SMP クラスタ向け分散共有メモリシステム SMS2
－設計と実装－

菊池 康祐・緑川 博子・飯塚 肇
成蹊大学大学院工学研究科情報処理専攻

1 はじめに

筆者らは，クラスタ型並列コンピュータシステム上に共有メモリプログラミング環境を実現するソフトウェア分散共有メモリ SMS の設計，実装を行ってきた．従来の SMS では，ノードの内外によらずソケットによるプロセス間通信を行って，ノード内 CPU 数に依存しない複数プロセスから成る並列プログラムを作成・実行していた．

今回，クラスタの各ノードにアプリケーションプロセス以外にデーモンプロセスを立ち上げ，SMP クラスタに適したプロセス，メモリ，通信の管理を行えるようにしてシステムの柔軟性を高め，ノード内プロセス間のデータ転送を効率化した．また，ノード内プロセスに共通のキャッシュ機構を導入することにより，ノード内プロセスの共有データアクセスを効率化した．

本稿では SMS2 の設計と基本関数及び数種類の基本通信性能の評価を行ったので報告する．

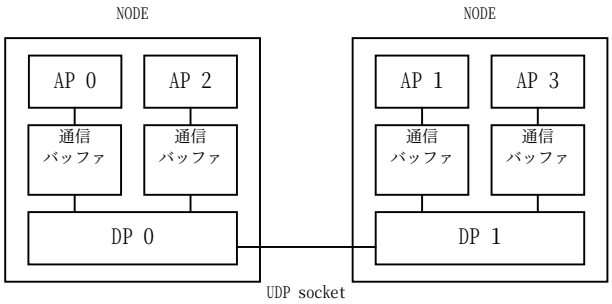
2 共有メモリのキャッシュ機構

従来の SMS ではユーザの要求したサイズ分のメモリを各プロセスに確保したので，SMS で使用できる共有メモリ領域のサイズは１ノードの物理メモリサイズによって制限されていた．SMS2 では共有メモリ領域をホームとキャッシュに分けた．共有メモリ割り当ては，メモリを確保するのはその領域のホームプロセスのみであり，他のプロセスがその領域にアクセスする場合は，ホームプロセスからページをキャッシュ領域にコピーしてからアクセスを行っている．１アプリケーションプロセスが使用するメモリ量はホームに割り当てる領域と，キャッシュに割り当てる領域なので，ホームを分散して割り付ければ，１ノード上に備えられている物理メモリより大きいサイズの共有メモリが使用できる．

3 SMS のデーモン化

SMS2 は，図１のようにデーモンプロセスとアプリケーションプロセスの全体で１ノードとして管理をしている．ノード外通信はデーモンプロセス間を UDP ソケットで接続して行っていて，ノード内通信は共有メモリを利用したバッファ経由で行っている．以下，本設計による高速化への寄与について述べる．

DSM System for SMP Clusters:SMS2
- Design and Implementation -
Kosuke Kikuchi, Hiroko Midorikawa, Hajime Iizuka
Department of Information Sciences, Seikei University



AP：Application Process　　DP：Demon Process

図１　プロセスの配置

・ノード内通信の高速化

従来の SMS では，１ノード上で複数のプロセスを実行する場合，ノード内とノード外のプロセスに対する通信を区別せず，UDP ソケットでプロセス間を全結合していた．しかし，SMS2 では通信をノード内に対するものとノード外に対するものに区別して，ノード内のデータ転送は図２のように直接バッファに読み書きすることによって低遅延の通信を実現し高速化を図った．

・ノード外プロセスに対する要求の減少

ノード外のプロセスへの要求によって得られるデータをノード内の他のプロセスが所有している場合は，要求するプロセスはノード外通信を行わないのでより高速にデータを得ることが可能である．

・ノード内通信時のデータコピーの減少

ページ転送のようなノード内の通信において，通信バッファへコピーを行わないで直接対象アドレス間でデータをコピーして送信することが可能である．

・ノード内の共有メモリの効率的な管理

ノード内のプロセス間で共有メモリのデータを共用することにより，プロセスのメモリスペースを節約やデータ転送回数を減らすことが可能である．

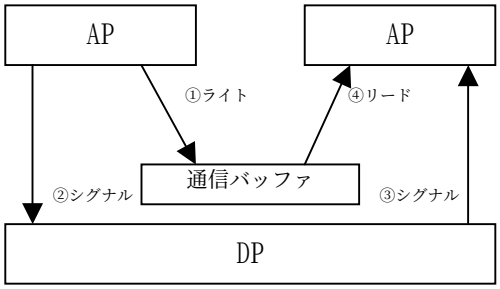


図２　ノード内プロセス間のデータ送信

4 性能評価

4.1 評価環境

測定は，表１のような環境で行った．

表1 評価環境

CPU	Intel PentiumIII 700MHz × 2
メモリ	256MB
ネットワーク	Intel PRO/1000 T 3Com SuperStack 3 Switch
OS	RedhatLinux7.2 kernel2.4-smp
台数	1〜4

4.2 基本関数

表２は各関数の実行時間である．SMS2 の実行コマンドは SMS と異なっていて smsrun を用いる．init 関数はデーモンによる初期化時間が含まれていないので，SMS に比べて速くなっている．barrier 関数は diff の転送が発生しないときの実行時間である．SMS2 の方が遅いのは SMS と実装方法が異なっているためであり，改善の余地がある．

4.3 通信性能

SMS2 では 4k バイト以上のメッセージは，4k バイトの複数メッセージに分けて送っている．図３は 4k バイト以下のメッセージの平均送受信時間である．ノード外通信はメッセージサイズに応じて送信時間がかかっているが，ノード内通信は通信時にコピーが１回しか生じないので送信時間にはメッセージサイズの影響がほとんどない．また，ノード外通信に比べて約半分の送信時間である．図４は 4k バイト以上のデータ送信時間である．転送データサイズが小さいときのノード内通信とノード外通信の送信時間の差はほとんど生じないが，実際の diff やページのような大きいサイズのデータは転送回数が増えるので差は大きい．

表2 基本関数の性能

単位:MicroSecond		N:ノード数 P:プロセス数					
		N:1 P:1	N:1 P:2	N:2 P:2	N:2 P:4	N:4 P:4	N:4 P:8
SMS	sms_init	40677	259562	241683	378521	305690	542585
	sms_calloc	6	132	156	248	233	754
	sms_barrier	20	218	268	417	383	633
SMS2	sms_init	943	1341	1087	1337	1247	1235
	sms_calloc	7	67	192	223	255	569
	sms_barrier	28	193	507	706	726	1519

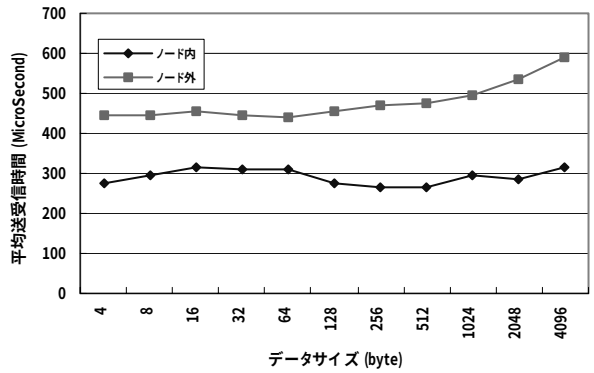


図3 4kバイト以下のメッセージの平均送受信時間

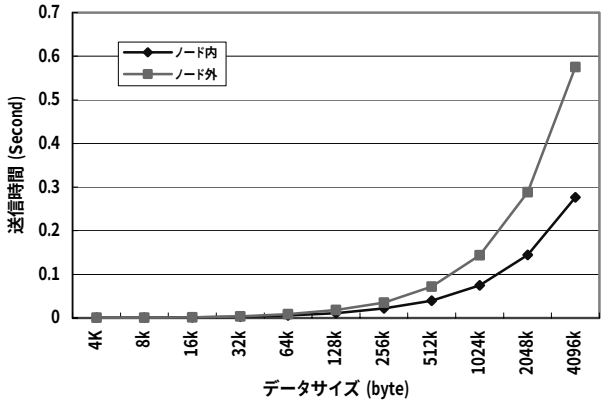


図4 4kバイト以上のデータ送信時間

表3 マンデルブローの実行時間 サイズ 512×512

単位:Second		N:ノード数 P:プロセス数					
		N:1 P:1	N:1 P:2	N:2 P:2	N:2 P:4	N:4 P:4	N:4 P:8
SMS	sms_init	0.043544	0.255744	0.240185	0.370683	0.304244	0.516823
	sms_calloc	0.000178	0.005584	0.002189	0.005516	0.004934	0.01165
	sms_barrier	0.000038	0.00021	0.000275	0.025546	0.02061	0.024365
	total	0.642772	0.602955	0.582539	0.631061	0.384205	0.631061
SMS2	sms_init	0.000838	0.001277	0.001204	0.002557	0.001209	0.001315
	sms_calloc	0.000333	0.000877	0.188383	0.182474	0.204594	0.229402
	sms_barrier	0.000128	0.000218	0.000574	0.026639	0.026485	0.021884
	total	0.600368	0.344118	0.530206	0.384205	0.403735	0.33104

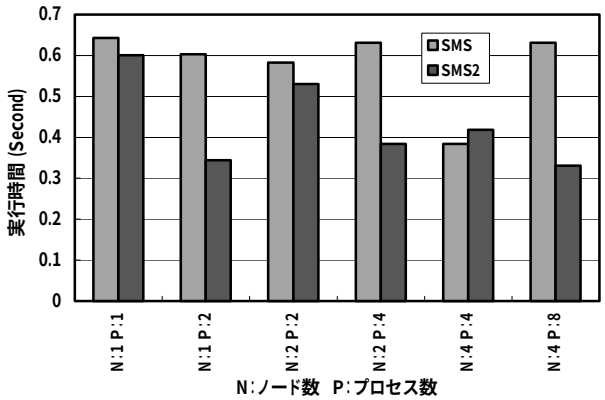


図5 マンデルブローの総実行時間 512×512

4.3 応用プログラム

表３は，マンデルブロー描画プログラム[1]の実行時の SMS と SMS2 の各成分の実行時間で，図５は総実行時間のグラフである．共有メモリの分散割付を行っているため diff の転送は発生していない．総実行時間は SMS2 の方が速いが，sms_init 関数は前述のように SMS に比べて見かけ上速くなっているだけなので，実際の計算時間はほとんど差がない．

5 結果

SMP クラスタに指向した SMS2 の設計・実装を行い，基本的な性能の評価を行った．今後の課題としては，barrier 関数の改善や lock 関数の実装，実装の最適化，実際の応用プログラムでの評価を考えている．

参考文献

[1]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 Vol.42,No.SIG9(HPS3),pp.170-190(2001,8)