計算機アーキテクチャ 119−15
（1996. 8. 27）

汎用超並列オペレーティングシステムと協調動作するギガビットネットワーク

國澤 亮太　　松本 尚　　平木 敬

東京大学 大学院理学系研究科 情報科学専攻

我々は汎用超並列オペレーティングシステムと協調動作することを目的とした高速かつ高機能なギガビットスイッチングネットワークを開発中である。マルチユーザ / マルチジョブ環境におけるユーザ間の通信に必要な機能をハードウェアで実現することにより、通信オーバヘッドが大幅に削減され、ワークステーションクラスタ環境においても効率の良い並列実行環境を得ることができる。本稿ではSunワークステーション用に実装したネットワークインターフェイスカードの鍵となる機能について述べる。

Gigabit Network with Cooperative Functions for General Purpose Massively Parallel OS

Ryota Kunisawa, Takashi Matsumoto, and Kei Hiraki

Department of Information Science, Graduate School of Science, University of Tokyo
7–3–1 Hongo, Bunkyo-ku, Tokyo 113

We are developping a high speed, enhanced gigabit switching network system which cooperates our general purpose massively parallel operating system. Communication overhead among user programs under multiuser/multijob environment is reduced by supporting hardware, so we can have efficient parallel execution environment on network of workstaions. We have implemented network interface hardware for Sun workstation, and describe the key functions of it in this paper.

1 はじめに

ワークステーション単体の性能が著しく向上し、スループットの高い通信経路が利用可能になった今日、ワークステーションクラスタ環境における並列処理が現実となりつつある。並列アプリケーションの効率良い実行のためにはユーザ間の高速な通信同期が必要であり、我々が開発中の超並列オペレーティングシステム SSS-CORE では、メモリベース通信 / 同期機構を可能な限り低オーバヘッドのソフトウェアで実装する方法を用いる [1]。

本稿で述べるネットワークはユーザ通信のオーバヘッドのさらなる削減のため並列計算機上の分散共有メモリシステムの実現のために提案された MBP(Memory-Based Processor) [2] の機能の一部 (リモートメモリアクセス、不可分リモートメモリアクセス、マルチキャスト、Ack のコンバイニング) をネットワークインタフェイスにおいて実現するものである。

1.1 関連研究

ワークステーションクラスタで並列処理環境を得るためのネットワークとして、Myrinet [4] が使用されはじめている。

Myrinet はスイッチとインタフェイスからなるネットワークである。パケットのヘッダに書かれている経路情報を用いてスイッチングを行なうので、1 対 1 通信のみをサポートする。インタフェイス上のコントローラはソフトウェアの書き換えによって任意のプロトコルのパケットを扱うことが可能である。インタフェイス上のバッファはユーザのアドレス空間からアクセス可能であり、送受信際のデータのコピー回数を削減できる。

2 ハードウェアによる支援

2.1 SSS-CORE のメモリベース通信機構

マルチユーザ / マルチジョブ環境においては、各プロセスにとってアドレス空間が仮想化されている。文献 [2] に述べられているように、ユーザ間の通信を軽くすることは、仮想化されたリモートメモリへのアクセスを物理レベルの通信手段にマッピングするコストをいかに削減できるかにかかっている。

仮想化されたリモートメモリアクセスの実現には、(1) 相手先のノード ID から通信路を決定する機構、(2) 与えられたネットワークアドレスから物理ページを決定する機構、(3) アクセスに関するプロテクションを調べる機構、が必要であるが、我々のネットワークでは以下のようにハードウェアで実装し、ソフトウェアによるオーバヘッドを削減する。

1. 相手先のノード ID のみで通信路を決定

   ページの移動やマイグレーションをユーザから仮想化するために、通信相手先のノード ID をユーザにとって仮想化する必要がある。

   我々のネットワークのスイッチングノードではパケットにネットワークの経路が書かれている必要があり、インタフェイスカードのコントローラは相手先の仮想ノード ID から経路を決定するためのテーブルを持つ。テーブルにヒットしなかった場合は OS に割込みをかけソフトウェアで解決する。

2. ネットワークアドレスから物理ページを決定

   SSS-CORE で使用しているネットワークアドレスは、プロセス ID とその中での仮想アドレスを組にしたものである。今回ネットワークインタフェイスを実装した SBus では仮想アドレスによるアクセスが可能なので、パケットが到着したコントローラでプロセス ID の比較を行ない、一致すれば仮想アドレスでアクセスする。

3. プロテクション

   SSS-CORE ではプロセス対してアクセス ID が用意されていて、アクセス ID が取得できたプロセスに対してリモートメモリアクセスが実行可能である。アクセス ID を取得するためには、自分のプロセス ID とアクセス ID を他人に通知する必要がある。

リモートメモリアクセスの順序管理には緩和されたメモリモデルを使用し、必要に応じてスレッドがメモリバリアを張ることで順序付けがなされる。メモリバリアは Acknowledge(Ack) ベースで実現され、non-blocking の手続きで同期完了か否かの結果はフラグに返され、Snoopy Spin wait [3] でそのフラグを利用する。

パケットの到着先でアクセス先プロセスがスケジュールされている場合には、アクセス先アクセス ID を比較する。一致すればアクセスは許可されるので、仮想アドレスから物理アドレスを引きアクセスを行なう。(ページ処理の時点で、リモートメモリアクセスされる可能性のある仮想アドレスと

物理アドレスの対応をコントローラのTLBに書いておくことが必要。今回インタフェイスを作成したSBusでは仮想アドレスでアクセスが可能なので、TLBは不必要。）成功した場合、すべての処理はハードウェアで行なわれソフトウェアオーバヘッドは存在しない。

1. アドレス変換が失敗した場合

   OSが物理ページを割当て[1]、データを転送する。

2. アクセスIDが一致しない場合

   OSは不正なアクセスを行なったプロセスのプロセスIDとアクセスIDをユーザに通知する。相手のアクセスIDを手に入れたユーザはシステムコールにより相手のプロセスを停止できる。パケット入っているアクセス発行元のプロセスIDとアクセスIDはユーザが改竄できないようにするため、送信元のコントローラが送信パケットのヘッダにハードウェアで書き込む。

3. プロセスIDが一致しない場合

   OSがそのプロセスの論理アドレスを解決し、当該ページを割当ててデータ転送をおこなう。

メッセージ送信の際にコントローラにプロセスIDとアクセスIDを書き込む方式を採用すると、ユーザがそれらを改竄不可能にするにはOSのシステムコールを呼ぶ必要があり余計なオーバーヘッドが課せられるので、コンテクストスイッチの際に書き込む方式を採用した。そのため、コントローラが独自のTLBを持たない今回の実装では、Sun SPARCstation 20のようにワークステーションに複数のプロセッサが存在する場合においてもリモートメモリアクセスを使用できるプロセッサは一つに限定されるという制約が課せられる。

リモートメモリアクセスがAckを必要とする場合、Ackが帰ってきたことを調べるためのSS-Waitフラグのネットワークアドレスをパケットに入れて送信する。このパケットを受信したノードは、リモートメモリアクセスを処理した後で送信元にAckを返送する。Ackの返送はAckなしのリモートメモリライトで実装される。

[1]実装の細かい話しになるが、SBus [5]ではリクエストのあった仮想アドレスに対して物理ページが存在しない場合、Translation PhaseでRerun Acknowledgeを生成し、その間に物理ページを用意することも可能である。

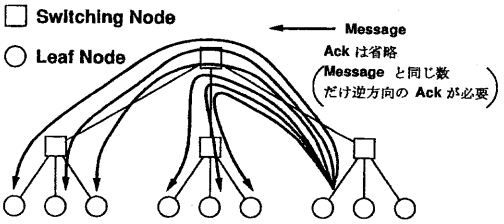

図 1: 1対1通信を使用したマルチキャスト

以上をまとめると、リモートメモリアクセスパケットを処理するためにコントローラに必要な情報は以下の通りである。

- パケットの型
- 相手先のネットワークアドレス (相手先のプロセスIDと仮想アドレス)
- 相手先のアクセスID
- 相手先のノードID
- データの先頭番地
- データのサイズ
- 自分のプロセスID
- 自分のアクセスID

2.2 チェイニングによるマルチキャスト

スケーラブルなネットワークは物理的に階層構造をしている場合が多い。クラスターを離れれば離れるほど (階層の上方でのネットワークのバンド幅が大きくない限り) 通信のスループットが下がるので、離れたクラスター間の通信の削減が必要である。1対1の通信であればお互いを近くにスケジューリングすれば良いが、多人数に対してマルチキャストを発生させると、人数に応じて離れたクラスタに対する通信が増加する。

1対1通信しかサポートしないネットワークでは、単純に離れたクラスタの数だけの通信が必要となる。

物理ネットワークが階層的であれば、スイッチイングノードにおいて入力先のパケットをコピーして複数の出力先に出力し、階層の上の方で必要なバンド幅を対数的に抑えることが可能である。

スイッチングノードにそのような機能がない場合、またはネットワークの物理構造が階層的でない場合のため、チェイニングによるマルチキャスト

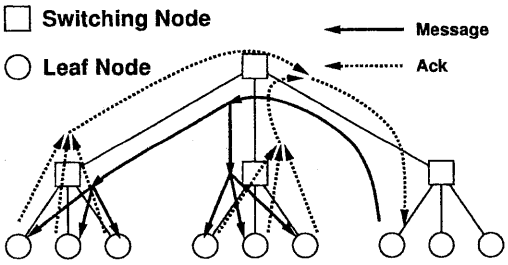

図 2: 階層マルチキャスト

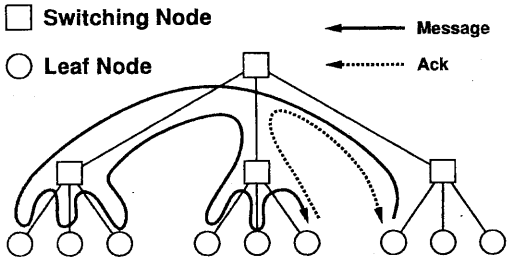

図 3: チェイニングによるマルチキャスト

の機能をインターフェイスカードに持たせる。インタフェイスカードは受け取ったパケットを自分の受信バッファにストアすると同時に、他人にフォワーディングする機能を持つ。ネットワークが階層的な場合、階層マルチキャストに比較して必要なバンド幅は 2 倍になるため、同様に対数的に抑えることが可能である。

マルチキャストが Ack を必要とするならば、パケットに自分の Ack もつけてフォワーディングする。そして、マルチキャストのパケットを受け取る最後のノードがまとまった Ack を送信先に返答する。必要なバンド幅は Ack 付きの階層マルチキャストと同じである。

階層マルチキャストとチェイニングによるマルチキャストではパケットが通る経路が異なるので、ヘッダに経路情報を入れる場合選択するマルチキャストの種類によってヘッダが異なる。スイッチングノードにおいてこの 2 種類のヘッダの間の変換は可能である。インタフェイスノードではチェイニングによるマルチキャストを階層マルチキャストに変換できるが[2]、逆を行なうにはパケットの送信元への経路情報が失われていないことが必要である。

[2]階層マルチキャストをサポートするために十分な機能を持つスイッチングノードではこの変換を実現できるので、インタフェイスノードで実現する必要はない。

2.3 リモートメモリに対する不可分操作

一連の不可分操作列を相手先ユーザに送信して実行すれば、通信と同期のパケットを減少させることが可能である。さらに、簡単な不可分操作 (Read-Modify-Write, Compare-and-Swap) をハードウェアでサポートするとオーバヘッドが減少する。

インタフェイスカードに演算器を実装し、I/O バス[3]に用意されているアトミックトランザクションを利用して不可分操作要求パケットを処理する。

3 実装

3.1 スイッチングノード

我々が構築中のネットワークはスケーラブルなスイッチングネットワークであり、以下の制約を満たす必要がある。

- スループットが高い
- ネットワークの物理構造を仮定しない
- 必要な場合に FIFO 性が得られる
- パケットを落さない
- エラー訂正がある

これらすべての問題を解決するため、ヘッダに経路情報を入れておき、スイッチでバーチャルカットスルーを行なう。Myrinet と異なり、前述の階層マルチキャストや Ack のコンバイニングをスイッチングノードで処理する必要があるので、パケットの型に関する情報をヘッダの先頭に入れる必要がある。

バーチャルカットスルーは受信ポートから送信ポートにデータがフォワードできない場合にはストアアンドフォワードに切替えるため、送信ポートに対して受信が可能であることを知らせるためのフロー制御が必要である。パケットの送信中にフロー制御を行なうと、トランシーバの送信路と受信路を同時に使用するのでネットワークのスループットを半減してしまう。パケットが受信できることをあらかじめ知らせておき、パケットの転送を途中で中断しないようにするには、一度に転送できるパケッ

[3]PCI [6] では、LOCK# 信号等を使用する。

SBus に関しては、IEEE1496 標準からは削除されたが、SBus Specification B.0 には存在しており、SBus を唯一実装しているといっていい Sun のワークステーションにはアトミックトランザクションは存在している。

トの大きさを制限する必要がある。また、送信がリモートライトの場合、パケットのデータがページ内に収まっていないと受信側では一つのパケットの処理中に属性の異なるページを処理する必要がある。(SBus でない場合、仮想アドレスから物理ページを引く作業をパケット処理中に複数回行なう必要がある。) 以上の理由により、パケット内のデータサイズは最大 4KB とし、ページ境界を超えないという制約をつける。

3.2 ネットワークインタフェイス

SBus カードとしてネットワークインタフェイスを作成し、Sun SPARCSation20 で使用している。参考までに、初めに各機能ブロックのスループットを示す。

光トランシーバは 53.125MHz を 1 サイクルとし、各サイクルに 20b[4]のデータを送受信でき、それぞれ 1.0625Gb/s のスループットを持つ。データは 8b を 10b にエンコード (「8B10B」 [7]) して転送されるので、実際に転送されるデータは 16b/ サイクルであり、850Mb/s である。

SBus の拡張転送はサポートしない。25MHz 動作の SBus では 1 サイクルに 32b のデータを転送可能 (100MB/s) である。

SS10/20 でサポートされているバーストの最大サイクル数は 8(32B 転送) なので、プロトコルに必要なサイクルを加えると 67MB/s のスループットであるが、バスは複数のデバイスにより共有されるので定常的に 67MB/s とは限らない。また、一般的にリアルタイムで I/O バスを利用できるとは限らないので、スループットの高い送受信バッファを用意する必要がある。

送受信バッファとしてライトサイクルが 20nS (50MHz) の SRAM(512K×8b) を使用した。32b のデータバスを確保するには 4 個の SRAM が必要であり、この場合のスループットは (20nS に 4b なので) 200MB/s である。送受信バッファはリングバッファとして使用し、ハードウェアによる管理機構を単純にする。

3.3 送信

まずヘッダの一部を SRAM に書き込み、コントローラの送信要求レジスタにデータ書き込む。その後、コントローラはバスを通してホストからデータ

[4]20 ビットの意味。以下、特に指定しない限り 'b' はビットを表し、'B' はバイトを表す。

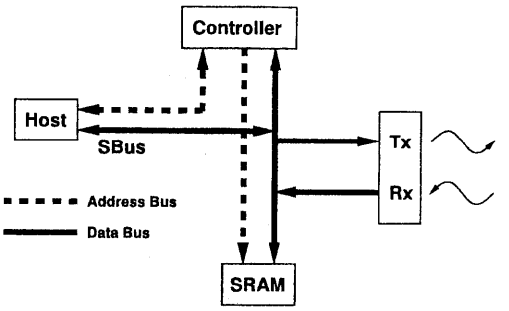

図 4: ブロックダイアグラム

を SRAM に転送し、CRC を計算する。リモートメモリアクセスでは、データがページの境界を超えないことを送受信の両方でチェックする。送信先の仮想ノード ID から経路を求めてヘッダを構築し、データを送信する。

3.4 受信

パケットのとりこぼしを防ぐため、トランシーバからの受信データをオンボードの SRAM に書き込む操作はすべての送受信に関わるデータ転送の中で一番優先度が高い。受信は到着順に処理する。

受信データ内のプロセス ID とアクセス ID を抜きだし、プロセッサに現在スケジュールされているプロセスのプロセス ID とアクセス ID と比較する。両方とも一致した場合には、仮想アドレスの変換を SBus にリクエストし、成功すれば DMA でデータを転送する。失敗した場合、OS に対して割込をかけ、原因に応じた対処を行なう。

3.5 機能ブロック

コントロール部、トランシーバ、送信データエンコード部、受信データデコード部、送受信バッファからなる。

コントローラ部分は DMA 転送回路、不可分操作のための演算器、プロセス ID やアクセス ID を比較するための比較器、仮想ノード ID から経路情報へのルックアップテーブルを持つ。

比較的低速で動作する FPGA で、約 18.8nS のサイクルでデータをエンコードすることは不可能である。このため、エンコードの回路は 2 セット用意し、32b を約 37nS でエンコードすることとした。(16b のデータは実際には 5b,3b,5b,3b のブロックに分割されていて、ブロック毎に計算したパリティ

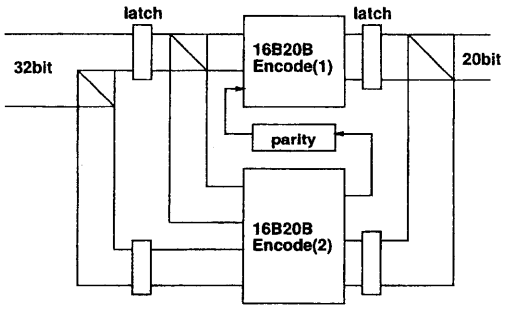

図 5: エンコード回路

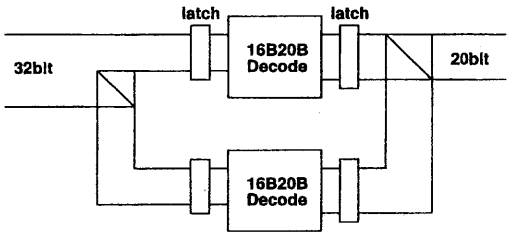

図 6: デコード回路

を元に次のブロックを計算する。パリティをプロパゲートしないためには、32b の最後のブロックでは、手前の 29b を 1 ビットにエンコードする必要がある。)

受信も同様に 1 クロックで 20b のデータを受け取る。これをコントローラ内のレジスタにラッチし、2 クロック毎に 32b のデータにデコードして SRAM に書き込む。

サイズ 4B リモートメモリライトを行なう場合、必要なヘッダとテイラは 30B+2B×n(途中のスイッチの数) であり、スイッチを 1 個経由するホスト同士の通信では通信経路の使用効率は $\frac{1}{9}$ で毎秒 24M 個のパケットを処理可能である。

4 おわりに

我々は汎用超並列 OS と協調動作可能なギガビットネットワークを構築中であり、そのためのネットワークインターフェイスカードを SPARCStation20 の SBus 上に実装した。このネットワークインターフェイスカードは高いスループットを持ち、メモリへの書き込み保護をハードウェアでサポートしているので、ユーザによる通信を低レイテンシで実現する。

今後、ネットワークのスイッチノードを実装し、ワークステーションクラスタを構築して実験をし、超並列 OS との協調動作の検証を行なう予定である。

謝辞

本研究は情報処理振興事業会（ＩＰＡ）が実施している独創的情報技術育成事業の一環として行なった。

参考文献

[1] 松本 尚, 平木 敬, “汎用超並列オペレーティングシステム SSS-CORE のメモリベース通信機能”, 情報処理学会 第 53 回全国大会 (September 1996)

[2] 松本 尚, 平木 敬, “超並列計算機上の共有メモリアーキテクチャ”, 信技報, CPSY92-96, pp.47–55 (August 1992)

[3] 松本 尚: “マルチプロセッサ上の同期機構とプロセッサスケジューリングに関する考察”, 計算機アーキテクチャ研究会報告 No.79-1, 情報処理学会, pp.1–8 (November 1989)

[4] Nanette J. Boden, Danny Cohen, Robert E. Felderman, Alan E. Kulawik, Charles L. Seitz, Jakov N. Seizovic, and Wen-King Su: “Myrinet – A Gigabit-per-Second Local-Area Network”, *IEEE-Micro*, Vol.15, No.1 pp.29–36 (February 1995)

[5] Susan A. Mason: “SBus Handbook”, *Sun Microsystems, Inc.* (1994)

[6] Tom Shanley, Don Anderson: “PCI System Architecture 3rd edition”, *MindShare, Inc.* (1995)

[7] A. X. Widmer, P.A. Franaszek: “A DC Balanced, Partitioned Block, 8B10B Transmission Code”, *IBM Journal of Research and Development*, Vol.27, No.5 (September 1983)