ハイパフォーマンス
コンピューティング 91−21
（２００２．８．２２）

Ethernet によるクラスタ上での分散共有メモリ OpenMP Omni/SCASH の性能評価

小島好紀[†1] 佐藤三久[†2] 原田浩[†3]
石川裕[†4] 朴泰祐[†2] 高橋大介[†2]

Omni/SCASH は，ソフトウエア分散共有メモリシステム SCASH を用いたクラスタ向けの OpenMP 処理系である．本稿では，PC クラスタで多く用いられている Ethernet で接続した PC クラスタ上で Omni/SCASH の性能評価を行った．基本性能を測定した結果，ページ転送，バリア同期などのコストが Myrinet に比べて，約 10 倍の差があることが分かった．このため，通信の少ない NPB EP では良好なスケーラビリティが得られたが，演算に比べて通信の多い laplace や NPB BT では，Myrinet に比べて，低い性能となった．また，SCASH を直接用いて書いたプログラムに比べて，Omni/SCASH では，データの配置がページ境界に配置されていないことがあり，false-sharing するなど，改善の余地があることが分かった．

Performance Evaluation of Omni/SCASH Software Distributed Shared Memory System on Ethernet-based Cluster

Yoshinori OJIMA,[†1] Mitsuhisa SATO,[†2] Hiroshi HARADA,[†3]
Yutaka ISHIKAWA,[†4] Taisuke BOKU[†2]
and Daisuke TAKAHASHI[†2]

Omni/SCASH is an OpenMP implementation for SCASH software distributed shared memory system. In this paper, we report the performance of Omni/SCASH on an Ethernet-based PC cluster, which is commonly used as a parallel computing platform. The results show that on the Ethernet-based cluster the basic performance such as page transfer and barrier synchronization is about ten times lower than that on Myrinet-based one. While the performance of NPB EP, which contain small communication, is scalable, the performance of Laplace and NBP BT is limited due to the high overhead of SCASH on Ethernet. By comparing the program using SCASH library directly, we found that Omni/SCASH can be improved so that the data is allocated at page alignment to prevent false-sharing.

1. はじめに

近年，マイクロプロセッサやネットワーク技術の進歩により，ワークステーションや PC をネットワーク結合したクラスタシステムが並列プラットフォームとして主流になりつつある．クラスタシステムは構築が容易であり，また安価であるという利点がある．

クラスタは分散メモリ型システムであるため，その上でのプログラミングは MPI や PVM などのメッセージ通信ライブラリを用いて行うのが一般的であるが，その場合メッセージ通信でプログラミングしなければならないため，プログラムが複雑になり，プログラミングのコストが高い．一方共有メモリ型マルチプロセッサでは OpenMP で並列化することができる．その場合逐次プログラム自体には大きな変更を加えずに並列化でき，また段階的な並列化が可能であるため，並列化のコストを低く抑えられる．

そこで SCASH[1] や TreadMark[2] などの，分散メモリマシンの上でソフトウェアで共有アドレスを実現するソフトウェア分散共有メモリ（Software Ditributed Shared Memory: SDSM）システムが研究・開発されている．

†1 筑波大学大学院理工学研究科
Master's Program in Science and Engineering, University of Tsukuba
†2 筑波大学電子・情報工学系
Institute of Information Sciences and Electronics, University of Tsukuba
†3 株式会社コンパックコンピュータ
Compaq Computer Ltd
†4 東京大学大学院情報理工学研究科
Graduate School of Information Science and Technology, The University of Tokyo

ノード間を結合するネットワークには Myrinet が一般に用いられている．Myrinet は高いバンド幅，低い通信遅延と信頼性を兼ね備えたネットワークであり，Myrinet で構成された SDSM システムは良好なスケーラビリティが得られることが報告されている[3]．しかし Myrinet は高価であるという欠点があるため，より安価なネットワークで SDSM システムを実現することが求められる．

本稿では低速であるが安価なネットワークとして普及している 100base-TX Ethernet によって構成されたクラスタ上で OpenMP Omni/SCASH[4] の性能評価を行い，Myrinet で性能評価を行った結果と比較する．

本研究の目的は，Ethernet で構成したクラスタ上で SDSM システムを実現したときの性能を示し，Ethernet を用いることの問題点を明らかにすることである．またそれらの考察を元に，低速なネットワークであっても良いスケーラビリティが得られるプログラムを見出すことである．

以下，2 章でソフトウェア分散共有メモリ SCASH の概要を述べる．次の 3 章で Myrinet と Ethernet の基本性能の比較を行い，4 章で Omni/SCASH の性能評価を行う．最後の 5 章で結論を述べる．

2. ソフトウエア分散共有メモリシステム SCASH

2.1 SCASH

SCASH は，RWC[5] で開発されたクラスタシステム SCore 上で提供されている SDSM システムである．Myrinet 及び Ethernet 上に低通信遅延かつ高通信帯域幅を提供する高速通信ライブラリ PM[6] を用いており，ユーザレベルのライブラリとして実現されている．

共有メモリ領域の一貫性維持は，オペレーティングシステムが提供するページ単位で行われる．一貫性モデルとして ERC（Eager Release Consistency）[7] を採用し，その実装にはマルチプルライタプロトコルを用いている．さらにページ単位の一貫性維持プロトコルとして，ページの無効化を通知する無効化プロトコルと，ページデータを送信し，ページの更新を通知する更新プロトコルの双方を実装し，実行時に選択できる．

以下に SCASH が提供している機能を示す．

- 共有メモリの初期化，割り当て，開放
  共有メモリを全ノード上で，割り当て，開放を行う．SCASH は共有メモリを全ノード上で等しいメモリ空間に割り当てる．
- 同期機構
  メモリバリアとロックの 2 つの同期機構を提供している．
- 一貫性制御機構
  書き込み共有メモリデータのホームノードへの書き戻し，ホームノードからの読み込みなどの，一貫性維持機構の一部をライブラリとしてユーザに開放している．
- その他
  グローバルメモリのデータをブロードキャスト，ページ単位でのホームノードの指定等の機能を提供している．

PM では，従来のメッセージパッシングによる通信の他に，送信元のユーザ空間から，送信先のユーザ空間へ直接メモリ転送を行う，リモートメモリ通信をサポートしている．

SCASH が提供するメモリモデルでは，プログラムやデータの各セグメントは，ノード毎に独立した分散メモリ上に割り当てられる．共有メモリは専用のアロケータを通してのみ提供され，実行時に確保する必要がある．

2.2 Omni OpenMP on SCASH

SCASH システム上で動作するプログラムは，OpenMP で並列化されたプログラムを Omni/SCASH コンパイラでコンパイルすることで得ることができる．Omni/SCASH は分散メモリシステム向けの Omni[8] OpenMP コンパイラであり，OpenMP で記述されたプログラムを分散メモリ型システムで実行可能なイメージにコンパイルする．

SDSM システムでは，各ページのホームノードの割り当てがプログラムの性能に大きく影響する．SCASH では，他のノードがホームノードになっているページにアクセスすると，ページフォルトが起きる．ページフォルトを起こしたページは，リモートメモリリードによってホームノードからページのデータを自ノードに転送する．また，バリア同期実行時に，ホームノードに対してページを更新するため，更新したメモリの内容を転送する．したがって，SCASH 上で性能を得るためには，できるだけ，データをアクセスするノードとデータのホームノードを一致させ，データの転送量を低減させる必要がある．しかし OpenMP の指示文には，データを特定のメモリ領域にマッピングする機能はない．このため Omni/SCASH では，独自の拡張としてデータ配置指示文を持ち，データをアクセスするノードとデータのホームノードを一致させることを可能にしている．

また，SCASH のユーザライブラリを用いてプログラムを記述することも可能である．その場合，データの確保や各ノードへの分配，ループの配置等はプログラマが明示的に指定しなければならない．データの配置やループの配置をより柔軟に決定できるが，逐次版のプログラムを変更しなければならず，OpenMP でプログラムを並列化するよりもプログラミングのコストは高い．

3. ネットワークの基本性能

3.1 評価環境

評価環境には 8 ノードからなる PC クラスタ *COSMO*（Cluster Of Symmetric Multi prOcessor）を用いた．各ノードは 4 つの Pentium II Xeon 450 MHz の共有メモリプロセッサである．各プロセッサは 1MB の 2 次キャッシュを持ち，各ノードの主記憶容量は 2GB である．ノード間は 100base-TX Ethernet Switch 及び 800Mbps Myrinet で接続されている．

クラスタ実行環境として RWC で開発された SCore-5.0.1[5] を使用した．OS には Linux 2.4.18，コンパイラには Omni OpenMP コンパイラ 1.4a を使用し，最適化オプションとして -O4 を用いた．

3.2 PM による基本通信性能

SCore で提供されている，PM を使用したネットワークの通信性能を測るベンチマーク pmbench で通信バンド幅，ピンポン転送の遅延時間を測定した．

その結果，バンド幅は Ethernet では理論上のピーク性能である 12.5 MB/s に近い値が得られ，Myrinet では 70 MB/s 以上のバンド幅が得られた．ピンポン転送の遅延はメッセージ長 1KB 時で，Ethernet で 450 μs，Myrinet で 60 μs であった．バンド幅，通信遅延共に Ethernet の性能は Myrinet の 1/6 以下であった．

3.3 SCASH のページ転送のコスト

SCASH のページ転送のオーバヘッドを測定した．長さが 8192 の 32 bit 整数型配列，すなわちサイズが 32 KB（= 8 ページ）の 1 次元配列をブロック分割して 8 ノードに分配し，マスタスレッドが他ノードのデータへ書き込みを行うのにかかる時間を測定した．書き込む範囲を変えながら，バリア同期を行い書き込みを終了するまでにかかる時間を測定した．これはマスタスレッドからの書き込みによる scatter 操作である．結果を**図 1** に示す．グラフの x 軸の値は，マスタスレッドがインデックス 0 番から x 番目までのデータに書き込むことを示す．

書き込みの範囲を広げると他のノードがホームであるページに書き込むことになり，バリア同期で転送が行われる．そのため実行時間は階段状に増え，1 ページの転送にかかる時間は階段状になっているグラフの 1 段分の高さに相当すると考えられる．図 1 より，Myrinet での 1 ページ転送のオーバヘッドは約 0.24 ms だと考えられる．Ethernet では明確な階段状のグラフは得られず，ページ毎の転送コストは不明である．あとの実験でも分かるように，バリアのコストが 9 ms と高く，これがページ転送毎の測定のゆらぎを大きくしていると考えられる．

また，子スレッドがマスタスレッドのデータを読み，自分の持つ領域にコピーするのにかかる時間を測定した．8 × 8192 の 32 bit 整数型配列，すなわちサイズが 256 KB（= 64 ページ）の 2 次元配列をブロック分割して 8 ノードに分割し，子スレッドがマスタスレッドのデータを読む範囲を変えながら，自分の領域へのコピーを終えるのにかかる時間を測定した．これは子スレッドからの読み込みによる scatter 操作である．データコピーのみにかかる時間と，処理の最後のバリア同期までにかかる時間を測定した．この場合他ノードのデータの更新は行わないので一貫性維持通信は必要ない．したがって，2 つの時間の差はバリア同期のみのオーバヘッドである．結果を**図 2** に示す．

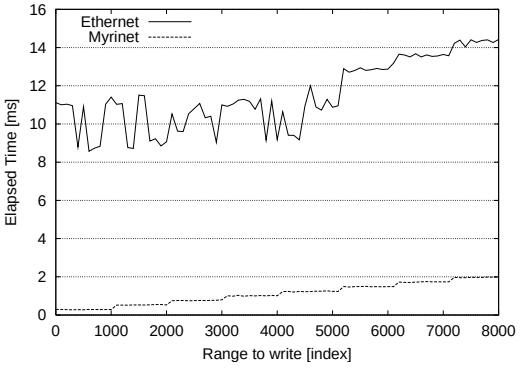

図 1 ページ転送のオーバヘッド

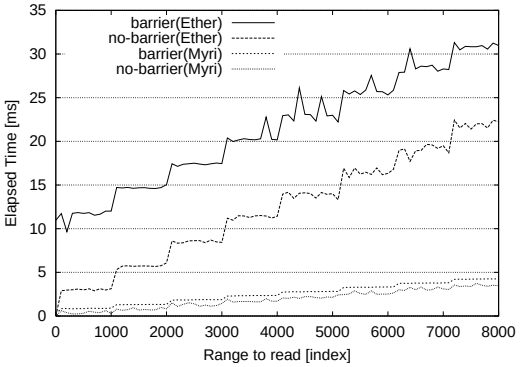

図 2 バリア同期のオーバヘッド

バリア同期のオーバヘッドは，barrier の値と no-barrier の値の差である．図 2 より，バリア同期のオーバヘッドは Myrinet では 約 0.7 ms，Ethernet では約 9 ms である．

4. 性能評価

4.1 対象プログラム

- **laplace**
  Jacobi 法による Laplace 方程式の解法．行列のサイズは 1024 × 1024，反復回数は 50 回とした．逐次版のコードを OpenMP で並列化したもの（以下 OMP-laplace）と，SCASH のユーザライブラリで直接記述したもの（以下 SCASH-laplace）の 2 つについて評価を行った．それぞれ

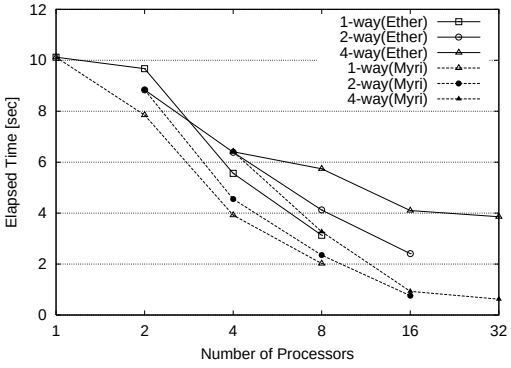

図 3 SCASH-laplace の実行時間

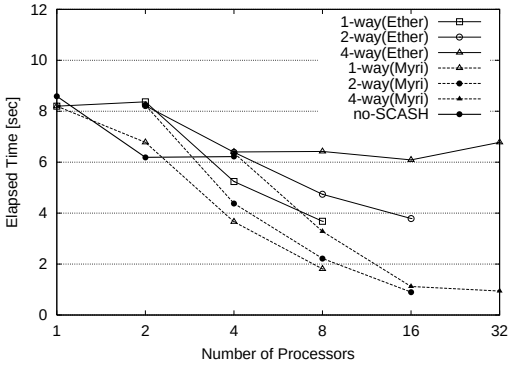

図 4 OMP-laplace の実行時間

Omni/SCASH 及び SCASH のテストプログラムとして提供されているものの C 言語版を使用した．これは Omni/SCASH による OpenMP からのオーバヘッドを比較するためである．それら 2 つはアルゴリズムに異なる部分があるため，一方に統一するよう変更を加え，評価を行った．具体的には，データの更新部分で OpenMP 版では配列間でデータコピーを行うのに対して，SCASH 版では配列の先頭アドレスの交換で済ませている．それを両方データコピーを行うように変更した．

- **NPB BT カーネル**
  NPB 2.3 の BT カーネルを長谷川らが OpenMP を用いて並列化したもの[3] を用いた．affinity スケジューリングを適用しているコードを使用した．問題のサイズは Class A を使用した．
- **NPB EP カーネル**
  NPB 2.3 の EP カーネルを OpenMP を用いて並列化したものを用いた．問題サイズは Class A を使用した．

4.2 評価結果

COSMO 上での性能評価において，実際に使用される物理的なプロセッサ数は，ノード数とノード内のプロセッサ数の積である．laplace，EP，BT ともに，ノード数を 1，2，4，8，さらに各ノード内で使用するプロセッサ数を 1, 2, 4 と変化させて計 12 通りの場合について測定を行った．

4.3 laplace

SCASH-laplace，OMP-laplace の測定結果をそれぞれ**図 3**，**図 4** に示す．図 4 において no-SCASH とは，SCASH システム上のプログラムとしてではなく，共有メモリマシン上の OpenMP プログラムとしてコンパイル・実行したものである．

両者とも Myrinet では比較的良いスケーラビリティが得られた．OMP-laplace では最大で 1 プロセッサの約 9.1 倍，SCASH-laplace では最大で約 16.3 倍の性能が得られた．一方 Ethernet では性能向上がほとんど得られず，OMP-laplace で最大約 2.2 倍，SCASH-laplace で最大約 3.2 倍の性能が得られる程度だった．

no-SCASH と SCASH を使用したものを比較すると，no-SCASH は 2 プロセッサのとき 1 プロセッサよりも性能が向上しているのに対して SCASH を使用したものはほとんど向上していない．SCASH を使用するとノード内の通信にも SCASH システムによる通信が行われ，そのオーバヘッドにより性能向上が相殺されてしまったと考えられる．

4.4 affinity スケジューリングの効果

OMP-laplace は各プロセッサへのデータの割り当ての境界と，ループの割り当ての境界が一致していない．そのため他のプロセッサが持つデータ領域に対して演算を行わなければならないことがある．そのためページフォルトが頻発し，それによりノード内通信，またはノード間通信が引き起こされ，それがオーバヘッドの一因となっていると考えられる．

Omni/SCASH は独自の拡張としてマッピング指示文を持つため，データの配置とループの配置を一致させる affinity スケジューリングを行うことが可能である．そこで，OMP-laplace を affinity スケジューリングを行うように変更し，affinity スケジューリングを行わない場合と比較した．

その結果，スケジューリングを行わないものと比較して最大で 10 %程度実行時間が短縮されたが，実行時間の傾向は変わらなかった．データの配置とループの配置の不一致がオーバヘッドの主な原因ではないと言える．

4.5 SCASH と Omni/SCASH の比較

OMP-laplace と SCASH-laplace の Ethernet 上での実行時間の比較を**図 5** に示す．同プロセッサ数で最も性能が良かったものを抜き出し，その比較を示す．

同じアルゴリズムのプログラムでありながらその性能は違ったものになった．まず 1 プロセッサのときに約 2 秒もの差があった．これは laplace で使用されるデータ構造である 2 つの配列の割り当てられ方の違いによるものと考えられる．OMP-laplace ではプログ

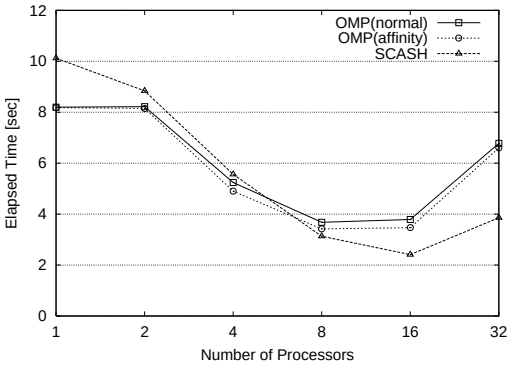

図 5 SCASH-laplace と OMP-laplace の比較

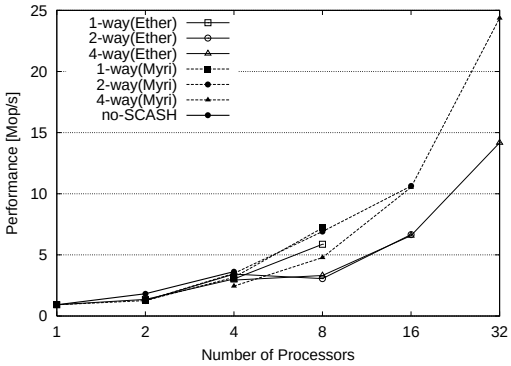

図 6 EP の性能

ラム上，静的な配列として確保しているが，SCASH-laplace は SCASH のユーザライブラリで確保している．そのためコンパイラで生成されるコードに違いが生じ， SCASH-laplace は配列アクセスのオーバヘッドが大きくなり，それが実行時間の差になっていると考えられる．

配列アクセスのオーバヘッドがあるにも関わらず，プロセッサ数が増えるに従って SCASH-laplace の方が良い結果になった．この原因を調べるため，ページフォルト回数を調べた．**表 1** にリードフォルトの回数，**表 2** にライトフォルトの回数を示す．スレッド ID 1 番のノードの回数を示す．データの初期化時のフォルト回数は除いてある．各ノード内の使用プロセッサ数を 1 に固定し，2，4，8 とノード数を変化させて測定を行った．

表 1 リードフォルトの回数

nodes	OMP	OMP(affinity)	SCASH
2	106	106	102
4	405	302	204
8	400	302	204

表 2 ライトフォルトの回数

nodes	OMP	OMP(affinity)	SCASH
2	102254	102258	104244
4	51299	51297	52404
8	25662	25677	26112

SCASH-laplace はライトフォルトが若干多いが，リードフォルトの回数は最も少なかった．リードフォルトはリモートメモリアクセスを引き起こし，オーバヘッドが大きいため，それが性能に影響していると考えられる．

フォルト回数が異なる一つの要因として両者のデータの配置の仕方の違いが考えられる．SCASH はデータを配置するとき，ページ境界に合わせて配置しているのに対し，Omni/SCASH はそれを行っていない．そのためフォールスシェアリングが生じ，性能を低下させていると考えられる．

4.6 NPB EP

EP の測定結果を**図 6** に示す．

Ethernet で測定した場合の速度向上率は，affinity スケジューリングを行った laplace で最大で約 1.9 倍であったのに対し，EP では約 15.4 倍もの性能向上が得られた．この違いは計算量に対する通信量の比率の違いによると考えられる．laplace は計算量に対する通信量の割合が比較的大きいため通信のオーバヘッドが大きく，速度向上が得られなかったと考えられる．それに対し EP はほとんどの計算はローカルで行われ，通信をほとんど必要としないため，Myrinet で最大約 26.5 倍，Ethernet でも最大約 15.4 倍と，良いスケーラビリティが得られたと考えられる．

4.7 NPB BT

BT の測定結果を**図 7** に示す．

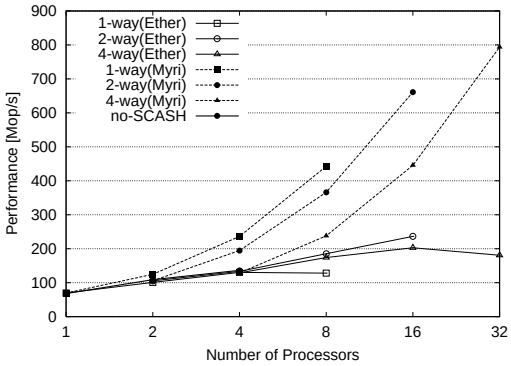

図 7 BT の性能

Myrinet では 32 プロセッサで 1 プロセッサの約 11.4 倍の性能が得られるなど良好なスケーラビリティが得られた．しかし Ethernet では 16 プロセッサで 1 プロセッサの約 3.4 倍の性能が得られる程度だった．

Myrinet では同プロセッサ数で比較した場合ノード数が多い方が性能が良く，ノード間通信が全くオーバヘッドにならないと言えるが，Ethernet では 8 プロセッサのとき 1-way 8 ノードよりも 2-way 4 ノードの方が良いなど，ノード間通信のオーバヘッドが大きいと考えられる．また 32 プロセッサでは逆に 16 プロセッサよりも性能が低下するという結果になった．

5. 結　　論

本稿では，Ethernet によるクラスタ上でソフトウェア分散共有メモリ OpenMP Omni/SCASH の評価を行い，Myrinet で評価を行った結果と比較した．

4-way SMP を 8 台接続した PC クラスタで測定した結果，laplace と NPB BT では数倍の性能向上しか得られなかったが，NPB EP では最大で約 15.4 倍の性能を得ることができた．この性能向上率の違いの原因は，演算量に対する通信量の違いによるものが大きいと考えられる．Ethernet は Myrinet と比較して通信のオーバヘッドが大きいため，通信を極力抑えることが Ethernet によって構成されたクラスタの性能を引き出すことにつながる．通信をほとんど必要としないプログラムか，または演算量と計算量のオーダが異なり，問題を大きくすることで通信を相対的に小さくできるプログラムならば Ethernet 上でも良いスケーラビリティが得られるのではないかと考えられる．

また，SCASH のライブラリで記述したものと OpenMP で並列化したものを比較した結果，SCASH のライブラリで記述した方が性能向上が大きいことが分かった．ページフォルトの回数を調べた結果，Omni/SCASH はフォールスシェアリングが生じているため速度が低下していると考えられた．そのため Omni/SCASH はページ境界に合わせた割り当てを行うなどの改善を行う余地があると言える．

本研究の今後の課題として以下のようなものがある．

- **Ethernet による SCASH の特性により適したプログラミング**
  計算量に対する通信量の比率が小さい EP では良いスケーラビリティが得られたが，他の通信が少ない，または演算量と通信量のオーダが異なるプログラムについても性能評価を行う必要がある．そのようなプログラムで実際にどの程度のスケーラビリティが得られるか評価を行い，Ethernet 上の SDSM システムに適したプログラムを見極める必要がある．
- **解析手法の確立**
  共有メモリシステムでは通信が暗黙的に行われ，プログラマには見えないため，その解析が困難である．性能向上のためにはどこがボトルネックになっているかを明らかにしなければならないが，上の理由によりそれも困難である．それを解決するためパフォーマンスカウンタやプロファイラなど解析を容易にするツールを整える必要がある．

謝辞　本研究に関する議論に参加して有益なアドバイスをして頂きました TEA グループの皆様に感謝致します．なお，本研究の一部は日本学術振興会科学研究費補助金基盤研究（A）（課題番号 14208026）による．

参 考 文 献

1) Harada, H., Tezuka, H., Hori, A., Sumimoto, S., Takahashi, T. and Ishikawa, Y.: SCASH: Software DSM using High performance network on commodity hardware and software, *Proc. ACM Eighth Workshop on Scalable Shared-memory Multiprocessors*, pp. 26–27 (1999).
2) Amza, C., Cox, A., Dwarkadas, S., Keleher, P., Lu, H., Rajamony, R., Yu, W. and Zwaenepoel, W.: Treadmarks: Shared Memory Computing on Networks of Workstations, *IEEE Computer*, Vol. 29, pp. 18–28 (1996).
3) 長谷川篤史, 佐藤三久, 石川裕, 原田浩: ソフトウエア分散共有メモリ上の OpenMP Omni/SCASH における NPB の最適化と性能評価, 情報処理学会研究報告 2001-HPC-85, pp.181–186 (2001).
4) 佐藤三久, 原田浩, 石川裕: ソフトウェア分散共有メモリシステム SCASH 上の OpenMP コンパイラ, 情報処理学会研究報告 2000-HPC-82, pp. 77–82 (2000).
5) 新情報処理開発機構: http://www.rwcp.or.jp.
6) Tezuka, H., Hori, A., Ishikawa, Y. and Sato, M.: PM: An Operating System Coordinated High Performance Communication Library, *Proc. 5th International Conference on High Performance Computing and Networking Europe (HPCN Europe 1997)*, Lecture Notes in Computer Science, Vol. 1225, Springer-Verlag, pp. 708–719 (1997).
7) Gharachorloo, K., Lenoski, D., Laudon, J., Gibbons, P., Gupta, A. and Hennessy, J.: Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors, *Proc. 17th Annual International Symposium on Computer Architecture (ISCA 1990)*, pp. 15–26 (1990).
8) Omni OpenMP Project: http://www.hpcc.jp/Omni.