PC クラスタにおける Ethernet による 高速ユーザレベルバリアの性能評価

岩崎 聖[†] 松岡 聡[†]
栄 純明[†] 小川 宏高[†]

PC クラスタ、特にバリア同期のように集中的な通信が求められる高性能なクラスタには、Myrinet などの高価で低レイテンシかつ高バンド幅のネットワークが不可欠だと信じられている。我々はこれらのネットワークを用いた場合と同等の性能のバリア同期をコモディティネットワーク、特に Fast Ethernet 上で実現するために、マルチキャストや複数ネットワークを利用する方法を提案し、詳細な検討を行った。我々はまず、VIA のように NIC に直接アクセスすることで低レイテンシを実現するとともに、マルチキャスト機能をサポートした実験用通信ライブラリを実装した。その上で、複数のバリアアルゴリズムを実装し、性能評価を行った。ベンチマークの結果、32 ノード時に Shuffle Exchange アルゴリズムでは 170μ 秒と Myrinet と比較して十分高速であった。マルチキャストを利用した場合には現状では 200μ 秒以上かかってしまったが、LogP モデルを用いて理論的に分析した結果、ライブラリの設計を見直すことで Shuffle Exchange より高速なバリアが達成できるという指針を得た。これらの結果から、コモディティネットワークはクラスタにおいて十分な性能を発揮し、より低コストにクラスタを構築できるという結論に達した。

Evaluation of Fast Barrier Synchronization on commodity PC Cluster connected with Ethernet

Satoru Iwasaki,[†] Satoshi Matsuoka,[†] Yoshiaki Sakae[†] and Hirotaka Ogawa[†]

It is still a typical belief that high-performance clusters require expensive networks with low latency and high bandwidth such as the Myrinet, especially for communication-intensive situations such as barrier synchronization. In order to achieve similar level of performance in barrier synchronization with commodity networks, in particular Fast Ethernet, we propose and investigate the design space of using multicasts and multiple networks. Our experimental library employs VIA-style low-latency access to Ethernet cards as well as supports multicasts, both of which are employed to construct several barrier algorithms. Benchmarks show that the Shuffle Exchange algorithm on our library can be low as 170μseconds with 32 nodes, almost matching the best performance on Myrinet. Although the use of multicast is found to be currently slower with 200μseconds, theoretical analysis using the LogP model reveals that better design of the library will likely yield even lower latency than Shuffle Exchange. The results show that commodity networks are sufficient for clustering, allowing lower cost and their wider acceptance as a result.

1. はじめに

近年、並列計算の分野では高価な超並列計算機に比べ、安価に構築できるクラスタ型計算機への関心が高くなっている。クラスタの各ノードの接続には比較的低速なネットワークを用いるため、超並列計算機に比べるとその性能はまだ水をあけられており、Myrinet[1)] や Gigabit Ether などの高速なネットワークを用いることでこのボトルネックを改善する方法が一般的にとられている。しかし、これらのネットワークの導入コストはまだまだ高く、安価に構築できるというクラスタの利点が半減してしまう。

我々は、より安価な、コモディティなネットワークで接続されたクラスタ上において、ソフトウェア技術で通信ボトルネックを低減することで、高速ネットワークを用いたクラスタや超並列計算機とどの程度比肩するパフォーマンスが得られるか明らかにするために様々な角度から研究を行っている。

本稿では Fast Ethernet で接続されたクラスタ上において、バリア同期をどの程度まで高速化できるかを研究した結果についての報告および考察を述べる。

分散メモリ型計算機上では Shuffle Exchange アルゴリズムを用いたバリア同期が一番効率良いとされて

† 東京工業大学
Tokyo Institute of Technology

いる。Myrinet で接続されたワークステーションクラスタ上で Shuffle Exchange を用い、32 ノードのバリアが約 90μ 秒で達成できるという研究報告もある [2]。

我々は Fast Ethernet 上において高速なバリア同期を実現する方法として Ethernet のマルチキャストを利用することに着目した。マルチキャスト利用のバリア同期は GAMMA[5)~8)] で取り入れられていたが、性能評価は Repeater hub で接続されたクラスタ上でのみ行われており、マルチキャストを利用しないバリアとの比較はなされていなかったため、バリア同期の高速化においてマルチキャストが有効であるかは明らかにされていなかった。この点を明らかにするべく、マルチキャスト送受信可能な低レベル通信ライブラリを構築、その上で数種のマルチキャストバリアを実装、Shuffle Exchange バリアとの性能比較を行った。また、クラスタの各ノードに複数の NIC を備えつけ、同時に扱えるパケット数を増やすことでより効率良くバリアのメッセージ伝達が行えるのではないかと考え、この点についても検証した。

その結果、Fast Ethernet 上でも低レイテンシな通信ライブラリを用いることで、32 ノードでのバリアが Shuffle Exchange アルゴリズムでは 170μ 秒程度、マルチキャスト利用 2 分木バリアでは 200μ 秒程度で行えることが分かった。

今回実装した通信ライブラリ上ではマルチキャスト・複数ネットワークの利用ともに目立った効率向上は得られなかった。しかし、LogP パラメータを用いた考察の結果、ライブラリの受信オーバヘッドを削減することでより効率の良いマルチキャストバリアを実現できるという指針を得た。

2. 低レベル高速通信ライブラリの実装

本研究では Ethernet のマルチキャストを利用したバリアアルゴリズムの性能評価を中心に行った。マルチキャストがサポートされている低レイテンシライブラリが見つからなかったこと、性能評価にあたりライブラリの実装の詳細について知っている必要があったことから、マルチキャストをサポートする低レイテンシ通信ライブラリを実装し、その上で各バリアアルゴリズムの実装、性能評価を行った。この節では、この通信ライブラリの実装について説明する。

2.1 通信ライブラリ MCD の概要

通信ライブラリの実装に際しては、マルチキャスト送信の可能な高バンド幅・低レイテンシな通信ライブラリを想定し、VIA[3),9)] の Linux 上での実装である M-VIA[4] を参考に実装を行った。

この通信ライブラリ（以下 MCD と記述）は、Linux の tulip NIC 用デバイスドライバに若干手を加えたドライバ部 (mcdtulip)、ライブラリが使用するリソースの管理などカーネルレベルの処理を行うカーネルモジュール部 (mcdk)、及びユーザアプリケーションにプログラミングインタフェースを提供し、カーネルモジュール部にシステムコールを通じて各種の要求を発行するユーザレベルライブラリから構成される (図 1)。

2.2 MCD での送信

アプリケーションは、ピンダウンされたユーザ空間のバッファにメッセージを作成し、このバッファを指すデスクリプタをデバイスの送信キューに post することで送信要求を行う。デスクリプタが post されると、デスクリプタの指し示すバッファからメッセージが直接 NIC 内の FIFO バッファへ DMA 転送され、順次ネットワークへ送出される（ゼロコピー送信）。各ノードは NIC の MAC アドレスによって番号付けされており、送信先には対応するノード番号と、受信側のポート（後述）を指定する。メッセージをマルチキャストで送信する場合には、宛先ノード番号に MCD_DST_MULTICAST を指定する。

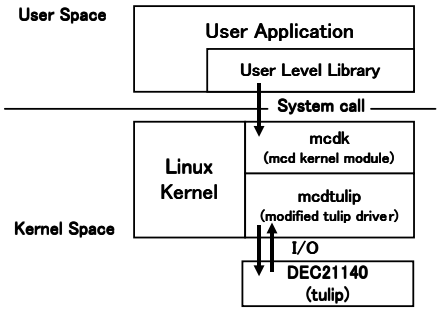

図 1 MCD 概念図

メッセージはノンブロッキング送信されるが、メッセージが完全に送信されるまでバッファを再利用することはできない。NIC はメッセージの送信を終えるごとに割り込みをかけ、割り込みハンドラ内でそのメッセージに対応するデスクリプタの送信済みフラグをセットする。ユーザはこの送信済みフラグがセットされたのを確認してからバッファ・デスクリプタを再利用することになる。送信済みフラグをチェックしバッファが再利用可能になるのを待つための API もライブラリに実装されている。

2.3 MCD での受信

メッセージの受信には、ポートの概念を導入する。各デバイスは 256 個のポートを持っており、アプリケーションは利用するポートを番号で指定し獲得しておく。各ポートは唯一つのプロセスが獲得することができ、到着したメッセージをどのプロセスのバッファにコピーするべきなのか判別するのに利用される。NIC はパケットを受け取るとカーネル内部の一時バッファへ DMA 転送を行い、割り込みをかける。割り込みハンドラでは到着したパケットのヘッダを調べ、MCD のパケットであった場合にはヘッダに書かれているポート番号を参照し、そのポートを獲得しているプロセスの受信バッファへとメッセージのコピーを行う（1 コピー受信）。MCD のパケットでない場合にはカーネルによって通常の処理が行われるようになっている。これにより、TCP/IP などのパケットと MCD のパケットの両方を同時に扱えるようになっている。

MCD ではフロー制御など信頼性に関する機能は一切実装しておらず、ネットワーク中でパケットがロストしてもアプリケーション側には通知されない。また、メッセージが届いた際に、宛先のポートに受信用バッファが用意されていない場合にはそのメッセージは捨

てられてしまう。

実際のアプリケーションで利用するバリアを実装する場合には Ack/Nack などによるフロー制御を行う必要がある。マルチキャストメッセージの信頼性を保証するのには若干の工夫が必要となるが、これに関してはいくつかのアルゴリズムを検討しており、特にメッセージがバリアを追い越さないことを利用した coloring による手法を検討中である。この手法を用いた場合のバリアに要する追加コストはほとんどないと思われる。

3. 実装したバリアアルゴリズム

この節では実装したマルチキャスト・複数ネットワークを利用したバリアアルゴリズムについて説明する。以降の説明では、ノード番号の一番若いノードをマスターノード、それ以外をスレーブノードと呼ぶ。

3.1 非マルチキャスト利用バリア

3.1.1 Shuffle Exchange

分散メモリ型計算機上では一番効率が良いとされているアルゴリズム。シャッフル交換の関係にあたるノード同士の 2 入力バリアを $log_2(n)$ 段接続する形でメッセージの交換が行われる。

3.2 マルチキャスト利用バリア

3.2.1 2 分木

"enter barrier" メッセージを 2 分木の形で集める。各スレーブノードは自分の子ノードからメッセージを受け取り、それを自分の親ノードに伝えていく。最終的にマスターノードが全てのノードのメッセージを受け取り、バリアの達成を "exit barrier" メッセージをマルチキャストすることで全スレーブノードに伝える。

2 分木ではマスターノードがスレーブノードの全メッセージを集めるのに $log_2(n)$ フェーズ、マスターノードからスレーブノードへのマルチキャストに 1 フェーズかかる。$log_2(n)$ のオーダーでバリアの達成が期待でき、Shuffle Exchange 同様比較的効率が良い。

また、Shuffle Exchange と比べ各フェーズの通信量が少ないため、Repeater hub で接続されたネットワークにおいてもノード数がある程度少なければ良い効率が期待できる。

3.2.2 3 分木

前述の 2 分木バリアよりもさらに通信フェーズ数を減らせるよう、二つの子ノードからのメッセージを親ノードが集めていく 3 分木型のアルゴリズム。

このアルゴリズムではマスターノードがスレーブノードからのメッセージを集めるのに $log_3(n)$ フェーズですむ。しかし、親ノードが 2 つの子ノードからのメッセージを受信することで、2 分木に比べ各フェーズの実行時間が長くなり、結果としてあまり性能が変わらない、あるいは性能が低下することも考えられる。

このアルゴリズムでは 2 分木にくらべ各フェーズでネットワーク内を流れるメッセージの量が増えるため、Repeater hub で接続されたネットワークにおいては 2 分木よりも性能の低下が予想される。

3.2.3 Concurrent

各スレーブノードはバリアに入ると一斉にマスターノードに "enter barrier" メッセージをポストする。

通信フェーズが最小で済むため、ロードバランスが悪いときには非常に良い効率が得られる。しかし、ロードバランスが良い場合にはマスターへのメッセージの到着が逐次化され、オーダーが n になってしまうため効率が低下する。Repeater hub を用いた場合にはコリジョンの発生によりさらに性能が悪くなる。

3.3 複数ネットワーク利用バリア

3.3.1 3 分木 with 2 networks

3 分木アルゴリズムにおいて、親ノードが 2 つの子ノードからメッセージを受け取る際に、別々のネットワークを利用することで親ノードの NIC の負荷減少を期待する。メッセージの受信をうまくパイプライン化できれば Shuffle Exchange と同等、あるいはそれ以上の性能が期待できるものと思われる。

3.3.2 Concurrent with 2 networks

Concurrent アルゴリズムにおいて、マスターが各スレーブノードからのメッセージを受け取る際に、複数のネットワークを利用することでネットワーク負荷を分散し、性能の向上が期待できるものと思われる。

4. 通信ライブラリおよびバリアの性能評価

4.1 評価環境

まず、今回の実験を行った環境について説明する。以降で述べる通信ライブラリの性能評価、およびバリア同期の性能評価は、表 1 に示すような PC 32 台を Fast Ethernet で接続した PC クラスタ上で評価を行った。各ノードは実験用に 2 つの tulip NIC を備えており、それぞれが独立したネットワークに接続されている。1 枚目の NIC は 32 ポートの Switching hub で接続されており、もう 1 枚の NIC は実験内容に応じて 32 ポートの Repeater hub または 16 ポートの Switching hub で接続した (表 2)。

CPU	Pentium II 350MHz
L1 Cache size	32KB(16KB+16KB)
L2 Cache size	512KB
Chipset	440BX
Memory	256MB
OS	Red Hat Linux 6.1 Kernel 2.2.14
NIC	Intel i82559(本実験では未使用)
	DEC 21140(tulip) × 2

表 1 ノード性能

Switch	PLANEX FHSW-3232NW
	PLANEX FHSW-1616NW
Repeater hub	PLANEX FDX-32S

表 2 ネットワーク環境

4.2 ライブラリの性能評価

Switching hub または Repeater hub で接続された 2 ノード間において通信ライブラリ MCD および M-VIA を用いて Ping-Pong 転送を行い、そのラウンドトリップタイム (RTT/2)、バンド幅を計測した結果を図 2 および図 3 に示す。MCD は M-VIA とほぼ同等の低レイテンシを達成しており、0 バイトメッセージ時のレイテンシは 30μ 秒ほどである。

なお、バンド幅のグラフで Switching hub の場合

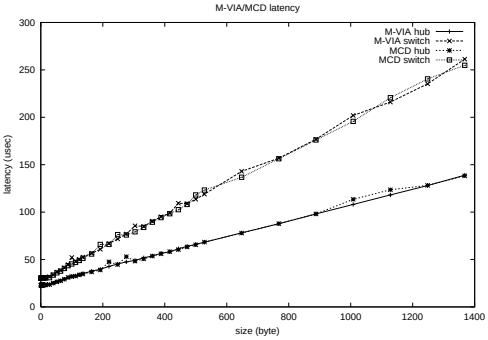

図 2 MCD/M-VIA レイテンシ (RTT/2)

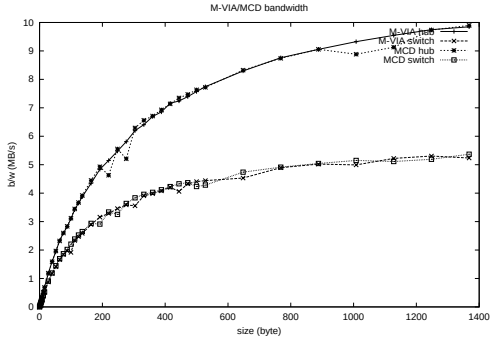

図 3 MCD/M-VIA バンド幅

に Repeater hub の半分程度の性能に見えるが、これは計測しているメッセージサイズが小さい (MCD が Ethernet の MTU(1500 バイト) までしか対応していない) ためであり、M-VIA においてはメッセージサイズを大きくしていくと差はほとんどなくなる。

4.3 マルチキャスト利用バリアの性能評価

以下では MCD 上に実装した各バリアアルゴリズムのマイクロベンチマークによる評価結果を示す。このベンチマーク (図 4) では、バリアと短いダミーループを 100 回繰り返す。関数 rdtsc() は Pentium のクロックカウンタを取得する関数で、これによりバリアに要した時間を計測する。各ノードで、100 回のバリアのうち最大、最小、および平均コストを求め、これらの値の全ノードでの平均をグラフに示した。

```
for (i = 0; i < 25; i++){
    rdtsc(start);
    barrier();
    rdtsc(end);
    dummy_loop(ABOUT_30_usec);
}
```

図 4 ベンチマークコード

4.3.1 Shuffle Exchange

図 5 が Shuffle Exchange を測定した結果のグラフである。Switching hub においては 32 ノードで平均 174μ 秒で、Myrinet に比較しても良い結果が得られている。Repeater hub においてはノード数が増えるにつれて大きく性能が低下しており、32 ノードで平均 3743μ 秒と Switching hub の場合の 22 倍以上遅い。

4.3.2 2 分木

図 6 は 2 分木アルゴリズムを測定した結果である。ノード数が少ない場合は Repeater hub でも Switching hub とほぼ同程度の性能が得られた。しかし、ノード数が増えるとコリジョンのために性能低下が起こっており、32 ノード時では Switching hub の平均 204μ 秒に対し Repeater hub では 728μ 秒である。全体的に Shuffle Exchange より劣っており、バリアの達成に 2 分木のほうが Shuffle Exchange よりも 1 フェーズ通信段数が多くかかることがそのまま表れている。

4.3.3 3 分木

図 7 が 3 分木バリアの測定結果である。2 分木バリアよりよい性能が得られるかと期待したが、実際には 2 分木バリアと同等の性能を示し、31 ノード時 Switching hub では平均 200μ 秒、Repeater hub では平均 675μ 秒であった。32 ノードになった場合、木のどこかでもう 1 ノード分のメッセージ受信が増えるため、もう 10μ 秒ほどコストが増すものと思われる。

4.3.4 Concurrent

図 8 が Concurrent バリアの測定結果である。ノード数に応じて linear にコストが増加していくことが見て取れる。Repeater hub ではノード数が増えるとコリジョンによって性能が低下している。32 ノードでは Switching hub において 309μ 秒、Repeater hub で 1059μ 秒と他のアルゴリズムと比較するとあまり性能が良いとはいえない。

4.4 複数ネットワーク利用バリアの性能

4.4.1 3 分木 with 2 networks

3 分木バリアにおいて、親ノードが 2 つの子ノードからのメッセージを受け取る部分を別々のネットワークを経由させることでさらに効率が上がることを期待していたが、結果は図 9 のようになった。

実験環境の都合上 Switch 2 つでの測定は 16 ノードまでしか行えなかったが、15 ノード時にネットワーク 1 系統では平均 159μ 秒だったのに対し、2 系統では 169μ 秒と若干性能が落ちている。

4.4.2 Concurrent with 2 networks

Concurrent についても 3 分木同様、ネットワークを複数系統使用することで性能の向上を期待していた。結果は図 10 のようになり、若干ではあるが性能の向上が見られるものの、Shuffle Exchange 並みの性能とまでは行かなかった。16 ノード時に 1 系統では 179μ 秒、2 系統では 147μ 秒となった。

5. バリア同期の評価結果についての考察

5.1 マルチキャストバリアについての考察

マルチキャストバリアの評価結果を行うにあたり、通信ライブラリの性能を LogP パラメータを用いてさらに細かく分析 [10),11)] する。

LogP モデルでは、Point-to-Point のメッセージ転送の性能を、**L**atency - 送信側ネットワークインタ

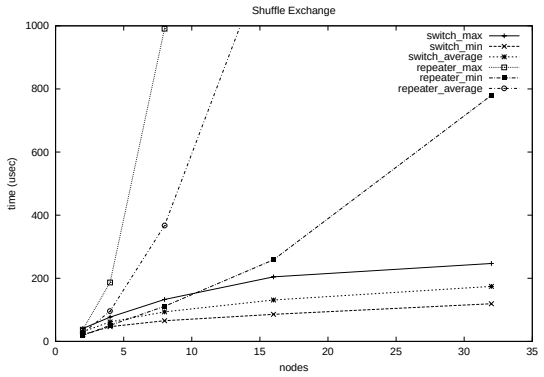

図 5 Shuffle Exchange

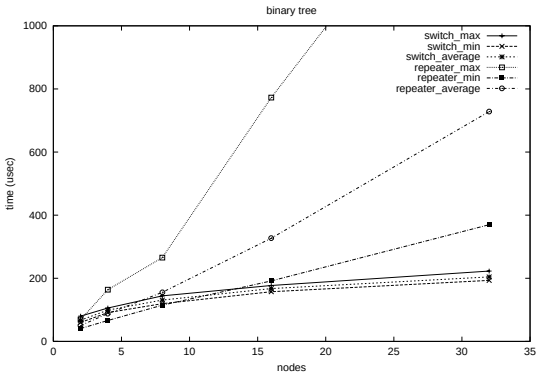

図 6 2 分木

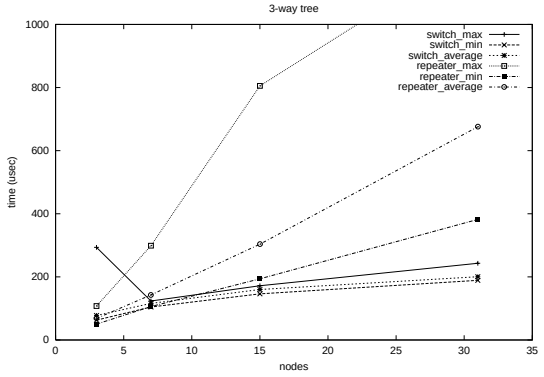

図 7 3 分木

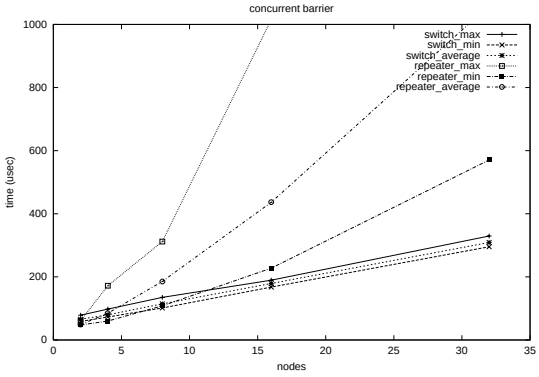

図 8 Concurrent

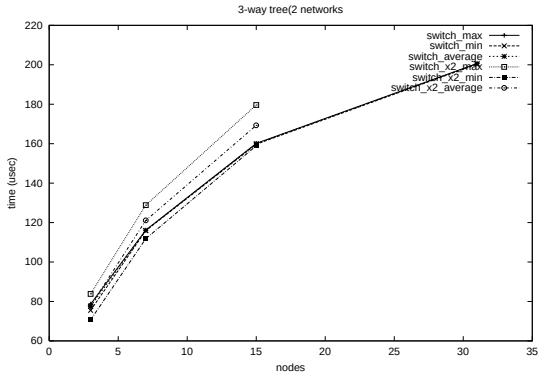

図 9 3 分木 (switch x 2)

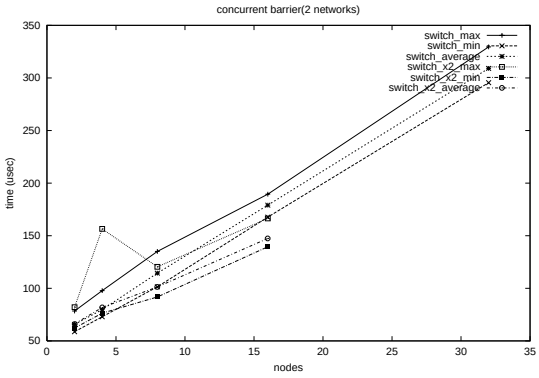

図 10 Concurrent(switch x 2)

フェースから受信側ネットワークインタフェースまで転送するのにかかる時間、**o**verhead - プロセッサが各メッセージの送信 (o_s) もしくは受信 (o_r) に従事している時間、**g**ap - プロセッサが連続した送信または受信を行う際の最小の時間間隔、および **P**rocessor - プロセッサ数、の 4 つのパラメータで特徴づける。これらパラメータを用いると、2 ノード間での Ping-Pong 転送のラウンドトリップレイテンシ RTT/2 は $RTT/2 = o_s + L + o_r$ と表すことができる。

5.1.1 LogP モデルによる通信ライブラリの分析

通信ライブラリ MCD に LogP モデルを適用した場合、4 バイトメッセージ時における RTT/2、約 30μ 秒 (Switching hub 接続の場合) を次のように分割できることが計測した結果から分かった。

o_s 非同期送信時のオーバヘッドは約 3μ 秒。システ

ムコールのオーバヘッドと、DMA 発行コストが送信側の CPU にかかる。

L NIC - NIC 間のレイテンシは約 17.5μ 秒。この間に、ユーザバッファ内のデータが tulip の Tx FIFO へ DMA 転送され、Tx FIFO からリンク・Switch/Repeater を経由して受信側 NIC の Rx FIFO へ送られ、Rx FIFO からカーネル内の一時バッファに DMA 転送される。これらの処理には送信側・受信側どちらの CPU も関与しない。

o_r メッセージ到着時のオーバヘッドは約 9.5μ 秒。メッセージの到着は割り込みによって通知され、かなりのコストがかかっている。

gap ライブラリの現実装ではメッセージが NIC の FIFO から転送されるたびに、対応するデスクリプタの送信完了フラグをセットするために NIC からホスト CPU に割り込みがかけられ、約 6μ 秒のオーバヘッドが生じる。連続送信時にもこの割り込みが定期的にかかるため、送信間隔は o_s+割り込みオーバヘッド=9μ 以上となる。

5.1.2 マルチキャストバリアの分析

LogP パラメータを用いて各バリアに要するコストを記述することを試みる。Shuffle Exchange の場合は 2 入力バリアが $log_2(n)$ 段接続され、このコストは $(o_s + L + o_r) \times log_2(n)$ となる。2 分木バリアでは同様に $(o_s + L + o_r) \times (log_2(n) + 1)$ となり、オーバヘッド o_s, o_r を減らすことができたとしても理論上 Shuffle Exchange より高速になることはあり得ない。3 分木バリアのコストは $(o_s + L + 2o_r) \times log_3(n) + (o_s + L + o_r)$。さらに、4 分木バリアを考えると、そのコストは $(o_s + L + 3o_r) \times log_4(n) + (o_s + L + o_r)$。

この 4 分木バリアにおいて、$n = 16$ のときを考えると、Shuffle Exchange では $4(o_s + L + o_r)$ のコストを要することになる。一方、4 分木では、$2(o_s + L + 3o_r) + (o_s + L + o_r) = 3(o_s + L + o_r) + 4o_r$ となる。これより、$3o_r < o_s + L$ の場合、4 分木のほうが Shuffle Exchange よりコストが低くなるという予測ができる。現状のライブラリでは $o_s = 3\mu, L = 17.5\mu, o_r = 9.5\mu$ であるためこの式は成り立たないが、メッセージの受信に割り込みを用いないよう実装し o_r を低減すれば、Shuffle Exchange バリアより 4 分木バリアのほうが効率がよくなる可能性がある。

5.2 複数ネットワーク利用バリアについての考察

本稿で性能評価を行った複数ネットワーク利用バリアでは、いずれも複数のメッセージを受け取る部分の並列化を試みていた。しかし、3 分木での複数ネットワーク利用バリアでは NIC を 2 枚利用し、各 NIC が 1 つずつメッセージを受信していた。この場合、それぞれの NIC から別々に割り込みが発生することになり、1 枚の NIC で 2 つのメッセージを処理する場合よりもかえって効率が低下していることになる。

一方、Concurrent での複数ネットワーク利用の場合は、片方の NIC からの割り込みによる処理を行っている間に、もう 1 方の NIC に複数のメッセージが溜まり、その NIC から発生する割り込みの回数が減少するために効率が上がったのではないかと予想する。

6. まとめ

本稿では Fast Ethernet 上で高速なバリア同期を実現するための手法としてマルチキャスト、複数ネットワークの利用に注目し、これらの有効性について検証した。マルチキャストを用いない Shuffle Exchange バリアでは 32 ノード時に 170μ 秒、マルチキャストを用いた中で一番高速だった 2 分木バリアでは 200μ 秒とわずかにおよばなかったが、これらの結果は Myrinet での結果と比較しても十分高速であると言える。

また、ライブラリの受信オーバヘッドを削減することで、ツリーバリアの効率をより向上させることが可能であると指針が得られたため、さらに高速なバリアを Fast Ethernet 上で実現できる可能性も期待できる。今後の課題としては、実際に受信オーバヘッドを削減したライブラリを用いた場合の Shuffle Exchange バリアとマルチキャスト利用バリアとの性能比較を行う予定である。また、リライアブルなバリアを実現するために coloring の手法を導入し、その性能の評価も行う予定である。

参考文献

1) http://www.myri.com/
2) 田中良夫, 久保田和人, 佐藤三久, 関口智嗣. 並列アルゴリズムにおける Collective 通信の性能比較. 情報処理学会研究報告, 96-HPC-62, pp.19-26, Aug. 1996.
3) http://www.viarch.org/
4) http://www.nersc.gov/research/FTG/via/
5) http://www.disi.unige.it/project/gamma/
6) G. Chiola and G. Ciaccio, GAMMA: a Low-cost Network of Workstations Based on Active Messages. In PDP'97 (5th EUROMICRO workshop on Parallel and Distributed Processing), January 1997.
7) G. Chiola and G. Ciaccio, Fast Barrier Synchronization on Shared Fast Ethernet. In CANPC'98 (Workshop on Communication, Architecture, and Applications for Network-based Parallel Computing, in conjunction with HPCA-4), February 1998.
8) G. Ciaccio, A Communication System for Efficient Parallel Processing on Clusters of Personal Computers. PhD Thesis DISI-TH-1999-02, 1999.
9) Intel Virtual Interface (VI) Architecture Developer's Guide. (Intel)
10) D.Culler, L.Liu, R.P. Martin, and C. Yoshikawa. LogP performance assessment of fast newwork interfaces. *IEEE Micro*, 1996.
11) Soichiro Araki, Angelos Bilas, Cezary Dubnicki, Jan Edler, Koichi Konishi, and James Philbin. User-Space Communication: A Quantitative Study. SuperComputing, November 1998.