサにメッセージを送信した後メッセージを受信するからである．

3.4 受信と演算のオーバラップ

ここでは受信と演算のオーバラップを実現するアーキテクチャと MLSim による評価について述べる．

受信と演算とのオーバラップでは相互結合網からメモリへのメッセージの転送オーバヘッドはまったくなくなる．プロセッサはメッセージのヘッダが到着するまで待つが，メッセージ全体を待つ必要はない．メッセージ受信要求がメッセージ到着よりも早い場合には

メッセージは指定されたユーザ領域に直接転送される．もしメッセージがすでに到着している場合にはメッセージバッファからユーザ領域にコピーされる．この場合もこの転送と演算とがオーバラップされ，転送開始とともに演算を開始することができる．

プロセッサがまだ到着していない領域へアクセスしたときは，アクセスフォールトが発生しプロセッサはその領域が利用可能になるまでアクセスを繰り返す．

このオーバラップを実現するために必要なハードウェア機能を以下に示す．

- 要求したメッセージと入ってくるメッセージとのメッセージ ID を比較する比較器
- 二つの DMA コントローラ（ネットワークからの転送とメッセージバッファからユーザ領域へのコピー）
- ユーザ領域のアクセスチェックのためのレジスタ（トップアドレスとボトムアドレス）と比較器
- すでに到着したメッセージのメッセージ ID をキューイングするメッセージ ID バッファ
- 要求したメッセージとすでに到着したメッセージ（メッセージ ID バッファにある）とのメッセージ ID を比較する比較器．

二つの DMA コントローラは，メッセージがネットワークからメッセージバッファに転送されている間にプロセッサから転送要求がきた場合に同時に動作する．同時に動作するときは，到着していないメッセージをユーザ領域へ転送しないように，メッセージバッファからデータ転送する DMA コントローラの転送カウントが，ネットワークからデータ転送する DMA コントローラの転送カウントを越えないように制御する．

また，アクセスチェックのためのトップアドレスレジスタは，ユーザ領域へデータ転送する DMA コントローラの転送のたびに更新されるものとする．

表3と図8にシミュレーションの結果を示す．

メッセージの受信と演算のオーバラップは大幅に通信オーバヘッドを削減するのがわかる．

この結果の最も興味ある点は，受信オーバラップが通信オーバヘッドを削減するだけではなく，アイドル時間も削減していることである．理由は次のように考えられる．受信オーバラップでは，メッセージ全体の到着ではなく，メッセージのヘッダだけの到着を待ち演算を開始することにより，アイドル時間の一部であるメッセージ全体を待つ時間を短縮することになる．また，多くの場合，プロセッサAがプロセッサBにメッセージを送信した後，プロセッサAはその後プロセッサBが送信するメッセージを待つ．メッセージ受信のオーバヘッドが削減されるとプロセッサBは早くプロセッサAにメッセージを送信することができるので，プロセッサAのメッセージ待ち時間が減ることになる．

これに対して，送信のオーバラップでは送信オーバヘッドが削減されてもアイドル時間が増えてしまうので全体の性能は向上しない．

4. 関連研究

Annaratone は，5種類の分散メモリ型計算機の性能に対して通信性能と同期オーバヘッドがおよぼす影響について定量的な評価を行っている[10]．しかし，隣接プロセッサ間通信のみ使用されており，また，メッセージのベクトル化も行われていない．

Hsu は，ハイパーキューブ並列マシンの性能評価とトレースドリブンシミュレーションについて述べている[11]．評価では，16プロセッサ以下のシステムを対象としており，大規模システムのアーキテクチャを反映していない．また，これらの論文では演算と通信とのオーバラップなどの効果については評価していない．

Johnson は，通信の局所性が与える性能への影響についてモデル化し，その影響を調査している[12]．この論文が対象としている通信はキャッシュコヒーレンシによる通信であり，メッセージパッシングの通信とはかなり異なる性質を持っている．

5. おわりに

本論文では，三種類の方法，送信と演算とのオーバラップ，直接メッセージ受信，受信と演算とのオーバラップの効果について定量的に評価した．評価にはメッセージパッシングアーキテクチャの評価のために開発したメッセージレベルシミュレータを用い，10種類の応用問題を対象にした．その結果，送信と演算とのオーバラップは通信オーバヘッドを削減するが，その効果はあまり大きくなかった．メッセージをユーザ領域へ直接受信する効果は応用問題の性質に大きく依存するが，例えば256プロセッサで実行したLINPACK では 22% 性能が向上した．受信と演算とのオーバラップでは，大幅に性能が向上した．これは，オーバラップにより通信のオーバヘッドだけではなく，アイドル時間も減少したからである．例えば

256 プロセッサで実行した LINPACK では約2倍の性能向上となった.

本シミュレーションでは,転送途中にプロセッサがまだ転送されていない領域にアクセスしないものとして転送と演算のオーバラップの評価を行った.今後,実際にアクセスするかどうかも含めたシミュレーションを行う必要がある.また,評価にもとづき,演算と通信のオーバラップを実現するハードウェアの詳細な検討を行う予定である.

本論文では,通信と演算とをオーバラップさせることにより通信オーバヘッドの削減を行った.一方,メッセージ同期によるアイドル時間の削減に対しては,以下の二種類の方法が考えられる.第一の方法は,送信可能になったデータをできるだけはやく送信し,受信側のメッセージ待ち時間を減少させるものである.しかし,この方法では,細かいメッセージが大量に転送されるため,メッセージハンドリングによる多きなオーバヘッドが生じてしまう可能性がある.第二の方法は,マルチスレッドにより一つのコンテキストが受信待ちの場合でも他のコンテキストを実行させアイドル時間を軽減するものである.この第二の方法については,現在シミュレーションによりその効果を評価している.

本論文では,ハードウェアによりバッファリングのオーバヘッドの削減あるいは通信と演算のオーバラップを実現することを考えた.それに対して,Eicken らはソフトウェア(プログラミングパラダイム)による通信と演算のオーバラップの実現やメッセージのバッファリングを不要とする方法を提案している[9].ソフトウェアによる実現とハードウェアによる実現との性能あるいはコストについての比較検討は今後の課題である.

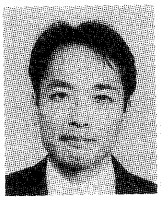
謝辞 日頃ご指導,ご助言いただく,並列処理研究センター石井センター長,白石担当部長,池坂主任研究員,佐藤主任研究員,ならびに研究室の同僚諸氏に感謝いたします.

参考文献

1) Dally, W. J., Chao, L., Hassoun, S., Horwat, W., Kaplan, J., Song, P., Totty, B. and Wills, S.: Architecture of a Message-driven Processor, *The 14th Annual International Symposium on Computer Architecture*, pp. 189-196 (May 1987).
2) Nikhil, R. S., Papadopoulos, G. M. and Arvind: *T: A Multithreaded Massively Parallel Architecture, *The 19 th Annual International Symposium on Computer Architecture*, pp. 156-167 (May 1992).
3) Sakai, S., Yamaguchi, Y., Hiraki, K. and Yuba, T.: An Architecture of a Dataflow Single Chip Processor, *The 16 th Annual International Symposium on Computer Architecture*, pp. 46-53 (May 1989).
4) Ishihata, H., Horie, T., Inano, S., Shimizu, T. and Kato, S.: An Architecture of Highly Parallel Computer AP 1000, *IEEE Pacific Rim Conf. on Communications, Computers, and Signal Processing*, pp. 13-16 (May 1991).
5) Shimizu, T., Horie, T. and Ishihata, H.: Low-latency Message Communication Support for the AP 1000, *The 19th Annual International Symposium on Computer Architecture*, pp. 288-297 (May 1992).
6) Horie, T., Ishihata, H. and Ikesaka, M.: Design and Implementation of an Interconnection Network for the AP 1000, *Information Processing 92*, Volume I, pp. 555-561 (1992).
7) Horie, T., Hayashi, K., Shimizu, T. and Ishihata, H.: Improving AP 1000 Parallel Computer Performance with Message Communication, *The 20 th Annual International Symposium on Computer Architecture* (May 1993).
8) Dally, W. J. et al.: The J-Machine: A Fine-grain Concurrent Computer, *Information Processing 89*, pp. 1147-1153 (1989).
9) Eicken, T., Culler, D. E., Goldstein, S. C. and Schauser, K. E.: Active Messages: A Mechanism for Integrated Communication and Computation, *The 19 th Annual International Symposium on Computer Architecture*, pp. 256-266 (May 1992).
10) Annaratone, M., Pommerell, C. and Ruhl, R.: Interprocessor Communication Speed and Performance in Distributed-Memory Parallel Processors, *The 16 th Annual International Symposium on Computer Architecture*, pp. 315-324 (May 1989).
11) Hsu, J. M. and Banerjee, P.: Performance Measurement and Trace Driven Simulation of Parallel CAD and Numerical Applications on a Hypercube Multicomputer, *IEEE Transactions on Parallel and Distributed Systems*, Vol. 3, No. 4, pp. 451-464 (1992).
12) Johnson, K. L.: The Impact of Communication Locality on Large-Scale Multiprocessor Performance, *The 19 th Annual International Symposium on Computer Architecture*, pp. 392-402 (May 1992).
13) Brent, R. P.: The LINPACK Benchmark on

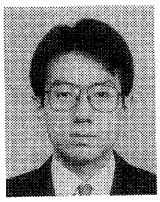
the AP 1000, *Fourth Symposium on the Frontiers of Massively Parallel Computation*, pp. 128-135 (Oct. 1992).
14) 速水 謙：SCG 法による拡散方程式の解法，コロナ社 (1989).
15) Brown, D. and Clarke, J. H. R.: Parallelization Strategies for MD Simulations on the AP 1000, *Proceedings of the Second Fujitsu-ANU CAP Workshop*, pp. L-1-L 10 (Nov. 1991).
16) Akemi, K. et al.: QCD on the Highly Parallel Computer AP 1000, *Nuclear Physics B*, Vol. 26, pp. 644-646 (1992).
17) Hatcher, P. J. and Quinn, M. J.: *Data-Parallel Programming on MIMD Computers*, The MIT Press (1991).
18) Ruehl, R.: Evaluation of Compiler Generated Parallel Programs on Three Multicomputers, *Proc. of the Sixth International Conference on Supercomputing* (June 1992).
19) Sato, H. et al.: Parallelization of AMBER Molecular Dynamics Program for the AP 1000, *Scalable High Performance Computing Conference 92*, pp. 113-120 (April 1992).
20) Gustafson, J. et al.: Slalom Update, *Supercomputing Review*, pp. 56-61 (March 1991).

(平成5年9月13日受付)
(平成6年1月13日採録)



堀江 健志（正会員）

1962年生．1984年東京大学工学部電気工学科卒業．1986年同大学院修士課程修了．同年(株)富士通研究所入社，現在に至る．並列計算機に関する研究開発に従事．1992年電子情報通信学会論文賞受賞．

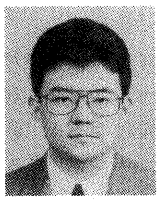


小柳 洋一

1966年生．1990年東京工業大学工学部情報工学科卒業．1992年同大学院修士課程修了．同年(株)富士通研究所入社，現在に至る．並列計算機に関する研究開発に従事．

今村 信貴（正会員）

1967年生．1990年九州大学工学部電子工学科卒業．1992年同大学院総合理工学研究科修士課程修了．同年(株)富士通研究所入社，現在に至る．並列計算機に関する研究開発に従事．



林 憲一（正会員）

1967年生．1991年東京大学工学部計数工学科卒業．同年(株)富士通研究所入社．現在並列処理研究センターにて，並列計算機アーキテクチャの研究に従事．

清水 俊幸（正会員）

1964年生．1986年東京工業大学工学部電子物理工学科卒業．1988年同大学院理工学研究科情報工学専攻修士課程修了．同年(株)富士通研究所入社，現在に至る．並列計算機に関する研究開発に従事．1992年電子情報通信学会論文賞受賞．電子情報通信学会会員．

石畑 宏明

1957年生．1980年早稲田大学理工学部電子通信学科卒業．同年(株)富士通研究所入社，現在に至る．並列計算機に関する研究開発に従事．現在，同社並列処理研究センターに勤務．元岡賞，電子情報通信学会論文賞受賞．