社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2006－ARC－169 (26)
2006／8／2

相乗り通信を利用したソフトウェアDSMの通信回数削減手法

坂口朋也[†] 今村昌之[†] 鈴木 祥[†]
大島聡史[†] 片桐孝洋[†]
吉瀬謙二[††] 弓場敏嗣[†]

PCクラスタにおいてソフトウェアで仮想的な共有メモリを実現するソフトウェア分散共有メモリ (S-DSM) システムの高速化について議論する．S-DSMシステムのクラスタを構成するノード間で，近い将来に転送されるページを動的に予測し，予め転送しておくことで通信オーバヘッドを削減し高速化を図る方法がある．本稿では，予測したページを予め転送しておく方式として，S-DSMシステムでやりとりされるメッセージにページデータを含ませる「相乗り通信」という転送方式を提案する．従来の転送方式のS-DSMシステムと比較して，提案手法を用いたS-DSMシステムは最大でMM(Matrix Multiply)では45.1%，SOR(Red-Black Successive Over Relaxation)で5.3%，LU(Parallel Dense-blocked LU Factorization, No Pivoting)で7.5%，IS(Integer sort)で4.8%の性能向上が得られた．

Ainori Communication: A Method to Reduce the Page Transfer of S-DSM Systems

Tomoya Sakaguchi,[†] Masayuki Imamura,[†] Sho Suzuki,[†]
Satoshi Ohshima,[†] Takahiro Katagiri,[†] Kenji Kise[†]
and Toshitsugu Yuba[†]

We discuss the method to speed up the software distributed shared memory (S-DSM) systems. By predicting a page which will be needed to be transferred in the future and prefetching it, we can speed up the S-DSM systems. In this paper, we propose a method to transfer the predicted pages together with a message used in S-DSM systems. We will call this method Ainori communication. We evaluate our implementation using four S-DSM benchmarks and show that it can decrease the number of communication and improve the performance.

1. はじめに

近年，計算機の高性能・低価格化とネットワークの高速化により，PCクラスタと呼ばれる汎用のPCを用いた安価で手軽な並列計算機環境の普及が進んでいる．

このようなPCクラスタ上で利用できる並列プログラミング環境の1つとして，仮想的な共有メモリを実現するソフトウェア分散共有メモリ (Software-Distributed Shared Memory, S-DSM) が提案，実装されている[1)2)3)4)]．

† 電気通信大学 大学院情報システム学研究科，東京都調布市
Graduate School of Information Systems,
The University of Electro-Communications,
Chofu, Tokyo
†† 東京工業大学 大学院情報理工学研究科，東京都目黒区
Graduate School of Information Science and Engineering, Tokyo Institute of Technology,
Meguro, Tokyo

S-DSMは，プログラマに仮想的な共有メモリを提供する．そのため，プログラマはクラスタを構成するPC(以後，ノードと呼ぶ) 上でのデータの配置や転送を考慮する必要がなくなり，容易な共有メモリモデルによる並列プログラミングが可能となる．

S-DSMは，分散メモリ間で一貫性を保証することにより仮想的な共有メモリを構築している．一貫性保証のためには分散メモリ間の通信をともなうデータ授受を頻繁に行う必要があるため，通信処理の大きさにより期待した性能が得られないという問題があることが知られている．そのため，通信処理のオーバヘッドを削減することで速度向上を向上させるS-DSMの通信オーバヘッド削減の研究[4)5)]が行われている．その1つとしてS-DSMの行うノード間でのデータの授受を予測し，予めそのデータを含むページを転送しておくことで通信オーバヘッドを削減し高速化を図る研究[6)7)8)]がある．

本研究では，予測したページを予め転送しておくための方式として，S-DSMシステムでやりとりされる

メッセージにページデータを含ませる相乗り通信という転送方式を提案しする．また，提案方式をS-DSMシステムの1つであるMocha[4]に実装し，性能評価を行う．

本稿の構成を示す．2章ではノード間におけるデータの授受を予測する手法と通信粒度調整機構について述べ，ページ通信回数を削減する方式を提案する．3章で提案方式をベンチマークプログラムを用いて評価する．4章で関連研究について述べ，5章で本稿をまとめる．

2. 予測に基づく通信粒度調整と相乗り通信

本章では，S-DSMシステム上で近い将来必要となるページを予測する方法について説明する．更に，相乗り通信に基づく転送方式を提案し，転送量を調整する通信粒度調整機構について述べる．

2.1 ページ番号予測

将来必要となるデータを予測し予め転送しておくことで，S-DSMの通信処理のオーバヘッドを削減する研究が行われている[6][7]．本研究では，これらの研究とは別の予測手法を用い，予測データの転送方法を工夫することで更なる速度向上を図る．

本研究では，S-DSMの1つであるMochaを研究環境として，それに適切な予測機構を設計・実装した．Mochaは8KBのページ単位で共有メモリを管理している．各ページには固有の番号が割り当てられており，それをページ番号と呼ぶ．本研究の予測機構はこのページ番号を予測する．

ページ番号の予測には，コンテキストベース値予測を用いている．コンテキストベース値予測とは，転送したページ番号とその直後に転送されるであろうページ番号との差分をまず予測し，その差分と転送したページ番号の和から直後に転送されるページ番号を予測する方法である[8]．

ここで，コンテキストベース値予測で最初に予測する差分の予測方法を説明する．プログラム実行中に，転送したページ番号とその直前に転送したページ番号との差分を常にもとめ，差分履歴テーブルに格納していく．この差分履歴テーブルの中から，最近の過去3回分の連続した差分の3つの値が連続して現れるパターンを見つけ出し，テーブルに置かれた値を差分の予測値とする．

差分の予測方法を図1を使って説明する．この図では，最近の過去3回分の連続した差分の3つの値を5，4，5とし，下の表を差分履歴テーブルとする．予測を行う際，まず最近の過去3回分の連続した差分の3つの値が差分履歴テーブルのパターンにないか探す．図の例では，差分履歴テーブル2行目のパターンが，最近の過去3回分の連続した差分の3つの値と一致している．従って，一致しているパターンの4列目の値

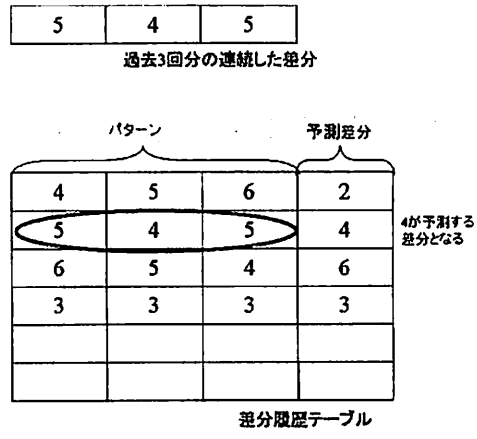

図1 コンテキストベース値予測の予測方法

が4であることから，次に転送されるページ番号は現在との差分が4であると予測される．

もし，差分履歴テーブルのパターンに最近の過去3回分の連続した差分の3つの値が存在しなかった場合は，予測を行わない．その後に発生した転送から新たな差分を求め，履歴テーブルのパターン部に3回分の連続した差分の3つの値，新たに求めた差分を予測差分として履歴テーブルに登録する．

2.2 通信粒度調整機構

MochaやJIAJIA[2]などのソフトウェア分散共有メモリの各ページには，固有のページ番号が付けられている．今回実装した機構では，ページ要求の際に，過去に要求されたページ番号のパターンを元に将来要求されるであろうページ番号を予測し，要求のあったページと共に，その予測したページ番号のページを1回にまとめて，すなわち通信粒度を大きくして転送する．この要求のあったページと共にまとめて転送したページが将来使われた場合，通信回数を削減したことになるので，その分速度向上が図れる．しかし，予測がはずれた場合には転送したページが無駄になり，転送時間を増加させてしまう．この無駄なページ転送の発生を抑えるため，予測のはずれが多い場合には予測したページの転送数を動的に減らし，予測の的中が多い場合には予測したページの転送数を動的に増やすという通信粒度調整機構を実装した．

この通信粒度調整機構は，予測して転送されたページが転送直後に使われた場合に，次に転送する予測ページ数を1増やし，転送された予測ページが転送直後に使われなかった場合に次に転送する予測ページ数を1減らすというものである．予測が的中した場合の動作例を図2に，予測がはずれた場合の動作例を図3に示す．

2.3 相乗り通信方式

S-DSMシステムでは複数種類のメッセージを使用

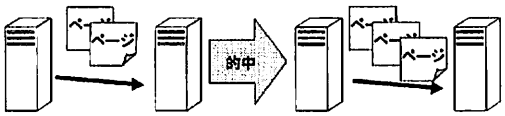

図 2 予測が的中した場合の動作例

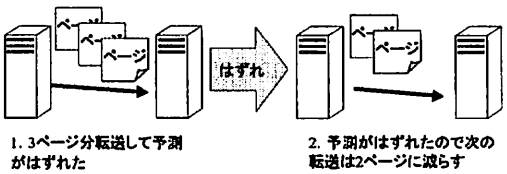

図 3 予測がはずれた場合の動作例

して，共有メモリの一貫性維持を行っている．

図 4 に S-DSM システム Mocha のメッセージの定義を示す．Mocha のメッセージはヘッダ部分とデータ部分からなる．データ部分は data という char 型の配列からなり，ここにページなどが格納される．ヘッダ部分は data 以外の変数からなる．

op はメッセージの種類，frompid はメッセージの通信元，topid はメッセージの転送先，size は data に格納されているデータのサイズ，seqno は何番目のメッセージか，temp はブロードキャストに使う変数，scope はページ要求時などに使われる変数，lock は lock-id をそれぞれ表す．本稿では，あるメッセージの data 部分に予測したページを格納することを，予測ページを含ませると表現する．

data に何も格納せずに通信を行ったり，まだ data に余分にデータを格納する余裕のあるメッセージが存在する．これらのメッセージに予測したページを含ませて 1 回の通信で複数ページ転送させることによって，通信回数を削減することができる．メッセージに予測したページを含ませることを相乗り通信と呼ぶことにする．

本研究では Mocha の 2 つのメッセージ，要求のあったページを転送するメッセージである OP_GETPGRANT と，バリアの通知を行う OP_BARRGRANT メッセージに予測ページを含ませて転送できるようにした．

OP_GETPGRANT メッセージを使用した相乗り通信をページ転送相乗り，OP_BARRGRANT メッセージを使用した相乗り通信をバリア同期相乗りと呼ぶ．

ページ転送相乗りは要求のあったページの直後に必要となるページを，要求のあったページと共にまとめて転送し，バリア同期相乗りはバリア後に必要となるページをまとめて転送する．

```
typedef struct Message{
  short op;            /* operation       */
  short frompid;       /* pid from        */
  short topid;         /* pid to          */
  short size;          /* data size       */
  unsigned int scope;  /* getpage etc.    */
  unsigned int lock;   /* lock-id         */
  unsigned int temp;   /* for broadcast   */
  unsigned int seqno;  /* sequence number */
  char data[Maxdatasize];
}
```

図 4 Mocha のメッセージ定義部

表 1 各評価アプリケーションのパラメータ

MM	行列サイズ 2048x2048，要素 double
SOR	行列サイズ M=5120，N=5120，iterations=100
LU	行列サイズ 1024x1024，blocksize32，要素 double
IS	問題サイズ LARGE(要素数 32768)

3. 評　価

本章では，S-DSM システム Mocha を用いて提案手法の性能を評価する．予測を行わない Mocha を normal，予測ページの転送方法としてページ転送相乗りのみを行う Mocha を ainori1，ページ転送相乗りとバリア同期相乗りの両方を行う Mocha を ainori2 と呼ぶ．

3.1 評価環境

評価には，16 ノードの PC をギガビットスイッチ (NETGEAR GS524T) で接続した PC クラスタを利用した．各ノードは Intel Pentium4 Xeon 2.8GHz を 2 個搭載する SMP 型の計算機で，1GB のメモリを保持する．クラスタのシステムソフトウェアとして，RedHat Linux 7.3 をベースとした SCore 5.6.1 を用いている．

本稿に示すデータは 1 台のノードに 2 つのプロセスを割り当てて（ノード当たり 2 個の CPU を用いて）測定したものである．それぞれの評価ベンチマークの実行時間は，3 回の実行時間の平均値である．

3.2 評価ベンチマーク

評価ベンチマークには，S-DSM の 1 つである JIAJIA 付属されているものを用いた．MM(Matrix Multiply)，SOR(Red-Black Successive Over Relaxation)，LU(Parallel Dense Blocked LU Factorization, No Pivoting)，IS(Integer Sort) を利用した．各評価ベンチマークのパラメータを表 1 に示す．

3.3 評価結果

動作速度を測定した結果を図 5 から図 12 に示す．図 5，図 7，図 9，図 11 は各評価アプリケーションの台

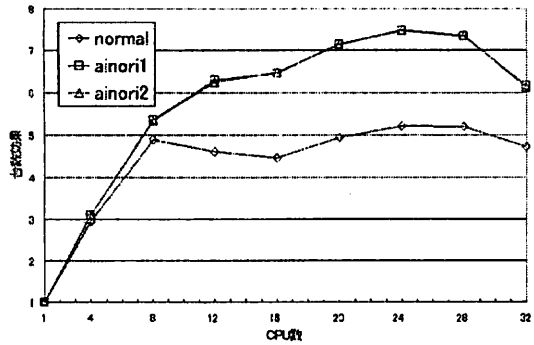

図5 MM での台数効果

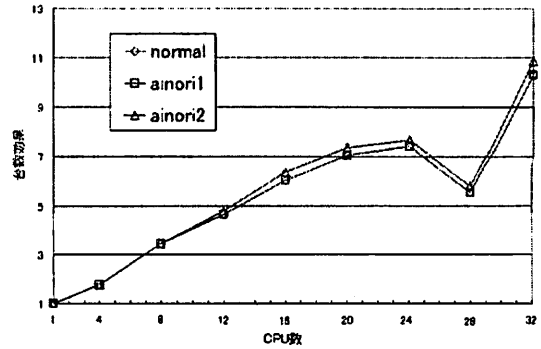

図7 SOR での台数効果

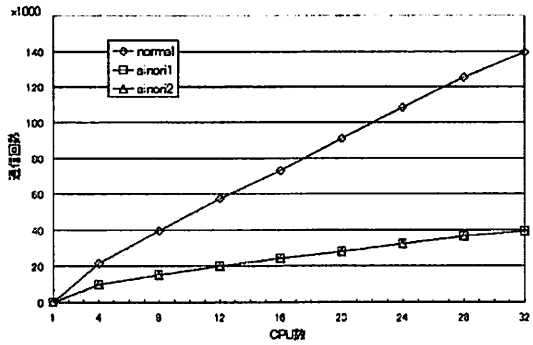

図6 MM での通信回数

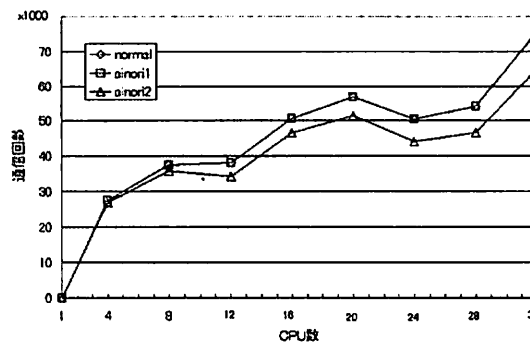

図8 SOR での通信回数

数効果を示しており，横軸には実行に用いた CPU 数，縦軸には速度向上率 (台数効果) をとっている．各グラフは CPU 数 1 での性能を 1 として正規化されている．図 6，図 8，図 10，図 12 は各評価の通信回数を示しており，横軸は実行に用いた CPU 数，縦軸はページデータの通信回数である．ページデータの通信回数はページ転送メッセージである OP_GETPGRANT とバリアメッセージである OP_BARRGRANT の総数の和である．

図 5 と図 6 の MM(Matrix Multiply) の結果について考察する．MM では，ainori1 と ainori2 の結果がほぼ一致していた．これはページデータの授受が頻繁に行われている MM の主計算部分ではバリア同期が行われないからである．ainori1 と ainori2 は normal と比べ，CPU 数 32 で 30.3%の性能向上を得た．また，CPU 数 16 では性能向上が最大となり 45.1%となった．通信回数では，ainori1 と ainori2 は normal と比べ CPU 数 32 で 71.9%の転送数の削減に成功している．

図 7 と図 8 の SOR (Red-Black Successive Over Relaxation) の結果について考察する．台数効果，通信回数の両方で ainori1 と normal はほぼ一致している．これは SOR がバリア同期を頻繁に行うので予測したページを予め転送してもバリア同期をまたいでしまうため使われず，通信粒度調整により通信粒度が 0 のまま増えないからである．この結果は，通信粒度調整で無駄な予測ページの転送を防いでいる事例といえる．ainori2 はバリア同期メッセージに含ませて転送するため，バリア同期直後に必要になるページも予め転送させておくことができる．その結果 normal と比べ性能向上を得ている．ainori2 は normal と比べ CPU 数 32 で 5.3%の性能向上を得ており，通信回数は 14.4%削減できている．SOR は，CPU 数 28 で台数効果が落ちている．CPU 数 28 のときの 1 度の通信にかかる時間が多いことが原因である．これは通信のコンテンションの影響ではないかと想定される．

図 9 と図 10 の LU(Parallel Dense Blocked LU Factorization, No Pivoting) の結果について考察する．CPU 数 16 までは normal，ainori1，ainori2 の結果はほぼ一致している．CPU 数 16 まで normal がスーパーリニアになっており，性能向上の余地がないためである．normal と比べ CPU 数 32 で ainori1 は 4.2%，ainori2 は 7.5%の性能向上を得た．CPU 数 32 で normal と比べ ainori1 は 40.0%，ainori2 は 54.5% の通

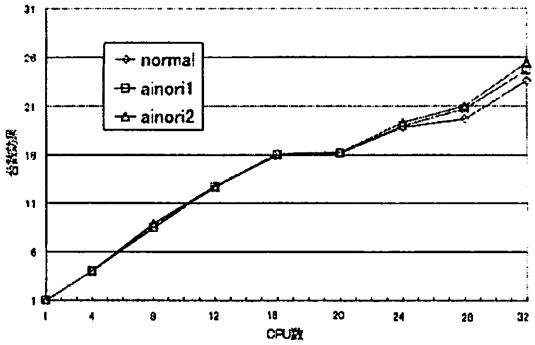

図 9 LU での台数効果

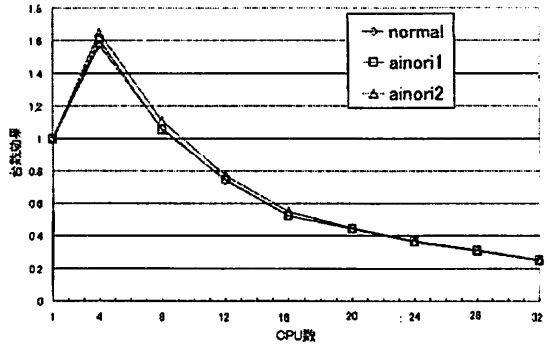

図 11 IS での台数効果

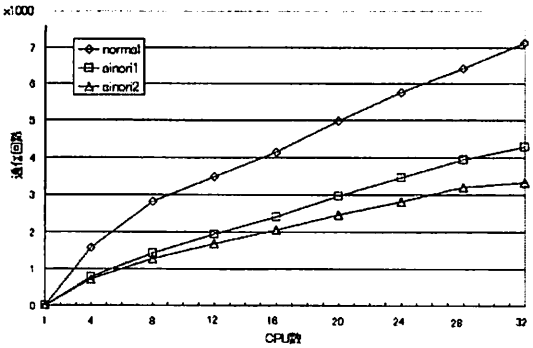

図 10 LU での通信回数

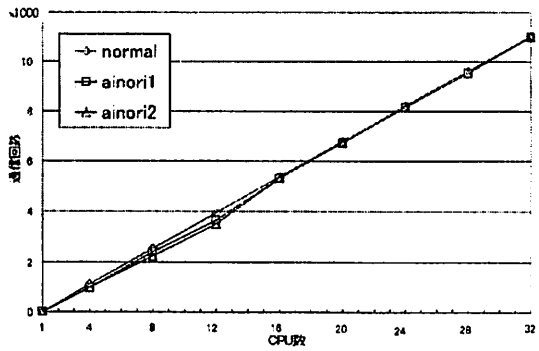

図 12 IS での通信回数

信回数の削減に成功している．

図 11 と図 12 の IS(Integer Sort) の結果について考察する．図 11 から，IS では CPU 数 4 で台数効果のピークを迎え，CPU 数 12 以上からは逐次処理より性能が低下してしまう．ainori2 は normal と比較して，CPU 数 4 のときに最大の性能向上 4.8%が得られた．図 12 から IS では適用した予測手法での通信回数削減の効果がほとんどないことがわかる．これは予測があまり的中していないことが原因である．ainori2 は normal と比較して PC 数が 4 のときに最も通信回数削減の比が高く，通信回数を 12.4%削減することに成功している．

4. 関連研究

他の予測手法を用いた S-DSM の高速化に関する研究について述べる．

田邊らの研究[6] では，連続した過去 2 回分の転送要求されたページの差分を用いて，次に転送するページを予測するストライド値予測を用いている．予測したページは，要求のあったページと共にまとめて転送している．

Kistler らの研究[7] では，予測手法にコンテキスト値予測を用いている．本研究で用いているコンテキストベース値予測はこのコンテキスト値予測を元にしている．Kistler らの手法では，本研究とは違い，送るページを予測するのではなく受け取るページを予測している．そして，1 回に複数のページ要求を行えるメッセージを用いることで，予測したページをもつノードに複数ページの一括転送を要求し，転送してもらう．

5. おわりに

本稿では，S-DSM システムの通信回数を削減し高速化するための手法について述べた．そしてその手法を S-DSM の 1 つである Mocha に実装し，評価した．評価には MM，SOR，LU，IS の 4 つのアプリケーションを用い，手を加えていない S-DSM システムと比較して，提案手法を用いた S-DSM システムは最大でそれぞれ通信回数を 71.9%，14.4%，54.5%，12.4% 削減することに成功している．そしてそれぞれ，45.1%，5.3%，7.5%，4.8%の性能向上を得た．

MM では，大幅な性能向上に成功した．ページのアクセスパターンが単純で予測しやすく通信回数を

71.9%と大幅に削減できたことと，実行時間に占める通信時間の割合が高く通信オーバヘッド削減による実行時間短縮効果が高いためである．

SORでは，通信回数の最大削減率は14.4%となった．予測ページはバリア同期相乗りでしか転送されていないことが，それほど効果が出ていない原因である．

LUでは，実行時間に占める通信時間の割合が低いので，通信回数を約50%削減してもMMと違い実行時間を少ししか減らせず，性能向上は約7%となっている．

ISでは，転送回数の最大削減率は12.4%だった．これは予測があまり的中しなかったことが原因である．

4つのベンチマークの結果から，相乗り通信は通信回数の削減手法として有効であるといえる．しかし，LUの評価結果などから，通信回数を大幅に削減しても，よい性能向上に必ずしも結びついていないことがわかる．通信回数を減らせていても，実行時間に占める通信時間の割合が高く無ければ，効果が薄いからである．このことから，実行時間に占める通信時間の割合が高く，ページ番号の予測が的中しやすいアプリケーションで相乗り通信の効果が期待できることがわかる．

今回は，ページ転送メッセージに予測ページを含ませるページ転送相乗りとバリア同期メッセージに予測ページを含ませるバリア同期相乗りをMochaに実装し，性能向上を得た．今後は，ページ転送とバリア同期以外のメッセージで相乗り通信を行えるよう実装したい．

また，今回はMM，SOR，LU，ISの4つの評価ベンチマークで評価を行った．今後は，さらに多くの評価ベンチマークで評価を行いたい．

参考文献

1) Keleher, P., Dwarkadas, S., Cox, A. L. and Zwaenepoel, W.: TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems, *Proc. of the Winter 1994 USENIX Conference*, pp.115–131 (1994).
2) Eskicioglu, M. R., Marsland, T. A., Hu, W. and Shi, W.: Evaluation of the JIAJIA Software DSM System on High Performance Computer Architectures, *Proc. of the 32nd Annual Hawaii International Conference on System Sciences(HICSS)*, Vol.8, p.8012 (1999).
3) 緑川博子，飯塚肇：ユーザレベル・ソフトウェア分散共有メモリSMSの設計と実装，情報処理学会論文誌ハイパフォーマンスコンピューティングシステム，Vol.42, No.SIG9(HPS3), pp.170–190 (2001).
4) 吉瀬謙二，田邊浩志，多忠行，片桐孝洋，本多弘樹，弓場敏嗣：S-DSMシステムにおけるページ要求時の受信通知を削減する方式，先進的計算基盤システムシンポジウムSACSIS2005論文集，pp.349–358 (2005).
5) 鈴木祥，坂口朋也，吉瀬謙二，弓場敏嗣：ソフトウェアDSM開発支援ツールを利用したアプリケーションの高速化，情報処理学会 研究報告，ARC-167, HPC-105, pp.181–186 (2006).
6) 田邊浩志，吉瀬謙二，本多弘樹，弓場敏嗣：通信粒度を動的に変更するソフトウェア分散共有メモリ，電子情報通信学会総合大会，No.D-6-5 (2002).
7) Kistler, M. and Alvisi, L.: Improving the Performance of Software Distributed Shared Memory with Speculation, *IEEE Transactions on Parallel and Distributed Systems*, pp.885–896 (2005).
8) 坂口朋也，鈴木祥，吉瀬謙二，弓場敏嗣：通信粒度予測機構を実装したソフトウェア分散共有メモリ，情報処理学会第67回全国大会，Vol.1, No.4ZB-2, pp.173–174 (2005).