# 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

中條拓伯† 藏前健治††
金田悠紀夫† 前川禎男†

本論文では，ワークステーション・クラスタ上においてOSのカーネルに手を加えず，ユーザレベルのソフトウェア制御のみによって，分散共有メモリ（DSM, Distributed Shared-Memory）の構築を試みた結果について報告する．分散共有メモリへのアクセスの高速化を図るためにアクセス遅延を隠蔽する方法として，ソフトウェア制御のコヒーレントキャッシュを実装する．本稿では，本システムの構成および我々が提案する，無効化に巡回型マルチキャストメッセージを用いたコンシステンシ・プロトコルについて述べ，基本的な性能評価を行うため，キャッシュへのミス/ヒットなどのアクセスタイプによる遅延時間を測定した結果を示す．さらに，実際の並列プログラムによりシステムを評価するために，行列の乗算と，SPLASHベンチマークプログラムを実行した場合の性能評価も示す．現有のイーサネットを用いたネットワーク環境では十分な性能を発揮することはできなかったが，今後の高速ネットワーク環境において，本システムの可能性について述べる．

# The Implementation and Evaluation of Software Distributed Shared-Memory (DSM) for Workstation Clusters

HIRONORI NAKAJO,† KENJI KURAMAE,†† YUKIO KANEDA † and SADAO MAEKAWA †

In this paper, we report an implementation and evaluations of Distributed Shared-Memory (DSM) for workstation clusters by using only standard UNIX user level processes. In order to hide a large amount of access latency under environment of local area network, the DSM system is equipped with a software controlled coherent cache memory. We explain a system configuration of the software DSM and consistency protocol which supports a travelling multi-cast message for invalidation. As a basic performance evaluation, we have measured access time to cache memory in the case of read miss or write hit and so on. We have also measured an each access time in executing some application programs, such as a matrix multiplication, SPLASH benchmark sets (mp3d, water) in our system. From these results, we cannot achieve high performance under the present Ethernet-based network environment. However, we confirm software DSM can be an expectable process communication mechanism for parallel processing on workstation clusters under high speed network environment in the future.

## 1. はじめに

近年，ワークステーションの低価格化とネットワークの整備により，多数の計算機ノード群の豊富なCPUパワーを活用し，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・クラスタの研究が注目されている．特に，メッセージ通信により並列処理を行うライブラリをユーザに対して提供するアプローチとして，PVM, p4, TCGMSGなどがある[1]．しかしながら，メッセージ通信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データ転送を明確に定めるなど，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．その欠点を補うために，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システムがいくつか提案されている[2)~4)]．

そこで我々は，IVYの固定分散Manager方式[3]に着目し，ワークステーション・クラスタ上における計算機ノードに共有メモリを分割して配置し，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

† 神戸大学工学部情報知能工学科
Computer and Systems Engineering, Faculty of Engineering, Kobe University
†† 松下電工株式会社東京研究所
Tokyo Research & Development Laboratory, Matsushita Electric Works, Ltd.

ュ・メモリを装備したソフトウェアDSM (Distributed Shared-Memory) システムを実現した．IVYではHomeページの概念はなく，分散したメモリはすべてキャッシュとして利用するCOMA (Cache Only Memory Architecture) 方式を取る．しかしながら，半導体メモリの低価格化・大容量化に加えて，COMA方式よりキャッシュ管理負荷が分散した，Homeページの存在する分散共有メモリ方式を採用した．

共有ページのコピーは，分散ディレクトリ法による書き込み無効化型/ライトバック方式のコンシステンシ・プロトコルにより管理を行うことで，共有データへのアクセスおよび管理負荷を分散させる．特に，バンド幅の狭いネットワークにおいて，無効化時のトラフィックを低く抑える必要があり，そのために一つのメッセージで複数のコピーを無効化できる巡回型マルチキャストによる無効化方式を考案した．

ここでは，将来の高速ネットワーク環境を睨んだ上で，並列処理におけるプロセス間通信方法としてのソフトウェアDSMの構成について説明し，コンシステンシプロトコルについて述べる．さらに基本的な性能評価に加え，SPLASHベンチマーク・プログラムを用いて評価を行い，システムの有効性・可能性を示す．また，現状での問題点，および今後の課題・方向性を検討する．

## 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，共有メモリへのアクセスおよびメモリ管理負荷を分散させるため，システム全体の共有空間をDSMとして分割し，各ノードに配置する．DSMを利用して通信を行うアクセスプロセスは，アクセス遅延を軽減するため，ページ単位でコピーされたキャッシュにアクセスを行う．

文献2)に，種々の分散共有メモリシステムの実装方式についてまとめてある．実装のレベルを分類すると以下のようになる．

- ●ハードウェアによる実現
  DASH, PLUSなど．
- ●カーネルによるサポート
  IVYなど． 仮想メモリ機構のページフォールトハンドラに手を加えて実装．
- ●ライブラリによる実現
  カーネルに手を加えずユーザライブラリの形で提供．実現が容易で移植性が高い．

そこで，我々は移植性を重視し，DSM自体はメインメモリ上に配列として確保し，ユーザライブラリの形での実装を行った．

### 2.2 ページ管理

本システムでは，Stanford大学のDASH[5]と同様に，共有メモリが分散配置されるノード内においてディレクトリを管理する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[6]．各ノードは自ノード内のDSMのみを管理すれば良いので，ノード数が増加した場合のディレクトリメモリの容量の制限は緩和される．共有空間へのアクセスにはアドレス変換テーブルを用い，アクセスプロセスに広大な共有空間を提供する．

DSMの各ページは，access (read/write), nodeset, lock (lock/unlock)の各フィールドを持つディレクト

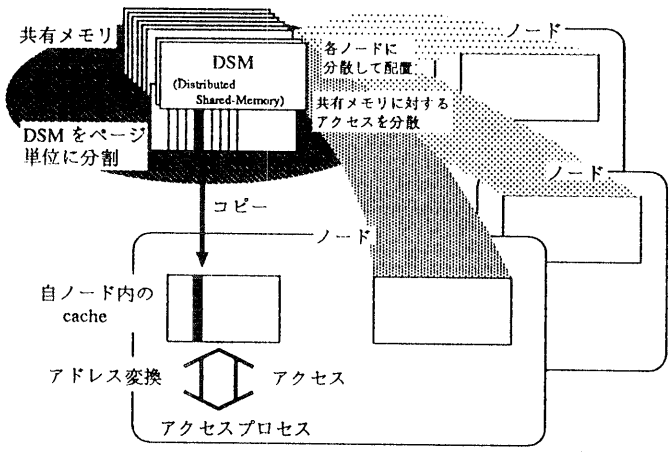



図1 ソフトウェアDSMの概念図
Fig. 1 Configuration of software DSM.

リによって管理される．accessフィールドがreadの場合，共有メモリ内のページの内容が最新であり，writeの場合は，他のノードにおいて書き込みが行われていることを示す．それぞれのページのコピーをどのノードが保持しているかをnodesetフィールドで管理する．lockフィールドは共有メモリの同一ページに同時にアクセスがあった場合に，各ノードからのアクセスの直列化を行うのに用いる．

### 2.3 システムを構成するプロセス

ノード内のシステム構成を図2に示す．ノード内にはMemory Managerと，Control Processと呼ばれる2つのプロセスが，それぞれ分散共有メモリとキャッシュメモリを管理する．Control Processは，アクセスプロセスのキャッシュに対するアクセスミス時に必要なデータ転送や，他のノードと共有しているページに対する書き込み時に，そのページを管理するノードに無効化を依頼するメッセージなどの作成を行い，その要求メッセージへの応答に対する処理を行う．

共有空間へのアクセスを実際のキャッシュメモリへのアクセスに変換するために，各アクセスはすべてアドレス変換テーブルを通り，ソフトウェア制御によるフルアソシアティブなマッピングを提供する．また，アドレス変換テーブルは，変換情報だけでなく，キャッシュ内の各ページの状態（Valid/Invalidなど）も保持する．Memory Managerは，ノード内のDSMをページ単位で管理し，各ページの状態に変更が生じたときの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う．

### 2.4 ソフトウェアDSMのユーザインタフェース

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．

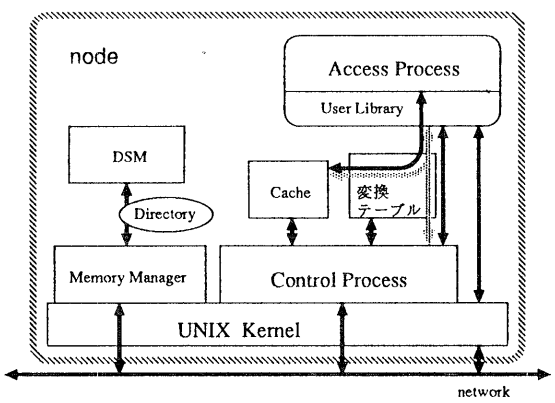



図2 ノード内のシステム構成
Fig. 2 Configuration of each node.

- int read_ss (index, c, data_size)
  共有空間へのリードアクセスを行う関数．
- int write_ss (index, c, data_size)
  共有空間へのライトアクセスを行う関数．
  引数は以下のとおりである．

```
u_int index;
  /* グローバルインデックス */
u_char * c;
  /* 読み込み/書き込みを行うデータの
     先頭ポインタ */
int data_size;
  /* データサイズ */
```

## 3. DSMにおけるコンシステンシ・プロトコル

### 3.1 アクセスの分類とシステムの動作

本システムにおいて，キャッシュ上のページの状態はValid/Invalid(有効/無効)，Clean/Dirty, Lock/Unlockに区分される．ページ状態がCleanの場合，共有メモリとページ内容が一致し，他のノードに同一ページのコピーが存在する可能性がある．したがって，Clean状態のページに対してライトアクセスを行う際，他ノードのキャッシュデータの無効化を，そのページを管理するノードのMemory Managerプロセスに要求する必要がある．Dirtyの場合，そのキャッシュ内のコピーのみが最新のデータであるので，そのままアクセスが可能である．他ノードがDirtyなページに対してリードアクスを行った場合，そのページを持つControl Processは要求されたページをリード要求元に転送すると同時に共有メモリへページを書き戻す．この時，状態は再びCleanに変わる．また，ページ単位で共有データをロックするためにLock/Unlockの状態を付加した．またリプレースは，キャッシュのコピー情報をリストにより管理することでLRU方式を実現している．

アクセス内容（リード/ライト），キャッシュのページ状態（Clean/Dirty）により，アクセスプロセスによるアクセスヒット/ミスを区分し分類すると，リードヒット，ダーティライト（Dirtyページへのライト），

表1 アクセスの分類
Table 1 Classification of accesses.

| | ヒット | | ミス |
|---|---|---|---|
| | Dirty | Clean | |
| リード | そのままアクセス | | ページデータ要求 |
| ライト | そのままアクセス | 他ノードの無効化要求 | 無効化<br>ページデータ要求 |

表 2 コンシステンシ・プロトコル
Table 2 Consistency protocol.

| | ミス／ヒットしたcacheの属性 | Control Process → Memory Manager | DSMのaccessフィールドの状態 | DSMのnodesetフィールドの状態 | Memory Manager → Control Process | Control Process → 他ノードのControl Process | 他ノードのcacheの属性 | 他ノードのControl Process → Control Process | 他ノードのControl Process → Memory Manager | 備考 |
|---|---|---|---|---|---|---|---|---|---|---|
| リードヒット | 変更なし | — | — | — | — | — | — | — | — | 通信不要 |
| ダーティライト | 変更なし | — | — | — | — | — | — | — | — | 通信不要 |
| クリーンライト | Clean→Dirty | IRM | read→write | 1つのIDのみをセット | IAM | IM | Clean→Invalid | — | — | 他のcacheを無効化 |
| リードミス | Invalid→Clean | RRM | —<br>write→read | IDを追加 | PDM<br>— | —<br>RBM | —<br>Dirty→Clean | —<br>RBPDM | —<br>DBM | read状態<br>write状態 |
| ライトミス | Invalid→Dirty | WRM<br>WRM<br>WRM, SAM | read→write<br>read→write<br>— | 1つのIDのみをセット | PDM<br>PDM<br>— | *<br>IM<br>SRM | *<br>Clean→Invalid<br>Dirty→Invalid | —<br>SPDM<br>SPDM | — | 他のcacheを無効化<br>cache間転送 |

IRM：無効化要求メッセージ　　RBPDM：リードバックページデータメッセージ
IAM：無効化応答メッセージ　　DBM：データバックメッセージ
IM：無効化メッセージ　　WRM：書き込み要求メッセージ
RRM：読み込み要求メッセージ　　SAM：シフト応答メッセージ
PDM：ページデータメッセージ　　SRM：シフトリクエストメッセージ
RBM：リードバックメッセージ　　SPDM：ソフトページデータメッセージ

クリーンライト(Cleanページへのライト)，リードミス，ライトミスの5つに分類される．**表1**に，アクセスの分類と，アクセス時に伴うシステムへの要求内容を示す．

それぞれのアクセスに対する，キャッシュおよびDSMのページ状態の変化，システムを構成する各プロセス間で送受信されるメッセージを**表2**に示す．

### 3.2 巡回型マルチキャストによる無効化

無効化型のコンシステンシ・プロトコルを用いた場合，無効化されるコピーの数は，せいぜい1～3ページに過ぎず，ブロードキャストによってすべてのノードへ無効化を発行する方法は，ネットワークに多大な負荷を与える．また，DASHプロトコル[5]では，point-to-pointで無効化メッセージを転送し，ライトアクセスを要求したノードには無効化すべきノード数を通知し，その数に等しいアクノリッジを受信した時点で無効化の完了とする．しかしながら，この無効化方式では，1回の無効化に伴う転送すべきメッセージ数は，無効化すべきコピーの数を$n$とした場合，$2\times n+1$個となり，バンド幅の限られたネットワークにおいて無効化が集中した時には，そのトラフィックは無視できないものと考えられる．

そこで，我々は，単一の無効化メッセージが，コピーを持つノード間を巡回することによって無効化を行う巡回型マルチキャスト方式を考案し採用した．すなわち，無効化を伴うライトアクセスを行ったノードのControl Processは，アクセスしたページが含まれるDSMのディレクトリを管理するノードに対して無効化要求を行う．要求を受けたノード内のMemory Managerは各ノード内のページを順に無効化するメッセージを1つ作成する．無効化メッセージはコピーを持つノードを巡回しながら無効化を行い，最後に要求元のノードへ無効化応答のメッセージとして到着する．この巡回型マルチキャストによる無効化方式は逐次的に無効化を行うため，無効化を行うノード数が多くなると無効化に要する時間が増大する半面，無効化時にMemory Managerが発行するメッセージ数は，DASHプロトコルの$n+1$個に対して1個のメッセージで済むことになり，Memory Managerの負荷を軽減できる．また，無効化時のメッセージ数も少なく，無効化が集中する場合に，ネットワークのトラフィックを緩和するという利点を持つ．

## 4. 基本アクセス時間の評価

Sun SPARCstationELC (21.0 MIPS, 33 MHz, 8 MB Memory) をEthernet (10 Mbps) により接続した環境で本システムを構築し，評価を行った．また，プロセス間通信にはUDPソケットを用いた．

3.1節で述べたアクセスの分類に基づき，ページサイズに対する各アクセス時間を計測した結果を**図3**に

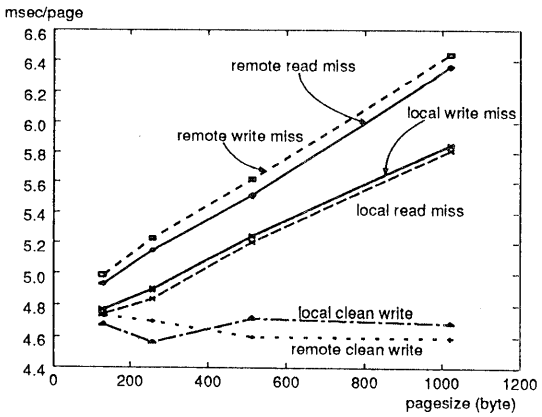



図3 アクセスの種類による遅延時間
Fig. 3 Latency of each access.

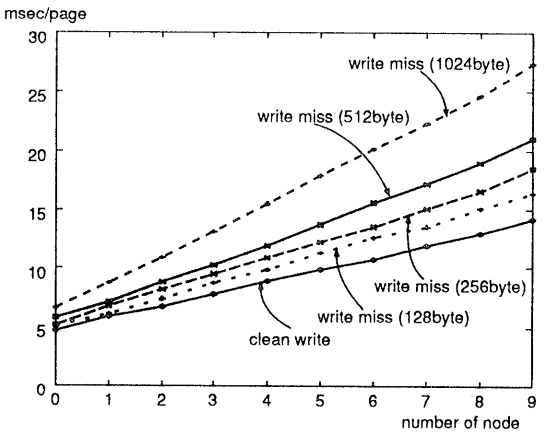



図4 無効化されるノード数に対するアクセス遅延
Fig. 4 Access latency with different number of node holding invalidated page.

示す．ここでは，DSMを管理するMemory Managerの存在する位置がアクセスプロセスと同じノード（ローカル）か，ネットワークを介した他のノード（リモート）かを区別して計測した．アクセスを行う際にControl ProcessやMemory Managerに対して要求が必要な場合（リードミス，ライトミス，クリーンライト）についても計測を行った．なお，この場合はページは共有されておらず，無効化は伴わない．

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40〜50 $\mu$ secであった．本システムのキャッシュはアクセスプロセスとControl Processで共有されるため，UNIX system callの共有メモリの形式で確保されているが，キャッシュにアクセスするにはセマフォを用いて排他制御を行わなければならず，そのシステムコールによるオーバヘッドが，キャッシュにヒットした場合のアクセス時間の高速化を妨げていると思われる．

図3より，リードミス，ライトミスの場合，アクセス時間はページサイズにほぼ比例し，ローカルのDSMに対するよりも，リモートへのアクセス時間が長いことがわかる．また，クリーンライトは，キャッシュ内にページが存在し，ネットワークを介した通信が不要なため，遅延の変化がページサイズに依存しない．

ライトアクセスの際にほかのノードのコピーページの無効化が必要な場合，無効化されるページを保持するノード数によって遅延の変化が異なる．無効化メッセージを受信するノード数に対するライトアクセスの遅延の変化を図4に示す．

本システムでは巡回型マルチキャスト・メッセージによる無効化を行うため，無効化されるページ数が増えるにしたがい，アクセス遅延は大きくなる．しかし，書き込み更新型と異なり，無効化型のプロトコルでは，無効化されるページ数はそれほど多くなく，大幅な遅延が生じる割合は少ない．また，クリーンライトの場合，ページ転送は必要なく，ページサイズにほとんど依存せず，無効化するノード数に対する遅延の増加度は低い．また，ライトミスの場合は，メッセージ中にコピーページを含むため，ページサイズが大きいほど遅延の増加度は大きくなっていることがわかる．

## 5. 並列プログラムによる評価

### 5.1 並列プログラム実行における台数効果の測定

本システムにおいて，100×100行列 $A, B, C$ を共有メモリ上に確保し，$AB=C$ の乗算を行った結果を図5に示す．台数にほぼ比例した良好な結果が得られている．これは行列 $A, B$ へのアクセスはリードアクセスのみであり，行列演算時間に対して通信時間が少ないからと考えられる．また，台数の増加にしたがい，台数効果が減少しているのは，ページサイズが大きいため，演算結果を格納する行列 $C$ へのライトアクセス時に，フォールスシェアリングによる競合が起こるためである考えられる．

SPLASH (Stanford Parallel Applications for Shared-Memory)[7]は，Stanford大で開発された共有メモリ型並列計算機のベンチマークプログラムセットである．ここでは，SPLASHの中からmp3d (3000 mol, 5-step)，およびwater (324 mol, 2-step) を実行し，評価を行った．mp3dは，モンテカルロ法を用いて大気中の粒子の状態を摸式化する流体力学シミュレーションであり，waterは立方体内の液体状態の水分子

の動きのシミュレーションをニュートン方程式によって求めるアプリケーションである．図6，7に実行結果を示す．

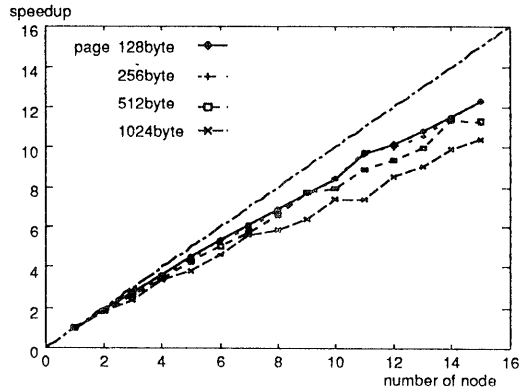



図5 100×100の行列の乗算
Fig. 5 Multiplication of 100×100 array.

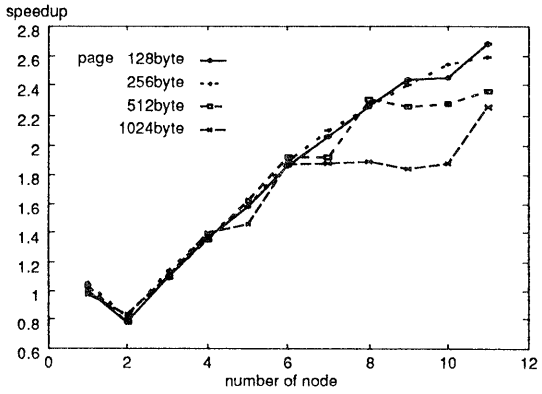



図6 mp3d (3000 mol 5-step)
Fig. 6 mp3d (3000 mol 5-step).

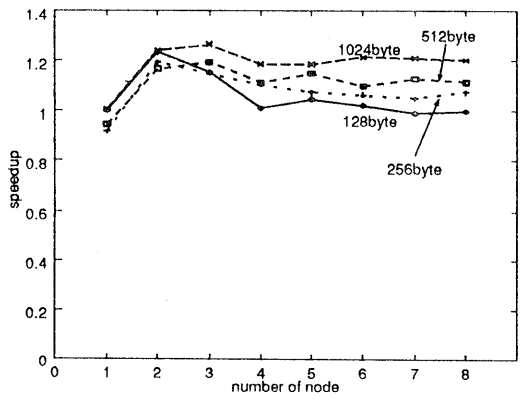


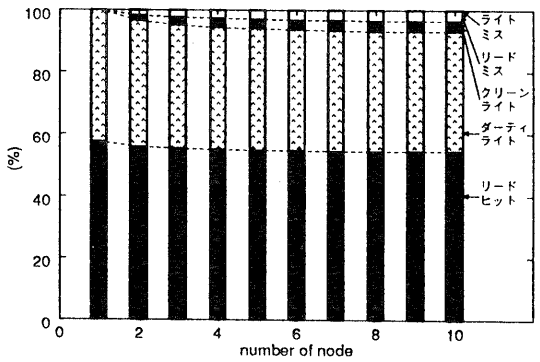

図7 water (343 mol 2-step)
Fig. 7 water (343 mol 2-step).

## 5.2 実行性能の解析

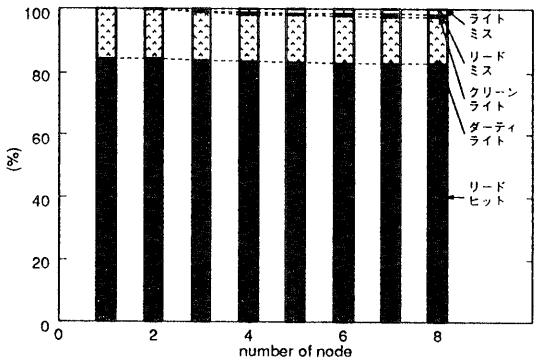
### 5.2.1 各アクセスの割合いと実行時間の内容

表1のアクセスの分類にしたがい，共有メモリに対して行われた各種アクセスの割合を図8(a),(b)に示す．図8(a)，(b)において，リードヒット，ダーティライト，クリーンライトの割合いの合計がヒット率となるが，クリーンライトに関しては，他にコピーが存在する可能性があるため，Memory Managerに無効化を依頼するのに時間を要する．mp3d, waterにおけるヒット率は充分に高く，またページサイズが大きい方がヒット率は高かった．しかし，実際には図6，7に示すように台数効果があまり得られていない．

図6，7の実行時間には，実際にアクセスプロセスがCPUを占有し処理を行っている時間，Control Processがメッセージの処理を行っている時間，通信時間等が含まれる．それぞれの内訳を図9(a)，(b)に示す．

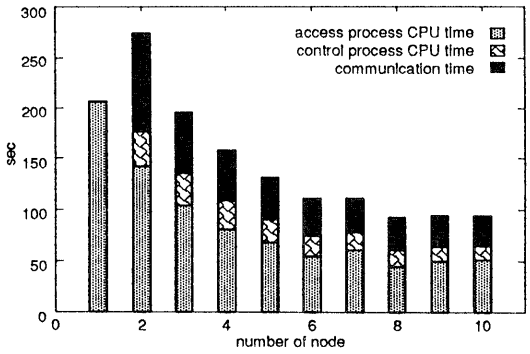



(a) mp3d, page 512 byte
(a) mp3d, page 512 byte.

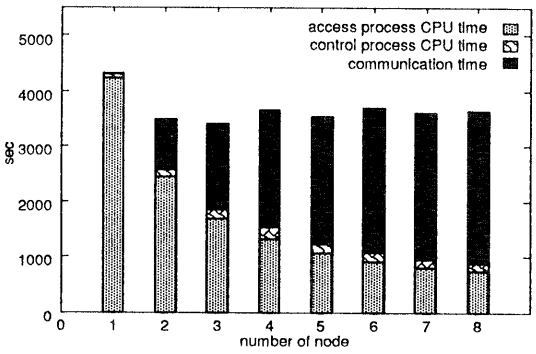



(b) water, page 512 byte
(b) water, page 512 byte.

図8 アクセスの種類の分類
Fig. 8 Classification of accesses.




(a) mp3d
(a) mp3d.




(b) water
(b) water.

図9 実行時間の内訳
Fig. 9 Processing time in detail.

図9(a)より，アクセスプロセスの処理時間は，データ分割により台数に応じて減少している．また，各ノードへの通信量も減少しているため，Control Processの処理時間も減少している．したがって，Control Processの負荷分散が行われていると言える．しかしながら，各ノードの通信時間が台数の増加に対して減少しなくなる．その原因として，Ethernetのバンド幅や，UDPに基づく通信プロトコルなどのOSのオーバヘッドがボトルネックとなっているためであると考えられる．

図9(b)では台数が増えるにしたがい，通信時間も増加している．その理由としては，waterプログラム中においてロックによる排他制御に用いる変数を含むページが，ホットスポットとなっていると考えられる．

### 5.2.2 同期の割合

実行時間に占めるバリア同期の割合を複数回計測した(スピンロックによる待ち時間を含む)結果を**表3**に示す．表3よりmp3dのプログラム実行中において，

表3 実行時間に占めるバリア同期の割合(mp3d, page 512byte)
Table 3 Percentage of barrier synchronization.

| node | max (sec) | min (sec) | average (sec) | rate (%) |
|---|---|---|---|---|
| 1 | — | — | 0.149 | 0.07 |
| 2 | 3.975 | 10.487 | 6.389 | 2.34 |
| 3 | 3.486 | 21.492 | 10.099 | 5.17 |
| 4 | 3.711 | 28.683 | 11.674 | 7.36 |
| 5 | 4.633 | 36.136 | 15.746 | 11.94 |
| 6 | 5.977 | 22.838 | 12.021 | 10.80 |
| 7 | 5.201 | 33.952 | 22.180 | 19.92 |
| 8 | 7.090 | 27.203 | 11.885 | 12.80 |
| 9 | 6.919 | 38.758 | 20.612 | 21.76 |
| 10 | 7.342 | 38.879 | 25.913 | 27.61 |

バリア同期に要する時間が大きいことがわかる．また，同じアプリケーションを実行するノード中で同期ポイントに達するまでのノード間のばらつきが大きいのは，通信時間がCPUの処理時間に比べて大きいため，ノード間におけるキャッシュのヒット率のわずかな差が大きく影響すると思われる．なお，waterについての結果については本稿では割愛したが，mp3dに比べバリア同期に要する時間が少ないものとなっていた．

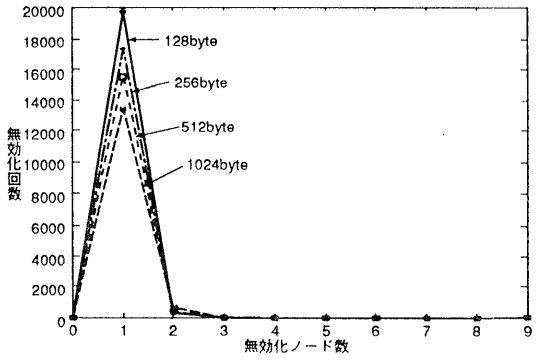
### 5.2.3 無効化とMemory Managerの負荷分散

ライトミス，クリーンライト時，にMemory Managerが無効化を行った平均ノード数の計測結果を**図10**(a)，(b)に示す．mp3dではノード間でのアクセス競合は少なく，全体の約95%が1台のノードに対する無効化であり，そのほとんどが初期値データを持つノードに対するものであると考えられる．また，バリア変数を含むページに対する無効化は比較的多かったが処理全体に及ぼす影響は少ない．

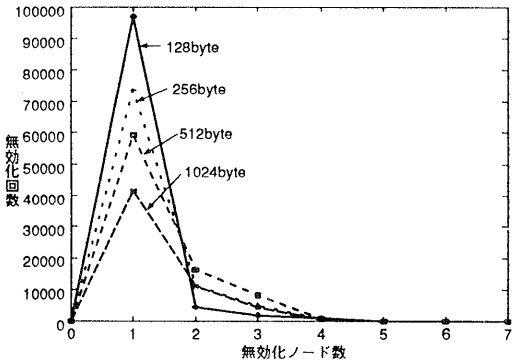
waterにおいては，ページサイズに対して共有データとして使用するデータのサイズが小さいため，mp3dに比べ多数のコピーを無効化してから書き込みを行うアクセスの割合が高いと思われる．

図10(a)，(b)の結果は，同一ページに対するアクセスの競合は少なく，ネットワーク環境においても無効化型プロトコルではコピーを保持するノード数が少なくなっていることがわかる．したがって，本システムで用いた，無効化するノードのみを選択してメッセージを送る巡回型マルチキャスト方式は有効であると考えられる．

DSMを管理するノード数に対する速度向上比を表4に示す．ただし，これらのノード上では並列アプリケーションの実行は行わず，Memory Managerプロセスのみが存在し，DSMとディレクトリの管理処理のみを行うようにした．




（a） mp3d，実行ノード数10
（a） mp3d with 10 nodes.




（b） water，実行ノード数8
（b） water with 8 nodes.

**図10** クリーンライト時における他ノードの無効化回数
Fig. 10 Count of invalidation in clean-write.

**表4** DSMを管理するノード数に対する速度向上
（8台のノードでmp3dを実行）
Table 4 Speedup for the number of DSM Managers (mp3d with 8 nodes).

| DSM node | 256 byte | 1024 byte |
|---|---|---|
| 1 | 1.000000 | 1.000000 |
| 2 | 0.998608 | 1.008524 |
| 4 | 1.001976 | 1.023995 |
| 8 | 1.016205 | 1.029715 |

（速度向上［対1］）

共有メモリへのアクセスと管理負荷の分散により，わずかではあるが速度向上が見られる．現時点では，通信コストの影響が大きく，Memory Managerの処理時間が全体に占める割合は小さいため，分散させる効果が十分に得られていない．

### 5.2.4 Control Processへの受信メッセージ数

Control Processへ到着したメッセージを種類別に分類した結果を**表5**（a），（b）に示す．

**表5** Control Processへ到着したメッセージの分類
Table 5 Classification of arrived messages to a Control Process.

（a） mp3d
（a） mp3d.

| | アクセスプロセスからの要求 | | | | | | 他ノードからのメッセージ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| node | RRM | IRM | ILRM | WRM | WLRM | ULRM | PDM | IAM | IM | RBM | SRM |
| 1 | 3 | 0 | 0 | 2 | 3 | 193 | 8 | 0 | 0 | 0 | 0 |
| 2 | 4545 | 4575 | 0 | 11 | 59 | 115 | 4663 | 4526 | 4526 | 4545 | 117 |
| 3 | 4133 | 4137 | 0 | 28 | 66 | 88 | 4296 | 4068 | 4120 | 4082 | 160 |
| 4 | 3505 | 3486 | 0 | 74 | 60 | 75 | 3719 | 3407 | 3497 | 3424 | 207 |
| 5 | 3007 | 2974 | 0 | 50 | 58 | 67 | 3191 | 2898 | 3002 | 2916 | 175 |
| 6 | 2638 | 2595 | 0 | 44 | 55 | 62 | 2817 | 2515 | 2640 | 2536 | 165 |
| 7 | 2318 | 2281 | 0 | 43 | 53 | 58 | 2495 | 2200 | 2315 | 2215 | 169 |
| 8 | 2124 | 2085 | 0 | 51 | 51 | 55 | 2313 | 1999 | 2124 | 2013 | 179 |
| 9 | 1899 | 1852 | 0 | 42 | 50 | 53 | 2071 | 1771 | 1903 | 1789 | 159 |
| 10 | 1451 | 1409 | 0 | 31 | 40 | 42 | 1589 | 1342 | 1454 | 1356 | 128 |

ILRM：ページ状態をClean/Unlock→Dirty/Lockにする要求
WLRM：ページ状態をInvalid→Dirty/Lockにする要求
ULRM：ページ状態をDirty/Lock→Dirty/Unlockにする要求

（b） water
（b） water.

| | アクセスプロセスからの要求 | | | | | | 他ノードからのメッセージ | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| node | RRM | IRM | ILRM | WRM | WLRM | ULRM | PDM | IAM | IM | RBM | SRM |
| 1 | 0 | 0 | 0 | 0 | 0 | 144093 | 0 | 0 | 0 | 0 | 0 |
| 2 | 4097 | 3902 | 0 | 2 | 21948 | 72059 | 26047 | 3902 | 3902 | 4097 | 21950 |
| 3 | 15153 | 14839 | 0 | 2 | 15327 | 48048 | 30482 | 14839 | 14958 | 14970 | 15328 |
| 4 | 18054 | 17341 | 0 | 1 | 12357 | 36042 | 30413 | 17341 | 17859 | 17440 | 12358 |
| 5 | 18755 | 15387 | 0 | 1 | 9728 | 28839 | 28484 | 15387 | 18560 | 15465 | 9728 |
| 6 | 18229 | 13605 | 0 | 1 | 8211 | 24086 | 26441 | 13605 | 18041 | 13678 | 8211 |
| 7 | 16374 | 11931 | 0 | 1 | 7138 | 20606 | 23512 | 11931 | 16177 | 11987 | 7138 |
| 8 | 15097 | 10540 | 0 | 1 | 6264 | 18034 | 21362 | 10540 | 14901 | 10589 | 6264 |

表より，リードリクエスト（RRM）とほぼ同数の無効化要求（IRM）があり，要求ページ（PDM）とほぼ同数の無効化（IM）が行われている．このことから，リードミスで一度読み込んだページに対しては，書き込み要求を行っている場合が多いと考えられる．

また，mp3dでは，アクセスプロセスが，書き込み時にページをロック/アンロック（WLRM, ULRM）する場合の多くはバリア同期処理であり，ページサイズ，台数に関係なくメッセージ数にほとんど変化はない．表5（b）では，アクセスプロセスからControl Processへのロック/アンロック要求（WLRM, ULRM）の割合がリードリクエスト（RRM）や無効化要求（IRM）に比べ非常に高く，dirtyページへのリード要求（SRM）も多くなっている．このことから，同一ページに対する排他的な書き込みの競合が起こっており，キャッシュ間においてページのピンポン現象が生

じていると考えられる．

## 6. まとめ

本研究では，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，実際の並列プログラムによる性能評価を行った．

実験結果から，行列乗算では，ある程度の速度向上が確認されたが，mp3d, waterでは十分な台数効果は得られなかった．その原因としては，ネットワーク自体のバンド幅や通信速度の遅さが考えられるが，今後のATM, FDDI, CDDI等の高速ネットワーク技術によって，この問題が改善されることにより，より一層の性能向上が期待される．

現在の問題点としては，管理する単位が比較的サイズの大きいページであることから，共有アドレス空間へのデータのマッピングによって，フォールスシェアリングや，同期ポイントに至るまでの処理時間のばらつきが大きくなるなど，性能向上に大きな影響を与えるということが考えられる．これらの問題に関しては，コンパイラによる支援が不可欠であり，並列化コンパイラの開発とともにその点を改善していきたい．

また，ヒット時のアクセス時間については，制御プロセスを軽量プロセスによるマルチスレッドで実現することにより，コンテキスト切り替えやシステムコールによるオーバヘッドを軽減することによって，高速化を図ることができる．今回は，巡回型マルチキャストによる無効化方式に関して，定量的な性能評価を行うことができなかったが，今後はネットワークの負荷を測定することによって，その有効性を示して行く予定である．

さらに，SPLASHに含まれるmp3d, water以外の並列プログラムによる評価を行い，高速ネットワーク環境において，ワークステーション・クラスタ上に構築されたソフトウェアDSMの有効性や，処理能力向上のためのシステムの改善点・問題点などを示していきたいと考えている．

**謝辞** 本システムの評価におきまして，有益なご助言をいただいた筑波大学電子情報工学系和田耕一助教授に感謝いたします．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」）による．

## 参考文献

1) Douglas, C. C. et al.: *Parallel Programming Systems for Workstation Clusters*, Yale University Department of Computer Science Research Report, YALEU/DCS/TR-975 (1993).
2) Singhal, M. and Casavant, T. L.: Distributed Computing Systems, *Computer*, Vol. 24, No. 8, pp. 12-15 (1991).
3) Li, K. and Hudak, P.: Memory Coherency in Shared Virtual Memory Systems, *ACM Trans. Comput. Syst.*, Vol. 7, No. 4, pp. 321-359 (1989).
4) Keleher, P. et al.: *TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems*, Rice COMP TR93-214 (1993).
5) Lenoski, D. et al.: The Directory-Based Cache Coherence Protocol for the DASH Multiprocessor, *Proc. 17th Int. Symp. Computer Architecture*, pp. 148-159, IEEE CS Press (1990).
6) 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，ARC 104, pp. 65-72 (1994).
7) Singh, J. P. et al.: SPLASH: Stanford Parallel Applications for Shared-Memory, *Tech. Report Computer Systems Laboratory*, Stanford University (1992).

（平成6年9月16日受付）
（平成7年2月10日採録）

**中條 拓伯**（正会員）

昭和36年生．昭和60年神戸大学工学部電気工学科卒業．昭和62年同大学大学院工学研究科電子工学修士課程修了．平成元年神戸大学工学部情報知能工学科助手．計算機アーキテクチャ，並列処理，並列入出力システムの研究に従事．電子情報通信学会，人工知能学会，システム制御情報学会各会員．

**藏前 健治**

昭和44年生．平成4年神戸大学工学部システム工学科卒業．平成6年同大学大学院工学研究科システム工学専攻修士課程修了．同年松下電工(株)入社．現在，同東京研究所にてネットワーク管理ソフトウェアの研究に従事．ATM管理システムなどに興味を持つ．

**金田悠紀夫**（正会員）

昭和15年生．昭和39年神戸大学工学部電気工学科卒業．昭和41年同大学大学院電気工学専攻修士課程修了．昭和41年電気試験所（現電総研）入所．電子計算機研究に従事．昭和51年神戸大学工学部システム工学科，現情報知能工学科教授．工学博士．コンピュータシステムのハードウェア，ソフトウェアの研究に従事．高級言語マシン，並列マシン，AIに興味を持つ．

**前川　禎男**（正会員）

昭和6年生．昭和29年大阪大学工学部通信工学科卒業．昭和34年同大学院工学研究科博士課程単位取得退学．大阪大学助手．昭和36年神戸大学工学部電気工学科助教授．昭和47年神戸大学工学部電子工学科教授を経て，現在，関西学院大学理学部物理学科教授．工学博士．システム理論，人工知能，自律移動ロボットなどの研究に従事．人工知能学会，電子情報通信学会，電気学会，システム制御情報学会などの会員．