計算機アーキテクチャ 114－5
（1995．10．19）

ネットワーク環境における分散共有メモリシステムの高速化と評価

薬師神 昌夫[1] 中條 拓伯[1] 藏前 健治[2] 金田 悠紀夫[1]

[1] 神戸大学 工学部 情報知能工学科
[2] 松下電工 株式会社 東京研究所

概要

我々はワークステーション・クラスタ上において，ソフトウェア制御の分散共有メモリ (DSM, Distributed Shared-Memory) の構築を行なった．アクセス遅延を減少させるため，コヒーレントキャッシュは UNIX System V の共有メモリシステムコールにより確保し，キャッシュにアクセスするユーザプロセスとコンシステンシ制御を行なうプロセスとの間において，セマフォを用いて排他制御を行なっていた．この排他制御を，ワークステーションに搭載された CPU の不可分な命令を用いたライブラリを利用して実現し，本システムの高速化を行なったので報告する．本稿では，まず本システムの構成について述べ，次に行列の乗算を実行した場合の性能評価を示し，その挙動の解析を行ない，今後の高速ネットワーク環境上における本システムの展望について述べる．

Performance improvement and evaluation of distributed shared-memory system on network environment

Masao Yakushijin[1] Hironobu Nakajo[1] Kenji Kuramae[2]
Yukio Kaneda[1]

[1] Computer and Systems Engineering, Faculty of Engineering,
Kobe University
[2] Tokyo Research Development Laboratory,
Matsushita Electric Works, Ltd.

Abstract

We describe an implementation and evaluations of Distributed Shared-Memory (DSM) system for workstation clusters. On the DSM system we implement Software-controlled coherent cache memory using shared-memory system call under UNIX System V. For the purpose of exclusive access to the shared-memory, we had utilized semaphore system call. This system call burdens heavy processing load to the system, so we could not achieve enough performance gain. Therefore, in order to acquire higher performance, we have adopted library function which uses atomic instructions of a CPU equipped with a workstation. In this paper, we describe a system configuration, results of matrix multiplication with various cache line size, and how much higher performance we have gained than the previous system. From these results, we confirm software DSM can be an expectable process communication mechanism for parallel processing on workstation clusters.

1 はじめに

近年のワークステーションの低価格化とネットワークの高速化により，LAN に接続された多数の計算機ノード群の豊富な CPU パワーやメモリを活用し，システム全体として処理能力向上を目指すワークステーション・クラスタ技術に関する研究が盛んに行なわれている．

一般にプロセス間の通信方式としては，メッセージパッシング型と共有メモリ型がある．メッセージパッシングは共有メモリを介した通信に比べ，プログラム中でデータ分割やメッセージ通信を明確に定めなければならないなど，ユーザが並列プログラムを記述しにくいという欠点がある．そこで，分散環境においても，メッセージ通信を用いて仮想的に共有メモリを実現する多数のシステムが提案されている [2][3]．ワークステーションクラスタにおいて共有メモリを介して通信を行なう場合，キャッシュメモリを用いてネットワークのアクセス遅延を減少させ，ネットワークのトラフィックを緩和することが必要となる．この時，ノード間のキャッシュコピーのコンシステンシをどのように維持するかが重要な問題となる．また，共有メモリを 1 箇所で管理する集中サーバ方式では，ノード数が増えるにしたがい，メモリサーバ自身がボトルネックとなり，台数増加による充分な性能向上 (Scalablity) が得られない．そこで，共有メモリ自身を複数のブロックに分割し，管理負荷も分散させた分散共有メモリ (Distributed Shared-Memory, **DSM**) が，Scalability の点においても有効であると考えられる．

そこで我々は，ワークステーション・クラスタ上の計算機ノードに，共有メモリとしてメインメモリの一部を確保し，各ノードにはソフトウェア制御によるコヒーレント・キャッシュ・メモリを装備したソフトウェア DSM システムを実現した．分散ディレクトリ法による書き込み無効化型の独自のコンシステンシ・プロトコルにより共有ページのコピーの管理を行なうことで，共有データへのアクセス及び管理負荷を分散させる．また，Ethernet のようなバス型のネットワークにおいては，ブロードキャスト方式による無効化などの制御メッセージの発行は有効であるが，FDDI や ATM などの今後のネットワーク環境の主流になるであろう point-to-point 型のネットワークにおいては，コンシステンシ制御時のメッセージトラフィックを低く抑える必要がある．そのために本システムでは，一つのメッセージで複数のノードにメッセージを転送できる巡回型マルチキャスト方式を考案し採用した．

本システムにおいて，アクセス遅延を緩和するコヒーレントキャッシュは UNIX System V の共有メモリシステムコールにより確保している．キャッシュにアクセスするユーザプロセスとコンシステンシ制御を行なうプロセスとの間での排他制御は，以前はセマフォを用いていたが，その排他制御の負荷が重く，充分な性能向上を得ることができなかった．そこで，計算機に搭載された CPU の不可分命令を用いたライブラリを利用してセマフォの P,V 命令を実現することにより高速化を行なった．これにより，行列の乗算などにおいて飛躍的な性能向上を達成した．

本稿では，第 2 章でソフトウェア DSM システムの構成について説明し，第 3 章で性能評価として，行列の乗算を実行した結果を示し，その挙動の解析を行なう．そして，第 4 章で実験の結果から考察を行い，現状での問題点や今後の高速ネットワーク環境における本システムの展望を示す．

2 システム構成

2.1 ソフトウェア DSM の構成

ソフトウェア DSM の概念図を図 1 に示す．図 1 において，共有メモリに対するアクセスおよび管理負荷を分散させるため，システム全体の共有メモリを **DSM** として各メインメモリ上に確保する．それぞれのノード上には自領域の **DSM** を管理するプロセス **(MemoryManager)** が存在する．

アクセスプロセス（システムを利用するユーザプロセス）は，ネットワークを介したアクセスを減らしアクセス遅延を減少させるため，キャッシュメモリを確保する．このキャッシュメモリは書き込み無効化型プロトコルを用いてコンシステンシを維持する．

本システムでは，Stanford 大の DASH [4] と同様に，分散ディレクトリ法によってフルマップ・ディレクトリを実現している．なお，ディレクトリの管理内容，キャッシュの状態については，文献 [5] で詳解している．

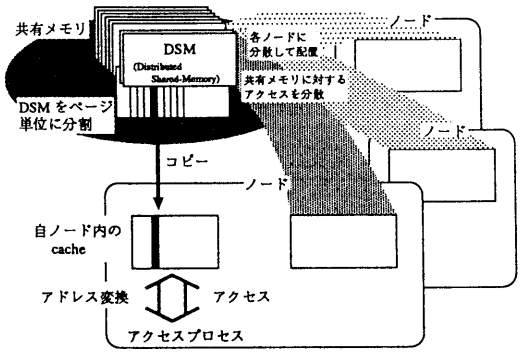

図 1: ソフトウェア DSM の概念図

MemoryManager プロセスは，自ノードの **DSM** をページ単位で管理し，ディレクトリの更新や依頼された無効化要求に対して，コピーを持つページデータの無効化を行なうなどの処理を受け持つ．**ControlProcess** は，キャッシュに対するページ転送や無効化を依頼するメッセージの作成を行なうなど，アクセスプロセスの要求に対する処理を行なう．システムを構成するプロセスの詳細については文献 [5] [6] に詳しく述べられている．

各ノード間の通信には UDP を用いた．キャッシュメモリは UNIX の共有メモリシステムコールを用いてメインメモリ上に確保し，アクセスプロセスと **ControlProcess** とがそれを共有する．アクセスプロセスと **ControlProcess** との間の排他制御には，琉球大学の新城靖氏が開発された SPARC チップの不可分命令を用いたライブラリを用いて高速化を図った．

2.2 ソフトウェア DSM へのアクセス

DSM 自体はメインメモリ上に配列として確保し，共有空間へのアクセスをキャッシュに対するアクセスに変換するインデックス変換テーブルを用いる．ユーザから共有空間へのアクセスには以下の関数を用いる．

```
int read_ss ( address, data, length )
int write_ss ( address, data, length )
    u_int address;
    u_char *data;
    int length;
```

インデックス変換テーブルは，変換情報以外にキャッシュの状態を保持し，**ControlProcess** は **read_ss, write_ss** を行なうたびにテーブルをチェックし，キャッシュが無効の場合は **DSM** からの転送要求を行なう．

3 性能評価

3.1 実験環境

本システムを，Panasonic Solbourne P2100(32MB Memory) を Ethernet(10Mbps) において通常のハブおよびスイッチングハブにより接続した環境で構築し評価を行なった．

3.2 基本アクセス時間の評価

アクセスプロセスと **ControlProcess** との間で共有するキャッシュの排他制御にセマフォを用いた以前のシステムにおいて，リードヒット，ダーティライトに要する時間は 1 アクセス 40〜50μsec であった．

SPARC の不可分命令ライブラリによる排他制御を用いた場合のアクセス時間を表 1に示す．

表 1: ヒット時のアクセス時間 (μ秒)

Page Size(Byte)	128	256	512	1024
dirty-write	15.73	15.71	15.83	15.72
read-hit	15.46	15.47	15.53	15.52

この表より，リードヒット，ダーティライトに要する時間は 1 アクセスあたり 15〜16μsec であり，約 3 倍の高速化がなされた．

また，キャッシュにヒットしなかったときのアクセス時間は，リードアクセスと無効化の伴わないライトアクセスについては，いずれもおよそ 20ms であった．あるノードがライトアクセスしようとしたとき，他のノードがそのページのコピーを持っている場合，それを無効化しなければならない．無効化するノード数によるライトアクセス時間の変化を図 2に示す．これを見ると，ノードが増えるにしたがって，階段状にアクセス時間が増加しているのが分る．本システムでは，無効化方式に巡回型マルチキャストというバケツリレーのような方式をとっており，無効化するノード数が増えるほど無効化メッセージの長さも大きくなる．このことにより，Ethernet のパケット数も増えるため，階段状にアクセス時間が増加すると考えられる．

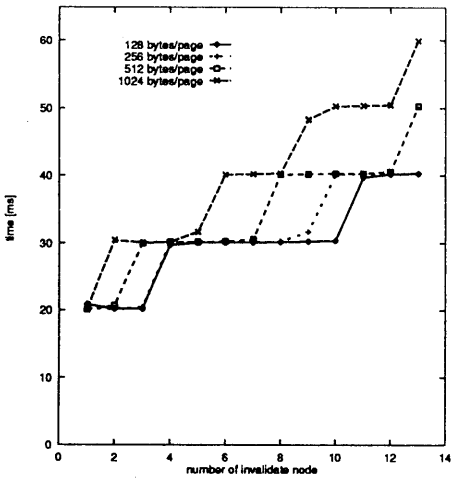

図 2: 無効化を伴うライトアクセス時間

3.3 行列演算による評価

本システム上において，100 × 100 行列および 200 × 200 行列の A, B, C を共有メモリ上に確保し，$AB = C$ の乗算を行なった．

行列の乗算は，乗算と加算を 100 次行列においてそれぞれ 100 万回，200 次行列においては 800 万回行なうものである．ひとつのノードから連続したアドレスに対するメモリアクセスが起こるので，並列環境の基本的な性能評価を行なうアプリケーションといえる．

3.3.1 処理時間と台数効果

表 2 に，セマフォを用いた以前のシステムと，SPARC チップの不可分命令によるライブラリ (atomic instruction libarary) を用いた場合の計算結果を示す．表 2 より，6 倍〜20 倍の高速化がなされたことがわかる．

表 2: 100 × 100 行列乗算の処理時間 (秒)

ノード数	1	2	4	8	15
semaphore system call	924.2	457.7	234.8	123.8	72.3
atomic instruction library (128 Byte)	58.4	34.2	21.8	15.6	12.9
(256 Byte)	47.1	25.8	15.6	10.3	8.7
(512 Byte)	41.0	21.5	12.6	8.3	6.9
(1024 Byte)	42.0	22.0	12.9	8.8	6.5

不可分命令によるライブラリを用いた場合の台数効果を図 3 にグラフとして示す．行列 A, B へのアクセスはリードアクセスのみであり，それらのデータを DSM からキャッシュに転送するときにネットワークを経由する．したがって，台数の増加にしたがい台数効果が減少している原因は，データのキャッシュへの読み込みや，演算結果を行列 C へ格納する時の通信オーバヘッドの影響が大きいからであると考えられる．

フォールスシェアリングの影響を排除するために，1 ページ内に行列要素を収めるデータ配置の最適化を行なったので，ページに対するアクセスの競合が起こらず，サイズが大きいほど台数効果が上がっている．

セマフォを用いた以前のシステム [7] では台数にほぼ比例した良好な結果が得られていたが，これはキャッシュへのアクセス遅延が大きく，通信時間の影響が少なかったためであると考えられる．

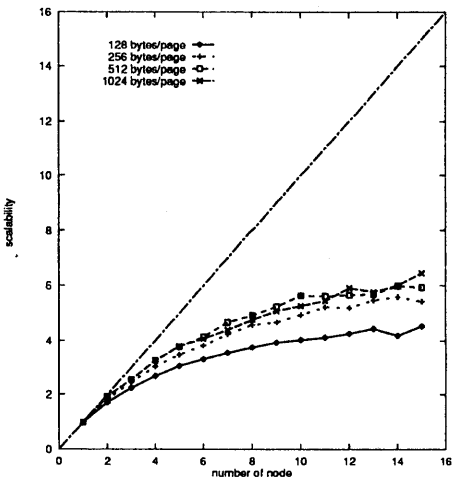

図 3: 100 × 100 の行列の乗算

図 4 には，乗算を行なうデータが格納されるページと書き込むべきページをあらかじめキャッシュにプリフェッチしておき，通信がほとんど起こらない状況での性能向上比を示す．当然のごとく，ページサイズに比例した良好な性能向上が得られている．このことから，コンパイラ等の支援を受けプリフェッチを行なうことにより，性能をリニアに向上させ得ることが予想できる．

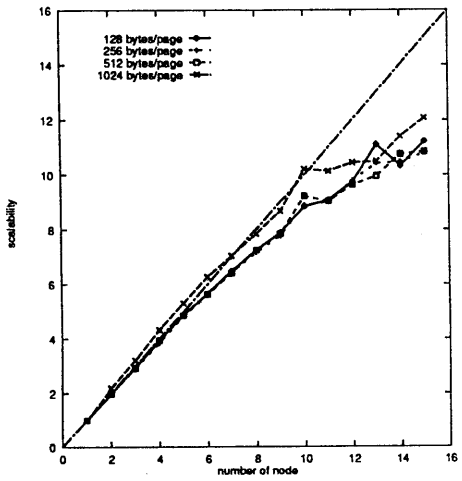

図 4: 100 × 100 の行列乗算（プリフェッチ）

表 3: 200 × 200 行列乗算の処理時間 (秒)

ノード数	1	2	4	8	15
128 Byte	335.1	179.1	105.1	66.1	51.7
256 Byte	305.8	160.5	89.2	52.6	38.7
512 Byte	282.5	145.5	76.9	42.6	29.4
1024 Byte	272.9	138.8	72.8	39.1	25.1

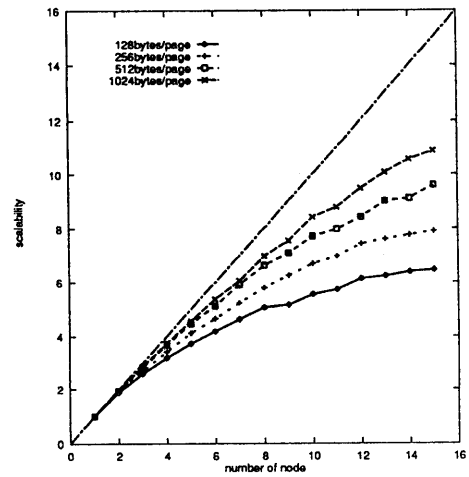

図 5: 200 × 200 の行列乗算

次に，200 × 200 行列の乗算を行なった結果を表 3と図 5 に示す．

この結果から，ページサイズが大きいほど高い性能向上比が得られているのがわかる．これは，ページサイズが大きいほど 1 回の転送で必要なデータを一度に転送することができ，また小さいほど転送回数が増えて，通信のオーバヘッドや Ethernet 上での衝突による影響が大きくなるものと考えられる．

なお，参考のために実験で使用したワークステーション 1 台の上で，100 × 100 行列と 200 × 200 行列の乗算を行なった場合の計算時間の比較を表 4 に示す．

表 4: ワークステーション単体との比較

行列サイズ	100 × 100	200 × 200
ワークステーション単体	3.0 秒	25.0 秒
atomic instruction library (15 node)	6.5 秒	25.1 秒

100 × 100 行列の計算は，ワークステーション単体の性能に及ばないが，次節に示すように現状での通信オーバヘッドが大きく，その時間を削減することにより，さらに性能を向上させることができると思われる．今後は高速ネットワークと通信プロトコルの軽減により，本システムの実用性を高めることができると考えられる．

3.3.2 実行性能の解析

図 6，7，8 に 100 × 100 行列の乗算での種々のキャッシュサイズに対する処理時間と通信時間を示す．

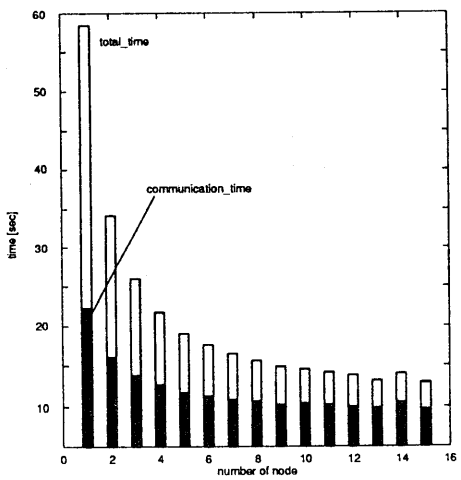

図 6: 通信時間の割合い (ページサイズ 128 Byte)

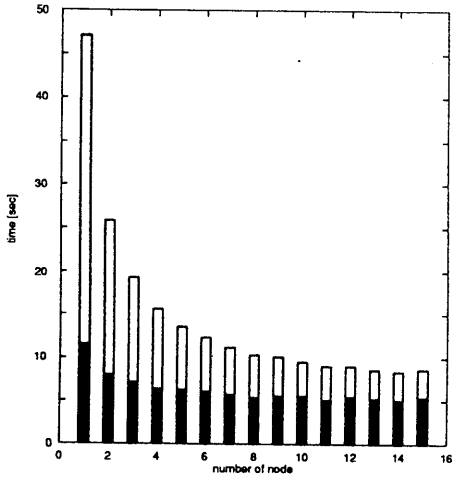

図 7: 通信時間の割合い (ページサイズ 256 Byte)

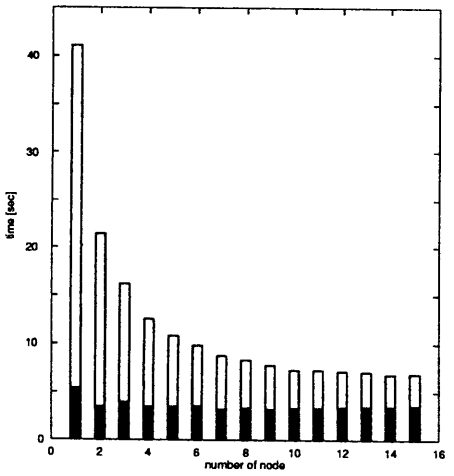

図 8: 通信時間の割合い (ページサイズ 512 Byte)

これらより，現状の Ethernet のバンド幅が問題となり，性能向上のためには通信時間の改善が必要であると考えられる．キャッシュサイズが小さいほど通信に要する時間が大きくなっているのは，ページ転送回数がその分増加し，Ethernet 上において衝突が生じているためであると考えられる．

3.3.3 スイッチングハブを用いた行列計算

サークル社製スイッチングハブ TurboSwitch2000 を用いて計算機をネットワークに接続し，100 次および 200 次正方行列の乗算を行なった．これは，ATM などの point-to-point 型の通信を用いたときの本システムのパフォーマンスを予測するためである．結果を図 9, 10 に示す．

通常のハブを用いてネットワークに接続した場合と比べて，ほぼ同じ結果が得られている．これは，行列の乗算はもともとページの競合が起こりにくいためにスイッチングハブを用いる効果が得られなかったと考えられる．ページの競合が発生しやすいアプリケーションで本システムを評価すれば，ネットワーク上での衝突などの影響を排除でき，良好な結果が得られるのではないかと思われる．

一方，スイッチングハブを用いることにより通信路の信頼性は向上した．パケットの取りこぼしが無くなり，システムの安定動作が確保された．

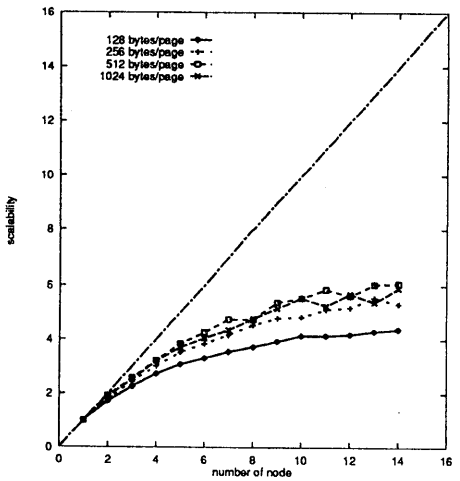

図 9: 100 次正方行列の乗算 (スイッチングハブ使用)

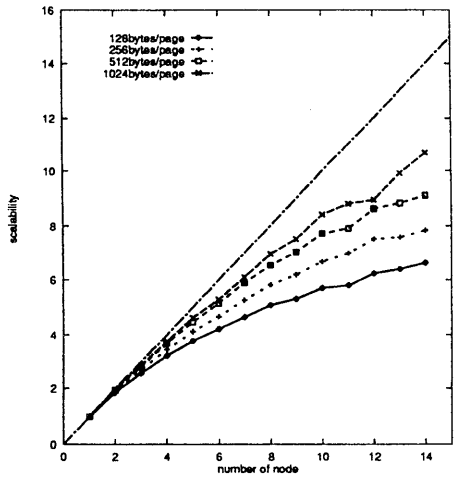

図 10: 200 次正方行列の乗算 (スイッチングハブ使用)

4 まとめ

我々は，ワークステーション・クラスタ上でソフトウェア制御のコヒーレントキャッシュを実装した分散共有メモリ（ソフトウェア DSM ）を実現した．さらに，SPARC チップの不可分命令を利用した排他制御ライブラリを導入することによって，従来のシステムを高速化することができた．

本システム上において行列計算を例に性能向上比を調べ，通信時間と処理時間を計測することによって，システムの評価を行なった．

実験結果から，100 × 100 行列の乗算を行なった結果，全体の処理時間に占める通信時間の割合いが高く，充分な速度向上を得ることができなかった．しかしながら，200 × 200 行列の乗算の場合には，計算処理の粒度が大きく，通信時間のオーバヘッドの割合いが相対的に低くなるため，ある程度の速度向上を得ることができた．

通信時間の問題に関しては，今後の ATM, FDDI, CDDI 等の高速ネットワーク技術による改善により，性能向上が達成できると考えられる．また，ワークステーションクラスタにおける並列処理のための専用ネットワークを用い，通信プロトコルを軽減することも考えられる．

巡回型マルチキャストによる無効化方式に関しては，Ethernet のようなバス型のネットワークではあまり有効ではなく，無効化にはブロードキャストやマルチキャストを用いる方が有効であると考えられる．巡回型マルチキャストは point-to-point 型のネットワークにおいては有効であると思われ，現在高速なシリアルリンクによりワークステーションを接続したワークステーションクラスタ環境において実装を進めつつある．

アプリケーションに関しては，今回は行列演算に焦点を絞って実験を行なったが，現在 SPLASH[9] および最近リリースされた SPLASH2 に含まれる並列プログラムによる評価を進めており，近々その評価結果が報告できる．

また，並列処理環境においてパフォーマンスをモニタリングすることは，性能解析において重要な要素であり，本システムにモニタシステムの実装も進めている．

謝　辞

本システムの高速化におきまして，SPARC の不可分命令を用いた排他制御ライブラリをご提供下さいました，琉球大学工学部情報工学科新城靖氏に感謝いたします．また，スイッチングハブを提供して頂いた，サークル株式会社の関谷裕行氏に感謝いたします．なお，本研究の一部は文部省科学研究費（重点領域研究（１）課題番号 04235130「超並列ハードウェア・アーキテクチャの研究」）によります．

参考文献

[1] CEAIG C. DOUGLAS, TIMOTHY G. MATTSON and MARTIN H. SCHULTZ: “PARALLEL PROGRAMMING SYSTEMS FOR WORKSTATION CLUSTERS”, Yale University Department of Computer Science Research Report YALEU/DCS/TR-975, August, 1993.

[2] Mukesh Singhal, Thomas L.Casavant: “Distributed Computing Systems”, COMPUTER, pp.12-15, August 1991.

[3] Pete Kelcher, Sandhya Dwarkadas, Alan L. Cox, and Willy Zwaenepoel: “Tread-

Marks: Distributed Shared Memory on Standard Workstations and Operating Systems", Rice COMP TR93-214, November 1993.

[4] D. Lenoski et al., : "The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor", Proc. 17th Int. Symp. Computer Architecture, IEEE CS Press, pp.148-159, 1990.

[5] 藏前健治, 中條拓伯, 前川禎男:"ネットワーク環境における分散共有メモリの実現と評価", 情報処理学会計算機アーキテクチャ研究会資料, ARC104, pp.65-72, January 1994.

[6] 藏前 健治, 中條 拓伯, 前川 禎男:"ソフトウェアDSMにおけるコヒーレント・キャッシュシステムの実装と評価", 並列処理シンポジウムJSPP '94 論文集, pp.303-310, May 1994.

[7] 藏前 健治, 中條 拓伯, 前川 禎男:"ソフトウェアDSMにおけるコヒーレント・キャッシュシステムの実装と評価", 情報処理学会論文誌 第36巻第7号, pp.1719-1728, July 1995.

[8] 吉井卓, 岩下茂信, 村上和彰:"仮想共有メモリの性能評価", 情報処理学会ハイパフォーマンスコンピューティング研究会資料, HPC55, pp.113-119, March 1995.

[9] Jaswinder Pal Singh, Wolf-Dietrich Weber and Anoop Gupta: "SPLASH: Stanford Parallel Applications for Shared-Memory", Computer Systems Laboratory, Stanford University, CA 94305.