社団法人 電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE

ソフトウエア分散共有メモリシステムの性能比較

緑川　博子†

†成蹊大学工学部　〒180-8633 東京都武蔵野市吉祥寺北町　3-3-1
E-mail:　†midori@is.seikei.ac.jp

あらまし　計算機クラスタ上にユーザレベルで簡単にインストールできるソフトウエア分散共有メモリシステムとして代表的な TreadMarks と JIAJIA，さらに我々の開発した SMS の３つのシステムを，同一ハードウエア・ソフトウエア環境で性能比較した．７種の応用プログラムについての実行性能や通信パターンを調べ，実装方式による違い，ユーザの使い勝手なども比較した．

キーワード　分散共有メモリ，性能評価，SDSM，クラスタ

A Performance Comparison of Software Distributed Shared Memory Systems

Hiroko MIDORIKAWA†

†Faculty of Engineering, Seikei University　3-3-1 Kichijoji Kita-machi, Musashino-shi, Tokyo, 180-8633 Japan
E-mail:　†midori@is.seikei.ac.jp

Abstract　The popularity of cluster computing gives SDSMs high potential as a new programming environment, but few realistic performance comparison between representative SDSMs is available. This report presents a performance comparison of three SDSMs, TreadMarks, JIAJIA and SMS, which includes a performance evaluation of 7 benchmark programs, a detail packet-level communication analysis and difference in user interfaces.

Keyword　Distributed shared memory, Evaluation, Software distributed shared memory, Cluster

1. はじめに

コストパフォーマンスが優れたクラスタによる並列処理が，最近盛んに行われるようになってきた．クラスタ並列処理の魅力は，安価なハードウエアとフリーソフトウエアを用いて，個々のユーザの要求／予算に応じた規模のマシンが簡単に構築でき，予算に応じた性能が得られることである．クラスタ上でのプログラミング環境としては，いまだMPIなどのメッセージパッシングによるプログラミングが一般的であるが，一方でクラスタ上に仮想共有メモリを構築する幾つかのソフトウエア分散共有メモリシステム（SDSM）も開発されてきている．これらSDSMの多くは，MPICHやLAMなどと同様にユーザレベルソフトウエアとして簡単にインストールでき，MPIに比べてプログラムの可読性や書きやすさに優れ，新しいプログラミングモデルとしての可能性を秘めている．

本報告では，SDSMでは草分け的存在であるTreadMarks[1] (Tmk)，代表的なフリーソフトJIAJIA[2]（JIA），我々が開発したSMS[3][4]について，ベンチマークプログラムの実行性能，実装方式の違い，ユーザの使い勝手などを比較した．現時点でのSMSが他の代表的SDSMに対しどの程度の性能で，どのような特徴があるかを調べた．SDSMの比較は同一ハードウエア・ソフトウエア環境での計測が必須であるが，実際に代表的な複数のSDSMを直接比較したこのような報告はほとんどない．

2. 性能評価環境

ここでは，ハードウエア環境，OS，コンパイラ，最適化オプション，ソースコードも同一にして比較した．同じアルゴリズムであっても，ソースコードの実行文の書き方で実行時間に影響を及ぼすこともある．幸いな事に，ここで扱うSDSMは，実装方式やメモリコンシステンシモデルは異なるが，APIはほとんど同じなので，同一のソースコードに対し，それぞれのSDSM固有の初期化関数，バリア関数，データ割付関数などを置き換える形式でプログラムを作成した．

さらに，扱うプログラムのデータ構造や実行手順がSMSに有利になることがないように，本報告のプログラムは，TmkとJIAのもとで作られたサンプルプログラムを用いた．JIAに添付されているプログラムはTmkのプログラムを元にしており，ほとんど同じであるが，若干書き換えを行っているプログラムがあるので，それらについてはJIA版とTmk版のそれぞれについて，SMSの計測を行った．SMSの特性を生かしたソースプログラムの計測については，今回は省いた．

通信量や通信パターンの解析に，各SDSMが独自に報告する統計量を直接比較することは不適切なので，ネットワーク上を流れる実際のパケットを，tcpdumpでローカルディスクにログをとり，実行後オフラインで集計解析した．解析には我々が開発した計算機クラスタ用通信解析ツール[5]を用い，各プロセスにおける入出力メッセージ数，通信量を時系列で調べた．tcpdumpがベンチマークプログラム実行に及ぼす影響は，少なくとも本実験対象においては小さく，各SDSMとも解析結果にはほとんど影響しない．

評価に用いたPCクラスタとソフトウエアのバージョンを表１，表２に示す．一部のプログラムについては参考のため

にPVMプログラムの性能も示した.

3. 応用プログラムの性能

3.1. ベンチマークプログラム

表３に示す７種のベンチマークプログラムについての性能評価を行った．epとfftはNAS並列ベンチマークを，water,luはSplashベンチマークを元にしている．tspは分枝限定法によるTraveling Salesperson問題でここでは19都市を解いている． sorは red-black 型 Successive Over Relaxation処理である．mmは行列積で，ブロック化をせずに単純に１次元方向に２つの行列の乗算を行っている.mmとluはJIAだけに添付されているもので,残りのプログラムはTmkで使用されてきたものである[6]．fft以外はサンプルプログラムとしてJIAに添付されている．ただし，mmはJIA添付のプログラムではfloat配列であったが,double配列に変更した．表３には各プログラムで使用する共有データサイズと,プロセス毎のバリアとロックのコール回数（実測）も示す．表中t,j,sはそれぞれTmk,JIA,SMSを示す．図中でSMSのバリアの数が１回少ないのは，Tmk,JIAとも，共有データ割付関数コール直後にバリアコールが必要であるが，SMSではデータ割付関数内部に同期機構が含まれているので不要なためである．またJIAとSMSにはデータの分散割付機構があるので，これを用いている．Tmkはバリア後の次の使用時まで更新情報が送られないlazyなプロトコルで，JIAはホームベース方式のinvalidプロトコルである.SMSではバリアとロックは別に扱われ，異なる方式で実装されているが，バリアはinvalid方式を用いた.

3.2. 主要計算時間の比較

７種のプログラムの主要計算部分の実行時間について,表５の逐次プログラムの主要計算部実行時間を１とした時の各SDSMの速度向上比を図１に示す.性能向上比はどれを基本の逐次時間値とするかで全く異なってくるが，ここでは，各SDSMでの１プロセス時の実行時間ではなく,SDSM固有の関数が全くない同じアルゴリズムの逐次プログラムとした．８プロセスで実行した場合の全体の通信データ量,メッセージ数,メッセージ長の平均，最大，最小を表４に示す．すべての応用で，他のシステムよりSMSのメッセージ数が多いのは，１メッセージ毎にACKメッセージを返すSMSの機構と,後述するようにページ転送単位が他のSDSMの1/2であるためである．このため，平均メッセージ長は計算上短くなる.

3.2.1. ep, lu, water

epは，計算量に比べ通信量が少ないので，いずれのSDSMでも理想の性能向上比を得ている．luも計算量が多いため,通信がボトルネックにならず,各システムでの通信量には大きな差があるものの，同等の高い性能向上比を得ている．waterはバリアとロックの両方を使用するが，通信量も中程度で，通信も一時期に集中しないために，性能向上比は８プロセス実行で７程度と高い.

3.2.2. mm

mmの性能や表４の総通信バイト量に大きな違いはないが,実際の通信パターンは３つのSDSMで大きく異なる．図３は,８プロセス実行時の各プロセスの総通信バイト量を時系列に示している.Tmkはproc0からページを転送しているのに対し，SMSとJIAは分散割付しているため，各プロセスから順番にデータ転送が行われている．SMSやJIAではproc0と他のprocとのメッセージ数との差がないのに対し,TmkではProc0は他のprocに比べ，バイト量で７倍，数で４倍程度も多い.総メッセージ数のうち100B以下の小さなメッセージが占める割合は約80%(JIA)，約90%(SMS)，約64%（Tmk)である．残りはページ転送で，TmkとJIAでは8kB単位で行っているのに対し，SMSは4KB単位で行っているためもあって，smsのメッセージ数が他に比べ多い．この応用では，割付／初期化後と計算後のバリアだけなので,いずれのSDSMでも初期ページ転送のみで，invalid更新方式にしたため diffの転送はない.

3.2.3. sor

SORでは，JIAの通信バイト量がTmkの３倍以上になっている．Tmkでは初期の短時間の内にSMSやJIAの４倍程度のデータ量を集中してproc0から送信しているが，JIAやSMSでは，送受信双方が起こり３倍程度の時間がかかっている．８proc時の速度向上比はTmk,SMS,JIAの順になっている.

3.2.4. fft

fftでは，いずれのSDSMも通信データ量が多く，全体的に向上比が低い．３Dデータのある次元に関して分割し並列処理するが，１回のFFT計算毎に各次元で１DFFTを行うために,データ分割方向とは異なる次元での計算の際にデータの転送量が多くなる．通信パターンを解析すると，初期化とデータ転送の部分と,後半の３DFFT演算の６回の繰り返し部分に分けられる.TmkではProc0で他のSDSMの２倍近い通信量を集中的に行い，各プロセスで同時にページを受けとり，FFT計算を早く開始している．一方JIAとSMSは，分散割付した各プロセスから順番にデータの転送が行われているが，JIAは受信の頻度とサイズが一定で飽和しており,結果として時間がかかっている.

3.2.5. tsp

tspは，表４にJIA版とTmk版の計測値を示すが，図１の性能向上比のSMSはTmk版の値である．８プロセス時の性能向上比は，PVM:7.6， SMS:5.5，Tmk:3.9，JIA:2.4で，各システムの差が最も大きい応用である．JIAの通信バイト量はTmkの8.8倍，SMSの3.4倍にもなっている．SMSの通信量がTmkの2.6倍であるにもかかわらず実行時間が速いのは，共有データサイズが大きく,短時間にlockを多用する応用であることが一因である.この応用ではデータ分割のようなアルゴリズムで処理できないので，すべてのSDSMで共有データをproc0にのみ割り付けている．このため図４(a)に示すように,SMSでは初期にproc0からの集中的なデータ転送が生じている.Tmkに比べproc1の送受信が多いのは,SMSではproc1がロック

表１　評価環境

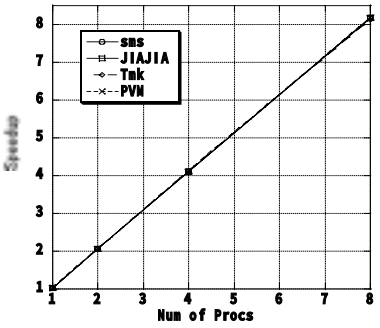
CPU	Intel PentiumIII-S 1.13GHZ
メモリ	512MB
ネットワーク	Intel PRO/1000T 3Com SuperStack3 Switch
OS	RedhatLinux7.1.2 kernel 2.4.7.10

表２ソフトウエアバージョン

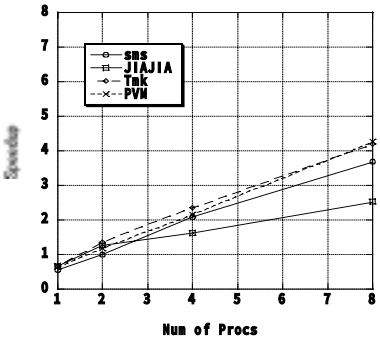
gcc version 2.96　-O3
SMS 0.4.16
JIAJIA 2.2
TreadMarks 1.0.3.2
PVM 3.4.4

表３　ベンチマークプログラム (8procs)

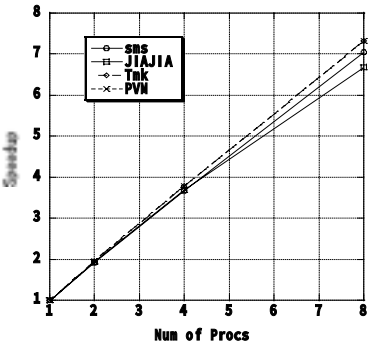
Prog	parameters	Data size	Barrier /proc	Lock /proc
ep	M=28,MK=10	44B	2	1
fft	128x 128 x 128 6 iterations	96MB	16 (t,j) 15(s)	0
water	1728 mols 5 steps	484KB	40	74
sor	2048x(1024x2) float 100 iterations	16MB	203	0
tsp	19 cities (19b)	100MB	4(j), 3(t,s)	75-122
lu	2048 x 2048 double 32blocks	34MB	135 134(s)	0
mm	2048 x 2048 double	96MB	3(s2)	0



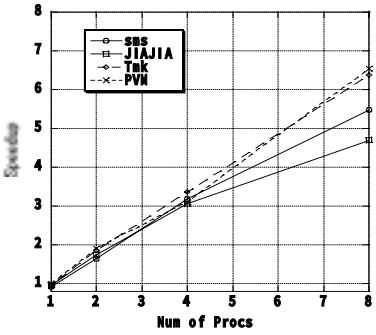
(a) ep



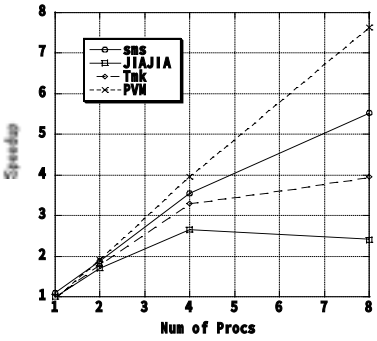
(b) fft



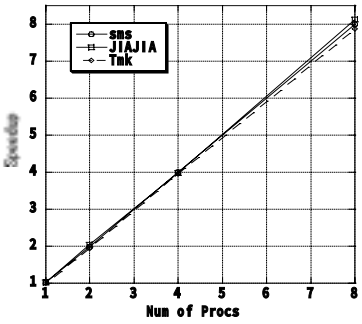
(c) water



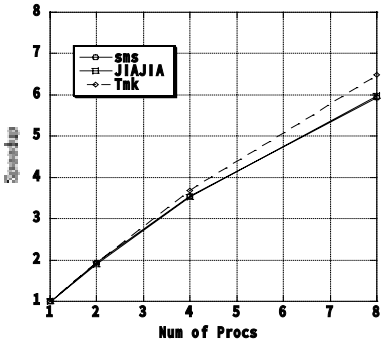
(d) sor



(e) tsp



(f) lu



(g) mm

図１　SMS, TreadMarks, JIAJIA における主要計算部の性能向上比

表４　８プロセス実行時の通信メッセージの概要

8 procs

Prog	Send/Rec Bytes(KB)			Send/Rec Msg Num			Msg Len Ave (B)			Msg Len Max (B)			MsgLenMin(B)		
	JIAJIA	SMS	Tmk	JIAJIA	SMS	Tmk	JIAJIA	SMS	Tmk	JIAJIA	SMS	Tmk	JIA	SMS	Tmk
ep	64	5	39	238	349	234	267	14	168	8,228	510	4,100	4	1	4
fft	569,320	410,229	337,375	273,179	847,503	83,671	2,084	484	4,032	32,740	4,100	8,204	4	1	4
water	28,131	30,467	26,081	17,124	46,828	11,877	1,642	650	2,195	31,276	32,707	31,884	4	1	4
sor	71,779	41,705	23,700	44,490	167,502	12,863	1,613	248	1,842	28,788	4,137	8,196	4	1	4
tsp		13,990	5,432		29,096	19,306		480	281		32,734	7,484	4	1	4
tsp-jia	47,617	14,168		35,694	15,057		1,334	940		13,452	32,745		4	1	4
lu	136,393	197,093	43,429	69,828	319,364	18,225	1,953	617	2,382	31,824	4,194	31,232	4	1	4
mm	268,933	233,100	262,702	125,549	624,985	89,321	2,142	372	2,941	8,228	4,100	8,196	4	1	4

表５　主要計算部
逐次実行時間

Prog	Seq (sec)
ep	224.11
fft	19.22
water	43.02
sor	8.30
tsp	8.52
lu	623.08
mm	94.25

マネージャーになっているためである．SMSでは通信の集中を避けるため，デフォルトではバリアマネージャーとロックマネージャーを別に割り当てる．このため，各プロセスが100回前後のロック/アンロック毎にメッセージと更新diffをproc１に送る．

特徴的なのは，図４(b)に示すメッセージ数で，Tmkが後半になって，小さいメッセージを多量に送受信しているのに対し，SMSでは後半のメッセージ数は少ない．これがTmkとSMSとの実行時間差の原因と考えられる．SMSは，ロック獲得時に，ロックマネージャを介しロック解放者から次の獲得者にそのロックに関わる最小の更新情報をupdateで渡す方式をとっている．一方，Tmkでは実際のデータアクセス時に，時系列的なdiffを各プロセスから集める方式をとっている．

3.3. 起動とデータ割付･初期化時間の比較

多くの性能評価論文で取り上げられるのは，4.2の主要計算部分のみであるが，実際にプログラムを起動して答えが得られるまでには，SDSMのプロセス起動や共有データ割付/初期化などを含む時間が必要である．そこで起動関数(sms_startup()，Tmk_startup()，JIA_init())の実行時間と，共有データ割付/初期化も計測し比較した．

起動時間は，応用プログラムによらずいずれのSDSMでも使用プロセス数に比例して長くなるが，SMSが最も高速でTmk,JIAの順となっている．２から８のプロセス起動で，smsは0.1〜0.6s，Tmkは0.1〜1.0s，JIAは3.3〜4.3sである．JIAは１プロセス起動の場合にも固定的に3.3sかかっている．

共有データ割付/初期化時間は，応用によって共有データサイズや割付関数コール数が異なるので，応用プログラム間での比較はできないが，同じ応用では各SDSMで同一コードなので，比較が可能である．初期化がなくデータの割付のみを行うepなどでは，プロセス２〜８で，SMSは2〜7ms，Tmkは1〜10msとミリ秒であるが，JIAでは固定的に１sかかる．割付後にデータ初期処理も行うwater，fft，sorでも，JIAの初期化時間が大きく，８プロセスで，［water：sms 0.8s，Tmk 0.7s，JIA 7.8s］，［fft： sms 4.4s，Tmk 3.0s，JIA 11.8s］，［sor：sms 3.5s，Tmk 3.0s，JIA 11.8s］である．

全体の傾向としてはTmkが割付初期化処理が最も高速で，次にSMS,JIAの順である．JIAは整数１つであっても，割付関数コール時にページをmapする手法を採っていることが遅さの一因である．これに対しTmkやSMSは，起動関数コール時に規定数分のページをmapしており，データ割付関数コール時には共有データ管理表のパラメタ設定のみしか行わない．

図２に，起動と割付初期化を含む総実行時間での性能向上比を示す．図１に比べて全体に向上比は低下するが，JIAの低下が著しい．特に主要計算部分の実行時間が小さい応用では，影響を受けやすく，並列処理高速化効果を全体性能では帳消しにするほどである．luは計算時間が長く，起動割付初期化時間の影響をあまり受けないので，図２では省略した．

4. ユーザインターフェースと使いやすさ

各SDSMのAPIは似ているものの，細かい点で違いがある．Tmkはデータ割付APIが他と異なり，割付関数はproc0のみが呼び，得られたアドレスをメッセージ転送に類似した関数でユーザが陽に他プロセスに伝える．JIAとSMSは，割付関数内部でアドレス伝搬を自動で行い，全プロセスが割り付け関数をコールするのが前提である．この違いは一長一短がある．

JIAはプロセス数を実行時にコマンドパラメタで変更できず，ホスト名とパスワードを含むファイルの変更が必要で，実行が煩雑であるばかりかセキュリティ上も危険である．

SMSとTmkは規定ページ数をあらかじめ起動時に割り付けるため，それを越える大規模データを扱う応用では，コマンドラインで毎回ユーザが初期ページ規定数を指定する必要があるが，JIAではこれが不要で使い勝手がよい．

5. おわりに

以上をまとめると，起動，割付，計算の各部とも，すべての応用で，JIAは他の２つ（SMS,Tmk）よりも劣っている．

SMSとTmkの比較では，起動関数はSMSが高速であるが，割付関数と初期アクセスは，SMSが２応用(sor，fft)で劣り，５応用で同程度である．計算部分は，SMSが３応用(sor，fft，mm)で劣り，３応用でほぼ同程度，１応用(tsp)で優れている．

Tmkは応用全体にメッセージ通信量が少なく，通信集中時のピーク転送性能がよい．データの分散割付をしなくとも性能が高い点で他(SMS,JIA)より優れている．

SMSは，全体としてTmkに比べ通信量が多く，ページ転送単位が他SDSMの半分なので，大容量データを扱う応用での初期ページ転送に不利なことが問題点である．ロックを多用する応用として，他にも何種かのtsp問題を行ったが,Tmkと同程度以上の性能が得られており，バリアのみの応用に比べ，SMS独自のロック実装方式が有効に働いていることがわかる．

文　献

[1] Peter Keleher, et al.: TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems, Proceedings of the Winter 94 Usenix Conference, pp. 115-131, (1994).

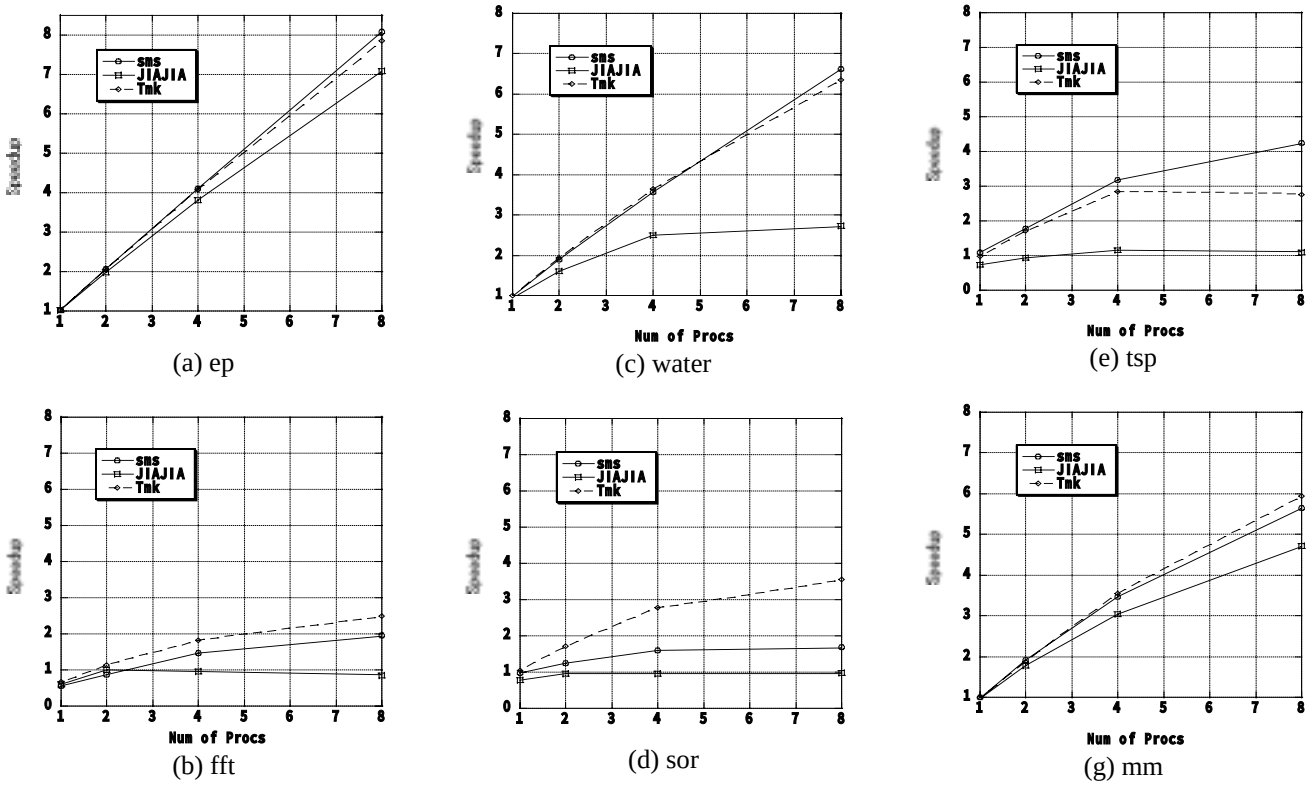
[2] M.R.Eskicioglu, et al.: Evaluation of the JIAJIA Software DSM System on High Performance Computer Architectures, Proc.of the 32nd Hawaii Int. Conf. on System Sciences, (1999)

[3] 緑川,飯塚, "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 HPC Vol.42, No.SIG9(HPS3), pp.170-190 (2001)

[4] 緑川，大橋，片野，飯塚, "ソフトウエア分散共有メモリ SMS －性能向上のためのロック実装と関数増設－",並列処理シンポジウム JSPP'02 p.127－130 (2002)

[5] 菊池，緑川，飯塚 ,"計算機クラスタ用通信解析ツールの開発",情報処理学会第６４回全国大会論文集(１)1－69（2002）

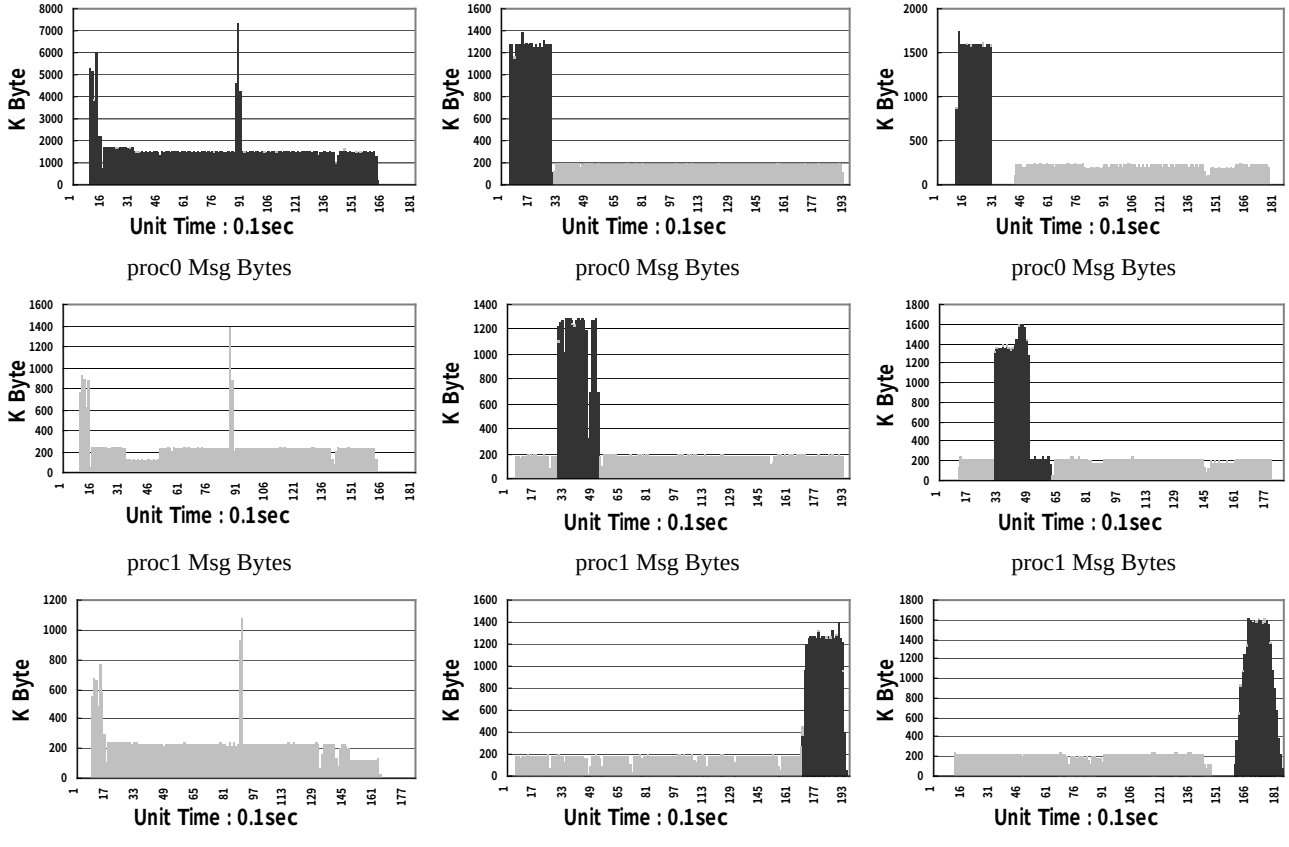
[6] H. Lu, et al.: Message Passing versus Distributed Shared Memory on Networks of Workstations, Proc. of Supercomputing '95 (1995)



(a) ep (c) water (e) tsp

(b) fft (d) sor (g) mm

図２　SMS, TreadMarks, JIAJIA における総実行時間の性能向上比



proc0 Msg Bytes　proc0 Msg Bytes　proc0 Msg Bytes

proc1 Msg Bytes　proc1 Msg Bytes　proc1 Msg Bytes

proc7 Msg Bytes　proc7 Msg Bytes　proc7 Msg Bytes

(a)　TreadMarks　(b)　SMS　(c)　JIAJIA

図３　TreadMarks, SMS, JIAJIA における mm の通信パターン（通信バイト量）（黒:送信, 灰:受信）

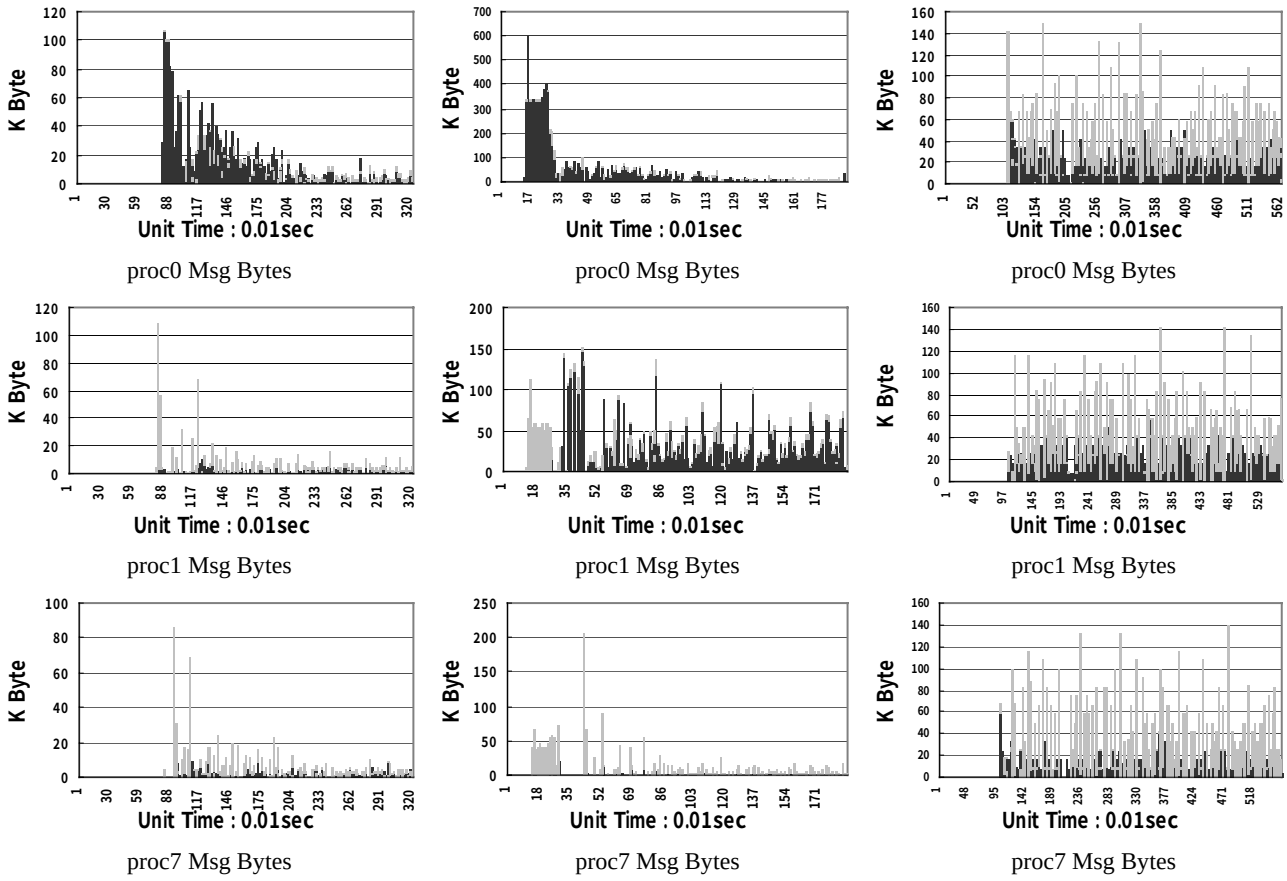
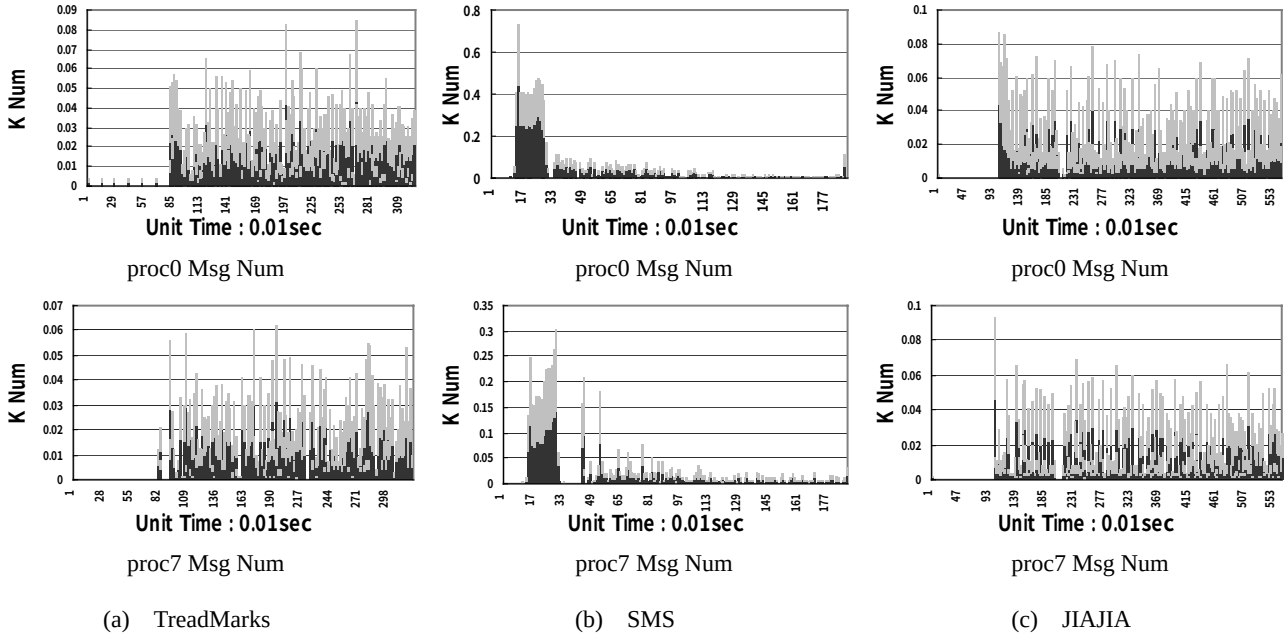


図４(a)　TreadMarks, SMS, JIAJIA における tsp 通信パターン（通信バイト量）（黒:送信，灰:受信）



(a)　TreadMarks　　(b)　SMS　　(c)　JIAJIA

図４(b)　TreadMarks, SMS, JIAJIA における tsp 通信パターン（メッセージ数）（黒:送信，灰:受信）