ホームベースソフトウェア分散共有メモリ上で Migratory Access を効率良く処理する権限委譲プロトコル

城田祐介[†] 吉瀬謙二[†]
本多弘樹[†] 弓場敏嗣[†]

コンシステンシモデルに Scope Consistency を用いたホームベースソフトウェア分散共有メモリシステムにおいて，同一のロック変数に関するロック操作とアンロック操作の中で起こる共有メモリに対する read&write および write を migratory access と呼ぶ．PC クラスタを対象とする従来のシステムでは，複数ノードで migratory access が頻繁に実行される場合，排他制御操作ごとのホームノードへのライトバックがオーバヘッドとなり，パフォーマンスが著しく低下する．このため，複数ノードで migratory access が頻繁に実行されるアプリケーションでは，スケーラビリティを得ることが困難だった．階層的なネットワーク構成を持つ SMP-PC クラスタではこの傾向がさらに顕著化する．本稿では，一連の migratory access を実行する複数ノード間でページを直接更新できる権限を一時的に委譲し，巡回させることで同アクセスにともなうライトバックのオーバヘッドを削減する新しいページコヒーレンシプロトコルとして権限委譲プロトコルを提案する．また，一連の migratory access の実行に先行して複数ノードから発行されるロックリクエストが，システムの処理が追いつかずにキューイングされる状態で同アクセスを自動的に検出し，権限委譲プロトコルで一括処理する機構を提案する．提案方式を既存のソフトウェア分散共有メモリシステム JIAJIA に実装し，ベンチマークプログラムで性能評価した結果，PC クラスタおよび SMP-PC クラスタ上で高い性能を得られることが明らかになった．

The Ownership Delegation Protocol for Migratory Access in Home-based Software Distributed Shared Memory

Yusuke Shirota,[†] Kenji Kise,[†] Hiroki Honda[†]
and Toshitsugu Yuba[†]

With the home-based scheme, Software Distributed Shared Memory (SDSM) systems' eager propagations of updates to the home node at release times can have a great impact on its performance, especially for applications with migratory access patterns. For efficient execution of these applications, we introduce a new Scope Consistency, multiple-writers sub-protocol, called Ownership Delegation Protocol, which collectively apply updates of some consecutive migratory accesses without causing writebacks. We also propose a mechanism that dynamically identifies migratory access and that adaptively switches between a proposed sub-protocol and a conventional home-based protocol which is appropriate for other data-access patterns. We implemented the proposed scheme on a home-based SDSM system called JIAJIA. Performance evaluations indicate that proposed scheme outperforms home-based-only approaches, and achieves considerable speedups for applications with heavy migratory access patterns on PCs clusters and SMP-PCs clusters.

1. はじめに

PC クラスタや SMP-PC クラスタの普及とともに，分散メモリ上に仮想的な共有メモリの構築をユーザレベルのソフトウェアライブラリで実現するページベースソフトウェア分散共有メモリ（SDSM）システム[1)~4)]が注目されている．オペレーティングシステムの仮想記憶で用いられるページ単位でコヒーレンシを維持するページコヒーレンシプロトコルのメモリアーキテクチャは，ライトバックの戻り先であるホームノードがページごとにある固定ノードに決められている方式とそうではない方式に分類される．近年の研究[1),5)]において，前者の固定のホームノードがページごとに存在する方式（以後，ホームベースプロトコル）を用いたホームベース SDSM システム[1),2)]が，後者

† 電気通信大学大学院情報システム学研究科
Graduate School of Information Systems, The University of Electro-Communications

のホームノードを持たずに diff 分散方式[3]を利用するページコヒーレンシプロトコルを用いたホームレスなシステム[3),4)]より総合的に高い性能をもたらすことが明らかになっていて，前者が決定版となっている．

しかし，前者は後者が持つ多くの問題点を解決しているものの，ホームノードを設ける副作用として，同期操作ごとのホームノードへのライトバックが新たなオーバヘッドになる．よって，メモリアクセスパターンに適合したデータ配置（ホームノードの配置）ができるかどうかがパフォーマンスに影響を与える[6)]．

なかでも，複数ノード間で頻繁にライトバックと読み出しが行われるメモリアクセスパターンでは，データ配置は難しく，ライトバックによって著しくパフォーマンスが低下する．また同メモリアクセスはロックで排他制御されるため，逐次化される．そのため，このようなメモリアクセスの実行回数が比較的多いアプリケーションはそもそもスケーラビリティが期待できないので，SDSM システムの適用対象外になっている．事実，これらのアプリケーションに限定すると diff 分散方式を利用したシステムが高い性能を示している[1),7)]．また，階層的なネットワーク構成を持つ SMP-PC クラスタではこの傾向がさらに顕著化する．なぜならホームノードへのライトバックのほとんどが SMP-PC ノード間を介したもので，SMP-PC ノードのローカリティを利用できないからである．

ロックを用いた排他制御は，ビジネスアプリケーションで頻繁に行われる集計処理[8)]，画像処理におけるソフトウェアパイプライン処理，数値計算におけるリダクション演算などにおいて典型的に引き起こされる．また，性能ヘテロなクラスタシステムや，1 つのアプリケーションで独占的に使用できないシステムなどにおいて，負荷をバランスさせるためにタスク並列にプログラムを記述することがある．この例では，タスクキューへのアクセスにおいてロックが利用される．

また，ロックを利用するとメモリアクセスが逐次化されるので大規模なクラスタシステムでは逐次部分の実行比率が大きくなってしまう．以上により，ロックで排他制御される共有メモリアクセスを効率良く処理することが，実用的な SDSM システムの実現に向けて重要な課題である．

ロックで排他制御される共有メモリアクセスのうち，排他制御操作ごとに実行されるホームノードへのライトバックを引き起こす可能性のあるものを migratory access[9)]と呼ぶことにする．本稿では，複数ノードが migratory access を頻繁に実行するアプリケーションに対してもホームベース SDSM システムを効率的に適用可能とすることを目的とする．

本稿では，ホームベースプロトコルを前提に，一連の migratory access を実行する複数ノード間でページを直接更新できる権限を一時的に委譲し巡回させることで同アクセスにともなうライトバックのオーバヘッドを削減する新しいページコヒーレンシプロトコルとして権限委譲プロトコルを提案する．また，一連の migratory access の実行に先行して複数ノードから発行されるロックリクエストが，システムの処理が追いつかずにキューイングされる状態で同アクセスを自動的に検出し，権限委譲プロトコルで一括処理する機構を提案する．提案方式を既存の SDSM システム JIAJIA version 2.1[1)]に実装し，評価した．

以下，本稿では，2 章で本稿が前提とするコンシステンシモデルとホームベースプロトコルについて述べ，その問題点を示す．3 章では権限委譲プロトコルと提案機構を説明する．また，4 章ではベンチマークプログラムによる提案方式の性能評価結果を示す．5 章で関連研究を概観し，6 章で今後の課題を述べる．

2. 従来のホームベースプロトコルの課題

2.1 Scope Consistency と Lock-Based ページコヒーレンシプロトコル

本稿で前提とするコンシステンシモデル Scope Consistency[10)]と同コンシステンシモデルを実装するページコヒーレンシプロトコル Lock-Based プロトコル[1)]（以後，特に断らない場合は，本稿におけるホームベースプロトコルは Lock-Based プロトコルを指すこととする）の概要を説明する．

Scope Consistency では，あるロック変数 i に対するロック操作（$\mathrm{Acq}(\mathrm{L}_i)$）とアンロック操作（$\mathrm{Rel}(\mathrm{L}_i)$）（以後，Scope i）の中で実行された write の値が，後続する Scope i の中で read できることのみを保証する．そのために，$\mathrm{Rel}(\mathrm{L}_i)$ を実行するノードは，対応する $\mathrm{Acq}(\mathrm{L}_i)$ を実行するノードに対し，それ以前に実行された write を次の方法で通知する．

まず，ロックマネージャと呼ぶロック変数ごとに決められたある固定ノードが，ロック構造体 (L_i) を用いて write の実行情報（もしくはページ無効化情報と呼ぶ）を管理する．$\mathrm{Acq}(\mathrm{L}_i)$ を実行するノードは，ロックリクエストをロックマネージャへ発行する．ロックマネージャでは，受信したロックリクエストを FIFO（以後，ロックリクエストキュー）に蓄えて，1 リクエストずつこれを処理（以後，サービス）していく．発行したロックリクエストが処理され，ロックマネージャより L_i を獲得したノードがページ n (p_n) に対して

write を実行すると，n は L_i に追加される．Rel(L_i) を実行すると，twin と呼ぶ p_n に対して write する直前の p_n のコピーと p_n の差分情報である diff が生成され，これを用いてホームノードへライトバックしアップデートする．ライトバックが完了したことを確認し，L_i をロックマネージャに対して解放する．これにより，それ以降に Acq(L_i) を実行し L_i を獲得するノードは，write が実行された p_n の検出と無効化を行うことができ，write された値は p_n のホームノードよりアップデートされたページをフェッチすることで read できる．

2.2 対象とするアクセスパターン

本稿では，migratory access を議論の対象とする．これまでにいくつかの定義[6),9)]がなされているが，本稿ではホームベース SDSM においてライトバックを引き起こす，オーバヘッドとなる可能性のある共有メモリアクセスを扱う便宜上，migratory access を以下のように再定義する．

[migratory access]

Scope Consistency において，同一のロック変数に関するロック操作とアンロック操作を用いて排他制御する共有メモリに対する read & write および write

2.3 migratory access を実行する場合の問題点

ホームベース SDSM 上で migratory access を実行する場合の問題点を明確にする．そのために，まず，ホームノードを設ける副作用であるアンロック操作ごとのホームノードへのライトバックのオーバヘッドとホームノードの配置の関係について，diff 分散方式と対比させて議論する．

ホームベースプロトコルが diff 分散方式に対してパフォーマンスが劣る可能性がある 1P-1C（one producer with one consumer）[6)] と呼ぶページ単位の共有パターンを用いて説明する．1P-1C の共有パターンにおいて，producer と consumer 以外のノードがホームノードである場合，diff は diff 分散方式では producer がローカルに保持するのに対し，ホームベースプロトコルではホームノードへライトバックされるため，これがオーバヘッドとなる．

1P-1C の共有パターンが交互に頻繁に複数ノードで実行される migratory access ではどのノードにホームノードを配置しても，ホームノードをアップデートするためにホームノード以外のノードはアンロック操作のたびに diff をホームノードに転送するので，これが過大なオーバヘッドとなる（**図 1**）．図中の太線はページの転送，細線は diff の転送を表している．

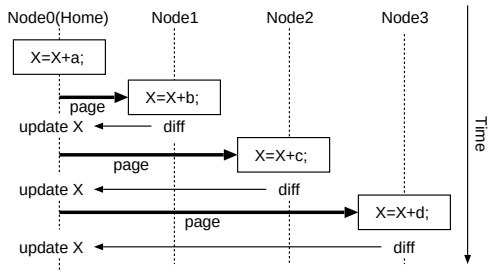

図 1 ホームベースプロトコル
Fig. 1 Home-based protocol.

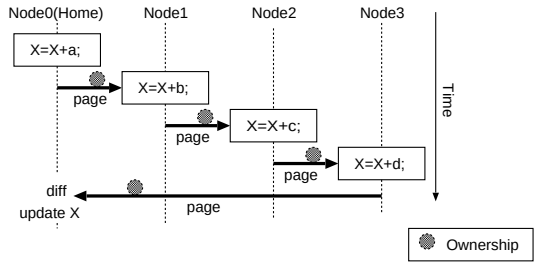

図 2 権限委譲プロトコル
Fig. 2 Ownership delegation protocol.

3. Migratory Access を効率良く処理するホームベース SDSM の提案

3.1 権限委譲プロトコル

複数ノードで migratory access が頻繁に実行される場合，アンロック操作ごとのホームノードへのライトバックがオーバヘッドとなり，パフォーマンスが著しく低下する．この一連の migratory access を実行する複数ノード間でページを直接更新できる権限の一時的な委譲を繰り返し，権限とページを巡回させる．そして，一連の migratory access の終了後，ページをホームノードにライトバックする．これにより，従来のホームベースプロトコルで必要とした diff/twin の生成と頻繁にホームノードにライトバックするオーバヘッドの削減が可能となる．

図 2 を用いて権限委譲プロトコルを説明する．まず，権限委譲プロトコルのために，新たにページのホームノードの権限（以後，ホーム権限）という概念を導入する．ホーム権限は「直接ページを更新できる（twin/diff を生成しライトバックする必要がない）」，ページごとに存在する，ロック変数に固有の権限で，初期状態ではページの本来のホームノードが所有している．図中の円形がホーム権限を表している．ホームノード以外のノードがホーム権限を与えられることで，同ノードはそのページの仮想的なホームノードと見な

される．これにより本来のホームノードが持つ「直接ページに書き込める」利点を持つことになる．

ホーム権限は，migratory accessが引き起こすページフォルトのときに，そのページに対して migratory accessを実行したノードに委譲される．例では，ノード 1が共有変数 X に対する read ミスに起因するページフォルトで X を含むページ p_n のホームノード 0に対してページリクエストを行う．図中の円形の移動は，ホームノード 0がノード 1に対してクリーンなページの転送（以後，ページサービス）を行うときに，ロック変数 i のページ p_n のホーム権限 $O_i^{p_n}$ を，一時的にノード 1に委譲することを示す．

$O_i^{p_n}$ を委譲されたノード 1は，p_n の仮想的なホームノードと見なされる．したがって，従来のホームベースプロトコルで行っていた twin/diff の生成と，同 diff を用いた本来のホームノード 0へのライトバックを行う必要がなくなる．また，ノード 1は，$O_i^{p_n}$ が一時的に委譲されたことを，ロック構造体 Li に付加してこれを解放する．それによって，次にノード 2が Li を獲得した際に，$O_i^{p_n}$ の委譲を検出できる．これにより，1P-1C のページ単位の共有パターンにおいて，producer がホームノードに配置されている関係を常時保ちながら，さらにホーム権限を次々に委譲することが可能となる．

最後に p_n に migratory access を実行するノード 3が複数ノード（ノード 1，ノード 2，ノード 3）で write が実行されたページをホームノード 0に転送し，$O_i^{p_n}$ も同ホームノードに戻す．ホームノード 0は，$O_i^{p_n}$ を委譲している間に p_n に対して行われた変更点を diff として抽出し，これを用いてホームノードをアップデートすることが必要となる．このために，ホームノード 0は，$O_i^{p_n}$ を委譲する直前のページのコピー（Home Twin $^{p_n}_{i}$）をあらかじめ作成しておき，同コピーとノード 3から転送されたページの diff を作成し，これを用いてホームノードのページをアップデートする．このように，一連の migratory accessにおいて，ホーム権限を巡回させることでホームノードを介することなく共有変数を更新できるので，一連の権限を委譲されたノード列の最後までライトバックのオーバヘッドを削減することができる．

3.2 権限委譲プロトコルとホームベースプロトコルを動的に切り替える機構

3.2.1 コンテンションの検出手法

まず，SDSM システムにおける実行時の特別な状態を利用することで，同システムの処理におけるコンテンションを動的に検出する手法を提案する．この実行時の特別な状態とは，複数ノードから発行されるページリクエストやロックリクエストが，システムの処理が追いつかずに同リクエストを受信したノードでキューイングされている状態である．この状態では複数ノードがリクエストが処理されるまでただ待っているので，コンテンションを検出しキューイングされている複数のリクエストを一括処理できればシステムを高速化することができる．

3.2.2 権限委譲プロトコルの適用方式

上記のコンテンションの動的検出手法を利用して，権限委譲プロトコルを適用する方式について論じる．

複数ノードで migratory access が頻繁に実行されるケースでは，同アクセスの実行に先行して発行されたロックリクエストがロックマネージャのロックリクエストキューに複数到着する．ロックリクエストが複数キューイングされている場合，キューイングされているリクエストの数と同数のノードはロックがサービスされるのをただ待っているので，権限委譲プロトコルが効果を発揮する．

つまり，オーバヘッドとなりうる migratory access の検出は，ロックリクエストキューの状態を検査しロックのコンテンションが高いことを検出することで可能となる．ロックリクエストが一定数以上キューイングされているときに，ホームベースプロトコルから権限委譲プロトコルに動的に切り替え（以後，権限委譲プロトコルを起動する），これらロックリクエストに関する一連の migratory access を一括処理し，同処理終了後に再度切り替える（以後，権限委譲プロトコルを終了する）ことでロックをサービスするスループットをあげる仕組みを提案する．また，権限委譲プロトコルの起動から終了までを「権限委譲の旅」と呼ぶことにする．

ホームベース SDSM 上に，権限委譲プロトコルを実装する**図 3** の例を使って説明する．例では，ノード 0以外のノードが，共有変数 X, Y を含む p_0 と，同 Z を含む p_1（ホームノードはいずれもノード 0）に対して，図中のような migratory access を実行する．

Acq(L_0) の実行で発行されるロックリクエストはロックリクエストキューに，(ノード 1 →ノード 2 →ノード 3）の順番でキューイングされている．この順番を，「権限委譲の旅の順番」と定義する．

例において，権限委譲プロトコルの起動条件を，ロックリクエストが 3つ以上キューイングされていることとする．ロックリクエストが 3つ未満のとき，ロックマネージャはホームベースプロトコルに従いロックのサービスを行う．

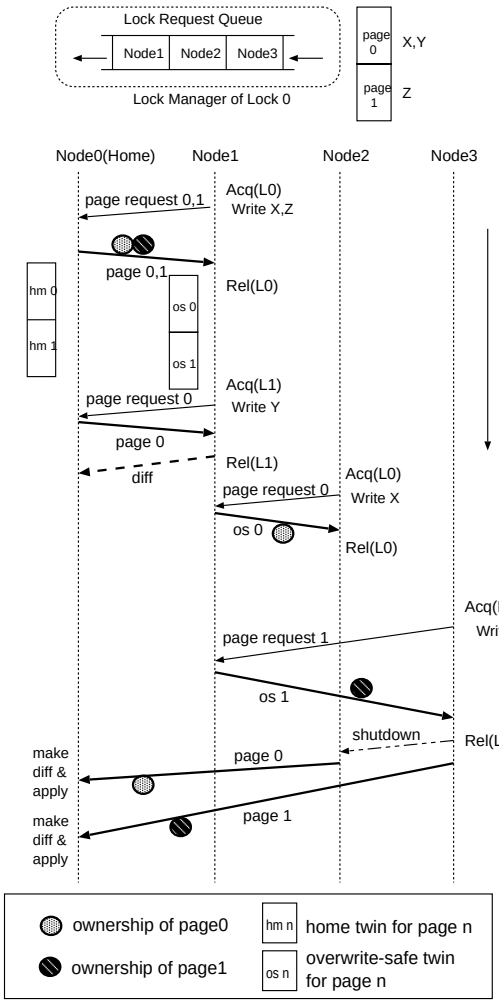

図 3 権限委譲プロトコルにおけるマルチプルライタ
Fig. 3 Ownership delegation protocol with multiple writers.

ロックマネージャは，ロックのサービスを行うときに権限委譲プロトコルの起動条件を評価し，条件が成立した場合，権限委譲プロトコルを起動し，以下の手順が実行される．

ロックマネージャは，ノード 1 に L_0 をサービスするときに，旅の順番も通知する．ノード 1 は，L_0 を解放するとき，ロックマネージャを経由せずに，通知された旅の順番において次点のノード 2 に同順番を付加した L_0 を直接転送する．同様に，ノード 3 が旅の順番が付加された L_0 を受信する．

旅の順番の最後であるノード 3 は，権限委譲プロトコルを終了させるために，以下の手順を実行する．ホームノードをアップデートするために，一時的に $O_0^{p_0}$ を委譲されているノード 2 と $O_0^{p_1}$ を委譲されている自分自身に権限委譲プロトコルの終了リクエスト（図中の shutdown）を発行する．これにより，ホームノードにホーム権限が戻り，アップデートも正しく行われる．同手順の終了後，L_0 をロックマネージャに転送する．

権限委譲の旅の間に発行されロックマネージャに到着するロックリクエストは新規にキューイングされる．また，Scope 0 の外で発行されるページリクエストは，上記の旅の間も本来のホームノードに到着し，同ノードがページをサービスする．

3.3 マルチプルライタの実現

3.3.1 Overwrite-safe Twin

False Sharing の問題を解決するためには，同一ページの書き込みを複数ノードに同時に許すマルチプルライタプロトコルの導入は不可欠である．本項では，権限委譲プロトコルにおいてマルチプルライタプロトコルを実現するために Overwrite-safe Twin というメカニズムを提案する．

図 3 の例を使ってこれを説明する．ここではノード 1 に注目する．ノード 1 は，L_0 を獲得し，X に write する．L_0 を解放後，続けて L_1 を獲得し，（L_1 のページ無効化情報により p_0 がダーティであれば p_0 を一度無効化し，p_0 を改めてフェッチしてから）Y に write を行うと，この後で $O_0^{p_0}$ が委譲されているノード 1 が p_0 をノード 2 にサービスするときに，migratory access の中で先に行った X への write のみが反映されていなければならないにもかかわらず，Y への write まで反映されてしまう．これを防止するために，ノード 2 にサービスする p_0 のコピー（以後，Overwrite-safe Twin）は別に保存しておき，ノード 2 から p_0 がリクエストされたら，同 Overwrite-safe Twin をサービスする．

3.3.2 Overwrite-safe Twin を作成するタイミング

上記のケースにおいて，ノード 1 が獲得した L_1 のページの無効化情報により p_0 が無効化されたり，ノード 1 が p_0 に write を行わなかったりするのであれば，ノード 2 に p_0 をそのままサービスすればよいので，Overwrite-safe Twin を作成する必要がない．Overwrite-safe Twin は必要になった時点で作成する．

3.4 積極的な権限委譲プロトコル

3.1 節では，ページがリクエストされた時点でページサービスを行う権限委譲プロトコル（lazy な実装）について議論してきた．migratory access を実行する複数ノードが同一ページを参照する可能性が高い場合には，ロックのサービスと同時にページサービスを行うことでさらなるオーバヘッドの削減を目指す積極的

（以後，eager）な実装が考えられる．具体的には，権限委譲の旅の順番において次点のノードにロック構造体を転送するときに，migratory access を行った複数のページもあわせて転送する．eager な実装にする利点は，ページごとのページリクエストがページのプリフェッチによりなくなることである．欠点は，あらかじめ転送しておいたページを利用しなかったときに，転送量が増えることである．

3.4.1 SMP-PC クラスタを対象とする権限委譲の旅の順番の最適化

SMP-PC クラスタは階層的なネットワーク構成を持つため，権限委譲の旅の順番を最適化することで SMP-PC ノードのローカリティを効率良く利用することが可能となる．3.2.2 項の図 3 の例で説明する．例では，ロックのリクエストはロックリクエストキューに,（ノード 1→ノード 2→ノード 3）の順番でキューイングされている．ここで SMP-PC クラスタシステムの構成において，ノード 1 とノード 3 が同じ SMP-PC ノードに属し，ノード 2 がこれとは別の SMP-PC ノードに属するとする．この場合，権限委譲の旅において，SMP-PC ノード間を 2 回通過する．旅の順番を（ノード 1→ノード 3→ノード 2）と変更することで，これを 1 回にすることが可能となる．この旅の順番の最適化の正当性を証明する．まず，各ノードからのロックのリクエストがロックマネージャに到着する順番はタイミング依存であること，また，ロックリクエストキューにキューイングされる各ノードからのロックリクエストはたかだか 1 つであることから，上記の旅の順番の変更は不都合を生じない．

4. 評　　価

4.1 評 価 方 法

既存の SDSM システム JIAJIA version 2.1 [1] をベースに提案方式を実装し，ベンチマークプログラムで性能を評価した．JIAJIA は，Lock-Based プロトコルを用いて Scope Consistency を実装している．

4.2 評価用ベンチマークプログラム

ベンチマークプログラムには，NAS Parallel Benchmarks（NPB）の IS と単純なサンプルプログラムを用いた．問題サイズは**表 1** のとおりである．

IS では，各計算ステップ終了時に，すべてのノードがいっせいに排他的に read&write を実行する．

サンプルプログラムは，タスク並列処理のアプリケーションにおけるタスクキューに対する非同期的な migratory access を想定している．migratory access の処理効率を検証することが目的である．プログラムは，タスクキューにみたてた共有メモリ領域（4 バイト）への排他的な read&write をすべてのノードで合わせて N 回繰り返す．

表 1 問題サイズ
Table 1 Problem size.

ベンチマークプログラム	問題サイズ
IS	鍵の数 $= 2^{26}$，鍵の最大値 $= 2^{14}$
サンプルプログラム	N = 320

表 2 評価実験環境
Table 2 Experimental platform.

	システム A	システム B
# Nodes	16(32CPU)	8(16CPU)
CPU	PentiumIII 866 MHz	PentiumIII 866 MHz
Memory	1 GB	640 M
Network	100 BASE-TX	Myrinet
OS	Red Hat Linux 7.1	Red Hat Linux 6.2
Compiler	gcc 2.96	gcc 2.91.66

4.3 評価実験環境

提案方式に対するネットワーク性能の影響を検証するために，実験には 2 つのプロセッサを持つ 2 種類の SMP-PC クラスタを用いた（**表 2**）．

4.4 評 価 項 目

4.4.1 単一プロセッサノードで構成される PC クラスタを対象にした評価

表 2 のそれぞれの SMP-PC クラスタの各ノードの 1 つのプロセッサを用いた 2 種類の PC クラスタ上で提案方式の評価を行った．評価に用いたページサイズは 4K バイト，権限委譲プロトコルの起動条件はロックリクエストキューにリクエストが 2 つ以上キューイングされていることとした（以降の評価においても同条件）．また，SDSM システム JUMP version 1.0 [11] との性能比較も行った．JUMP が実装している Migrating-Home プロトコルではページサービス時にホームノードの再配置を行う．

4.4.2 SMP-PC クラスタを対象にした評価

SMP-PC クラスタで提案方式の評価を行うために，JIAJIA を SMP-PC クラスタで動作させた．SMP-PC クラスタ上の実行においては，SMP-PC クラスタの各 SMP-PC ノードで 2 つのプロセスを起動させ，SMP-PC ノード内通信にはノード間と同様に UDP を用いている．

4.4.3 SMP-PC クラスタを対象にした権限委譲の旅の順番の最適化手法の評価

SMP-PC クラスタ上で提案方式における権限委譲の旅の順番の最適化手法の効果の評価をするために，カットオフのある粒子シミュレーションプログラムを利用した．このプログラムは，2 万個の粒子間の相互

表 3 権限委譲プロトコルの基本コスト
Table 3 Basic operation cost.

操作	コスト [μsecs]	
	システム A	システム B
ページフォルト処理（SMP ノード間）	725	311
ページフォルト処理（SMP ノード内）	219	236
各種 twin 作成	33	28
diff 作成	57-175	52-171
diff 適用	1-65	1-72

作用の計算を 60 ステップ行うものである．プログラムの概要を説明する．カットオフの判定は空間分割法で粒子をセルに割り当てることで行う．各計算ステップが終わるとバリア同期をとり，セル間を移動した粒子を最適なセルに再度割り当てる．バリア同期をとる直前にローカルに計算した粒子の加速度を足し込む migratory access を実行する．セルをサイクリックに各プロセッサに割り当てることでアプリケーション側で負荷分散もしているが，それでもプロセッサごとに計算量は異なる点が IS と異なる．これにより，各ステップごとに権限委譲の旅の順番が異なる．

4.5 権限委譲プロトコルに関する基本コスト

権限委譲プロトコルに関する基本的なコストを，対象とするクラスタシステムごとに**表 3** に示す．ここで，ページフォルト処理のコストは，ページアクセスによるページフォルトによりホームノードからページをフェッチし，ページアクセスが再開するまでの時間である．

4.6 評価結果

4.6.1 単一プロセッサノードで構成される PC クラスタを対象にした評価

JIAJIA version 2.1，JUMP version 1.0 および提案方式を実装したシステムの 3 つのシステムについて IS の実行時間を測定した結果を**図 4** と**図 5** に示す．各図にはロック操作のコストも示す．ロック操作のコストとして，Scope に入る以前の read/write が終了した時点から Scope の中の read/write を開始するまでの時間を測定する．提案方式の eager な実装の場合には，ロック操作時間に積極的に転送されたページの処理が含まれる．図に示すのは，ベンチマークプログラムの実行においてロック操作の実行時間が最も長かったノードのもの（以降の図についても同様）である．

JIAJIA の評価は，version 2.1 の新機能である Home Migration，Write Vector Technique，Adaptive Write Detection [1]を用いたものである．逐次実行では，JIAJIA ライブラリはリンクされているが，オーバヘッドはほとんどない（以降の評価についても同様）．IS の逐次実行時間を**表 4** に示す．

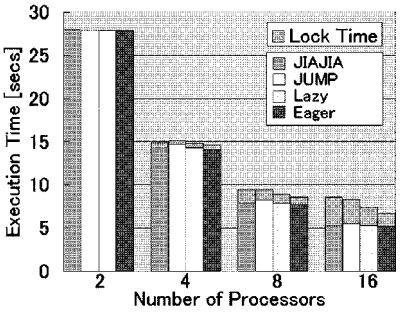

図 4 PC クラスタ A における性能比較（IS）
Fig. 4 Performance comparison for IS on PCs cluster system A.

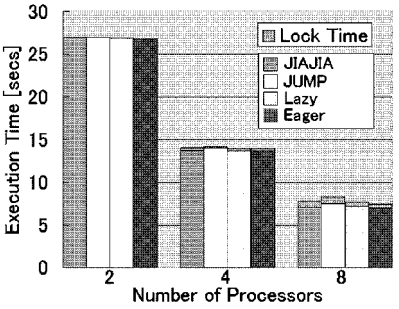

図 5 PC クラスタ B における性能比較（IS）
Fig. 5 Performance comparison for IS on PCs cluster system B.

表 4 逐次実行時間
Table 4 Sequential execution time of IS.

操作	逐次実行時間 [secs]	
	システム A	システム B
IS	55.2	53.8

表 5 IS の各種イベント実行回数
Table 5 Number of events for IS.

operations	JIAJIA	lazy	eager
page requests	4800	4960	2720
diff updates	2400	320	320

まず IS の評価結果について説明する．図 4，図 5 より，提案方式では migratory access を効率良く処理してロック操作時間を短縮し，高い性能を得ていることが分かる．IS を JIAJIA と提案方式を用いて 16 ノードで並列実行した場合のページリクエストの実行回数と，ホームノードのアップデートに利用された diff の数を**表 5** に示す．ページリクエストの実行回数を JIAJIA と比較する．eager では積極的に転送されたページを利用できたことで実行回数を削減しているが，lazy ではこれが増加してしまっている．これは，ホーム権限が委譲されているノードに対してページの

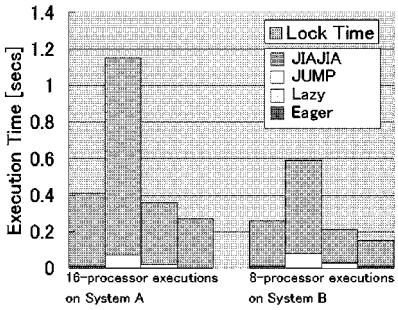

図 6 migratory access の処理時間（サンプルプログラム）
Fig. 6 Performance comparison for sample program.

表 6 サンプルプログラムの各種イベント実行回数
Table 6 Number of events for sample program.

operations	JIAJIA	lazy	eager
page requests	315	334	22
diff updates	300	23	23

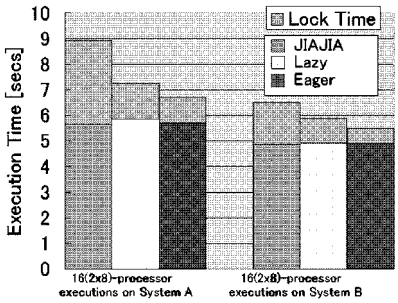

図 7 SMP-PC クラスタにおける性能比較（IS）
Fig. 7 Performance comparison for IS with 8 SMP-PCs nodes.

本来のホームノードが同ページとそのホーム権限を取り戻すためにページリクエストを行うからである．しかしホームノードをアップデートする diff の数が大きく削減できているため性能が向上している．

次に，サンプルプログラムの実行時間（**図 6**）を比較する．サンプルプログラムは migratory access のみを抽出したプログラムであり，権限委譲プロトコルの基本性能と考えることができる．

PC クラスタ B の 8 ノードにおいては，eager では JIAJIA と比較して 46.1%，JUMP と比較して 76.2%の処理速度の向上が得られた．サンプルプログラムを JIAJIA と提案方式を用いて PC クラスタ A の 16 ノードで並列実行した場合のページリクエストの実行回数と diff の数を**表 6** に示す．IS の場合と同様の傾向がみられることが確認できる．

4.6.2 SMP-PC クラスタを対象にした評価

SMP-PC クラスタの 8 ノード（16 プロセッサ）において IS を評価した結果を**図 7** に示す．評価結果を図 4 の PC クラスタ A の 16 ノードで並列実行した場合と比較する．JIAJIA では，単一プロセッサノードで構成される PC クラスタ A を用いた場合の性能が高い．しかし，権限委譲プロトコルを用いると SMP-PC クラスタ A を用いた性能が高い．これは，IS ではロックリクエストが複数ノードで規則的な順番で発行され，権限委譲の旅の順番が最初から最適になっているからである．これにより，SMP-PC ノードのローカリティを利用できていることで，性能が向上している．

表 7 逐次実行時間
Table 7 Sequential execution time of N-Body problem.

逐次実行時間 [secs]	
システム A	システム B
164.7	214.0

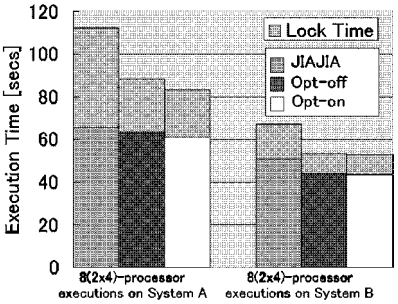

図 8 権限委譲プロトコルの旅の順番の最適化手法の効果
Fig. 8 Effect of itinerary optimization.

4.6.3 SMP-PC クラスタを対象にした権限委譲プロトコルの旅の最適化の評価

粒子シミュレーションプログラムの評価結果を示す．逐次実行時間を**表 7** に示す．2 種類の SMP-PC クラスタの 4 ノード（8 プロセッサ）を用いた実行時間を**図 8** に示す．図 8 では，JIAJIA と，権限委譲プロトコル eager に関して最適化を行った場合とそうでない場合で比較したものである．最適化を行うことにより，SMP-PC クラスタ A で 5.65%の性能向上を得た．SMP-PC クラスタ B では，SMP-PC ノード内と SMP-PC ノード間で通信性能に大きな違いがないことから最適化の効果は小さく，性能向上は 0.74%にとどまった．

4.7 提案プロトコルの改良に関する検討

上述においては，ホーム権限委譲の旅を平等に扱っていた．しかし，中には，権限委譲プロトコルのオーバヘッドが逆に性能を低下させる旅（以後，悪い旅）がありうる．たとえば，旅の中で migratory access が実行されない場合，Home Twin をつくることはオーバヘッドとなる．このようなことに対処するために，権限委譲プロトコルを起動するか否かを前回の旅の評

価によって決定し，オーバヘッドをなくす方法などが考えられる．このように悪い旅を省略するだけでよいので，オーバヘッドを容易になくすことが可能である．

5. 関連研究

一部のホームベースSDSMシステム[1),2)]では，ホームノードを動的に再配置することによりdiffの転送量を減らすよう工夫している．しかし，再配置はバリア同期時に行われるため，migratory accessによる問題を解決するものではない．

migratory accessの処理の効率化に対して，アクセスパターンに応じてシステムがシングルライタプロトコルとマルチプルライタプロトコルを選択するシステムが提案されている[12)]．同方式は，ページ全体を更新するようなmigratory accessのみを対象としており，その適用範囲は狭い．

JUMPで提案されているMigrating-Homeプロトコルでは，ページサービス時にホームノードの再配置を行うことによりホームノードの再配置をmigratory accessパターンにまで適合させる点で，権限委譲プロトコルと類似している．しかし，権限委譲プロトコルは実際にホームノードの再配置を行わないでホーム権限のみを委譲する点で異なっている．JUMPでは，ホームノードの再配置には，それを全ノードに通知するブロードキャストをともなうため，ホームノードの再配置が多発するmigratory accessではこれが新たなコストとなりスケーラビリティが得られない．一方，権限委譲プロトコルでは，ホーム権限の一時的な委譲のみを行い，ホームノードの再配置は行わないので，余計な通信オーバヘッドがない．また，権限委譲プロトコルでは，適用範囲をmigratory accessに限定し，同アクセスによるオーバヘッドを削減することで，ホームノードを再配置する際に，配置ミスすることがない．SMP-PCクラスタを対象にした場合，上記オーバヘッドによる問題はさらに顕著化すると考えられる．さらに，提案方式では一連のmigratory accessを一括処理することで，SMP-PCクラスタを対象に有効な最適化を行うことを可能としている．

また，JUMPとの比較において Overwrite-safe Twinを作成するオーバヘッドがあるが，Overwrite-safe Twinの作成はクリティカルパスの外で行われるので，オーバヘッドは大きくない．

6. 今後の課題

本稿ではJIAJIA，JUMPとの性能比較を行ったが，今後はdiff分散方式を利用したTreadMarks[3)]などのシステムとの性能比較も必要である．また，提案方式は，ホームノードの再配置は行わないため，本来のホームとホーム権限が一時的に委譲されている複数の仮想的なホームノードとが混在することが可能となる．したがって，マルチプルライタの状況で，ホーム権限委譲の旅が複数同時に実行される場合，最も効果を発揮することが期待される．よって，ソフトウェアパイプライン処理を実行する実アプリケーションなどについての適用可能性などについても検討していく．

提案方式はmigratory accessを多用するアプリケーションほど効果が得られやすいが，一方でそのようなアプリケーションは並列化の効果が得られにくい．このように，実アプリケーションによる性能評価は，migratory accessをいかに効率良く処理できたかというより，そのアプリケーションについてmigratory accessが占める割合に大きく依存してしまう．migratory accessの処理の性能向上を正確に計るためには同アクセスだけを抽出し，これに対する処理の効率化をみる必要がある．このために，実アプリケーションを用いた評価に加えて，様々なmigratory accessのパターンを表現できるようにパラメータ化したベンチマークの作成と評価が必要である．

またクラスタ専用の高速ネットワークを利用することで，従来のネットワークではmigratory accessの実行比率が大きすぎて十分な並列処理効果を得ることができなかったアプリケーションもSDSMシステムで適用できるようになる．本稿ではMyrinetで提案方式の有効性を示したことで，同方式によりSDSMシステムで適用できるアプリケーションがさらに増えることが期待される．

今回はPCクラスタおよびSMP-PCクラスタで評価を行ったが，今後は，複数のクラスタを使用するクラスタ・クラスタなどのGrid計算環境が利用可能になると考えられる．クラスタ・クラスタでは，クラスタノード間とクラスタノード内で通信性能が大きく異なる．権限委譲プロトコルが有する性能の高いホームベースプロトコルベースにブロードキャストを要せずにmigratory accessを効率良く処理できること，および，権限委譲プロトコルの適用方式により権限委譲の旅の順番の最適化が可能であるという階層性に強い2つの利点をクラスタ・クラスタ上に展開することが今後の最大の課題である．

また，クラスタ・クラスタなどでは本稿で提案しているSDSMシステムのコンテンションの動的検出手法をページリクエストにも適用することで性能向上が期待できる．

参 考 文 献

1) Hu, W., Shi, W. and Tang, Z.: JIAJIA: A Software DSM System Based on a New Cache Coherence Protocol, *HPCN Europe*, pp.463–472 (1999).
2) 原田 浩, 石川 裕, 堀 敦史, 手塚宏史, 住元真司, 高橋俊行：ソフトウェア分散共有メモリ SCASH におけるページ管理ノードの動的再配置機構の実装と評価, 情報処理学会研究報告, Vol.1999, No.77, pp.89–94 (1999).
3) Keleher, P., Dwarkadas, S., Cox, A.L. and Zwaenepoel, W.: TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems, *Proc. Winter 1994 USENIX Conference*, pp.115–131 (1994).
4) Keleher, P.: The Relative Importance of Concurrent Writers and Weak Consistency Models, *Proc. 16th Int'l Conf. on Distributed Computing Systems* (*ICDCS-16*), pp.91–98 (1996).
5) Zhou, Y., Iftode, L. and Li, K.: Performance Evaluation of Two Home-Based Lazy Release Consistency Protocols for Shared Memory Virtual Memory Systems, *Proc. 2nd Symp. on Operating Systems Design and Implementation* (*OSDI'96*), pp.75–88 (1996).
6) Iftode, L., Singh, J.P. and Li, K.: Understanding Application Performance on Shared Virtual Memory Systems, *Proc. 23rd Annual Int'l Symp. on Computer Architecture* (*ISCA'96*), pp.122–133 (1996).
7) Cox, A.L., de Lara, E., Hu, C. and Zwaenepoel, W.: A performance comparison of homeless and home-based lazy release consistency protocols in software shared memory, *Proc. 5th High Performance Computer Architecture Conference*, pp.279–283 (1999).
8) Hirayama, H., Honda, H. and Yuba, T.: Scalable data mining with log based consistency DSM for high performance distributed computing, *Proc. 6th IEEE Int. Conf. on Engineering of Complex Computer Systems*, pp.143–150 (2000).
9) Weber, W.-D. and Gupta, A.: Analysis of cache invalidation patterns in multiprocessors, *Proc. 3rd International Conference on Architectural Support for Programming Languages and Systems* (*ASPLOS III*) (1989).
10) Iftode, L., Singh, J.P. and Li, K.: Scope Consistency: A Bridge between Release Consistency and Entry Consistency, *Proc. 8th ACM Annual Symp. on Parallel Algorithms and Architectures* (*SPAA'96*), pp.277–287 (1996).
11) Cheung, B.W.-L.: A Migrating-Home Protocol for Implementing Scope Consistency Model on a Cluster of Workstations.
12) Amza, C., Cox, A.L., Dwarkadas, S., Jin, L.-J., Rajamani, K. and Zwaenepoel, W.: Adaptive Protocols for Software Distributed Shared Memory, *Proc. IEEE, Special Issue on Distributed Shared Memory*, Vol.87, No.3, pp.467–475 (1999).
13) 城田祐介, 吉瀬謙二, 本多弘樹, 弓場敏嗣：プログラマの意図により複数のキャッシュコヒーレンスプロトコルの利用を可能とするソフトウェア分散共有メモリ, 情報処理学会研究報告, Vol.2001, No.144, pp.7–12 (2001).
14) 城田祐介, 吉瀬謙二, 本多弘樹, 弓場敏嗣：Migratory Access を対象とするホームベース分散共有メモリ, 並列処理シンポジウム JSPP2002 論文集, Vol.2002, No.8, pp.119–126 (2002).

(平成 14 年 6 月 7 日受付)
(平成 14 年 10 月 9 日採録)

城田 祐介

2001 年電気通信大学情報工学科卒業．同年より同大学大学院情報システム学研究科修士課程在学中．分散メモリシステムにおける並列処理に関する研究に従事．

吉瀬 謙二（正会員）

1995 年名古屋大学工学部電子工学科卒業．2000 年東京大学大学院情報工学専攻博士課程修了．工学博士．同年，電気通信大学大学院情報システム学研究科助手．計算機アーキテクチャの研究に従事．

本多 弘樹（正会員）

1984 年早稲田大学理工学部電気工学科卒業．1991 年同大学大学院理工学研究科博士課程修了．1987 年より同大学情報科学研究教育センター助手．1991 年より山梨大学工学部電子情報工学科専任講師．1992 年より同助教授．1997 年より電気通信大学大学院情報システム学研究科助教授．並列処理方式，並列化コンパイラ，並列計算機アーキテクチャ，グリッド等の研究に従事．工学博士．電子情報通信学会，IEEE-CS，ACM 各会員．

弓場 敏嗣（正会員）

1966 年神戸大学大学院工学研究科修士課程修了.（株）野村総合研究所を経て，1967 年通商産業省工業技術院電気試験所（現，産業技術総合研究所）に入所．以来，計算機のオペレーティングシステム，見出し探索アルゴリズム，データベースマシン，データ駆動型並列計算機等の研究に従事．その間，知能システム部長，情報アーキテクチャ部長等を歴任．1993 年より電気通信大学大学院情報システム学研究科教授．並列処理の科学技術一般に興味を持つ．工学博士．電子情報通信学会，日本ソフトウェア科学会，日本ロボット学会，ACM，IEEE-CS 各会員.