ソフトウェア分散共有メモリにおけるアクセス履歴に基づくホーム移動

岡本秀輔[†] 阿部拓弥[†,☆]

ソフトウェア分散共有メモリシステム構築のための，ホーム移動の一手法の提案と，その基本性能評価について述べる．提案手法は，ホームノードにおいて収集した，ページに対する read/write のアクセス履歴を，ホーム移動の判定に用いる．ホームノードは，他の計算ノードからのページ要求や，ページへの変更データを受け取るたびに履歴を更新し，そのエントリ数が閾値を超えたところで，履歴を用いたホーム移動の判定を行う．本論では，提案手法の特徴，および既存の手法との違いを明らかにし，既存の手法を含めた基本的な性能評価について述べる．性能評価には，8 台構成の PC クラスタと，SPLASH-2 ベンチマークのいくつかを用いた．その結果，提案手法は，他の手法と同等以上の効果があり，ホームを固定した場合と比べて，最大で 2.25 倍の高速化を確認した．

A Moving Home Method with Access Record in Home-based Software DSM System

Shusuke Okamoto[†] and Takuya Abe[†,☆]

This paper describes a new moving home method for home-based software Distributed Shared Memory systems and its evaluations. In our system, a home node uses records for access to the shared page to determine the moving its role of home. Each time a home node receives a service request for the shared page from a non-home node, it records the node ID and the type of access. If the number of the access records exceeds the threshold of request counts, the home node decides whether the non-home node which sends the request is suitable for a new home node of the page. To evaluate our moving method, we have implemented a prototype of DSM system from scratch. The result of our performance evaluation using SPLASH2 benchmark programs shows that our method improved the performance of the system by 2.25 times over the fixed home node method.

1. はじめに

ソフトウェア分散共有メモリ（以下ソフトウェア DSM）システムを構築する方法の 1 つとして，ホームノードを設ける方法[1)~5)] がある．計算ノード間で共有するメモリ領域を仮想記憶のページ単位に分割し，それぞれの最新状態を管理する計算ノードをあらかじめ決めておく方法である．ページ X の管理を行う計算ノードをページ X のホームノード，それ以外の計算ノードを非ホームノードと呼ぶことにすると，ホームノードを用いたソフトウェア DSM システムの一般的な特徴は次のように説明できる．ホームノードはページ X の原本と見なせるデータを保持し，非ホームノードは，実行状況により，ページ X の複製を持つことができる．非ホームノードが複製にアクセスする際には，ホームノードと通信を行うことで，ホームノード上にあるページ X の原本との一貫性を保つ．

具体的な例として，Unix OS 上で write-invalidate プロトコルと multiple-writer プロトコル[6)]，ロックベースの ScC（Scope Consistency）モデル[7)] を採用するソフトウェア DSM システムについて考える．このシステムでは，次のような通信がホームノードと非ホームノードとの間でなされる．

クリティカル・セクションの初めでロックを取得した非ホームノードは，そのロックによって定義されるスコープ情報を用いて，自分の持つページ X の複製が有効であるかを判断する．無効と判断した場合には，そのページの保護設定をアクセス不可にする．その後，その計算ノードが複製にアクセスすると，SIGSEGV

† 茨城大学大学院理工学研究科情報工学専攻
Graduate School of Science and Engineering, Ibaraki University
☆ 現在，NEC ソフト株式会社
Presently with NEC Soft, Ltd.

ハンドラに処理が移る．ハンドラはホームノードに対して，複製を最新状態にするためのデータを要求し，ホームノードから受け取ったデータを用いて，複製を最新状態に更新する．そして，**twin**[6]と呼ぶ別のページXの複製を作成し，元の複製の保護設定をアクセス不可からread/write可にして，ハンドラの処理を終える．クリティカル・セクションの終わりでは，変更を加えた複製とtwinから，**diff**[6]と呼ぶ，クリティカル・セクション中のページXに対する変更差分を切り出し，それをホームノードに送る．ホームノードでは，送られてきたdiffをもとに，ページXの原本を更新する．非ホームノードでは，diffを送ったすべてのホームノードの更新を確認し，最初に取得したロックのスコープを更新する．そしてロックを開放してクリティカル・セクションを終える．

このように，非ホームノードによるページへのアクセスでは，複製を最新状態にするデータの要求（以後，update要求と呼ぶ）と，その要求に応答として送られる更新データ（以後，update応答と呼ぶ），そしてdiffが，ホームノードと非ホームノード間で送られる．一方で，ホームノードが自分の管理するページにアクセスする場合には，ローカルメモリに直接アクセスするだけでよい．そのため，ホームノードを設けるソフトウェアDSMシステムでは，ホームノードをどの計算ノードとするかが全体性能を決める重要な点となる．

ホームノードの割当てには，ユーザやコンパイラが行う静的な方法と，ホーム移動と呼ばれる，システムが実行時に割当てを変更する動的な方法がある．並列アプリケーションのメモリアクセスの性質を，ユーザなどが完全に把握でき，静的な割当てをうまく行えるならば，動的な方法は必要がない．しかし，メモリアクセスの性質が実行状況によって変化する場合や，共有メモリコンピュータ用の並列アプリケーションを，なるべく手を加えずに動作させようと考える場合には，動的な方法が有用となる．

本論ではホーム移動の一手法を提案する．以下では，提案手法の特徴，既存の手法との違い，および既存の手法との比較を含めた提案手法の性能評価を述べる．これにより提案手法の有効性を示す．

2. アクセス履歴に基づくホーム移動

この章では，我々の提案するアクセス履歴に基づくホーム移動[9),10)]の手法について述べる．

2.1 概　　要

アクセス履歴に基づくホーム移動の手法では，ページに対するread/writeのアクセス履歴をホームノードにおいて収集し，それを用いてホーム移動の判定を行う．履歴としてread/writeの別と，それを行った計算ノードを記録する．ホームノードは，update要求やdiffを受け取った時点で履歴を更新し，後述の条件の下で移動判定を行う．移動判定では，update要求を送った計算ノードに，ホームを移動するか否かを判断する．また，ホーム移動の際には，履歴を新ホームノードに引き継ぐことで，連続的な情報を判定に用いる．

2.2 read/writeの検出

非ホームノードによるページへのread/writeアクセスは，ホームノードが受け取ったupdate要求とdiffから判断する．ホームノードがdiffを受け取ったときには，送り主によるページへのwriteとして履歴を残す．これに対し，update要求を受け取ったときには，readまたはwriteに分けて履歴を残す．これは，無効な複製を持つ非ホームノードのアクセスでは，read/writeの別にかかわらずupdate要求を送るためである．つまり，update要求はread用とwrite用の2種類があり，非ホームノードにread/writeの別を指定させることで，それぞれに区別した履歴を残す．

一方，ホームノードによるreadアクセスは，ホームノードがdiffを受け取った後に，ページの保護設定を一時的にアクセス不可とすることで検出する．また，ホームノードによるwriteは，部分ページ更新[5)]という手法の一部として検出する．

部分ページ更新とはupdate応答の転送量を減らす手法である．ページの複製を更新するためのupdate応答には，ページ全体を用いる方法が最も簡単である．しかし，これは通信量を増大させる．そこでホームノードにおいて，計算ノードごとに送るべきページの範囲を管理し，毎回ページ全体を送らないようにする．送るべき範囲を特定するために，ホームノードによる当該ページへの変更も調べる．これは非ホームノードにおけるdiffの作成と類似の処理で行う．結果として，ホームノードにおける処理は増加するが，通信量の削減により，全体性能は向上する．

提案手法でもこの部分ページ更新の手法を採用し，同時にホームノードによるwriteを検出する．なお，ホーム移動時には，部分ページ更新のための情報も，履歴とともに新たなホームノードに引き継ぐ．

2.3 ユーザ指定のパラメータ

ユーザ指定のパラメータにはWSZ（Window SiZe）と$MINT$（Move INTerval）がある．WSZは移動判定に用いる履歴の個数で，最新のWSZ個の履歴を判定に用いることを指定する．移動判定には古い履

歴よりも新しい履歴が重要であるが，その境目はアプリケーションごとに異なるため，これをユーザが指定する．一方 $MINT$ は，ホーム移動の判定を開始するまでに，ホームノードが収集すべき履歴の個数である．ホーム移動後の新ホームノードでも，判定の再開までに同数の履歴を収集する．これによりホーム移動が頻繁に起こることをユーザが抑制する．

2.4 移動判定条件

非ホームノード N からの update 要求により，履歴を登録したとき，そのホームノードで新たに収集した履歴が $MINT$ 個以上である場合には移動判定を行う．このとき有効な履歴の個数 x は $MINT \leq x \leq WSZ$ であり，これらを調べて，次のどちらかの条件を満たすならば計算ノード N にホームを移動する．

- update 要求が write 用の場合：ホームノードがページを変更中でなく，かつ，履歴中の計算ノード N の write の数が $MINT$ 個以上である．
- update 要求が read 用の場合：履歴中にホームノードの read/write はなく，かつ，計算ノード N の read の数が履歴中で最大である．

2.5 ホーム移動の例

図 1 に移動の例を示す．計算ノード A，B，C があり，ホームノードは初めの時点で A，WSZ は 3，$MINT$ は 2 とする．履歴はサイズ 3 のキューとして表現している．細い斜めの矢印が計算ノード間の通信，太矢印がホーム移動の変遷を示す．時刻は図の上から下に向かって進む．計算ノード B，C，A，C の順にアクセスし，それぞれ，計算ノードの識別子と read/write の別が履歴となる．最初の B による read アクセスでは，履歴の個数が $MINT$ 未満なので移動判定を行わない．次の C のアクセスでは移動判定を行うが，C による write の回数が $MINT$ 未満なので移動の条件を満たさない．ホームノードによる write アクセスの後，2 回目の C による write アクセスの時点で B の履歴を破棄し，C の write を履歴に加えたうえで，再度 C に対する移動判定を行う．ここでは移動条件が成立し，C にホーム移動を行う．同時に履歴を C に送る．

2.6 ホーム移動の通知

ホーム移動によって，新ホームノードがどの計算ノードとなったかを他に知らせる方法は，ホーム移動の手法では重要な要素である．その方法は一般的に 3 つあるとされる[4]．それらはブロードキャスト，集中管理，リンクリストを用いる方法である．リンクリストを用いる方法とは，非ホームノードが旧ホームノードにたずねることで，移動の変遷をたどっていくものである．

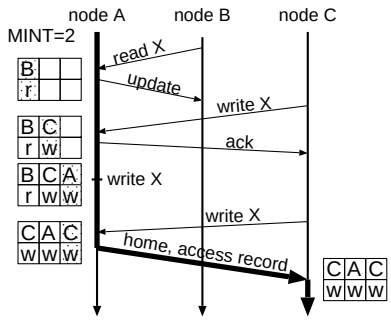

図 1 アクセス履歴に基づくホーム移動の例
Fig. 1 Example of moving home with access record.

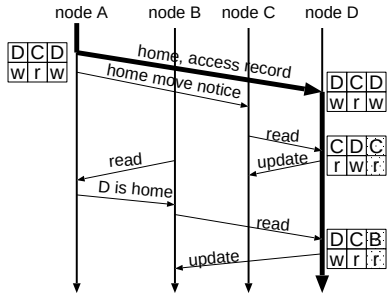

図 2 マルチキャストとリンクリストによるホーム移動通知
Fig. 2 Home move notice with multicast and linked-list.

提案手法では，アクセス履歴を利用したマルチキャストとリンクリストの双方を行う．**図 2** にあるように，ホームノードは，ホーム移動の処理として，アクセス履歴に含まれる計算ノードに対し，移動通知のマルチキャストを行う．計算ノード B のような通知を受けなかった他の計算ノードは，リンクリストをたどる．

3. 関 連 研 究

この章では性能評価で用いた 3 種の既存の手法の特徴，注意すべき点，提案手法との違いを述べる．

3.1 JIAJIA のホーム移動手法

JIAJIA [1),2)] はロックベースの ScC モデル[7)] を採用したソフトウェア DSM システムである．

3.1.1 移動手法の特徴

JIAJIA におけるホーム移動の判定は，バリア同期マネジャが行い，ホーム移動を示す情報をバリア同期完了のメッセージとともに送る．ホーム移動は，前回のバリア同期から当該のバリア同期の間で，唯一の計算ノードがページを変更した場合に行う．そして，唯一変更した計算ノードが，そのページの新ホームノードとなる．複数の計算ノードがページを変更した場合には，そのページのホーム移動は行わない．

新たにホームノードとなる計算ノードのみが当該ページを変更しているので，その計算ノードの保持するページの複製は最新状態にある．そのため，ホーム移動に際しては，新ホームノードで当該ページの状態を複製から原本に変更し，旧ホームノードでページの状態を原本から複製にするだけで，これらの計算ノード間では通信を行わない．また，バリア同期の時点で，すべてのノードが新ホームノードについて知るので，移動通知のための通信も行わない．

3.1.2 注意すべき点

この手法では，バリア同期の時点で多数のホーム移動がいっせいに発生する可能性がある．ホーム移動の対象となった新旧のホームノードでは，当該ページの管理情報を更新し，他のすべての計算ノードでは，新ホームノードを登録する．それぞれの計算ノードの処理量は，バリア同期の時点で保持しているページの複製や原本の数によって異なる．さらに，新ホームノードとなる計算ノードは，管理情報を更新し終えるまで，当該ページに対する update 要求や diff を処理できない．したがって，状況によっては，バリア同期後の処理の再開が計算ノードによりまちまちとなり，それが性能低下の要因となりうる．

性能に関係する他の注意すべき点は，update 応答にある．新旧のホームノード間で通信を行わないため，前述の部分ページ更新を導入しても，新ホームノードでページ範囲の情報を得ることができない．その結果，ホーム移動後の update 応答に，一度はページ全体を含めなければならない．ページ範囲の情報を新ホームノードに送るようにすると，多数のホーム移動が発生したときにその通信が衝突し，性能低下につながる．

3.1.3 提案手法との相違

この手法はバリア同期の時点にホーム移動が限定され，唯一ページを変更した計算ノードのみが新ホームノードとなるため，提案手法とは移動のタイミングと条件で大きく異なる．また，移動処理が集中する可能性は提案手法よりもこの手法の方が大きい．

3.2 JUMP のホーム移動手法

JUMP [3] は，ホーム移動を行わない初期の JIAJIA システムに対し，別の作者がホーム移動の手法を加えたソフトウェア DSM システムである．

3.2.1 移動手法の特徴

JUMP の手法では，ホームノードの移動判定によって，update 要求を送った計算ノードが，可能な限り新たなホームノードとなる．

JUMP のホーム移動の特徴的な点は，false sharing の対処にある．**図 3** は対処の例である．ページ X のホーム移動の後に，旧ホームノードとなった計算ノード PREV は，新ホームノード NEW からの移動通知を受け取るまで，OTHER のような他の計算ノードからの update 要求に応える．このとき PREV は，すでにページ X のホームノードではないため，新ホームノードが NEW であるという情報とともに update 応答を送る．これによりページ X 用の diff が NEW に正しく送られる．NEW と OTHER が同時にページ X を変更でき，false sharing に対処する．

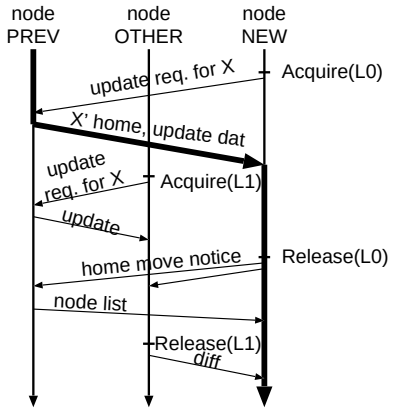

図 3 JUMP のホーム移動
Fig. 3 Moving home on JUMP system.

ページ X の新ホームノード NEW は，ロックを開放する際に移動通知をブロードキャストする．移動通知を受け取った旧ホームノード PREV は，ホーム移動後に update 応答を送った計算ノードをリストにして NEW に送る．NEW は，このリスト中のすべての計算ノードから diff を受け取るまで，次のホーム移動を行わない．すべての diff を受け取った後は，update 要求の受け取りによってホーム移動を行う．

3.2.2 注意すべき点

文献 3) には明記されていないが，JUMP では update 要求を 2 種類に区別している．非ホームノードが read の時点で update 要求を旧ホームノードに送った場合には，後にページを変更せずに diff を送らないことがある．diff が新ホームノードに届かない場合には，そのページの以後のホーム移動はいっさい行われない．したがって，旧ホームノードは，read のために update 要求を送ってきた計算ノードをリストに入れないようにしなければならない．

3.2.3 提案手法との相違

update 要求を受け取ったときにホーム移動を行う点では，この手法と提案手法は同じであるが，ページへのアクセス状況を考慮せずに可能な限り移動を行う点

が異なる．この手法では，アプリケーションとの相性によって性能が極端に変わると思われる．また，頻繁にホーム移動が行われ，そのたびごとにブロードキャストがなされる点は性能低下の要因となる．

3.3 ORION のホーム移動手法

ORION[5] はマルチスレッドで実現されたソフトウェア DSM システムで，Eager RC（Release Consistency）モデルを採用する．ページの更新方法に write-update と write-invalidate の双方のプロトコルを切り替えて用い，ホーム移動は当該ページを write-invalidate プロトコルで管理している場合にのみ行う．

3.3.1 移動手法の特徴

ORION の手法では，ホームノードによるページへのアクセスがない間に，そのページを頻繁にアクセスする非ホームノードが現れた場合に，ホーム移動を行う．その状況を調べるために RUC（Remote update count）と PAC（Page access count）と呼ぶ 2 つの配列を管理する．RUC 配列の k 番目の要素の値は，ホームノードがアクセスするまでに，非ホームノード k が diff を送った回数を示す．ホームノードのアクセスによってすべての要素の値を 0 とする．一方，PAC 配列では，非ホームノード k が update 要求を送った回数を $PAC_{[k]}$ で示し，ホームノードが diff を受け取った後に，当該ページにアクセスしたか否かを $PAC_{[HOME]}$ の値として示す．PAC 配列を用いて移動判定を行った場合には，PAC 配列のすべての要素の値を 0 にする．

ホーム移動の判定は，非ホームノード k から diff または update 要求を受け取ったときに，対応する配列を更新したうえで行い，その非ホームノード k にホーム移動を行うかどうかを判定する．

diff の受け取り時には毎回移動判定を行い，閾値 $HMTL$（home migrating triggering limit）を用いて，$RUC_{[k]} > HMTL$ を満足したときに移動を行う．これはホームノードのアクセスがなく，非ホームノードが一定回数以上ページを更新している状況でのホーム移動となる．一方，update 要求の受け取り時には，PAC 配列の合計が閾値 $PRSP$（page request sampling period）以上のときに判定を行い，移動の条件は $PAC_{[HOME]} = 0 \wedge PAC_{[k]} > HMTL$ である．これは，ホームノードが diff を他に転送するだけのバッファとして動作している状況でのホーム移動となる．

ホーム移動のコストは，update 応答または diff への応答メッセージに付加される．ホーム移動後の通知方法は文献 5) に示されていない．

3.3.2 注意すべき点

PAC 配列はホームノードのアクセスによって 0 クリアしないが，これを使った判定条件に $PAC_{[HOME]} = 0$ があるので，事実上ホームノードのアクセスでクリアしているのと同じである．両配列ともホームノードのアクセスによって一度無効となることから，$HMTL$ の値を大きくすると移動条件をほとんど満たさなくなり，$PRSP$ の値を大きくすると update 要求を受け取ったときの移動条件を満足しなくなる．

3.3.3 提案手法との相違

ORION の手法は，update 要求と diff を受け取った数を利用する点で，提案手法と類似する．しかし，この手法では，ホームノードのアクセスによって 2 つの配列が事実上無効となる．結果として，ホームノードの割当てが望ましくない状態のときに，一度のホームノードのアクセスによって，移動が起こらない可能性がある．また，配列を長い期間の履歴情報として残すことも不可能である．

これに対して提案手法では，ユーザの指定する個数だけの履歴を保持し，履歴の収集でホームノードが特別扱いとなることはない．結果として，ホームノードの割当てが悪い場合にも，ホームノードのアクセスを優先しすぎることはない．また，ホーム移動の際に履歴を引き継ぐので，長い期間の履歴を残すことも可能である．他の相違点としては，提案手法では，update 要求を 2 つに分けて，ページに対する read/write のアクセスをより強調している点があげられる．

4. 性能評価

既存のホーム移動の手法とアクセス履歴に基づくホーム移動の手法の性能を調べるために，Linux カーネル用のソフトウェア DSM システムを構築した．このシステムでは，1 章で述べたプロトコルを SIGIO と SIGSEGV ハンドラにより実現し，通信には UDP/IP を用いている．このベースシステムに提案および既存のホーム移動の手法を実装した．ORION の移動通知の方法は不明なので，ホームノードが移動を決定した時点で，$RUC_{[k]}$ の値が非零の計算ノード k に移動通知を送り，他の計算ノードはリンクリスト方式で新ホームノードをたどるようにした．

4.1 評価の環境と設定

評価環境には 8 台の PC クラスタを用いた．各々は Intel Celeron 1.2 GHz，128 MB RAM，Gigabit Ethernet（PCI 32 bit）を備える．Linux カーネル 2.4.18 を用いて，各 PC で 1 プロセスを実行させた．

SPLASH-2 ベンチマークプログラム[8] から，いく

表 1 ベンチマークプログラムの特性と設定値
Table 1 Characteristics of benchmarks and parameters.

Code	Problem Size	Sequential time(sec)	swap	Mem (MB)	Barrs	Locks	FIXED msg count	FIXED msg amt(GB)	AccessRecord MINT	AccessRecord WSZ
LU	2048^2	25	no	32	70	0	25129	0.67	2	8
LU	4096^2	287	yes	128	134	0	95315	4.70	2	8
Ocean	514^2	26	no	57	1692	742	409997	4.39	1	8
Ocean	1026^2	2226	yes	224	1808	798	1097062	16.41	1	8
Water-Nsq	4096 mole.	601	no	5	231	554	241992	0.98	4	9
Water-Nsq	6859 mole.	1889	no	8	231	554	401000	1.58	5	9

つかのプログラムを選び，ソフトウェア DSM 用に移植した．**表 1** にベンチマークの特性と設定値を示す．表 1 で Sequential の time は逐次プログラムの実行時間を示し，swap はメモリスワップの有無を示す．Mem，Barrs，Locks は，並列実行における共有メモリ領域のサイズ，バリア同期の回数，ロック操作の回数をそれぞれ示す．これらはホーム移動の手法に左右されない値である．これらから選んだベンチマークの特徴が，バリア同期のみのもの（LU），バリア同期がロック操作よりも多いもの（Ocean），ロック操作がバリア同期より多いもの（Water-Nsq）と分かる．

Water のプログラムは，手動により，各計算ノードのローカルアクセスが増えるようにプログラムを調整し，必要なロックの数も計算ノード数にあわせた．したがって，ホーム移動なしで当初から高い性能を示す．なお，ホームノードの初期割当てはページごとにラウンドロビン方式で行った．

表 1 の FIXED msg は，ホーム固定の並列実行の特性を示す．count は，全計算ノードが行った diff と update 応答の転送回数の総計を示し，amt は，対応する総転送量をギガバイト単位で示したものである．

アクセス履歴の手法では，$MINT$ と WSZ のパラメータにより移動判定の条件を調節できる．これらのパラメータは試行錯誤の結果，表 1 の最右列にある値を用いた．LU と Ocean のベンチマークでは $MINT$ の値が性能に対して支配的であり，WSZ の値は PC の数より増やしても性能は大きく変化しなかった．Water においては，ほぼ $MINT >= WSZ/2$ ならば，表 1 の値より大きな値でも性能は変わらなかった．手動のホームノードの割当てにより，これより性能を向上させるのは難しく，$MINT$ の値で頻繁なホーム移動をおさえるだけとなっていたと考えられる．

また，ORIONに対しては，$HMTL = 1$, $PRSP = 8$ がどのベンチマークでもつねに良く，このパラメータでの結果を以下の比較に用いる．ORION の LU と Ocean では，$HMTL$ の値が性能に対して支配的であるが，$HMTL = 1$ という小さな値は 3.3.2 項で述べた ORION のパラメータの特徴を示している．

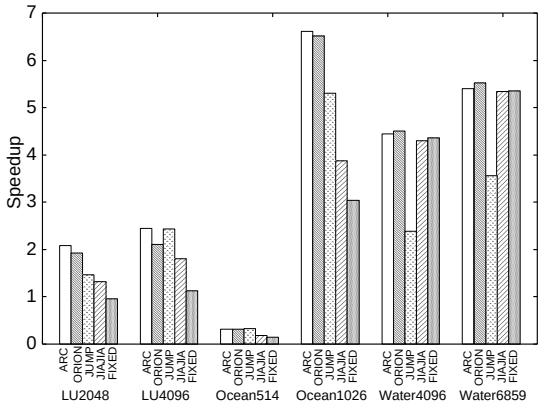

図 4 8 プロセッサにおける逐次実行に対するスピードアップ
Fig. 4 Speedup on 8 processors against sequentials.

4.2 評価項目

以下では，まず，逐次実行に対するスピードアップの比較を示すことで，提案手法の相対的な性能を示す．そして，スピードアップ値の違いの原因を探るために，ホーム移動の回数，データ転送回数，データ転送量を比較する．最後に，JUMP の移動通知の方法を変更し，ブロードキャストを行わずに旧ホームノードのみに通知した場合の性能を調べる．ORION 以外に JUMP の移動通知の方法も提案手法に近づけることで，ホーム移動の判定における履歴の扱いの違いが，どのように性能に影響を与えるかを明らかにする．

4.3 結果と考察

以下の図において，ARC（Access ReCord）はアクセス履歴の手法を，ORION，JUMP，JIAJIA は同名のシステムの手法を用いた場合，FIXED はホーム固定の場合の値をそれぞれ示す．

4.3.1 スピードアップの比較

図 4 は，表 1 の逐次プログラムの実行時間に対する並列実行の時間の比をスピードアップとして示す．実行時間はベンチマーク中に元々指定されている方法で計測し，データの初期化の時間を含まない．

ARC のスピードアップ値は，すべての場合で最高

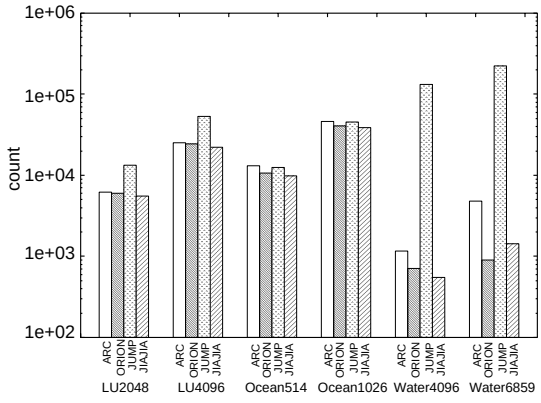

図 5 ホーム移動の回数
Fig. 5 Number of moving home.

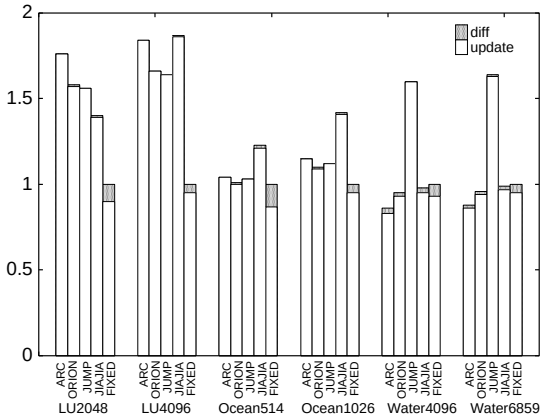

図 6 ホーム固定に対するデータ転送回数の比
Fig. 6 Rate of message count (FIXED=1).

値ではないものの，全体として上位の結果となっている．FIXED に対する ARC のスピードアップ値は，1.01〜2.25 である．ORION の手法も全般的に良いが，LU4096 の結果だけはあまり良くない．JUMP は Water の結果が極端に悪い．JIAJIA は FIXED よりも良いが，全般的に FIXED よりも大きく向上していない．Water における FIXED の結果から，プログラムの調整によって，初期のホームノードの割当てが良かったことが分かる．

4.3.2 ホーム移動回数の比較

図 5 に，ホーム移動の回数を示す．ARC と ORION を比べると，LU ではわずかだが，どのベンチマークでも ARC の移動回数が多い．これは ORION で配列を頻繁に 0 クリアすることで，移動が起こり難いという特徴が現れていると思われる．ただし，この特徴は Water では有効に働いており，移動回数が少ない ORION が ARC よりも性能が高い．また JUMP では，逆に，多数の移動回数で性能低下となっている．

4.3.3 転送回数と転送量の比較

図 6 は，各手法における update 要求と diff の転送回数を，表 1 の FIXED msg count の値に対する比で示す．**図 7** は，update 応答，diff，ホーム移動の転送量の合計を，表 1 の FIXED msg amt の値に対する比で示す．これらは初期化の処理を含めた値である．JIAJIA 以外では，ホーム移動が update 応答または diff の応答とともに行われるので，図 6 にホーム移動の回数を含めていない．

すべてにおいて diff の転送回数は少ない．LU と Ocean において，FIXED の diff の転送量の割合が大きいことから，少ない回数で大量のデータを転送していたことが分かる．ARC，ORION，JIAJIA の FIXED に対する性能向上は，diff の転送量の削減に

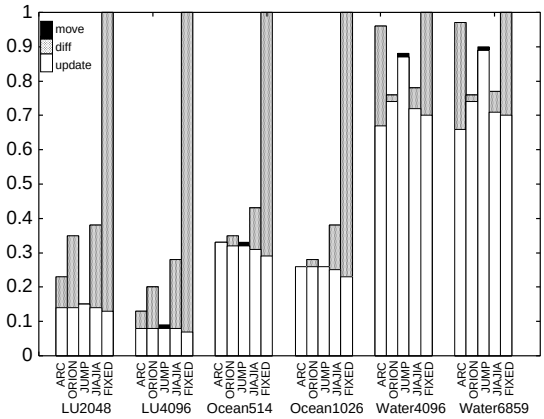

図 7 ホーム固定に対するデータ転送量の比
Fig. 7 Rate of data transfered (FIXED=1).

関係しているといえる．JUMP も同様に diff の転送量を削減しているが，ARC や ORION ほど性能が向上していない．これは後に述べる移動通知との関係にある．

Water では，ORION と JIAJIA が diff の転送量を削減し，ARC が update 応答の転送量を削減している．ORION が最高性能を示しているが FIXED との性能差は小さい．JUMP は diff の転送量を最小に削減しているが，逆に update 応答の転送量を増やして性能が低下している．したがって，ホーム移動の手法の違いは，性能向上よりも性能低下として現れている．

4.3.4 履歴使用の有無による比較

JUMP の移動通知を旧ホームノードのみとして性能を調べた．逐次実行に対するスピードアップ値は，LU と Ocean514 では変化がなかったが，Ocean1026 は 6.68 と大きく向上し，Water4096 で 2.67，Water6859

で 3.78 とやや向上した．これらから移動通知のブロードキャストが衝突して，性能向上を妨げていたと考えられる．しかし，Water の 2 つに関しては改善された性能が依然として FIXED よりも低い．

4.3.5 評価結果のまとめ

JIAJIA はホーム移動の回数が少なく性能向上が少ない．JUMP は diff を削減する特徴を持つが，頻繁にホーム移動を行うため，ブロードキャストの衝突やベンチマークとの相性によって，性能が向上しないことがある．ORION は，ホームノードを ARC より優先するために，LU や Ocean のようなホーム移動を必要とするベンチマークで，移動回数が減り，diff をより削減した ARC の方が高い性能を示した．

5. おわりに

ソフトウェア DSM システムにおけるホーム移動の手法の 1 つとして，アクセス履歴に基づく方法を提案した．提案手法は，update 要求と diff の受け取りによって，ページへの read/write のアクセス履歴を作り，それを移動判定に用いる．そして，ホーム移動後に履歴を引き継ぐことで連続的な情報を保つ．

update 要求と diff の受け取り回数を利用する既存の手法には，ORION がある．この手法は，ホームノードのアクセスのたびに回数情報をクリアするため，ホームノードの割当てが悪い場合に，移動が起こり難い．また，長い期間の履歴情報を残すことができない．

性能評価においては，JIAJIA および JUMP といった履歴と無関係な手法よりも提案手法の方が良い性能を示した．ORION の手法に対しては，ベンチマークにより優劣が分かれたが，ホーム固定のプログラムの実行性能が悪い LU と Ocean において，提案手法が ORION の手法よりも良く，ホーム固定の場合に比べて最大で 2.25 倍の性能となった．ホーム移動が必要とされるベンチマークにおいて性能の優位さを示したことで，提案するアクセス履歴の手法に有効性があると思われる．しかし，今回の性能評価は包括的なものとはなっていない．今後，詳細な性能評価を行うことで，提案手法の有効性をより正確に把握する．

参考文献

1) Hu, W., Shi, W. and Tang, Z.: JIAJIA: An Software DSM System Based on A New Cache Coherence Protocol, *Proc. High Performance Computing and Networking* (*HPCN'99*), LNCS 1593, pp.463–472, Springer (1999).
2) Hu, W., Shi, W. and Tang, Z.: Home Migration in Home-Based Software DSMs, *Proc. 1st Workshop on Software Distributed Shared Memory* (*WSDSM'99*) (1999).
3) Cheung, B.W., Wang, C. and Hwang, K.: A Migrating-Home Protocol for Implementing Scope Consistency Model on a Cluster of Workstations, *Proc. International Conference on Parallel and Distributed Processing Techniques and Applications* (*PDPTA'99*) (1999).
4) Chung, J.W., Seong, B.H., Park, K.H. and Park, D.: Moving Home-based Lazy Release Consistency for Shared Virtual Memory Systems, *Proc. 1999 International Conference on Parallel Processing* (*ICPP'99*) (1999).
5) Ng, M.C. and Wong, W.F.: ORION: An Adaptive Home-based Software Distributed Shared Memory System, *Proc.7th International Conference on Parallel and Distributed Systems* (*ICPADS'00*) (2000).
6) Carter, J.B., Bennett, J.K. and Zwaenepoel, W.: Techniques for Reducing Consistency-Related Communication in Distributed Shared Memory Systems, *ACM Trans. Comput. Syst.*, Vol.13, No.3, pp.205–243 (1995).
7) Iftode, L., Singh, J.P. and Li, K.: Scope Consistency: A Bridge between Release Consistency and Entry Consistency, *Proc. 8th Annual ACM Symposium on Parallel Algorithms and Architectures* (1996).
8) Woo, S.C., Ohara, M., Torrie, E., Singh, J.P. and Gupta, A.: The SPLASH-2 Programs: Characterization and Methodological Considerations, *Proc. 22nd Annual International Symposium on Computer Architecture* (*ISCA'95*), pp.24–36 (1995).
9) 阿部拓弥，岡本秀輔：ソフトウエア DSM におけるホーム移動に関する一考察，情報科学技術フォーラム（FIT2002）講演論文集，pp.167–168 (2002).
10) Abe, T. and Okamoto, S.: A Moving Home-based Software DSM System, *IEEE Pacific Rim Conference on Communications, Computers and Signal Processing* (*PACRIM'03*), pp.17–20 (2003).

(平成 15 年 10 月 10 日受付)
(平成 16 年 1 月 30 日採録)

岡本　秀輔（正会員）

昭和 40 年生．平成 6 年成蹊大学大学院博士後期課程修了．平成 6 年電気通信大学大学院助手．平成 9 年同大学大学院講師．平成 11 年茨城大学工学部，同大学大学院講師．並列処理プログラミング，プロセッサアーキテクチャの研究に従事．電子情報通信学会，日本ソフトウェア科学会，IEEE Computer Society，ACM 各会員．

阿部　拓弥（正会員）

昭和 54 年生．平成 13 年茨城大学工学部卒業．平成 15 年同大学大学院博士前期課程修了．同年 NEC ソフト入社．