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

2007－ARC－172 (18)
2007－HPC－109 (18)
2007／3／1

ソフトウェアDSM MochaとMPIの並列ベンチマークを用いた性能評価

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

ソフトウェア分散共有メモリ (S-DSM) システムによるプログラミングは，一般に普及しているメッセージパッシングインタフェース (MPI) によるプログラミングよりも容易であるとされている．本稿では，MPI のひとつの実現ライブラリである MPICH と，S-DSM である Mocha の性能を比較をした．4 種のアプリケーション (MM, SOR, IS, LU) をベンチマークとして使用した．S-DSM システムのオーバヘッドを測定するため，割り込みハンドラの実行時間を測定した．その結果，S-DSM の性能が MPI の性能に追いつくためには：1. ページフォルト時より前に共有データをプリフェッチ; 2. ロック取得の高速化; 3. バリア同期の高速化; の順番に改善をおこなうべきであることが明らかとなった．

A Performance Comparison of Software-DSM Mocha and MPI Using Parallel Benchmarks

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

Software distributed shared memory (S-DSM) system is more friendly and easier to do programming compared with message passing interface. In this paper, we compared the performance of Mocha which is one of S-DSM systems, and MPICH which is a widely-used parallel programming library with message passing interface. Four applications (MM, SOR, IS, LU) were used for the parallel benchmarks. To measure S-DSM system overhead, the execution time of interrupt handlers was measured. The result shows that the followings should be performed to archive high performance in the S-DSM system compared with MPI: 1. Introdaction of a pre-fetching for shared data before a page fault; 2. Improvement of acquiring lock performance; 3. Improvement of barrier synchronization performance.

1. はじめに

並列計算のための環境として，コストパフォーマンスに優れた汎用の PC (Personal Computer) をもちいたクラスタシステム (PC クラスタ) が近年，普及してきている．ソフトウェア分散共有メモリ (Software Distributed Shared Memory, S-DSM) は，PC クラスタなどの分散メモリ型並列計算機環境に対して，ソフトウェアによる仮想的な共有メモリ型並列計算機環境を提供する．S-DSM はすでにいくつかのシステムが提案され，実装されている[1)2)]．

† 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
†† 東京工業大学 大学院情報理工学研究科
Graduate School of Information Science and Engineering, Tokyo Institute of Technology

S-DSM は，仮想的な共有メモリ環境を構築するために，分散メモリ間のデータ一貫性を保つための通信をおこなう．この通信は非同期におこなわれており，アプリケーションプログラマが意識することはない．そのため，並列アプリケーションプログラムの作成において，プログラマは分散したメモリ空間に配慮する必要がない．反面，プログラマが細かく配慮して通信処理を実装する性能チューニングを困難にしている．また，S-DSM は並列アプリケーションに対し独立して仮想的な共有メモリ環境を構築しているため，プログラマの意図しない性能低下が現れることがある．

以上により，分散メモリ型並列計算機環境を意識して作成された並列アプリケーションプログラムに比べ，S-DSM を用いると性能が劣るとされる[3)]．しかし，どのような部分において S-DSM プログラムの性能が劣るのか詳しく比較を行った報告は少ない．そこで本報告では，S-DSM で並列プログラミングを行う

場合に，どのような事由により性能低下を起こすのかをメッセージパッシングインタフェース（MPI）を用いた並列プログラムを用いて比較検証する．

本研究では，4種の挙動が異なるベンチマークを用いて，DSMプログラムとMPIプログラム両者の実行時間を詳細に計測することで問題を明らかにする．S-DSMシステムとして電気通信大学並列処理学講座で開発したMocha[2]を用いた．MPIとしては広く利用されているMPICH[4]を用いた．

本論文の構成は以下のとおりである．2章で比較評価を行う場合の問題点と手法について述べる．3章で，性能評価結果と考察について述べる．4章で関連研究について述べ，最後に本研究で得られた知見をまとめる．

2. 性能比較の前提と手法

2.1 S-DSMプログラムで検討すべき項目

ホームベース方式を採用しているS-DSMは，（1）仮想的な共有メモリに対するアクセス；（2）バリア同期；（3）排他制御；をもちいてデータの一貫性を保持することで，仮想的な共有メモリ型並列計算機環境の実現をしている．この3つは，MPIをもちいた並列プログラム（以降，MPIプログラム）にはなく，S-DSM向けに記述された並列プログラム（以降，DSMプログラム）の性能を決定する要因である．

2.2 MPIプログラムで検討すべき項目

MPIは，分散メモリ環境でのデータ通信を行うための並列プログラミングインタフェイスである．プリミティブな機能のみ提供するため，並列プログラムの記述方法に関して多くのバリエーションをもつ．そこでS-DSMプログラムと比較する場合，MPIプログラムの実現方法に関してDSMプログラムとの共通性をもつ制約を設ける必要がある．そこで本報告では，MPIプログラム作成時において，以下の基準を設定した．

(1) 計算アルゴリズムは同一とする．
(2) 各ノードが計算開始時に保持しているデータは同一とする．
(3) MPIプログラムにおける通信処理は，DSMプログラムにおけるアクセスパターンと同一になるようにする．

基準(1)と(2)を設定することによって，DSMプログラムからS-DSM機構をはずしたプログラムと，MPIプログラムから通信インタフェイスをはずしたプログラム，すなわち逐次プログラムはほぼ同一となる．このことで，計算カーネルの性能差をなくせるので，S-DSM機構自体の性能評価がおこないやすい．基準(3)を設定することによって，S-DSMプログラムが，共有データの転送時間がMPIプログラムと同一になることが期待できる．

なお基準(1)－(3)では，MPIプログラムにおける通信方式の制約はない．そこで本報告のMPIプログラムでは，非同期通信をもちいて通信と計算のオーバラップを行う．

表1 DSMプログラムの実行時間詳細内訳

比較手法	構成要素		
実行時間内訳	待ち時間	システム時間	計算時間
DSMプログラムの実行時間詳細内訳	バリア同期待ち時間	バリア同期システム時間	――
	ロック取得待ち時間	ロック機構システム時間	――
	一貫性処理待ち時間	アンロックシステム時間	――
	共有データ受信待ち時間	ページフォルトハンドラシステム時間	――
		IOハンドラシステム時間	――

2.3 比較手法

評価の基準としては，プログラムの実行時間をもちいる．ただし，並列プログラム全体の実行時間だけでは，詳細な比較評価をおこなえない．そのため実行時間を，以下の3つの観点から測定する：(1) 台数効果，(2) 実行時間内訳，(3)S-DSMプログラムの実行時間詳細内訳．以下で上記3つの説明をおこなう．

2.3.1 台数効果

プログラム全体の実行時間による比較である．台数効果の観点からの性能差がDSMプログラムとMPIプログラムにあるのか概観できる．

2.3.2 実行時間内訳

台数効果で明らかになった性能差が，どこにあるのか同定するためにもちいる．そのため，対象プログラムの実行時間を以下のように分離する．

計算時間 本来の計算をおこなう部分である．2.2節で述べたとおり，MPIプログラムとDSMプログラムの性能を比較するときに，計算時間は同一問題サイズに対し，双方で同じになるようにそろえた方がよい．もし双方の計算時間が異なるのならば，ベンチマーク自体に性能差があることとなり，公平な評価にならない．

待ち時間 通信によってプログラムが停止している時間である．

システム時間 並列計算を行う際に，計算以外の動作(バリア同期，データ通信によるIOハンドラなど)を行っている時間である．上記の待ち時間は，本システム時間には含まれない．分散メモリ型並列計算機環境においては，実行時間に占める待ち時間は重要であるので別項目とした．

2.3.3 DSMプログラムの実行時間詳細内訳

DSMプログラムの実行時間詳細内訳により，実行時間内訳を解析する際，その要因がDSM固有のどの処理によって引き起こされたのかを同定できる．そのため，DSMプログラムの待ち時間とシステム時間を詳細に項目別にする．待ち時間，システム時間をどのように分けたのかを表1にまとめる．

2.4 実行時間の計測

Mochaのバージョン0.2において，前節で述べた時間を計測するための機構を追加した．計測するのは，

Mocha 上でのプログラム実行中における以下の項目部分である：(1) 待ち時間，(2) 割り込みハンドラのシステム時間，(3) インターフェイスのシステム時間．

(1) は，データ送受信の完了，およびデータの一貫性保持を待つためのビジーウェイト時間である．(2) および (3) は，Mocha によるシステム時間を割り出すのに使用する．実行時間から (1)，(2) を引いたものを計算時間とした．ベンチマークプログラムの計算カーネル部には，本計測による変更は一切ない．

使用した MPICH のバージョンは，1.2.1 である．また，Mocha の時間計測に合わせるため，非同期通信のみをもちいて，以下の時間を計測する．

(i) MPI_WAIT および MPI_WAITALL を呼び出し，返ってくるまでの時間；

(ii) 上記以外の MPI 関数の実行時間；

(i) の MPI_WAIT，MPI_WAITALL は，前もって呼び出された非同期通信の終了を待つ関数である．通信による待ち時間が存在しない場合の MPI_WAIT および MPI_WAITALL の呼び出し時間は，通信時間に対して無視できるほどに小さい．実行時間から (i)，(ii) を引いたものを計算時間とした．この実現のため，Mocha の場合とは違い MPICH に変更を加えていない．ベンチマークプログラム上に記述をすることで実現した．このことで，ベンチマークプログラムの計算カーネル部には計測のための変更があるが，性能に関する影響は少ないと思われる．

Mocha と MPICH とも，初期データ配置の初期化などを含まない並列計算部分のみについて計測した．

2.5 比較に用いたベンチマーク

本報告で比較に用いた MM，SOR，LU は，既発表の報告[5] で使用しているものと同一であり，MPI による実現方式も同一なので説明を割愛する．ここでは，本論文で追加した IS のみ説明をする．

2.5.1 大規模整数ソート IS

int 型の整数配列を，バケットソート法によってソートするベンチマークである．データの初期配置は，ノード数で整数配列とキー配列をブロック分割した形になっている．ここでは，要素数が 2^{26} の問題サイズで性能評価をおこなう．

IS ベンチマークを DSM プログラムとして実現する場合，共有メモリ上に排他領域が存在する．排他制御によるデータ一貫性管理のための性能を評価するベンチマークとして有用である．

3. 性能評価結果と考察

3.1 評価環境

性能評価にもちいた PC クラスタ環境を表 2 に示す．使用した C コンパイラは，Mocha，MPICH ともに gcc のバージョン 3.2.3 である．最適化オプションを 2 レベル (-O2) にしてコンパイルをおこなった．

表 2 評価環境

ノード数	16
CPU	Intel Xeon 2.8GHz
OS	Cent OS 4.4
メモリ	1GB/ノード
ネットワーク	ギガビットイーサネット

台数効果を調べる際に基準としたのは，JIAJIA バージョン 2.2 に添付されているベンチマークにおいて，並列インタフェース部分とデータ分散部分を取り除いた逐次プログラムの実行時間である．

3.2 行列積 MM

3.2.1 結　　果

図 1 は，行列サイズ 2048×2048 での台数効果をあらわしている．また，図 2 では，全ノードのうち計算カーネルの終了がもっとも遅かったノードの実行時間の内訳を示している．以降，実行時間内訳は全て，計算の終了がもっとも遅かったノードのものである．

図 1 では，MPI プログラムが DSM プログラムよりも高い台数効果を示した．全てのノード数において，計算時間が MPI プログラムの方が短い．16 ノードにおいて，DSM プログラムは，MPI プログラムに比べて 57%の台数効果しかない．また S-DSM(Mocha) では，実行時間の内訳について，ホスト数が多くなるにつれ待ち時間が全実行時間に対して大きな割合を占めるようになる．このことから，ノード数が増加するに従い，待ち時間の差が性能差に直結している．

DSM プログラムによる，16 ノードにおける実行時間詳細内訳を表 3 に示す．詳細内訳の中で，0 秒の項目は載せていない．以降の DSM プログラムの実行時間詳細内訳についても同様である．

3.2.2 考　　察

図 2 の計算時間においては，ノード数が少ないときに差が出ている．ノード数が増加するにしたがって，計算時間については，MPI プログラムと S-DSM プログラムの間に差がなくなる．この理由は，各ノードが保持するデータ量が減ることによって，キャッシュヒットによる効果の差がなくなるからと考えられる．

MM では表 3 の項目で一番の実行時間を占めている共有メモリに対するアクセス機構が性能差をもたらす原因であることがわかる．MPI プログラムでは，通信処理をアプリケーションのデータアクセスパターンに添った形で自由に記述ができるので，非同期通信を使用して通信と計算をオーバラップするように記述できる．そのため，図 2 のように，通信による待ち時間がない．

一方，DSM プログラムでは，データアクセス特性とは無関係に，ページフォルトの際に必要なページを入手するためのデータ通信をおこなうためにデータの受信を待つ時間が発生する．

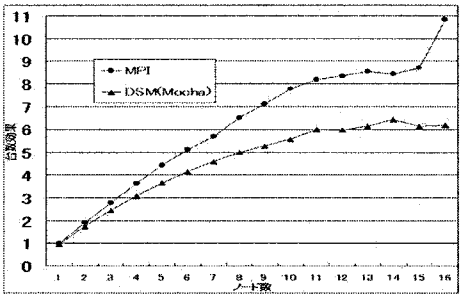

図 1　MM(行列サイズ 2048) の台数効果

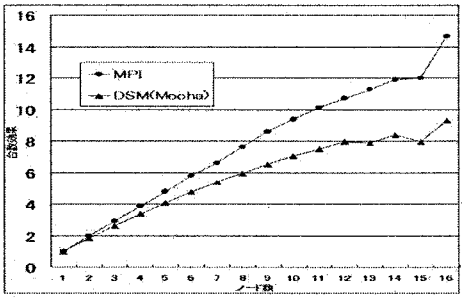

図 3　SOR(行列サイズ 10000) の台数効果

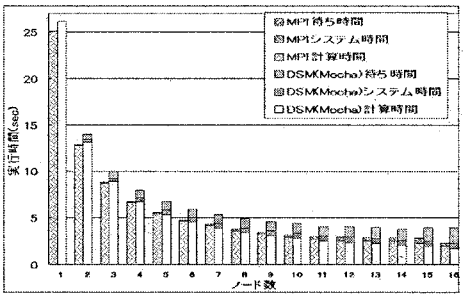

図 2　MM(行列サイズ 2048) での実行時間内訳

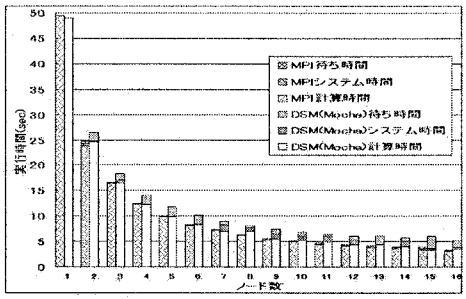

図 4　SOR(行列サイズ 10000) での実行時間内訳

表 3　MM(2048 × 2048) での DSM プログラムの実行時間詳細内訳

待ち時間 (sec)	システム時間 (sec)	
共有データ受信待ち	ページフォルトハンドラ	IO ハンドラ
1.64	0.20	0.36

表 4　SOR(10000 × 10000) での DSM プログラムの実行時間詳細内訳

待ち時間 (sec)		
共有データ受信待ち	バリア同期待ち	
0.27	1.44	
システム時間 (sec)		
バリア同期	ページフォルトハンドラ	IO ハンドラ
0.02	0.07	0.14

3.3　逐次過剰緩和法 SOR

3.3.1　結　　果

図 3 は，10000 × 10000 の行列サイズでの台数効果，図 4 は，実行時間の内訳をあらわしている．

MM と同じく，MPI プログラムが DSM プログラムよりも高い台数効果を示した．全てのノード数において，計算時間が MPI プログラムの方が短い．16 ノードにおいて，DSM プログラムは，MPI プログラムに比べて 63%の台数効果しか達成できない．MM と違い，ノード数が増加してもそれほど待ち時間が増えていない．16 ノードについて調べたものを表 4 に示す．

3.3.2　考　　察

表 4 をみると MM とは違い，同期による待ち時間が実行時間の大半を占めている．また，バリア同期を用いた一貫性保持は，それほど性能に影響していないことが分かる．

待ち時間の差は以下の理由による．SOR で 1 ループごとに黒格子と赤格子の操作が終わった後で，共有データが更新される．そのため，同期をとらなければならない．このとき，MPI では実装により，ある程度の時間幅を持たせて同期をとるような実装ができる．このことにより，同期をとるために待つノードの数を減らし，さらに待ちノードについても，待ち時間を減らすことができる．しかし，S-DSM では，一斉に同期をとるバリア同期を記述することが一般的である．このことで，余分な待ち時間が積み重なる．その結果，待ち時間が台数増加に従い増える．

3.4　大規模整数ソート IS

3.4.1　計 測 結 果

図 5 は，問題サイズ 2^{26} での台数効果をあらわしている．また，図 6 は問題サイズ 2^{26} での実行時間の内訳をあらわしている．

MPI プログラムと DSM プログラムで，性能差が顕著である．ノード数が増えるに従い，DSM プログラムの待ち時間が大きくなっている．DSM プログラムの実行時間は 16 ノードで，MPI プログラムの 62%の台数効果しか出ていない．DSM プログラムの実行時間詳細内訳について，差が良く現れている問題サイズ 2^{26}，16 ノードのデータを表 5 に示す．

3.4.2　考　　察

表 5 で，ロック取得待ち時間が半分を占めている．

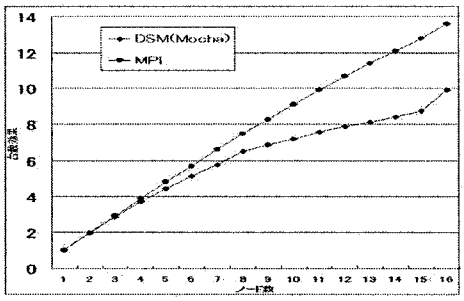

図 5 IS(問題サイズ 2^{26}) の台数効果

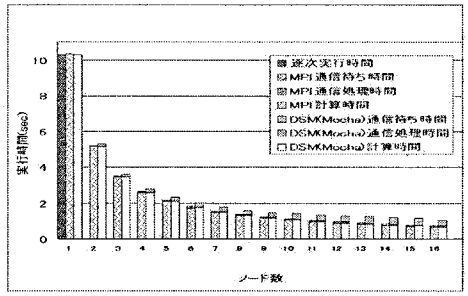

図 6 IS(問題サイズ 2^{26}) の実行時間内訳

表 5 IS(サイズ 2^{26}) での DSM プログラムの実行時間詳細内訳

待ち時間 (sec)		
ロック取得待ち	データ一貫性処理	共有データ受信待ち
0.15	0.03	0.08
システム時間 (sec)		
アンロック	ページフォルトハンドラ	IO ハンドラ
0.01	0.01	0.05

ロック制御のための通信時間が多くかかっていることがわかる．MPI プログラムも DSM プログラムも同じアルゴリズムをもちいているため，排他領域で操作をしている他ノードを待つ時間は MPI プログラムにも存在する．それにもかかわらず，これだけの差が生じるのは，ロックを取得するために，ロック管理をしているリモートノードへの通信時間が性能低下の原因と推察される．

3.5 LU 分解 LU

3.5.1 計測結果

図 7 は，ブロックサイズ 64 × 64，行列サイズ 512 × 512 における台数効果である．また，図 8 は図 7 と同様のサイズでの実行時間の内訳である．

台数効果に関し，MPI プログラムも S-DSM プログラムもほとんど同じ傾向を示した．実行時間内訳から，実行時間のほとんどが待ち時間のノードが混在する．この理由を調べるため，ノード数 13 における DSM プログラムの実行時間詳細内訳を表 6 に示す．

3.5.2 考　　察

実行時間のほとんどが待ち時間のノードがあるのは，今回，実行時間内訳を調べたノードが計算をおこなわ

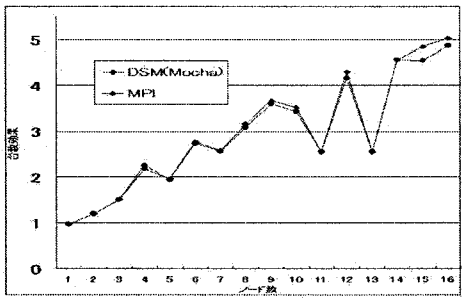

図 7 LU(行列サイズ 512) の台数効果

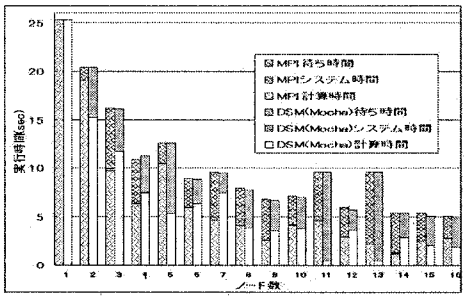

図 8 LU(行列サイズ 512) での実行時間内訳

表 6 LU(512 × 512) での DSM プログラムの実行時間詳細内訳

待ち時間 (sec)
バリア同期待ち
9.40

ないノードだからだということが表 6 から分かる．

前節の 3 つのベンチマークに比べ，LU は通信の隠蔽をおこないにくい構造となっている．LU の MPI プログラムは一般に，通信と計算がオーバラップできるアルゴリズムを採用できる．ところが今回使用している LU では，データが到着するまで計算を停止しなければならない構造となっている．これは表 6 で，各ノードがバリア同期で待つ時間が長いことからも推察できる．また，DSM プログラムでバリア同期をおこなう箇所も，MPI プログラムではブロードキャストによる通信をおこなう必要がある．このことから MPI においても，通信時間を削減しにくい．

以上のことから，同質の通信をおこなう DSM プログラムと MPI プログラムで，実行時間の差異がほとんどないことが分かる．MPI をもちいることを前提に LU 分解のアルゴリズムを変更できれば，MPI においては，より良い性能が得られると考えられる．

4. 関連研究

DSM プログラムと MPI プログラムを比較評価した研究として，Treadmark，MPICH，および PVM

を比較したPaulらの研究がある[3]．この研究では，並列アプリケーションを，(1) 並列化しにくいもの，(2) 同期 (SIMD) 型，(3) 半同期型の3つに分類し，それぞれ台数効果の計測をおこなっている．その結果，アルゴリズムが複雑で並列化しにくいアプリケーションでは，DSMプログラムとMPIプログラムの性能が拮抗した．その他の2種類の型では，MPIプログラムが優位であることが示されている．

本研究で使用したベンチマークは，上記の分類では全て同期型に属す．本評価から，同期型のアプリケーションでもDSMとMPIが拮抗する場合があることが明らかとなった．またPaulらの評価では，台数効果だけを評価基準としている．本評価では，性能差がどこに現れるのか，実行時間の内訳を詳細に調べた．

5. まとめ

本稿では，S-DSM(Mocha) が共有メモリを構築する際に必要となる機構の性能について調べるため，4つのベンチマークを選び，またどの程度の性能低下を起こすのか定量的に評価した．その際，分散メモリ環境を指向して構築されたMPIプログラムと比較することによって，より実用的な観点から性能評価をおこなった．

本評価から得られた知見について，共有メモリに対するアクセスの観点から述べる．まずS-DSMの核となる機構の性能が，全体の性能に直結する．4種のベンチマークを通して得られた性能低下の理由として，以下が挙げられる．

(1) ページフォルト段階になって初めてリクエストを出すため，データ受信のため待ち時間が多く発生する．
(2) プログラム記述時にデータアクセスパターンを考慮して記述するのが困難．
(3) S-DSMシステムが並列プログラムと独立しているため，S-DSMシステムがユーザアプリケーションの特徴を得てチューニングすることが困難．

1に関しては，プリフェッチをおこなう機構を導入することにより回避や軽減ができる．(2)，(3) については，アプリケーションがどのような通信をおこなっているのか分かれば性能低下を防げる．

次にバリア同期と排他制御について述べると，一貫性保持機構はそれほど性能低下に影響を及ぼさない．しかし，ノード間通信をおこなう必要のある排他制御やバリア同期は，性能に大きく影響する．多くの場合，一貫性保持のために使われるのはバリア同期である．また，排他制御と違いバリア同期は多くのノードが関わる．

以上の結論として，S-DSMの性能がMPIの性能に追いつくためには，(1) ページフォルト時より前に共有データをプリフェッチする機構の導入，(2) ロック取得機構の高速化，(3) バリア同期関数の高速化 の改善をおこなうべきである．

参考文献

1) Amza, C., Cox, A., Dwarkadas, S., Keleher, P., Lu, H., Rajamony, R., Yu, W. and Zwaenepoel, W.: TreadMarks: Shared Memory Computing on Networks of Workstations, *IEEE Computer*, Vol. 29, No. 2, pp. 18–28 (1996).
2) 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, 情報処理学会論文誌コンピューティングシステム, Vol. 46, No. SIG 12(ACS 11), pp. 170–180 (2005).
3) Werstein, P., Pethick, M. and Huang, Z.: A Performance Comparison of DSM, PVM, MPI, *Proceedings of the Fourth International Conference on Parallel and Distributed Computing, Applications and Technologies(PDCAT03)*, pp. 476–482 (2003).
4) Gropp, W., Lusk, E., Doss, N. and Skjellum, A.: A high-performance, portable implementation of the MPI message passing interface standard, *Parallel Computing*, Vol. 22, No. 6, pp. 789–828 (1996).
5) 今村昌之, 鈴木祥, 坂口朋也, 大島聡史, 片桐孝洋, 吉瀬謙二, 弓場敏嗣: MPIとの比較によるソフトウェアDSMの性能評価, 情報処理学会 研究報告 (ARC-169), pp. 157–162 (2006).