分散メモリ型ネットワークインタフェースの提案と評価

† 奥野 弘之, ‡ 井口寧, † 堀口 進

2001 年 8 月 2 日

IS-RR-2001-017

北陸先端科学技術大学院大学

† 情報科学研究科 ‡ 情報科学センター

〒 923-1292 石川県能美郡辰口町旭台 1-1

hirono@jaist.ac.jp, inoguchi@jaist.ac.jp, hori@jaist.ac.jp

ISSN 0918-7553

要旨

WSやPCをネットワークで接続したクラスタによる並列処理が盛んに研究されている．ギガビットレベルのネットワークの登場により，この様なクラスタでボトルネックとなっていたノード間のメッセージ通信性能が向上する一方，メッセージ通信時のノード内データ転送速度が新たなボトルネックとして挙げられるようになってきた．この問題を解決するため，本論文ではネットワークインタフェース上にメモリを搭載し，主記憶の一部とする分散メモリ型ネットワークインタフェースを提案し，その概要と予備評価について述べる．性能評価のためシミュレーションによる行列演算を行った結果，分散メモリ型ネットワークインタフェースはメッセージ通信時のバスボトルネックを削減できることが分かった．

1 はじめに

計算機の処理能力向上やネットワーク速度の向上によって，WSやPCをネットワークで結合し，クラスタ計算機を構成する研究が盛んに行われている[1][2]．しかしこの様なクラスタ計算機では，計算機間のデータの送受信を行うメッセージ通信が性能向上を阻む大きな要因となっていた．

10Base-T, 100Base-Tといった低速なネットワークでは，クラスタノード間のデータ転送に大きな時間を費やし，実行規模に見合う性能向上を得ることは不可能であった[3]．近年のMyrinetやGigabit Ethernetといった，ギガビットレベルのネットワークの登場によって，メッセージ通信におけるボトルネックの大きな要因であるクラスタノード間のデータ転送速度は飛躍的に向上したが，逆に計算機内部のバス速度がこれらネットワークやCPU処理速度の向上に追従できていないことが問題として挙げられるようになってきた[4][5]．

そこで本研究では，主記憶の一部として扱うことのできるメモリを搭載した，分散メモリ型ネットワークインタフェース（NIC）を提案する．NIC上にメモリを搭載し主記憶の一部として扱うことによって，NICがメッセージ通信のためのデータを持つことが可能となる．それによってメッセージ通信時にボトルネックとなる主記憶とNIC間のデータ転送を削減でき，クラスタ計算機の性能を改善することが期待できる．

本論文の構成は以下の通りである．2章において関連研究についての議論を行い，3章で提案する分散メモリ型NICの概要とその仕様について述べる．4章では，性能評価のために行ったシミュレーションとその結果について議論する．最後に5章で結論と今後の課題を述べる．

2 関連研究

クラスタ計算機でのメッセージ通信時におけるノード内のバスボトルネックを解消する研究としては，主に以下のような研究が行われている．

MINI[6]では，ATMネットワークインタフェースをI/OバスではなくSIMMのメモリスロットに組み込む提案がされている，このネットワークインタフェースを用いたクラスタを構築し，約1Gbitのスループットを達成したが，SIMMの衰退によって一般的なものとはならなかった．

Memory Channel[7], Memory Channel2[8]は，PCIカードのNICに，共有メモリシステムのためのアドレス変換機構とデータ送受信用のバッファとコントローラを搭載し，低レイテンシと高いバンド幅を得た．しかし，ノード間の接続に専用のスイッチが必要で，

大規模な構成を取ることができない．

MEMOnet[4] は，前述したMINIと同様にメモリスロットにネットワークインタフェースを搭載する手法を提案した．SIMMソケットだったMINIに対して，DIMMソケット利用したDIMMnet-1[9] によって，低レンテンシ，高帯域幅を達成している．一方，MINIと同様に本来メモリ用のスロットにNICを搭載しているため，メモリスロットの規格に左右されやすい，スロット数が制限されるなどの問題が残されている．

また，SCIMA[5] では，メモリの一部をCPUと同一のチップ上に搭載し，CPUとメモリ間の高速なデータ転送を実現している．CPUとメモリ間のデータ転送に注力しているためノード内部のバスボトルネックを更に削減させる方法として注目できるが，ノード間通信などのようなCPUチップ外へのデータ転送性能は十分とは言えない．

いずれの研究も計算機内のバスボトルネックを解消する手法について提案しているが，本研究はNIC上にメモリを搭載し，主記憶の一部として扱うアプローチでこの問題に取り組む．PCIバスを使用することにより高い汎用性を得られ，メッセージ通信ではNICメモリ上からデータ送受信が行われるため，主記憶からデータを読み出して送信/受信してデータを主記憶に書き込む，従来型のメッセージ通信に比べて高速なメッセージ通信が行えると考えられる．

3　分散メモリ型ネットワークインタフェース

3.1　ネットワークインタフェースの概要

本研究で提案する，分散メモリネットワークインタフェース（NIC）の概要について述べる．1にその概念図を示す．提案するNICはPCIバス上に搭載される．このNIC上にメモリを搭載し，主記憶と同じメモリアドレス空間を与える．以降，NIC上に搭載したメモリのことをNICメモリと称する．NICに搭載するメモリ容量は数MByte～数十MByteを検討しており，現在のメモリ集積技術から見れば妥当な量と考えられる．今回は8MByteとして評価を行う．このNICメモリに，メッセージ通信されるデータが主記憶から転送されて配置される．転送されたデータは再び主記憶へ戻すことも可能である．現段階では，これらの処理はソフトウェアで行われる．

NIC上のメモリは主記憶の一部として扱うので，同じデータが主記憶，NICメモリ上共に存在することは無い．つまり主記憶とNICメモリ間のデータのコヒーレンシに関する問題を考慮しなくて済む．また，主記憶とNICメモリ間のデータ転送処理が行われない限り，データはNICメモリ上に存在できるため，バッファを搭載しているNICのようにメッセージ通信のたびに主記憶からデータをコピーする必要もない．

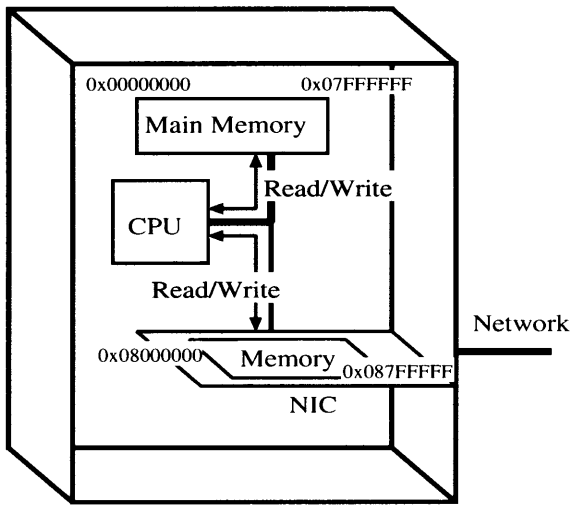

図 1: 分散メモリ型ネットワークインタフェースの概念図

3.2 データ転送プロトコル

提案したNICを実現するため，必要に応じて主記憶上からデータをNICメモリへ転送し，メッセージ通信におけるセットアップ時間を減少させる機能が必要となってくる．このための主記憶とNICメモリ間のデータ転送を行うPut, Backについて本節では述べる．Put, Back操作の概念を2に示す．

Put: 主記憶からNICメモリへの転送 Putは，主記憶上にあるデータをNICメモリへ転送する機能である．メッセージ通信処理において，参照するデータがNICメモリ上に存在しない場合，自動的に実行される．

Putの処理手順の詳細を3に示す．メッセージ通信時にデータが主記憶上にある場合（B1），処理を中断し，主記憶からデータを読み出し（B2），NICへ転送し（B3），NICメモリへデータを書き込み（B4），メッセージ通信処理を再開し，NICメモリからデータを読み出しメッセージの送信（B5），或はNICメモリへデータを書き込み受信（B5）を行う．

データがNICメモリ上にある場合（A1）は，NICメモリからデータを読み出して送信（A2），或はNICメモリへデータを書き込んで受信を行う（A2）．

Back: NICメモリから主記憶への転送 Backは上に述べたPutの逆に，NICメモリ上にあるデータを主記憶へ転送する．ノード内での演算時にデータが主記憶上に存在しない場合，Putと同様，自動的に実行される．

Backの処理手順を4に示す．NICメモリ上にデータがある場合（D1），主記憶に対

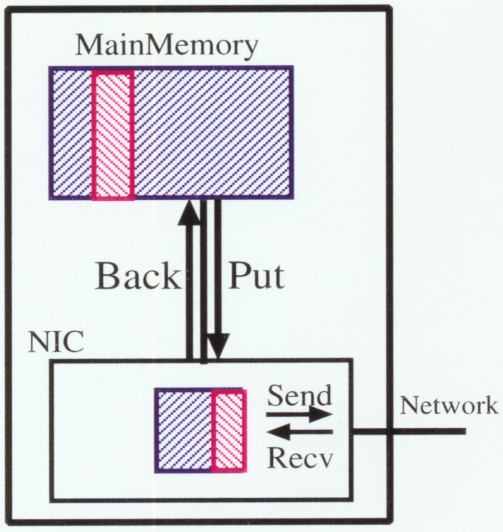

図 2: Put, Back 操作の概念図

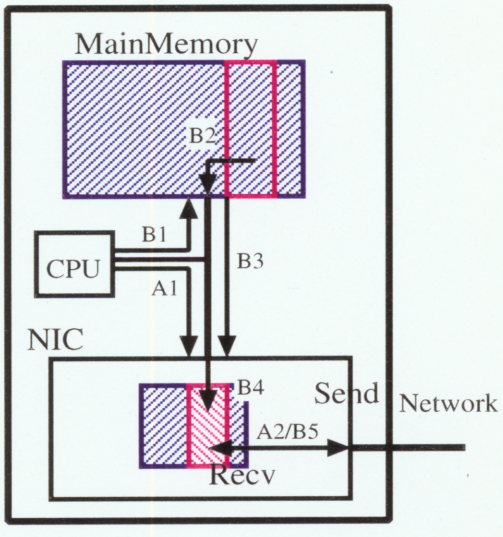

図 3: Put 操作におけるデータの流れ

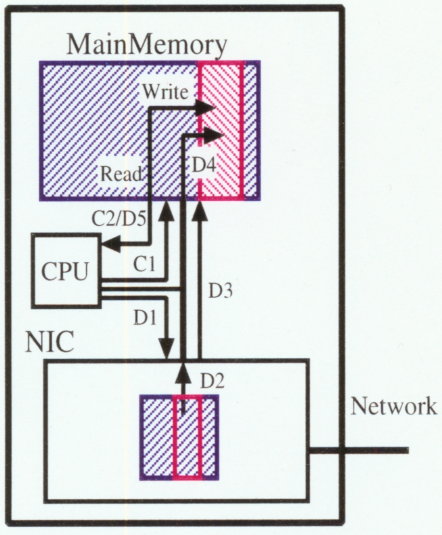

図 4: Back 操作におけるデータの流れ

する読み出し，書き込みを一旦中断して，NIC メモリからデータを読み出し（D2），主記憶へ転送し（D3），主記憶へデータを書き込み（D4），読み出し，書き込みが再開され，データの読み出し/書き込みが行われる．

主記憶上にデータが存在する場合（C1）は，通常の計算機で行われる読み出し/書き込みが行われる（C2）．

処理中で一度だけメッセージ通信されるようなデータを Put した場合に対して，この Back を使用し主記憶へ書き戻せば，無駄な NIC メモリアクセスを削減可能である．

Put, Back のいずれも，主記憶, NIC メモリ容量の不足によって実行できなくなる事が起こり得る．このような場合には，LRU などのメモリ置き換えアルゴリズムを利用して転送先のメモリ容量確保を行い，Put, Back を実行可能にすることが考えられる．但し今回は，Put, Back を実行する際の，主記憶と NIC メモリ間のデータ転送について議論するため，詳細な仕様も含めたこの点に関する議論は今後の課題とする．

このように，メッセージ通信を行うデータを NIC メモリ上に集めることによって，メッセージ通信時に NIC メモリから直接データを送信/受信し，主記憶と NIC 間のデータ転送を不要にすることができると考えられる．しかし，主記憶と NIC の，物理的に異なる場所にデータが存在するため，以下のトレードオフについて議論する必要がある．

- CPU と主記憶間のアクセスは速い

  ↔ CPU と NIC メモリ間のアクセスは遅い

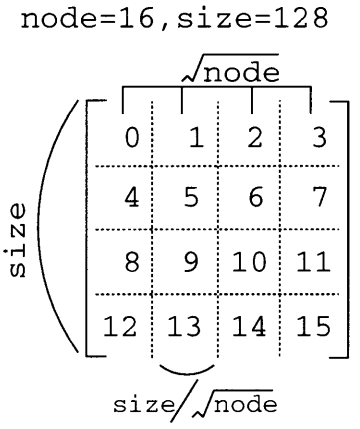

図 5: 16 ノードによるサイズ 128x128 行列演算時の計算領域割り当て

- NIC メモリからのメッセージ通信は速い

  ↔ 主記憶からのメッセージ通信は遅い

次章では，提案した分散メモリ型NIC をてシミュレーションによる性能評価実験を行い，上記のトレードオフの効果を評価，議論する．

4 シミュレーションによるNICメモリの評価

4.1 シミュレーション条件

提案した分散メモリ型NIC の性能を評価するため，サイズ 128×128 及び 256×256 の行列の乗算（$A \times B = C$）について，4, 16, 64 ノードによる実行をシミュレーションで行った．A, B, C 各行列の計算領域は 5 のように各ノードへ割り当てられる．サイズ: *size* の行列は，ノード数: *node* の領域に分割される．これより 1 ノードあたりが担当する行列の大きさは，$size/\sqrt{node} \times size/\sqrt{node}$ となる．5 の場合，ノード数 16 で行列サイズ 128×128 なので，1 ノードあたりが担当する行列の大きさは，$128/\sqrt{16} \times 128/\sqrt{16} = 32 \times 32$ となる．行列の乗算は，まずこの担当領域の乗算を行い，終了すると，上下左右のノードと担当領域の交換を行い乗算を続ける．この交換は，$\sqrt{node} - 1$ 回行われる．

シミュレーションでは，クラスタ内の 1 ノードにおける行列演算実行時のデータメモリ参照履歴を取る．主記憶，NIC メモリの読み出し/書き込み状況について，Put のみと，Back も併用した場合それぞれについて計測した．行列演算実行時のみのデータメモリ参照履歴を取っているので，演算に至るまでの，各種変数初期化時などに発生している参照履歴は計測していない．Put のみについての計測は，Back 併用時に対する比較のために

行った．またシミュレーションでは，主記憶は十分な大きさを持つものとし，主記憶からのディスクなどへのページアウトについては考慮しない．

4.2 主記憶，NIC メモリの参照状態

シミュレーションで計測する主記憶，NIC メモリの参照状態について述べる．主記憶，NIC メモリに対するアクセスは次の 4 種類に分類することができる．

- 主記憶に対するアクセス
  - 内部読み出し時にデータが主記憶上に存在する（rmh）/しない（rmf）
  - 内部書き込み時にデータが主記憶上に存在する（wmh）/しない（wmf）

  この時，データが主記憶上に存在せず，NIC メモリ上にある場合は Back が実行され，主記憶上にデータが転送された後，改めて読み出し/書き込みが行われる．Back の実行は wmf として扱い，その直後の読み出し/書き込みはそれぞれ rmh/wmh として扱うものとする．

- NI メモリに対するアクセス
  - メッセージ送信時にデータが NIC メモリ上に存在する（rnh）/しない（rnf）
  - メッセージ受信時にデータが NIC メモリ上に存在する（wnh）/しない（wnf）

  この時，データが NIC メモリ上に存在せず，主記憶上にある場合は，Put が実行され，NIC メモリ上にデータが転送された後，改めて送信/受信が行われる．Back と同様に，Put の実行は wnf として扱い，その直後の送信/受信はそれぞれ rnh/wnh として扱うものとする．

前章で挙げたトレードオフについて議論するためには，主記憶，NIC メモリがどのように参照されているかを明らかにする必要がある．

4.3 主記憶に対する参照結果

前節で定義したメモリ参照に対する結果として，行列サイズ 256×256 実行時の主記憶/NIC メモリのヒット/ミス比率を示す．

6 に，主記憶に対する参照の結果を示す．

6 より，Put のみを実行した場合は，主記憶に対する rmf（主記憶上にデータが存在しない）の比率が 50%以上という結果が得られた．実行した行列乗算は，各ノードが行列の一部のみを所持しているため，メッセージ通信で各ノードが所持している行列を送受信し

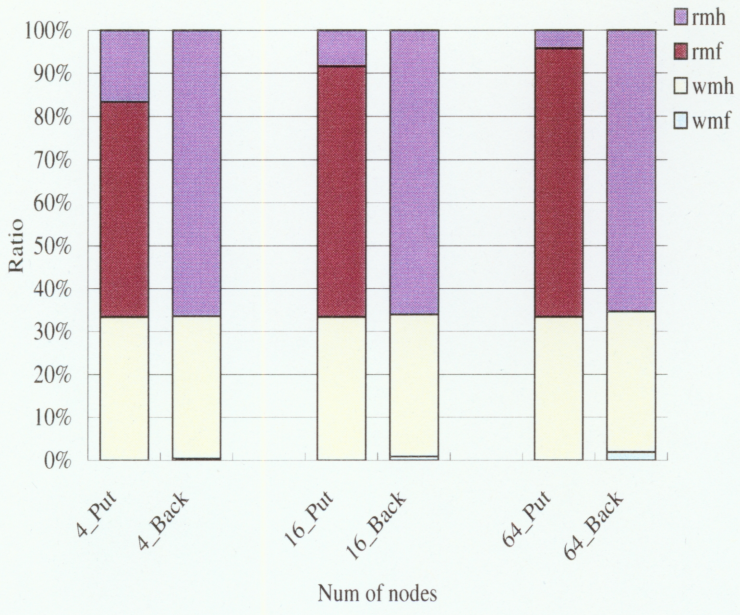

図 6: 行列乗算時の主記憶参照におけるヒット/ミス率

合う必要がある．そのため，乗算に用いる行列 A と B がメッセージ通信時 Put によって全て NIC メモリ上に移動してしまい，演算時に A と B を全て NIC メモリから読み出さなければならなくなったことが大きなミス率の原因である．このことから，Put のみを使用した場合は，アクセスの遅い CPU と NIC メモリ間のデータ参照が発生していることが分かる．また，ノード数が多いほど rmf の比率が高くなる理由は，ノード数の増加によって 1 ノードあたりが繰り返す部分行列の乗算回数が増えたためである．

一方 Back も使用した場合は，6 から分かる通り，主記憶へのアクセスのほぼ全てが rmh で占められるようになった．これは，演算時にデータが NIC メモリ上に存在している行列 A, B に対して Back が実行され，データが主記憶へ書き戻されるため，演算を実行する時点では A, B のデータ読み出しが全て rmh となるためである．

これらの結果より，Put と Back を併用することで速度の遅い CPU と NIC メモリ間のアクセスを，速度の速い CPU と主記憶間のアクセスに変え，処理速度を高められることが分かった．

4.4 NIC メモリに対する参照の結果

NIC メモリに対する参照結果を 7 に示す．Put のみでは，送受信される全てのデータが NIC メモリ上に存在するため，rnh, wnh（NIC メモリ上にデータが存在する）の比率がノード数の増加につれて大きくなる結果となった．この比率の増加は，ノード数の増加に

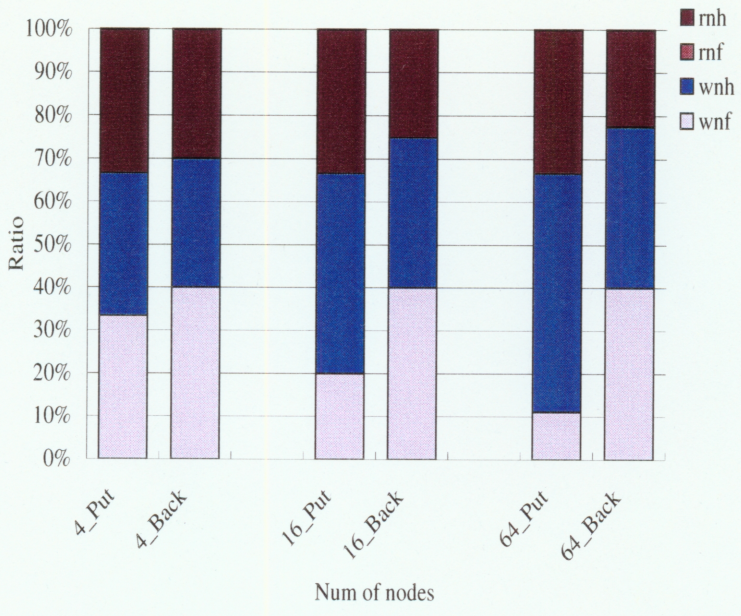

図 7: 行列乗算時の NIC メモリ参照におけるヒット/ミス率

よって 1 ノードあたりが行うメッセージ通信回数が増加するためである．

一方 Back を併用した場合は，7 より，逆に wnf が増加する結果となった．これは，各ノードが担当領域の乗算が終る度にメッセージ通信で新たな計算領域を得るため，Put と Back がメッセージ通信と行列演算のたびに行われるためである．つまり，Back されたデータが再びメッセージ通信されるため，速度の遅い主記憶からのメッセージ通信が，Back が行われた分だけ発生していることが分かる．

以上のことから，Back は，メッセージ通信とノード内部での計算を同じだけ行う処理の場合，ノード数の増加に伴って，アクセス速度の遅い wnf の比率を高めてしまう事が分かる．この点についての考察を行う．

4.5 Back の有効性に関する考察

Back を行うことによって，Put で NIC メモリへ移動してしまったノード内で読み書きするデータを，再び主記憶からアクセスすることが可能になる．しかし，前述したように，メモリリード/ライトとメッセージ通信を交互に行うような処理では，Put, Back が交互に実行され，その結果速度の遅い主記憶からのメッセージ通信や，CPU と NIC メモリ間のアクセスによる，主記憶と NI メモリ間のデータ転送がボトルネックになり，逆に性能低下を招く可能性が考えられる．この点について考察を行う．

4.5 は，Put のみ利用した結果に対して，行列サイズ 128, 256 時における Back 利用時

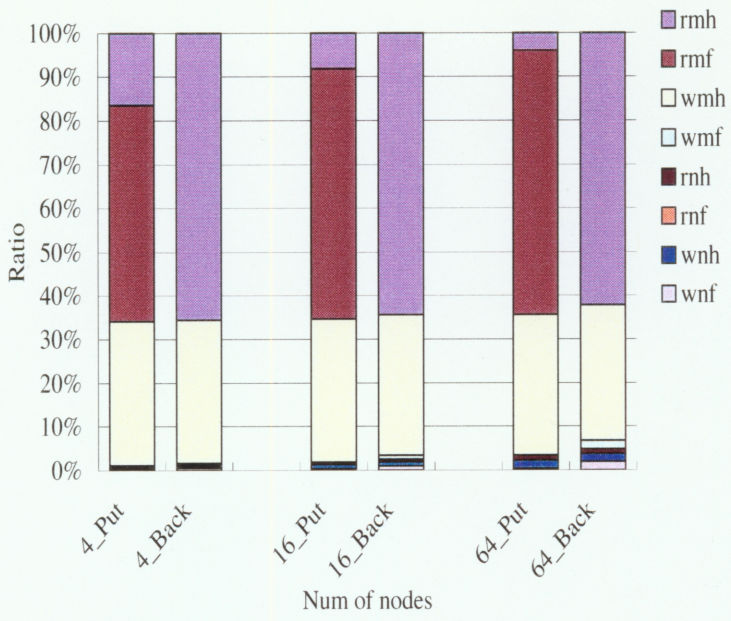

図 8: 行列乗算時の主記憶/NIC メモリ参照における合計ヒット/ミス率

の Put, Back, Read Hit の増加数を，各実行ノード数毎に計測したものを示す．また，6 及び 7 を合計したものを 8 に示す．

5 及び 4.1 節において述べた行列演算の手順より，Put や Back の元となるメッセージ通信は，ノード数によってのみ変わるので，各実行ノード数の，行列サイズ 128 から 256 への変化による Put, Back 回数の増加は無い．また，ノード数の増加に伴って rmh の増加がいずれも小さくなっているが，これは 1 ノード当たりが担当する行列のサイズが rmh を決めるためである．例えばサイズ 256 の行列において 4 ノードでは担当行列サイズは $128 \times 128 = 16384$ 要素となるが，64 ノードでは $32 \times 32 = 1024$ 要素となる．このため，ノード数が多いほど rmh の増加が少ない値となっている．

ノード数が増加するに従い，Put, Back の実行数も増加している．これによって主記憶

表 1: 行列演算における Back 実行時の Put, Back, Read Hit の増加数

ノード数	4	16	64
Put 数	+1	+5	+13
Back 数	+3	+7	+15
rmh（size 128）	+790528	+230400	+61696
rmh（size 256）	+6307840	+1839104	+492544

表 2: Put, Back 増加分によって移動したデータ量（単位:Byte）

ノード数	4	16	64
担当行列要素数	16,384	4,096	1,024
Put 分	65,536	81,920	53,248
Back 分	196,608	114,688	61,440
合計	262,144	196,608	114,688
ページ数換算	64 ページ	48 ページ	28 ページ

と NIC メモリ間で新たなデータの送受信が発生することとなる．これがどの程度の影響を及ぼすか評価する．計算するデータ型を単精度浮動小数点（float）とし，ページサイズを 4KB とすると，Put, Back の増加分によって移動されたデータ量を 2 に示す．2 より，増加分は最大でも 4 ノードの 256KByte で，64 ノードでは 112KByte しかない．

一方，乗算の計算では，行列の要素単位で数えると，$(size/\sqrt{node})^3 \times \sqrt{node}$ 回の読み出しが，行列 A, B それぞれに発生する．これは最も少ない 64 ノード時でも 2MB 分のデータが読み出されることに相当する．更に，A, B は担当領域を計算し終る度にメッセージ通信を行い，計算する部分行列 A, B の内容が変わるため，A, B は読み込み直す必要がある．その為，64 ノード時のようにキャッシュに入り切る様なデータサイズであったとしても Put, Back を行うのと同様である．また，8 から分かるように，新たな Put, Back のために発生する wnf や wmf は，これで得られる rmh の増加に対して非常に小さな割合となっている．このことからも Back による性能低下の影響はほとんど無いといえる．

以上の結果から，今回の行列演算のような，メッセージ通信し，かつノード内でも頻繁に参照されるデータが存在する場合は，Back を利用して事前にデータを主記憶へ戻す事で，性能を低下すること無く無駄な NI メモリ参照を削減することが可能である．

4.6 NIC メモリに関する考察

今回のシミュレーションでは，NIC メモリ容量を 8MByte とした．この容量の妥当性について議論する．実行した行列演算において各ノードが NIC メモリに転送するデータは，行列 A, B のそれぞれ担当する領域である．演算に用いるデータ型を単精度浮動小数点とすると，この各ノードに割り当てられる行列 A, B の担当領域に使用される合計メモリ量は，**“担当する領域の要素数 × データ型サイズ ×2（A, B の二つ）“**で求められる．これにより，ノード数，行列サイズを変化させた場合のメモリ量を示す 9 が得られる．

9 より，4 ノード構成でもサイズ 2048 × 2048 までの行列乗算が，NIC メモリ容量をオー

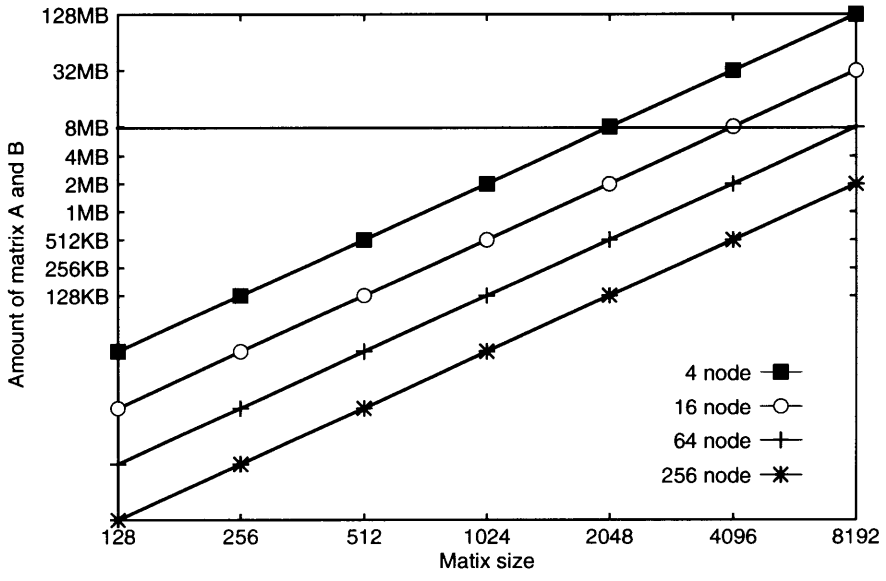

図 9: ノード数，行列サイズの変化に対する行列 A, B のメモリ使用量

バーさせること無く実行可能であることが分かる．また，256 ノード使用時では，16384 × 16384 までの規模の乗算が NIC メモリ 8MByte で計算できることが分かる．

以上から，メモリ容量 8MByte でも大規模な行列乗算を行うために十分な容量であると考えられる．

5 おわりに

本研究ではメモリ搭載型ネットワークインタフェースについて提案し，シミュレーションによる実験で評価を行った．主記憶の一部を NIC 上に搭載し，Put, Back によって主記憶と NIC メモリ間のデータ転送を行い，メッセージ通信時における主記憶と NIC 間のデータ転送を削減する手法を提案した．シミュレーションによる実験では，行列演算を行い，Put, Back をそれぞれを導入した場合における主記憶，NIC メモリの参照状態を計測した．その結果から NIC メモリの有効性について評価し，本手法がアクセス速度の遅い主記憶と NIC メモリ間のデータ転送を削減することが可能であるという結果を得た．

主記憶の一部を NIC 上という離れた場所に配置し，状況によってそれぞれのメモリにデータを転送する手法は，データ転送に関するペナルティが逆に性能の低下を招くことも懸念される点だったが，シミュレーションの結果によってその影響は大きなものではないことが分かった．

今後の課題としては，NIC メモリ容量オーバー時の性能評価や，詳細な数値による NIC メモリの性能考察，様々なアプリケーションによる評価が挙げられる．これらの結果から，

よりクラスタ計算機の性能向上を得られるNICメモリの仕様を決定することが可能であると考えられる.

参考文献

[1] http://now.cs.berkeley.edu/

[2] http://www.llnl.gov/asci/

[3] 奥野弘之，堀口進: ワークステーションクラスタにおけるネットワーク通信性能の評価，情報処理学会研究報告，HPC-68, pp.21-26(Oct. 1997)

[4] 田邊昇，山本淳二，工藤知宏: メモリスロットに搭載されるネットワークインタフェースMEMnet，情報処理学会研究報告，ARC-134, pp.73-78(Aug. 1999)

[5] 中村宏，近藤正章，大河原英喜，朴泰祐: ハイパフォーマンスコンピューティング向けアーキテクチャSCIMA, 情報処理学会論文誌，Vol.41, No.SIG5(HPS 1), pp.15-27(2000).

[6] Ron Minnich, Dan Burns and Frank Hady: The Memory Integrated Network Interface, IEEE Micro, Vol.15, No.1,(1995.2)

[7] James V.Lawton, John J. Brosnan, Morgan P. Doyle, Seosamh D. ORiordain, Timothy G. Reddin: Building a High-performance Message-passing System for MEMORY CHANNEL Clusters, Digital Technical journal Vol.8, No.2 (1996)

[8] M. Fillo, R. B. Gillett: Architecture and Implementation of MEMORY CHANNEL 2, Digital Technical Journa, Vol.9, No.1(1997)

[9] 田邊昇，山本淳二，工藤知宏: メモリスロット搭載型ネットワークインタフェースDIMMnet-1における細粒度通信機構，情報処理学会研究報告，ARC-137, pp.65-70(Mar. 2000)