

Myrinet 用いた分散共有メモリにおける メモリバリアの実装と評価

原田 浩[†] 手塚 宏史[†] 堀 敦史[†]
住元 真司[†] 高橋 俊行[†] 石川 裕[†]

既存 Unix ベレ ティングシステムと低通信遅延かつ高通信バンド幅を有するネットワークシステム上に SCASH と呼ぶソフトウェア分散共有メモリを開発している。SCASH では、更新プロトコルによるメモリバリアを実行時に選択できる。SCASH ではリモートメモリアクセスによるページ転送の採用により、ページコピーのオーバーヘッドを解消している。その結果、従来のメッセージパッシングによるページ転送との比較で、メモリバリア中のページ転送速度で 2.46 倍、SPLASH2 の LU(CONT) ノード数 64 台における実行速度で、1.89 倍の性能向上を達成している。

Implementation and Evaluation of Memory Barrier on Software Distributed Shared Memory on Myrinet

HIROSHI HARADA,[†] HIROSHI TEZUKA,[†] ATSUSHI HORI,[†]
SHINJI SUMIMOTO,[†] TOSHIYUKI TAKAHASHI,[†]
and YUTAKA ISHIKAWA[†]

We have been developing a software distributed shared memory system called SCASH on top of a Unix with a low latency and high bandwidth network system. SCASH enables the users to select an update protocol or an invalidation protocol for the page update mechanism. The update protocol has been implemented using both the remote memory access and message passing mechanisms supported by the underlying network system. The page transfer latency and SPLASH2 benchmark LU (CONT) using the remote memory access mechanism are 2.46 times and 1.89 times faster than that of using the message passing mechanism, respectively.

1. はじめに

我々はギガビットネットワークの一つである Myrinet¹⁾ 上に低通信遅延かつ高通信帯域幅を提供する高速通信ライブラリ PM^{2),3)} を用いて、オペレーティングシステムのメモリ管理機能を利用した SCASH⁴⁾ と呼ぶソフトウェア分散共有メモリを実現している。

使用しているプラットフォームは 128 台の Intel PentiumPro プロセッサ (200MHz) と Myricom Myrinet ネットワークから構成される PC クラスタで、各ノードプロセッサのカーネルとして Linux⁵⁾ を使用している。

SCASH のメモリバリアは、一貫性維持プロトコルとして、メモリバリア時にページデータを送信し、ページの更新を通知する更新プロトコルと、メモリバリア時にはページ無効化メッセージを送信し、ページにアクセスが行われてから、ページを転送する無効化プロ

トコルを選択できる。

従来の高通信遅延、小通信帯域幅のネットワークを用いてソフトウェア分散共有メモリを実現する場合、通信量の削減によるパフォーマンスの向上を計るために、ページの一貫性維持プロトコルとして、無効化プロトコルを採用する例が多い。

しかしこれまでの我々の測定では低遅延、高帯域幅ネットワークである Myrinet 上の PM 通信ライブラリを用いて、更新プロトコルを用いても、無効化プロトコルに近い台数効果が得られることを確認している⁴⁾。

そこで本論文では、メッセージパッシングとゼロコピー通信の 2 種類のページ転送プロトコルを比較し、更新プロトコルを用いたメモリバリアの詳細を測定、評価する。また行列のラブラシアンを求めるアプリケーションと SPLASH2⁶⁾ の中から LU を用いて比較を行っている。その結果、従来のメッセージパッシングによるページ転送との比較で、メモリバリア中のページ転送速度で 2.46 倍、SPLASH2 の LU(CONT) ノード数 64 台における実行速度で、1.89 倍の性能向上を達成している。

[†] 新情報処理開発機構つくば研究センター
Tsukuba Research Center, Real World Computing
Partnership

本論文の構成は以下の通りである。第 2 章では、SCASH の概要とメモリバリアについて述べる。第 3 章では、ゼロコピーを用いたページ転送の実現と問題点について述べる。第 4 章ではメモリバリアの基礎評価と、メモリバリアで記述された共有メモリアプリケーションを用いて SCASH の評価を行う。第 5 章で評価結果を示し、第 6 章では評価結果を基に、SCASH のメモリバリア、ページ転送方式、問題点について検討する。第 7 章で関連研究を紹介し、第 8 章でまとめと今後の課題について述べる。

2. SCASH

2.1 SCASH の概要

SCASH は、Myrinet¹⁾ 上に低通信遅延かつ高通信帯域幅を提供する高速通信ライブラリ PM^{2),3)} を用いたソフトウェア分散共有メモリである。オペレーティングシステムのメモリ管理機能を利用し、ユーザレベルのライブラリとして実現されている。

共有メモリ領域の一貫性維持は、オペレーティングシステムが提供するページ単位で行われる。一貫性モデルとして ERC (Eager Release Consistency)^{7),8)} を採用し、その実装にはマルチプルライタプロトコル⁹⁾ を用いている。さらに、ページ単位の一貫性維持プロトコルとして、ページの無効化を通知する無効化プロトコルと、ページデータを送信し、ページの更新を通知する更新プロトコルの双方を実装し、実行時に選択できる。

以下に SCASH が提供している機能を示す。

- 共有メモリの初期化、割り当て、開放
共有メモリを全ノード上で、割り当て、開放を行う。SCASH は共有メモリを全ノード上で等しいメモリ空間に割り当てる。共有メモリを割り当てるメモリアドレスは、初期化時に指定可能である。
- 同期機構
SCASH はメモリバリアとロックの 2 つの同期機構を提供している。メモリバリアは、全ノードでバリア同期を取り、共有メモリの全内容が、全ノード上で同一の最新内容に更新されることを保証する同期機構である。ロックはブロッキングロックが提供される。ロックは分散ロックキューによって実現されている。
- 一貫性制御機構
SCASH では、書き込み共有メモリデータのホームノードへの書き戻し、ホームノードからの読み込みなどの、一貫性維持機構の一部をライブラリとしてユーザに開放している。
- その他
SCASH は、グローバルメモリのデータをブロードキャスト、ページ単位でのホームノードの指定等の機能を提供している。特に、局所性が高い共

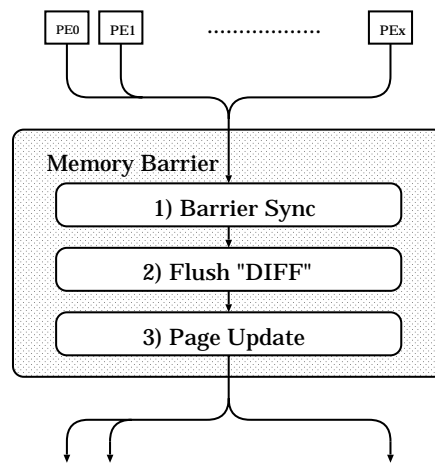


図 1 メモリバリアの実行手順

有メモリ領域に関しては、ホームノードを指定を指定することによって通信量を削減し、実行性能を向上させることができる。

現在 SCASH は、新情報処理開発機構の PC クラスタ 2 号機上で稼働している。PC クラスタのノード数は 128 台であるが、ページディレクトリの制限から、SCASH の実行ノード数は最大 64 台である。

2.2 SCASH のメモリバリア

SCASH のメモリバリアは、一貫性維持プロトコルとして、ページデータを送信し、ページの更新を通知する更新プロトコルを実行時に選択できる。更新プロトコルを用いた、メモリバリアの実行の概要を図 1 に示す。メモリバリアの実行手順は以下の通りである。

- (1) Barrier Sync
最初に全ノード間でバリア同期を取り、全ノードが、共有メモリへのアクセスを中断し、メモリバリアの実行を開始したことを確認する。バリア同期は、PM のメッセージ通信を用いて実現している。
- (2) Flush DIFF
全ノードで、書き込みが行われた全ページについて、書き込みが行われる以前のページと書き込みが行われたページを比較して、差分 (以下 DIFF) を作成する。作成した DIFF はそれぞれの該当ページのホームノードへ送信される。DIFF を送信した後、ページのプロテクションを読み書き可から読み込み可に変更する。ページのプロテクションを読み込み可に変更するのは、ページに対する書き込みアクセスによってページ例外処理ルーチンを起動させるためである。起動されたページ例外処理ルーチンは、ページの複製を作成し、ページプロテクションを読み書き可に戻す。全ての DIFF の送信が終了した時点でバリア同期を取る。

(3) Page Update

各ページのホームノードは、DIFFを受信した全てのページについて、ページディレクトリからページを共有しているノードを検索し、ページを共有しているノードのページを更新する。ページの更新には、メッセージパッシングによるページ転送と、ゼロコピーによるページ転送を選択することができる。全てのページを更新した後に、バリア同期をとって処理を終了する。

3. ページ転送方法

一貫性維持プロトコルとして更新プロトコルを採用した場合、メモリバリア中にページ転送が発生する。本論文では、以下の2つのページ転送方法を実装し、比較する。

3.1 メッセージパッシング

メッセージパッシングによるページ転送は、ページデータを格納したメッセージを送受信することによってページを転送する方法である。

メッセージパッシングによるページ転送では、送信側はページを通信バッファに、受信側は通信バッファからページに、それぞれデータをコピーするオーバーヘッドがある。

3.2 ゼロコピー

PM^{2),3)}では、従来のメッセージパッシングによる通信の他に、送信元のユーザ空間から、送信先のユーザ空間へ直接メモリ転送を行う、ゼロコピー通信をサポートしている。PMのゼロコピー通信では、送信元から送信先へ直接メモリ転送を行うので、ページコピーのオーバーヘッドは発生しない。PMのゼロコピー通信機能を用いたページ転送を実現することによって、ページデータをコピーするオーバーヘッドをなくし、Myrinetの高帯域幅を活かした効率的なメモリバリアが実装できると考えられる。

ゼロコピー通信を行うには、メモリ空間の転送元、転送先の双方が、ピンダウンされていなくてはならない。更に転送元のノードは、メモリ転送時に転送先のピンダウンアドレスを指定しなくてはならない。ページ転送を行うにあたり、逐一、転送元、転送先のページをピンダウンし、ピンダウンアドレスをメッセージパッシングによって交換しては、効率的なページ転送を期待出来ない。

そこで、SCASHでは共有メモリの初期化時に全ての共有メモリ領域をピンダウンし、ピンダウンアドレスを他の全ノードへブロードキャストしている。SCASHでは共有メモリ領域は全ノードで共通のアドレスに配置されているので、ページの送信ノードは、ページのアドレスと初期化時にブロードキャストされたピンダウンアドレスから、送信先のページのピンダウンアドレスを求めることが出来る。そのため、SCASHでは

ページ転送の為に、逐一ページのピンダウンと、ピンダウンアドレスの交換をせずに、即座にゼロコピー通信によりページ転送が可能である。

但し、共有メモリの初期化時に、全共有メモリ領域をピンダウンするため、共有メモリ領域の大きさは、PMがピンダウンできるメモリ空間の大きさに制限されるという欠点がある。

4. 評価方法

4.1 メモリバリアの基本評価

メモリバリアの実行において顕在化する最低限のコストを測定する。

メモリバリアは図1のように実現されているので、以下の3つ、すなわち、共有メモリ領域にアクセスが行われていない場合のメモリバリアのコスト、ページのDIFFを作成送信のコスト、及びページを送信して、ページを更新するコストを測定する。

4.1.1 メモリバリアの最小コスト

PE数ごとのメモリバリアの最小コストを測定する。共有メモリ領域にアクセスを行わずに、単純にメモリバリアを複数回ループ実行した場合のメモリバリアの実行時間を測定し、メモリバリアの最小コストとする。すなわち、DIFFの作成、送信、及びページの更新を全く行わないメモリバリアの実行時間をメモリバリアの最小コストとして測定する。測定はノード数2台から64台まで、メモリバリアを1万回ループさせて行う。

4.1.2 DIFFの作成送信コスト

DIFFを作成してホームノードへ送信する最小コストを測定する。具体的には、1ページについて1ワードのみ更新されたページに対して、DIFFを作成し、作成したDIFFをホームノードへ送信するコストを測定する。すなわち、図1のFLUSH DIFFに相当するコストである。

4.1.3 ページ更新コスト

最後に更新されたページを、ホームノードから共有ノード1つへ送信するコストを測定する。すなわち図1のPage Updateを実行するコストである。この測定は、従来のメッセージパッシングによるページ転送とゼロコピー通信によるページ転送を実行した場合のページ転送コストを測定する。この測定により、メッセージパッシングによるページ転送とゼロコピー通信によるページ転送速度を比較する。

4.2 アプリケーションによる評価

SPLASH2⁶⁾からLU(CONT)、行列のラプラシアンを求めるLAPLACEの2つのアプリケーションの実行速度と、メモリバリアに要するオーバーヘッドを測定する。

測定に用いる問題サイズは表1の通りである。

LU、LAPLACE共にゼロコピー、メッセージパッ

表 1 問題サイズ

LU (CONT)	2048x2048 Matrix, 64x64 Block
LAPLACE	1022 x 1022 Matrix, Iteration 50

シングルの2種類のページ転送を用いた場合の実行時間、及びメモリバリアの実行時間の詳細を測定する。

LU、LAPLACE 共バリア同期のみで記述されているので、メモリバリアとページ例外処理以外のオーバーヘッドは存在しない。

LU、LAPLACE 共にノード数を1台から64台で動作させた場合の実行時間を測定する。ノードが1台の実行時間は、LUはSPLASH2のnullmacroを用いた実行時間、LAPLACEはシングルスレッドでの実行時間を測定した。LU、LAPLACE共にノード数1台の実行では、SCASHのライブラリは呼び出されないで、SCASHのオーバーヘッドは存在しない。

プログラムの測定は全て10回行い、平均値を採用する。LU、LAPLACE共にページ転送が削減されるように、共有メモリ領域中のページのホームノードを、ライブラリを用いて明示的に変更するよう、プログラムを改良している。

4.3 評価環境

測定は全て我々が開発したPCクラスタ2号機上で行う。PCクラスタ2号機の主な仕様を表2に示す。Myrinet用通信ライブラリPMの最小遅延時間と最大帯域幅はそれぞれ、7.2μsec、117.6 MBytes/secである。

表 2 PC クラスタの主な仕様

# of Node	128
CPU	Intel PentiumPro 200Mhz
Cache	512KBytes
Chipset	440FX
Memory	EDO 256MBytes/Node
Network	Myrinet 1.28GBits/sec
Node OS	Linux

5. 評価結果

5.1 メモリバリアの基本評価

メモリバリアの最小コストの測定結果を表2に示す。DIFFの作成送信の最小コストはページ当たり134(μsec)である。

ページ更新のコストの測定結果を表3に示す。表3中のCostは1ページ当たりの転送コストを示す。Bandwidthは、転送コストから求めた、転送帯域幅を示す。測定結果から判るように、ゼロコピーを用いた場合、従来のメッセージパッシングによるページ更新と比較して、2.46倍の性能を示している。

5.2 LU(CONT)

LU(CONT)の実行時間とメモリバリアの実行時間

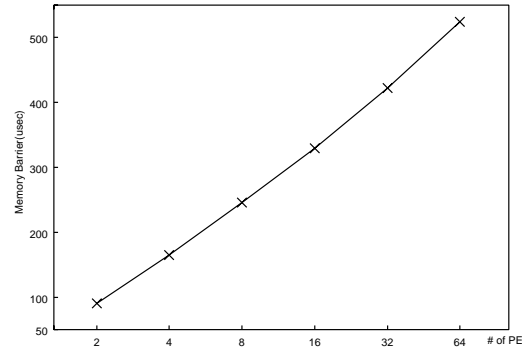


図 2 メモリバリアの最少コスト

表 3 ページ更新コスト

Transfer Method	Cost (usec)	Bandwidth (MBytes/sec)
Message Passing	122	31.9
Zero Copy	49.6	78.8

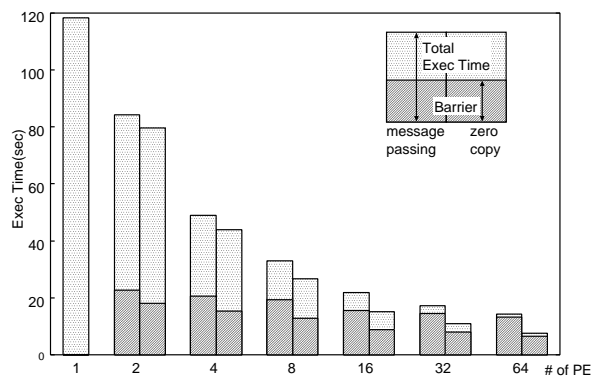


図 3 LUの実行時間

を図3に示す。2本並んでいる縦棒の、左がメッセージパッシング、右がゼロコピー通信によるLUの実行時間である。縦棒は、それぞれがLU全体の実行時間を示している。うち下の部分が、全体の実行時間に占めるメモリバリアの実行時間である。

メモリバリアの実行時間の詳細を図4に示す。2本並んでいる縦棒の、左がメッセージパッシング、右がゼロコピー通信によるメモリバリアの実行時間である。メモリバリアのコストは、下から順に、バリア同期、DIFFの作成送信、ページ更新の3つに分類して示している。メモリバリアの実行回数は33回である。

5.3 LAPLACE

LAPLACEの実行時間とメモリバリアの実行時間を図5に示す。2本並んでいる縦棒の、左がメッセージパッシング、右がゼロコピー通信によるLAPLACEの実行時間である。縦棒は、それぞれがLAPLACE全体の実行時間を示している。うち下の部分が、全体の実行時間に占めるメモリバリアの実行時間である。

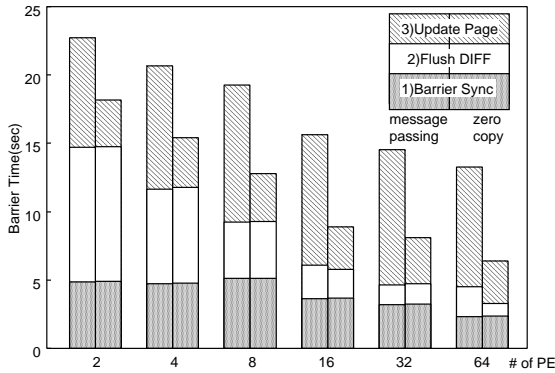


図4 LU:メモリバリアの実行時間

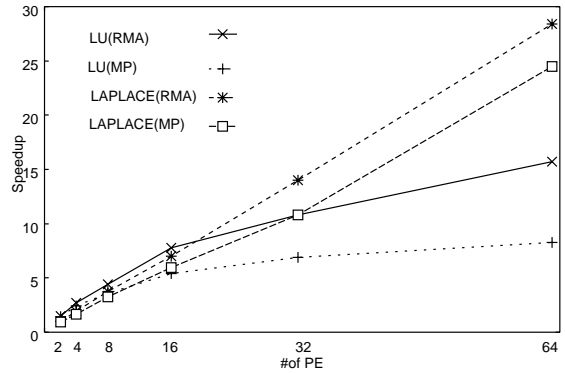


図7 台数効果

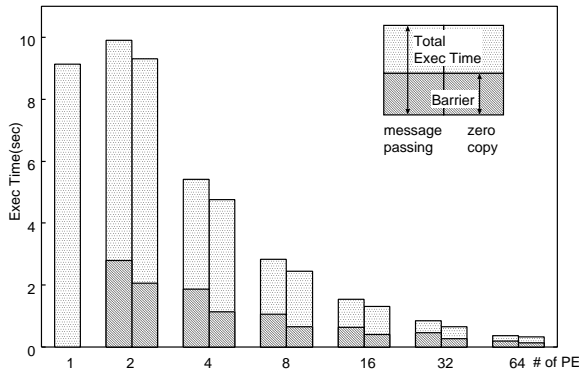


図5 LAPLACEの実行時間

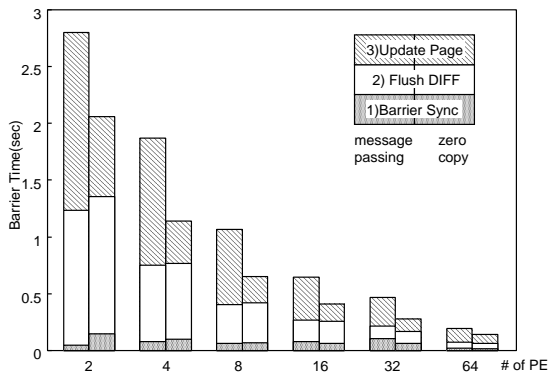


図6 LAPLACE:メモリバリアの実行時間

メモリバリアの実行時間の詳細を図6に示す。2本並んでいる縦棒の、左がメッセージパッシング、右がゼロコピー通信によるメモリバリアの実行時間である。メモリバリアのコストは、下から順に、バリア同期、DIFFの作成送信、ページ更新の3つに分類して示している。メモリバリアの実行回数は50回である。

5.4 台数効果

測定結果から得られたLU、LAPLACEの台数効果を図7に示す。

6. 検 討

6.1 LU

ページ転送方式別に、LUの実行時間を比較する。64ノードで実行した場合、ゼロコピーによるページ転送のほうが、1.89倍高速である。台数効果としては、64ノードの実行で、ゼロコピー、メッセージパッシングでそれぞれ、15.7倍、8.29倍のスピードアップを得ている。

どちらの転送方式を採用した場合でも、ノード数が増加するに従い、実行時間に占めるメモリバリアの実行時間が増加する傾向にある。64ノードの実行では、実行時間に占めるメモリバリアの割合は、ゼロコピー、メッセージパッシングでそれぞれ、84.8%、93.0%である。LUの実行時間は、メモリバリアの実行時間に依存していると言える。

LUの実行時間を短縮するには、メモリバリアの実行時間を短縮しなくてはならない。メモリバリアの実行時間を比較すると、ノード数が増加するにつれて、メモリバリアの実行時間に占めるページ更新の割合が増加していることがわかる。

64ノードでLUを実行した場合、メモリバリアの実行時間のうち、最も大きな時間を占めているのがページ更新の実行時間であり、ゼロコピー、メッセージパッシングでそれぞれ、48.9%、66.0%を占める。

ゼロコピーによるページ転送方式のほうが、ページ更新に要する時間が小さく、LUの性能向上の原因であることがわかる。64ノードでLUを実行した場合、ページ更新に要する時間を比較すると、ゼロコピーを用いた方が、メッセージパッシングに対して、2.79倍高速である。

バリア同期の実行時間については、64ノードでLUを実行した場合に、メモリバリアに占める割合は、ゼロコピー、メッセージパッシングでそれぞれ、37.1%、17.5%である。バリア同期の実行時間が大きいということは、各ノード間の負荷バランスが取れていないを

示している。各ノード間の負荷バランスを取ることで、更なる性能向上が期待できる。

6.2 LAPLACE

ページ転送方式別に、LAPLACEの実行時間を比較すると、64ノードで実行した場合、ゼロコピーによるページ転送のほうが、メッセージパッシングによるページ転送を行った場合との比較で1.16倍高速である。台数効果としては、64ノードで実行した場合に、ゼロコピー、メッセージパッシングでそれぞれ、28.4倍、24.5倍のスピードアップを得ている。

どちらの転送方式を採用した場合でも、ノード数が増加するに従い、実行時間に占めるメモリバリアの実行時間が増加する傾向にある。64ノードの実行では、実行時間に占めるメモリバリアの割合は、ゼロコピー、メッセージパッシングでそれぞれ、44.2%、52.9%である。LAPLACEの性能はLUほどではないが、メモリバリアの実行時間に大きく依存していると言える。

メモリバリアの実行時間を比較すると、ノード数が増加するにつれて、メモリバリアの実行時間に占めるページ更新の割合が増加していることがわかる。64ノードでLAPLACEを実行した場合に、メモリバリアの実行時間に占めるページ更新の割合は、ゼロコピー、メッセージパッシングでそれぞれ、62.4%、54.5%である。

ゼロコピーによるページ転送方式のほうが、ページ更新に要する時間が小さい。そのためメモリバリアの実行時間も、ゼロコピーによるページ転送のほうが短い。64ノードでLAPLACEを実行した場合のページ更新に要する時間を比較すると、ゼロコピーを用いた方が、メッセージパッシングに対して、1.58倍高速である。

また、全実行時間に占めるバリア同期の割合はLUよりも小さく、各ノードの負荷バランスが取れていると言える。

6.3 更新プロトコル

これまでのソフトウェアDSMの実装では、無効化プロトコルが多く採用されてきた。従来の低通信帯域のネットワークを用いてソフトウェアDSMを実現する場合、通信による性能低下を防ぐために、可能な限り通信量を削減する手法が研究され採用されてきた。無効化プロトコルも、通信量を削減する手法の一つである。

しかし、現在ではMyrinet、ギガビットイーサネットに代表されるような、低通信遅延、高通信帯域幅を有する高速ネットワークを容易に利用することができる。こうした、高速ネットワークを用いてソフトウェアDSMを実現する場合、通信量による性能低下を低く抑えられる可能性があるため、更新プロトコルを採用しても十分な性能を得られる可能性がある。実際、ゼロコピー通信を用いた場合、LU、LAPLACEを64ノードで実行させるとそれぞれ15.7倍、28.4倍のス

ピードアップを達成した。

6.4 ゼロコピー通信の無効化プロトコルへの応用

ゼロコピー通信を使ったページ転送は、インバリデイトプロトコルでも有効のはずである。しかし、インバリデイトプロトコルでは、メモリバリア実行中ではなく、アプリケーションが実際にページにアクセスを検出した時点で、ページが転送される。すなわちページを共有するノードがホームノードへページを要求して、始めてページ転送が行われる。

現在のPMでは、ゼロコピー通信は、データの送信ノード側からしか起動できない。すなわち、リモートメモリに対する書き込みは可能であるが、読み込みは出来ない。そのため、無効化プロトコルを採用した場合、現在のPMでは、ページを共有するためにページを受信したいノードは、ホームノードへページ転送を要求し、ページ転送要求を受信したホームノードは、ページ転送の終了を確認した後、要求ノードに対して、ページの転送終了を通知しなくてはならない。

ページ転送を行うために、ページ転送を行うゼロコピー通信の他に、2つのメッセージ通信が必要となってしまう。更に、ホームノードの実行を中断させる必要があるため、ゼロコピー通信の利点を活かし切れないと思われる。リモートメモリ読み込みを用いて、ページを必要とするノードがホームノードから直接ページを読み込めれば、上記の問題を解決できると考えられる。

ゼロコピー通信を用いた無効化プロトコルの高速化は、今後の検討課題の一つである。

7. 関連研究

ソフトウェアによるDSMシステムの性能評価はこれまでも多数行われている。リモートメモリアクセスによるソフトウェアDSMの実現としては、論文¹⁰⁾がある。上記論文ではMBCF¹¹⁾と呼ばれるメモリベース通信機能を用いたソフトウェア分散共有メモリについて論じている。

Myrinetを用いた分散共有メモリの性能評価としては、論文¹²⁾がある。上記論文では、複数のメモリー貫性モデルと、複数のページサイズの粒度についてSPLASH2を用いた性能評価を行っているが、16ノードまでの性能評価に留まっている。また、ページへのアクセスチェックには専用のハードウェア¹³⁾を用いており、SCASHのようにソフトウェアで全てを実現しているわけではない。

マルチプルライタプロトコルを用いたソフトウェア分散共有メモリの性能に関しては、Keleher¹⁴⁾が、シングルライタとSCによるDSMの実装とマルチプルライタとLRCによる性能の比較評価を行っている。上記論文は一貫性モデルとしてLRCを採用しているため、ページの更新に更新プロトコルは採用されてい

ない。

上記の全ての論文は、ノード数が、8 台又は 16 台までの性能評価しか行われていないが、本論文では、64 ノードまでの性能評価を行っている。

8. まとめと課題

最新の高速ネットワーク技術を用いて、SCASH と呼ぶソフトウェア分散共有メモリを PC クラスタ上に開発している。

SCASH はページの一貫性維持プロトコルとして、更新プロトコルを選択できる。更新プロトコルを採用した場合、メモリバリア中にページ転送を行う必要がある。メモリバリア中に行われるページ転送を、ゼロコピー通信、メッセージパッシングの 2 種類の通信方法で実現し、比較、評価を行った。

メモリバリア、ページ転送の評価として、メモリバリアの基本的なコストの計測と、アプリケーションによる、比較評価を行った。アプリケーションには、行列のラプラシアンを求める LAPLACE と SPLASH2 の LU を採用した。

メモリバリア中のページ転送では、ゼロコピー通信によるページ転送の方が、メッセージパッシングによるページ転送の 2.46 倍高速である。

従来の高通信遅延、低通信帯域幅のネットワークでは、ほとんど採用されることがなかった更新プロトコルであるが、LAPLACE、SPLASH2 の LU の評価では、64 ノード、メッセージパッシングによるページ転送でそれぞれ、24.5 倍、8.29 倍の台数効果を得ることが出来た。更に、ゼロコピー通信によるページ転送を採用することによって、LAPLACE、LU それぞれ、64 ノードの実行で 28.4 倍、15.7 倍の台数効果を得ることが出来た。

以上の通り、性能向上に有効なゼロコピー通信であるが、共有メモリ領域の初期化時に一括して共有メモリ領域をピンダウンしているため、共有メモリ領域の大きさが、PM がピンダウン可能なメモリの大きさに制限されるという問題点もある。メモリが廉価になり、100メガバイト単位のメモリを搭載する計算機が当たり前となっている現状では、上記の制限を取り除き、かつ、効率的なページ転送を実現する手段が求められる。

更新プロトコルだけでなく、無効化プロトコルにおいてもゼロコピー通信によるページ転送を採用し、性能向上を求めることが課題となる。

今回の測定では、LAPLACE、LU 共にノード数 64 台までの性能評価を行い、64 台まで台数効果が得られることを確認した。SCASH の動作ノード数の制限を緩和し、更に大規模なクラスタシステムで性能評価を行うことも今後の課題となる。

謝 辞

本研究における実装、評価に関して貴重な御意見を頂いた新情報処理開発機構工藤 知宏氏、山本 淳二氏に感謝いたします。

参 考 文 献

- 1) <http://www.myri.com>.
- 2) Hiroshi Tezuka, Atsushi Hori, Yutaka Ishikawa, and Mitsuhiro Sato. PM: An Operating System Coordinated High Performance Communication Library. In *High-Performance Computing and Networking '97*, 1997.
- 3) 手塚, 堀, O'Carroll, 原田, 石川. ピンダウン キャッシュを用いたユーザレベルゼロコピー通信. 情報処理学会研究報告. 情報処理学会, August 1997.
- 4) 原田, 手塚, 堀, 住元, 高橋, 石川. Myrinet を用いた分散共有メモリシステムの評価. ハイパフォーマンスコンピューティング研究会資料, 98-HPC-73, pp. 73-78. 情報処理学会, October 1998.
- 5) <http://www.linux.org>.
- 6) Steven Cameron Woo, Moriyoshi Ohara, Evan Torrie, Jaswinder Pal Singh, and Anoop Gupta. The SPLASH-2 Programs: Characterization and Methodological Considerations. In *In Proceedings of the 22nd International Symposium on Computer Architecture*, pp. 24-36. ACM, June 1995.
- 7) K. Gharachorloo, D. Lenoski, J. Laudon, P. Gibbons, A. Gupta, and J. Hennessy. Memory Consistency and Event Ordering in Scalable Shared-Memory Multiprocessors. In *Proceedings of the 17th Annual Symposium on Computer Architecture*, pp. 15-26, May 1990.
- 8) J.B Carter, J.K. Bennett, and W. Zwaenepoel. Implementation and Performance of Munin. In *Proceedings of the Thirteenth Symposium on Operating Systems Principles*, pp. 152-164, October 1991.
- 9) Amza C. Cox A. L. Dwarkadas, S. and Zwaenepoel W. Software DSM Protocols that Adapt between Single Writer and Multiple Writer. In *Proc. of the 3rd IEEE Symp. on High-Performance Computer Architecture (HPCA-3)*, pp. 261-271, February 1997.
- 10) T. Matsumoto, T. Komaarashi, S. Uzuhara, and K. HIRAKI. The Asymmetric Distributed Shared Memory Using Memory-Based Communication Facilities (in Japanese). 情報処理学会 コンピュータシステムシンポジウム, pp. 37-44. 情報処理学会, November 1996.
- 11) Takashi MATSUMOTO and Kei HIRAKI. M

B C F : a protected and virtualized high-speed user-level memory-based communication facility. In *1998 International Conference on Supercomputing*, pp. 259–266, Melbourne, June 1998. ACM.

- 12) Y. Zhou, L. Iftode, J.P. Singh and K. Li, B.R. Toonen, I. Schoinas, M.D. Hill and D.A. Wood. Relaxed Consistency and Coherence Granularity in DSM Systems: A Performance Evaluation. In *Proceedings of the 6th ACM Symposium on Principles and Practice of Parallel Programming*, June 1997.
- 13) Robert W. Pfile. Typhoon-Zero Implementation: The Vortex Module Technical Report. Technical report, Wisconsin University, CS department, 1995.
- 14) P.J. Keleher. The relative importance of concurrent writers and weak consistency models. In *Proceedings of the IEEE COMPCON'96 Conference*, February 1996.