ソフトウエア分散共有メモリSMS
ー性能向上のためのロック実装と関数増設ー

緑川 博子　大橋 祐介　片野真吾　飯塚 肇
成蹊大学 工学部

ユーザレベルソフトウエアによる分散共有メモリシステムSMSにおいて、今回新たにロック更新情報をロックマネージャー集中型にする方式を実装し、さらに実行時のロックマネージャー変更、非排他ロック、非ブロック型ロック獲得など、ロック機能を強化した。また多量な更新情報を通信する際における流量制御機構、通信エラー検知機構などを付け加えた。さらにデータ分割による並列処理によく見られるメモリアクセスパターンに有効なバリア関数を提供した。

Software Distributed Shared Memory SMS
- A new lock implementation and new functions for performance upgrade-

Midorikawa H., Ohashi Y., Katano S., Iizuka H.
Seikei University

A new conveyance scheme of memory update information from a lock releaser to a lock acquirer is incorporated into our software distributed shared memory system, called SMS. Moreover, various kinds of lock functions, such as non exclusive lock acquire and non block lock acquire, a communication flow control and a communication error detection are newly supported. To avoid a significant overhead of sigsegv handler for shared data access detection, we proposed a new dedicated barrier function to be available for one of the common memory access patterns in parallel processing.

１．はじめに

筆者らは、OSに手を加えたり、特殊な通信機構を前提とすることなしに、ユーザレベルのソフトウエアだけで容易に構築できる分散共有メモリシステムSMSを提案、実装してきた[1][2][3]。SMSは、Linux/FreeBSD上にUDP/TCPで実装されており、メモリコンシステンシとしてIBEC(Implicit Binding Entry Consistency)モデル[1][3]を用いた、ページベースのシステムである。今回新しいロック実装方式とロック関連関数の強化、sigsegv抑制バリア関数、通信フロー制御とエラー検知などを導入し改良したので、報告する。

２．ソフトウエア共有メモリシステムSMS

SMSは以下のような特徴を持つ。

- **ユーザレベルソフトウエアによる実装**
- **ネットワークを選ばないTCP/UDP通信**
  特殊な通信機能を用いず、ギガビットイーサ、MyrinetなどTCPもしくはUDPをサポートする通信媒体であれば、どのようなものでも使用することができる。
- **ハードウエア独立で柔軟な仮想プロセス実行**
  単一CPUから、複数CPUのPCクラスタまで、ハードウエア上のPC数、CPU数に独立にプログラムを作成することが可能。
- **新しい緩和型メモリコンシステンシモデル**
  暗黙バインディングエントリコンシステンシモデルIBECを採用している。従来のエントリコンシステンシとは異なり、共有データ変数と同期変数との関連付けをプログラム中で明示する必要がない。現在の実装ではアップデート方式で、共有変数更新情報はロック獲得者に渡す方式をとっている。
- **柔軟かつ多様なユーザライブラリ関数**
  SMSでは、共有変数割付関数、条件変数、様々なロック関数とバリア関数、さらに共有変数やロックを管理するページマネージャーやロックマネージャーの実行時変更関数などを提供している。

３．ロック機能の強化

・ロックマネージャーによる更新情報集中管理

従来方式では、あるロックに関連する共有変数の更新情報は、最後にそのロックを獲得したプロセスがすべて保持していた。ロック獲得・解放時にロックマネージャープロセスにより指令を受け、最後にロック獲得していたプロセスが、次にそのロックを獲得して共有変数を使用しようとするプロセスへ、更新情報を直接渡す。このためメッセージ数は少なく、ロック獲得したプロセスがその後も続けてその共有変数をアクセスする際には、更新情報を他に転送する必要がなく、効率的であった。ロックマネージャープロセスは、単にロックキューの管理と排他制御のみを行い、更新情報はそのロックに関わる共有変数を使うプロセスが持つという思想である。

しかし、応用によっては、一つのバリアインターバル中に特定のプロセス間で、何度も同一

ロックの獲得解放が繰り返される場合がある。あるプロセスにとっては、前回にロック獲得した後の変更情報さえ得られればよいにもかかわらず、そのロックに関わる時系列的なすべての更新情報が繰り返し受け渡される。このためメッセージサイズが大きくなり、全体性能を落とすこともある。

そこで、ロック解放者はいったん更新情報をロックマネージャーに返し、次のロック獲得者はロックマネージャーから更新情報を受け取るというロックマネージャーによる更新情報集中管理方式を実装した。これにより、ロック獲得者は自分の現在のタイムスタンプをロック要求メッセージに付け、ロックマネージャーはそのタイムスタンプ以後の更新情報のみを、ロック獲得者へ渡す。ロック解放者と獲得者の間にロックマネージャーが介在することになり、一般に従来方式よりメッセージ数は増加するが、メッセージサイズは少なくなる。しかし、多量な更新情報がある場合には、後述するように適当なサイズのメッセージパケットに分割して複数メッセージで送信しているので、新方式により更新情報が増大化しないことで、メッセージ数を減らせることもある。

・ロックマネージャー変更関数

今回、ロックマネージャーによる更新情報集中管理方式に変更したのに伴い、実行時におけるロックマネージャーの変更関数を新設した。従来は、更新情報そのものをロック獲得者間で受け渡していたために必要がなかったが、ロック実装方式の変更により、ロックマネージャーは最もそのロックの使用頻度の高いプロセスにすることが有利になるためである。この関数を呼び出した次のバリアで、マネージャーが異動し有効になる。

・非排他ロック関数

ロックを獲得して共有変数にアクセスする場合、実際の応用プログラムでは、共有変数のリードのみしか行わない場合が多くある。すなわち、マルチプルリーダシングルライターというような状況である。このような場合、リーダーはロックを獲得しても排他制御をする必要はなく、ロック獲得時の最新データさえもらうことができれば、その後は他のロック要求を排除しない。すなわち複数の非排他ロックの同時使用が可能である。しかし通常の排他ロックが使用されている場合には、非排他ロック要求はキューに入れられる。非排他ロック解放関数は、通常のロック解放関数と違い、ロックマネージャーに通知されず、単にそれ以後は、そのロックに関わる共有変数にアクセスしてもデータが最新でないというマークに使用される。

・非ブロック型ロック関数

ロック獲得要求で、ロックが獲得できなかった場合に-1を返して、ブロックしない関数を増設した。これによりロック獲得できなかった場合には、他の処理を行うというようなプログラム記述も可能になった。

４．通信フロー制御とエラー検知

SMSでは、メッセージデータのほとんどが１バイトから８バイト程度のメッセージタイプ毎の固定メッセージ長である。固定長メッセージで最もメッセージ長が大きいのはページ転送メッセージである。可変長メッセージなのは、ロック解放・獲得やバリア時に発生する更新情報メッセージでのみである。したがって、更新情報メッセージに限り、通信ソケットバッファを越えない一定のサイズにメッセージを分割し、メッセージ送信毎に受信側からackを返して、オーバーフローしないように制御している。

また、SMS構築時に通信エラー検知モードを指定すると、メッセージを送るたびにタイマーをセットし、一定時間を過ぎてもackが返ってこない場合には、データの不達としてエラー表示し終了する。

５．シングルリードライト用バリア関数

複数プロセスでデータ領域を分割して並列に処理するというのは、典型的な並列処理のパターンである。その中でも各プロセスの独立性が高く、分割データ領域の隣接境界部分以外は、担当プロセスのみがリード/ライトするという場合がよくある。このような一見、非常に独立性が高く、それぞれのプロセスへの割り当ても最適に近い並列処理を、OSのメモリ保護機構を共有変数アクセス検知に用いたシステムで行うと性能劣化が起こることがある。これは、sigsegvのオーバーヘッドが予想以上に大きいためである。ハードウエアによるメモリアクセス検知機構を持たない多くのソフトウエア共有メモリシステムでは、ロックやバリアで更新情報を得るためや、ローカルにないメモリページなどの転送要求を知るために、共有変数へのアクセス検知にsigsegvを使用している。しかし、上述のような応用で、明らかに他のプロセスからの更新情報の要求がないと分かっている時に、オーバーヘッドの大きいsigsegvを用いて、アクセス検知を行い更新情報作成のための準備（ページのコピーなど）をするのは無駄である。そこで、二つのバリア間でライターが１プロセスで他のプロセスにコピーがないRW（リードライト可）の

ページは、本来バリア時にRO（リード専用）にページ状態をリセットするのであるが、上記条件を満たすページはそのままRWにしておき、次回のバリアまでsigsegvを抑制し、よけいな更新準備をさせないシングルリードライト用バリア関数を設けた。SMS内部に、バリア時にsigsegv抑制バリアに切り替えるか通常バリアのままにするかの自動切り替え機構を組み込んだり、予想に反し他からの更新データの要求がされた際の処理を組み込むことも可能であるが、プログラマーが一番その応用の性質を知っているので、通常バリア関数を使うか、シングルリードライトバリアを使うかはユーザが選択できるようにした。これにより、複数プロセスからアクセスされるデータ境界ページは、通常どおり、sigsegvでアクセスを検知し、更新情報を得るための準備をするが、１プロセスのみがアクセスするページに関しては、初回を除いてRWのままでバリアを通過する。これによりsigsegvハンドラは起動されず不要な処理を省く。

６．性能評価結果

・新ロック実装方式の効果

表１の評価環境1において、1つのロック変数を用いて、lock/unlock間で1変数のみをインクリメントしていく処理を2プロセスでそれぞれ1000回繰り返した場合の性能差を図１に示す。従来方式の(1)を1としている。

図1の(1)は、2つの作業プロセス中1プロセスがロックマネージャーを兼ねた場合である。この場合、片方のプロセスがロックを獲得するときは通信が発生するが、もう片方のプロセスがロックを獲得するときは通信が発生しない。(2)は、2つの作業プロセス以外のプロセスがロックマネージャーの場合である。この場合、どちらのプロセスがロックを獲得しても通信が発生する。ロックマネージャーが作業する場合のほうが更新情報の転送量は少ない。このような通信パターンの場合、新方式は従来方式に比べ1.6倍から2.2倍の速度向上が見られた。

また、8プロセスで各プロセスがそれぞれ8つのロック変数繰り返しlock/unlockした場合の性能差を図2に示す。この場合においても新方式のほうが優れていること、またロックマネージャーをロックidごとに分散させたほうがより効果があることがわかる。

ここで、新方式の性能を、後述する代表的なソフトウェアDSMであるTreadMarks(ver.0.10.1.2)[4]と比較してみる。通信パターンは、1つのロックidを2つのプロセスがロック/アンロックする。SMSではロックマネージャーとページマネージャーの指定を明示的に行えるため、ロックとそのロックに関連づけられた共有変数を使用するプロセスをそれぞれのロックマネージャーとページマネージャーとした。

この場合の各プロセス間のメッセージ数とメッセージ量を図3、表3に示す。表3は、ページマネージャーが単一な場合と複数で分散している場合を示しており、図3はページマネージャーが分散されているときのメッセージ量である。

ページマネージャーが単一プロセスの場合であっても、メッセージ量では若干TreadMarksに劣っているものの、メッセージ数はTreadMarksに比べかなり少なくなっている。ページマネージャーを分散させた場合、メッセージ量、メッセージ数ともにTreadMarksを大きく下回ることができた。pc9-pc10間、pc11-pc12間といった、ロックidに関わる通信は、SMSとTreadMarksで大差ない。しかし、TreadMarksの場合、これらの通信とは別に、pc9からその他すべてのプロセスへの通信が発生している。これは、SMSではページマネージャーを明示的にそのページが必要なプロセスに割り振ることにより、不要なページ転送などを減らした結果と思われる。以上のテストプログラムは、ロックの獲得・解放を多数繰り返す処理で、新ロック実装方式が有利になる例であるが、従来のロック実装方式では、更新データが膨大になり性能の上がらなかった、応用についても、TreadMarksと同程度の性能を得ることができた。図4は、表２の評価環境２においてSPLASH2のWATER（1728mole）を処理した時の性能向上比である。

表1:評価環境1

CPU	Intel Celeron 400MHz
メモリ	128MB
ネットワーク	Intel PRO/1000 T 3Com SuperStack 3 Switch
OS	TurboLinux Server 6.1
台数	1～8

表2:評価環境2

CPU	Intel MMX Pentium 166MHz
メモリ	64MB
ネットワーク	Intel PRO/100+ corega FSW-16L
OS	FreeBSD 2.2.8-RELEASE
台数	1～8

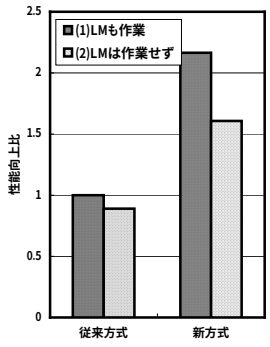

図1　1ロック、2プロセス

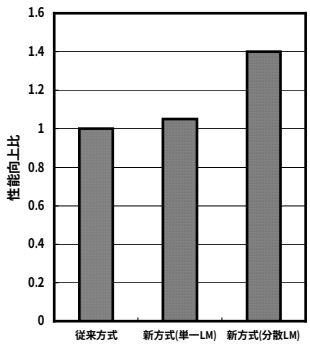

図2　8ロック、8プロセス

lock/unlock1000 回繰り返し
(sms-0.4.4c&sms-0.4.5b&sms-0.4.5c、環境 1)

表３　TreadMarks とのメッセージ量の比較

		SMS	Tmk
単一ページマネージャー	メッセージ量(byte)	226,777	195,598
	メッセージ数(個)	2,090	3,402
分散ページマネージャー	メッセージ量(byte)	65,697	188,158
	メッセージ数(個)	2,196	3,397

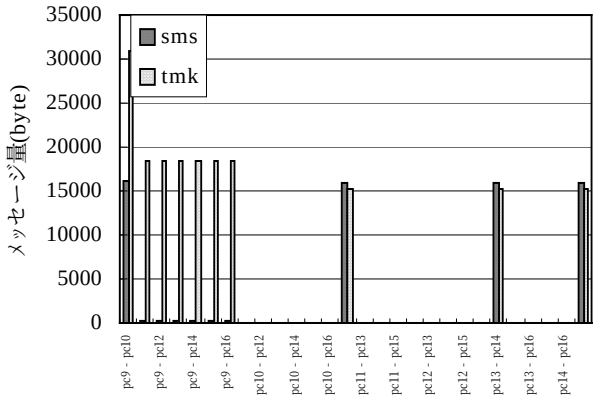

図3 TreadMarksとのメッセージ量の比較

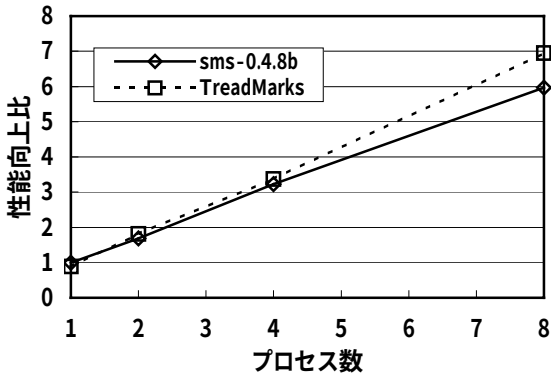

図4 WATERの性能向上比(1728mole　環境2)

・シングルリードライト用バリア関数の効果

評価環境２において、TreadMarksを開発したRice大学のベンチマークSORの処理結果を、TreadMarksと比較した。SORはデータ領域分割による並列処理の典型であるが、シングルリードライト用バリア関数を用いてsigsegvを抑制するとことで、性能が非常に良くなることがわかる。実際に、８プロセス実行時でユーザプログラムコード実行以外の時間に占めるsigsegvハンドラの割合は、49%から9%まで減少した。これは、sigsegvハンドラの時間だけでなく、命令パイプライン、キャッシュなどへの割り込みによる悪影響を抑制したことにもよる。

６．おわりに

新ロック実装方式とsigsegv抑制バリアの効果が高いことを確認した。現在、各種実装方式の構築、コードの最適化などを行っており、さらに、柔軟で、高性能なシステムを目指している。

参考文献

[1]伊藤，緑川，飯塚:"PCクラスタにおける共有メモリの実装（２）-ロックの実装-",情処全国大会 2H-2(2000,3)

[2] 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 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 (2001,8)

[3] 大橋，緑川，飯塚:"PCクラスタにおける共有メモリの(4)ーロック実装の違いによる性能への影響ー”,情処全国大会 2ZB-5(2002,3)

[4] Peter Keleher, et.al.: “TreadMarks: Distributed Shared Memory on Standard Workstations and Operating Systems”, Department of Computer Science, Rice University, Huston, TX 77251-1892

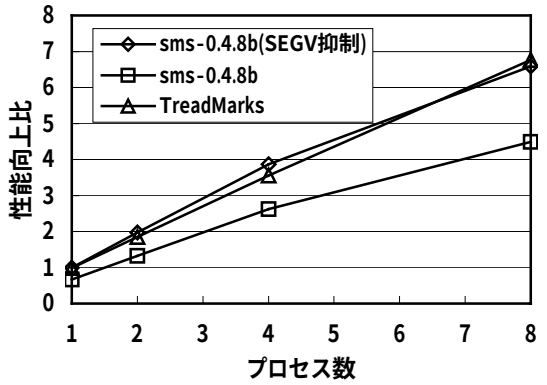

図5 SOR性能向上比(1024*1536　環境2)