システムソフトウェアと
オペレーティング・システム 65−19
（１９９４．７．２２）

Machの外部ページャによる分散共有メモリサーバの評価

斎藤 彰一† 中村 素典† 大久保 英嗣† 大野 豊† 白川 洋充††

†立命館大学理工学部情報学科

††近畿大学理工学部経営工学科

我々は，ネットワークワイドに唯一つの仮想記憶空間をユーザに提供する分散仮想記憶システムの開発を行っている．分散仮想記憶システムによって，位置透過性を始めとした各種の透過性を容易に実現することができる．本論文では，分散仮想記憶システムの基本部分の１つである分散共有メモリサーバの処理方式について述べる．特に，そのプロトタイプとして実現したMachオペレーティングシステム上の分散共有メモリサーバの評価について述べる．

An Evaluation of Distributed Shared Memory Server by Mach External Pager

Shoichi Saito† Motonori Nakamura† Eiji Okubo† Yutaka Ohno†
Hiromitsu Shirakawa††

†Department of Computer Science,
Faculty of Science and Engineering, Ritsumeikan University
1916 Noji, Kusatsu, Shiga 525, Japan

††Department of Industrial Engineering,
Faculty of Science and Engineering, Kinki University
3-4-1 Kowakae, Higashi-Osaka 577, Japan

We have been developing the Distributed Virtual Memory System(DVMS) which provides users with single network-wide virtual memory. It becomes easy to implement various transparency such as location transparency. In this paper, we describe the Distributed Shared Memory Server(DSM server) which is one of basic parts of DVMS. In particular, an evaluation of a prototype system of DSM server on Mach operating system is described.

1 はじめに

我々は, ネットワークワイドな共有メモリをユーザに提供する分散共有メモリサーバ (Distributed Shared Memory Server, 以下 DSM サーバ) [1][2] を Mach オペレーティングシステム [3] 上に構築している. 本 DSM サーバは, 我々が現在開発中である分散仮想記憶システムの基本サーバとして位置付けられる.

DSM サーバは, 各マシンの主記憶を分散共有メモリのキャッシュとして使用し, そのキャッシュに分散共有メモリのコピーの全体あるいは一部を配置する. さらに, コピーに対して一貫性制御を行い, 各マシン間で共有メモリの内容の同一性を保証している. 一貫性制御には, write-invalidate 方式を採用している. write-invalidate 方式では, 読み出しは同時に複数のマシンから可能であり, 書き込みは同時には唯一つのマシンでのみ可能である.

DSM サーバでは, 仮想記憶のページ単位に 7 つのアクセス状態を設定し, このアクセス状態に基づいて, ページイン/ページアウト処理, ページのコピーや無効化の処理を行っている. また, 複数のマシンからのアクセス要求を, ページオーナと呼ばれる管理マシンに集中させることで, 一貫性制御や排他制御を実現している.

以下, 本論文では, 2 章で DSM サーバの全体構成について述べ, 3 章で分散共有メモリマネージャによる一貫性制御及び排他制御の詳細について述べる. 4 章でページングマネージャによるページング用 2 次記憶の管理方式について述べる. 最後に, 5 章では DSM サーバの評価結果を示す.

2 分散共有メモリサーバ

分散共有メモリを実現するために, 我々はオペレーティングシステムとして Mach を採用した. Mach は, 従来のオペレーティングシステムではカーネルレベルでしか行えなかった仮想記憶管理を, ユーザレベルで行なえる機能を提供している. これは**外部ページャ** (External Pager)[4][5] と呼ばれている. Mach カーネルと外部ページャとの間のインタフェースを**外部メモリ管理インタフェース**という. これは, 外部ページャがページイン/ページアウト処理を行うために使用するインタフェースであり, MIG(Mach Interface Generator) によって実装されている.

DSM サーバは, 各マシンに配置され互いに協調しながら, マシン間でのメモリオブジェクトの共有を実現している. また, 各マシンの 2 次記憶の容量を超えるような処理も可能としている.

DSM サーバは外部ページャを利用した**分散共有メモリマネージャ(MM)**, 共有メモリオブジェクトの管理を行う**共有オブジェクトマネージャ(OM)**, ページング用 2 次記憶 (ページングファイルと呼ぶ) の管理を行う**ページングマネージャ(PM)** の 3 つのタスクで構成されている (図 1参照). 以下, それぞれについて説明する.

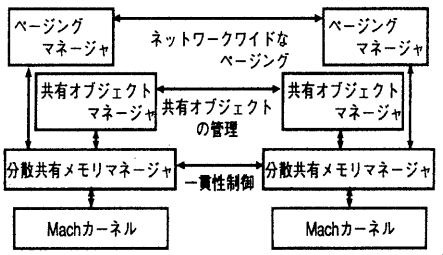

図 1 分散共有メモリサーバの構成

・分散共有メモリマネージャ(MM)

外部メモリ管理インタフェースを備えた外部ページャである. 主記憶上でページフォルトが発生した場合, カーネルからのメッセージに従ってメモリのページインとページアウトを行う. また, 他のサイトのメモリマネージャとの間で, メモリオブジェクトの一貫性制御を行う.

・共有オブジェクトマネージャ(OM)

共有メモリオブジェクトの管理を行う. 本来メモリオブジェクトはマシンローカルなものである. 本マネージャでは, これを各サイト間で協調しながら管理を行ない, ネットワークワイドな共有メモリオブジェクトとして扱うことを可能にしている.

・ページングマネージャ(PM)

ページング処理とそのための 2 次記憶の管理を行なう. 本マネージャでは, 各々のサイトが提供する 2 次記憶の領域をネットワーク上のすべてのサイトで共有している. 従来のページング用 2 次記憶はマシンローカルな存在であった. このために, 十分な 2 次記憶を確保できないマシンでは, 大きな仮想空間を利用するような処理を行えなかった. しかし, マネージャによって, ページングファイルをすべてのサイトで共有することにより, 各マシンの 2 次記憶よりも大きな仮想空間を必要とする処理も行うことが可能となっている.

3 分散共有メモリマネージャ

本マネージャは, 主記憶上でページフォルトが発生した場合, カーネルからのメッセージに従ってメモリのページインとページアウト処理を行う. また, 他のサイトのマネージャと協調して, メモリオブジェクトの一貫性制御を行う.

3.1 共有ページとアクセス状態

DSM サーバでは, 共有メモリの基本単位として仮想記憶のページを用いている. 同時に一貫性制御の最小単位もページとなる (我々が実装を行った マシンでは 1 ページは 4KB である). これを**共有メモリページ**という.

共有メモリページの状態は**アクセス状態**によって表される (図2参照). アクセス状態には, 複数のマシンが読み出し可能な **READ**, 唯 1 つのマシンが書き込み可能な **WRITE**, 排他制御機構によりページがロックされている **LOCK** の 3 つの状態がある. これらを特に**確定状態**という.

ページは確定状態に対して**移行状態**を持つ. 移行状態とは, ある確定状態に移行中の状態を表す. 移行中とは, ページフォルト発生からページの供給完了までの間を表し, その間にページの無効化や移送が行われる. 移行状態は, 例えば, READ 状態へ移行中ならば READ-shift と表す.

移行状態でページアウトが発生した場合のアクセス状態は, 各々の確定状態へ遷移する. また, 確定状態でページアウトが発生した場合のアクセス状態は READ 状態に移行する (後述). ただし, ページアウト発生時のアクセス状態が LOCK 状態の場合のみ **LOCK-out** へ移行する. これは, ページアウトによってページが 2 次記憶に書き出された状態においても, LOCK 状態を保持する必要があるためである. 以上によって, ページの状態は図 2に示すように 7 通りで表される. これらの状態は, ページへのアクセスの種類によって遷移する. アクセスの種類には, 読み出し, 書き込み, 排他制御のためのロック要求と, ページアウトの 4 種類がある. 読み出し, 書き込み, ページアウトの 3 つは, 外部メモリ管理インターフェースによって発生する. ロック要求は, ユーザタスクからのタスク間通信によって発生する.

3.2 ページオーナとホーム

各々の DSM サーバに対応するページの要求は, ページオーナに送られて処理される. ページオーナとは, 各時点で当該ページへの書き込み権を所有するマシンを表す. ページオーナは, 各ページ毎に設定され, ページへの書き込みよってマシン間を移動する. ページオーナは, ページの要求を処理するために, アクセスの種類別に待ち行列 (アクセスキュー) を持つ. 読み出し用に READ キュー, 書き込み用に WRITE キュー, ロック用に LOCK キューの 3 つがある. アクセスキューは, ページオーナが変更されると新しいページオーナへ移動する.

ページオーナはその時々によって異なるため, すべての MM が, すべてのページのページオーナを常に把握することは難しい. 従って, DSM サーバでは各共有オブジェクト毎に特定の MM によってページオーナを管理する方法を採用している. この管理を行なう特定の MM をホームという. ホームは, その共有オブジェクトを始めに宣言したマシンの MM がなる. ホームは, ページフォルトが発生したマシンからページオーナへのページ供給要求の中継を行う, また, 各 MM からのページオーナ検索の問い合わせに応答する.

3.3 一貫性制御

分散環境でメモリオブジェクトを共有するためには, 共有を行う各マシンがメモリオブジェクトのコピーをローカルのアドレス空間に所有する必要がある. しかし, そのコピーに対するアクセスの結果は, どのマシンでも常に同じでなければならない. メモリオブジェクトの読み出しは内容を変更することはないので, コピーに対する特別な制御は必要ない. しかし, メモリオブジェクトの書き込みは内容を変更する. この時, 書き込みの発生したマシン以外が所有するコピーのすべてを無効化するか, すべてを同時に変更する必要がある. すべてを無効化する方式を **write-invalidate 方式**といい, すべてを更新する方式を **write-through 方式**という. DSM サーバでは, write-invalidate 方式を採用している. 現在, write-through 方式を含めて他の方式の実装を検討中である.

write-invalidate 方式

この方式ではページフォルトの発生したページに対して, 2 つのフェーズに分けて処理が行われる. 当該ページのコピーの無効化を行う**無効化フェーズ**と,

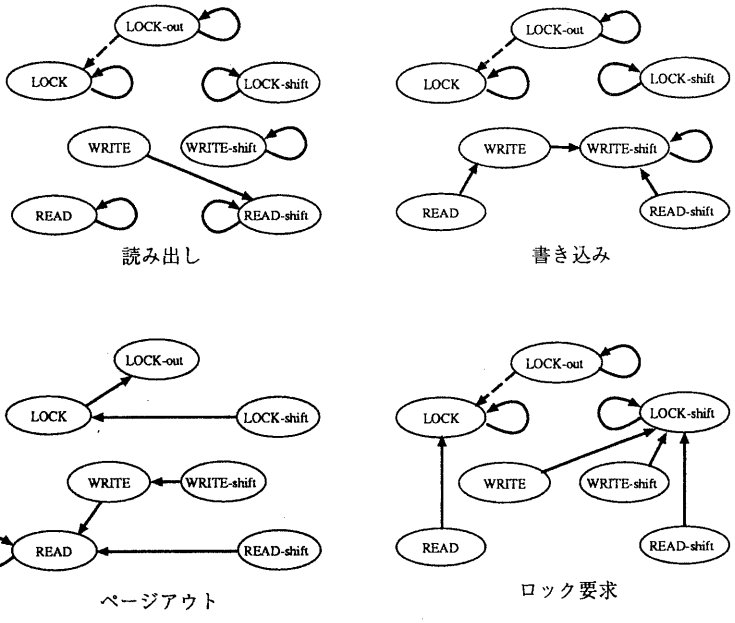

図 2　共有メモリページの状態遷移

ページフォルトの発生したマシンへページの供給を行い, 書き込みを可能にする**更新フェーズ**である.

無効化フェーズ

無効化フェーズは, ページフォルトが発生してから, すべてのコピーの無効化が終了するまでの間を示す.

クライアントのメモリアクセスによって発生したページフォルトは, Mach カーネルから外部メモリ管理インタフェースを用いて, そのマシンの MM にページ要求として伝わる. MM はその要求を, 当該ページのページオーナへの要求に変換する. ここで, ページオーナへの要求は, 一旦その共有オブジェクトのホームにページオーナ検索のために送られる. その後, ホームによって, 当該ページのオーナへ送られる. ここでは以下に示す 3 つの処理が行われる. ただし, どの処理が行われるかについては, 発生時のアクセス状態とアクセスの種類によって異なる. それぞれの場合の処理内容を表 1に示す (表中の番号は, それぞれ以下に示す番号に対応する).

(1) アクセスの種類に応じたアクセスキューへのキューイングを行う.

(2) アクセス状態を, アクセスの種類によって新しい状態へ移行させる (状態遷移については図 2を参照).

(3) LOCK-out 状態のページを主記憶上に再配置する.

(4) ページの無効化を要求する.

表 1の READ 状態における読み出しと書き込みの場合に, (1) の処理が記載されていない. これは, 要求されたページの最新の内容が, 既にページングファイルに書き込まれており, ページの更新処理を行う必要がないことによる. 従って, ページオーナは直ちに要求を PM へのページイン要求へ変換する. さらに, 読み出し要求の場合は, そのマシンをそのページを読み出しているマシンのリスト (READER リスト) へ追加し, 書き込み要求の場合は, ページオーナの変更処理を行う (変更処理については更新フェーズを参照).

移行状態では, それ以前の要求によってすでに無効化が行われており, さらに無効化が行われることはない. 移行状態でのアクセス状態の遷移は, 現在のアクセス状態よりも強いアクセスが発生した場合にのみ発生する. DSM サーバでのアクセスの優先度は, ロック

要求 > 書き込み > 読み出しの順になっている．例えば，WRITE-shift 状態において，読み出しと書き込みはキューイングが行われるのみであるが，ロック要求の場合はその状態を LOCK-shift 状態へ移行する．

LOCK-out 状態でのページ要求は，ページフォルトが発生したマシンによってその処理が異なる．そのページをロックしているマシン (ページオーナ) 以外からの要求の場合は，キューイングを行うのみである．ページオーナからの要求の場合は，LOCK 状態へ移行し PM へページイン要求を行う (図 2では，点線で表わしている)．

実際に無効化が必要となるのは，表 1中に (4) の処理が記載されている 3 つの場合のみである．アクセス状態が WRITE の場合は，書き込みを行っているマシン (通常はページオーナ) へ無効化の要求が行われる．また，READ 状態の場合は，READER リストの各マシンへ無効化の要求が行われる．

更新フェーズ

無効化フェーズにおいて，**書き込みが行われたページの無効化が行われた場合**に更新フェーズの処理が行われる．

無効化されたページは，外部メモリ管理インタフェースを介して，そのマシンの MM によってページアウトが行われる．ページオーナは，ページアウトされたページに対して次の 2 つの処理を行う．

(1) ページングファイルの内容を更新する．

(2) 当該ページへのアクセスを待っているマシンへ，ページを供給する．

ページアウトされるページは，すべてページングファイルへ反映しなければならない．このためにページオーナ MM は，PM に対してページアウト処理を要求する．ページの供給先は，その時点でのページのアクセス状態によって決まる (表 2参照)．ページオーナ MM は，アクセスキューを，LOCK キュー，WRITE キュー，READ キューの順に参照し，始めにキューイングされていたマシンに対してページの供給を行う．ただし，READ キューのみにキューイングされていた場合は，READ キューにキューイングされているすべてのマシンにページを供給する．最後に，ページが供給されたマシンは，アクセスキューから取り除く．

アクセス状態が移行状態にある場合は，無効化要求によってページアウトが行われる．確定状態の場合は，DSM サーバの無効化要求とは無関係にページアウトが発生する．すなわち，主記憶上に新たなページを配置できなくなったために，Mach カーネルがページ置き換えアルゴリズムに従って，ページアウトを要求したものである．この場合，MM は PM にページアウト要求を行い，アクセス状態を READ 状態へ移行させる．ただし，LOCK 状態でページアウトが発生した場合のみ，LOCK-out 状態へ移行する．これは，ページアウト中でも LOCK 状態を維持する必要があるためである．また，LOOK-out 状態では，ページはすでにページアウトされていので，ページアウトが発生することはない．

アクセス状態が LOCK-shift と WRITE-shift の場合には，書き込みを行うマシンが変わるために，ページオーナの変更が必要である．ページオーナの変更時には次の 2 つの処理が行われる．

(1) ページオーナの変更をホームに連絡する．

(2) 新しいページオーナに，アクセスキューとアクセス状態を移送する．

ページオーナでは，アクセスキューとアクセス状態によってページアクセスのための処理を行う．このために，ページオーナが変更された場合には，新しいページオーナにアクセスキューとアクセス状態を移送しなければならない．

3.4 排他制御

DSM サーバの排他制御は，一貫性制御の一機構 (ロック要求の処理) として実現している．MM が一貫性制御に用いている write-invalidate 方式は，同時には唯一つのマシンのみを書き込み可能にする方式である．MM が行う排他制御は，この方式の「唯一つのマシンによるアクセス」に着目したものである．しかし，通常の処理では，書き込みの途中でも，他のマシンから要求があれば MM はその権利を放棄し，他のマシンに権利を移さなければならない．そこで，書き込み権の制御をユーザに委ねることで，特別な処理を行うことなく排他制御を実現している．ユーザがロック要求を行うと，そのユーザによってロックが解除されるまで，他のマシンからのアクセス要求はキューイングされ続ける．実装において書き込みと異なる点は，アクセス状態のみである．従って，ロック要求は，書き込みと同一のコードを使用している．

このように排他制御は一貫性制御を利用することによって実現している．しかし，Mach のメモリオブ

表 1　無効化フェーズ

アクセスの種類	アクセス状態	ページオーナの処理
読み出し	READ	
	WRITE	(1), (2), (4)
	LOCK	(1)
	READ-shift	(1)
	WRITE-shift	(1)
	LOCK-shift	(1)
	LOCK-out	(1), (2), (3)
書き込み/ロック要求	READ	(2), (4)
	WRITE	(1), (2), (4)
	LOCK	(1)
	READ-shift	(1), (2)
	WRITE-shift	(1), (2)(ロック要求のみ)
	LOCK-shift	(1)
	LOCK-out	(1), (2), (3)

表 2　更新フェーズ

アクセス状態	ページオーナの処理
READ-shift	READ キューの各マシンへページを供給する.
WRITE-shift	WRITE キューの先頭のマシンへページを供給する.
LOCK-shift	LOCK キューの先頭のマシンへページを供給する.
READ, WRITE	READ 状態へ移行する.
LOCK	LOCK-out 状態へ移行する.

ジェクトの管理はマシン単位で行われる. 従って, 同一マシン上で複数のタスクが共有オブジェクトを使用していた場合, あるタスクを書き込み可能にすると, その他のタスクも書き込み可能になってしまう. このため, 同一マシン上のタスク間の排他制御は, Mach のメモリオブジェクト管理とは別に, MM が行う必要がある. しかし, Mach のメモリオブジェクト管理では, 特定のタスクのみをアクセス可能にすることはできない. 従って, MM では, ロック要求を行ったスレッドをメッセージ待ち状態にして書き込み可能になるまで中断させている. しかし, この方法では, 排他制御されているページに, 他のスレッドがロック要求を行わずにアクセスを行うことが可能となってしまう. 従って, 本システムでの排他制御の使用は, ユーザの厳密な管理が前提になっている.

4　ページングマネージャ

ページングマネージャは, 各マシンが所有するページング用 2 次記憶 (ページングファイルと呼ぶ) をネットワーク全体で共有するための機構である. これにより, あるマシンのページングファイルに新たなページを書き込む空きがなくなった場合でも, 他のマシンのページングファイルに空きがあれば, そのマシンへページングを行うことが可能になる. これをリモートページングと言う. PM がリモートページングを依頼するマシンは, PM の起動時に静的に定まる. この依頼先へのリンクは, システム全体としてサイクルを構成するように設定される (図 3参照). これを **PM** リンクと言う. PM のリモートマシンへの要求は, すべて PM リンクを介して行う.

PM は MM からの要求を受けて, ページイン処理とページアウト処理を行う. これらの処理は, いずれ

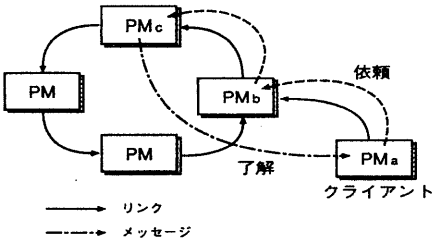

図 3　PM リンク

の場合も以下に示す手順によって行われる.

(1) ページングファイルテーブルの検索を行い, 当該ページのページングファイル内の位置を確定する.

(2) ページの位置が確定すると, 当該ページのページングを行う.

(3) ページングができない場合, PM リンクを介して他のマシンへ要求の転送を行う.

ページングファイルテーブル (以下 PFT) は, ローカルマシンのページングファイルのページと 1 対 1 に対応する配列構造のデータである. PFT で管理されるデータは, 共有メモリオブジェクトの ID(object_number) と, 共有メモリオブジェクト内でのオフセット (page_offset) である. PM はこの 2 つのデータに基づいてページングファイルへアクセスを行う. ページイン/アウト処理は, PFT と PM リンクによりすべてのマシンのページングファイルを対象に行われる.

5　評価

本章では, DSM サーバにおける各種処理のオーバヘッド, 特に一貫性制御のオーバヘッドについて述べる. 評価環境は, i486(66MHz), 主記憶 16 メガバイト, 外部キャッシュ256 キロバイト, SCSI HDD のマシン 4 台を EtherNet で接続した構成になっている.

無効化フェーズ

本フェーズにおけるページの無効化に要する時間は, ページオーナに配置してあるコピーについては 1 ミリ秒未満である. これは, ローカルの処理のみが行われ, ネットワーク通信のコストがかからないためである. また, ページオーナ以外のマシンのコピーについては, 1 個につき 10 ミリ秒となり, コピー数に比例して処理時間が増加する. これは, ページオーナ以外のマシンに対して, 個別に無効化のメッセージを送信しているためである. これをブロードキャストやマルチキャストに置き換えることで, 台数に依存しない処理が可能になると思われる [6].

また, ページフォルトが発生したマシンの MM からホームまで要求を転送するための時間は, 5 ミリ秒となっている. 従って, ホームを経由してページオーナへ要求を送るためには, 2 回分の要求の転送が必要となるので, 結果として 10 ミリ秒となる. しかし, ホームでページフォルトが発生した場合や, ホームがページオーナであった場合には, 1 回の転送で十分なので 5 ミリ秒となる.

更新フェーズ

更新フェーズでは, ページオーナから他のマネージャへのメッセージ送信が必要となる. これらのメッセージ通信に, Mach の copy on reference によるポートを使用している. 従って, 更新フェーズの処理時間は, 3.3節で述べたページオーナ MM での処理に要する時間と, copy on reference による通信時間を合わせたものになる. 更新フェーズでの処理時間を表 3に示す. ここで, 「ページオーナ MM の処理」以外はすべて通信時間である. このように, 一貫性制御の処理時間の多くが通信時間であることから, 一貫性制御の高速化には通信処理の高速化が重要になる.

表 3　更新フェーズの処理時間

処理内容	処理時間 (ミリ秒)
ページオーナ MM の処理	10
ホームへの伝達	5
アクセスキューの移送	5
ページの供給	25
合計	45

アクセスコスト

マシン上にページのコピーを持たない時に書き込みを行った場合, そのページにアクセス可能になるまでに必要な時間を表 4に示す. また, READ 状態と WRITE 状態での一貫性制御の処理時間の内訳を表 5に示す. READ 状態でのアクセスは, 無効化/更新

フェーズによらずに処理が行われるので,表4のコピー数による処理時間の差は,コピーの無効化に起因するものである.WRITE状態では,ページオーナのみがコピーを所有しているのでに,ローカルマシンに対する無効化のみで十分である.このため,多数のコピーがキャッシュされている場合のREAD状態と比較すると,更新フェーズが必要な書き込みの方が高速な結果を得ている.

表4 ページングの総処理時間

アクセス状態	コピー数	処理時間(ミリ秒)
READ	1	40
	2	50
	3	60
WRITE	-	50

表5 READ/WRITE状態における一貫性制御の処理時間

処理内容	処理時間(ミリ秒)	
	READ状態	WRITE状態
ページオーナへの要求	5または10	5または10
無効化	10 * コピー数	1未満
ページの供給	25	-
更新フェーズ	-	45
合計	40以上	50または55

6 おわりに

本論文では,Machの外部ページャを用いたDSMサーバの構成と評価について述べた.

DSMサーバでは,一貫性制御方式としてwrite-invalidate方式を採用している.これは,Machの外部ページャと外部メモリ管理インタフェースを用いた実装に適していることによる.write-through方式では,主記憶にキャッシュされたメモリオブジェクトの内容が変更されたことをDSMサーバが知る必要がある.外部ページャを使用してそれを知るためには,書き込みを行うマシンに対して,常に無効化を要求しなければならない.無効化のコストは,本論文で示したように,決して低いものではない.従って,効率良くwrite-through方式を実現するためには,カーネルの変更が必要になる.

DSMサーバでは,排他制御も一貫性制御の一環として行っている.これは,排他制御機構を実現する際に必要となる処理と,write-invalidate方式における処理に共通して使用できる部分が多いためである.しかし,本論文で述べたように,厳密な排他制御を実現するにはこれでは不十分である.現在,分散環境で,スレッド単位での排他制御を実現するシステムを検討中である.

参考文献

[1] Ming-Chit Tam, Jonathan M. Smith, and David J. Farber: *A Taxonomy-Based Comparison of Several Distributed Shared Memory Systems*, Operating System Review, Vol. 24, No. 3 (1990).

[2] 斎藤彰一, 廣瀬宰平, 大久保英嗣, 大野豊, 白川洋充: Machの外部ページャを利用した分散共有メモリサーバとその応用, 情報処理学会研究会報告 93-OS-61, Vol. 93, No. 68, pp. 41-48 (1993).

[3] Mike Accetta, Robert Baron, David Golub, Richard Rashid, Avadis Tevanian, and Michael Young: *Mach: A New Kernel Foundation for UNIX Development*, 1986 Summer USENIX Conference (1986).

[4] Richard Rashid, Avadis Tevanian, Jr., Michael Young, David Golud, Robert Baron, David Black, William Bolosky, and Jonathan Chew: *Machine-Independent Virtual Memory Management for Paged Uniprocessor and Multiprocessor Architecture*, Proc. of 2nd Architectural Support for Programming Languages and Operating Systems (1987).

[5] David Black: *External Memory Manager*, 1991 USENIX Mach symposium tutorial (1991).

[6] Henri E. Bal, M. Frans Kaashoek, and Andrew S. Tanenbaum: *Replication Techniques for Speeding up Parallel Applications on Distributed Systems*, Concurrency Practice & Experience, Vol. 4, No. 5, pp. 337-355 (1992).