計算機アーキテクチャ 132−21
システムソフトウェアとオペレーティング・システム 80−21
ハイパフォーマンスコンピューティング 75−21
（1999. 3. 5）

PC間ネットワークによる共有アドレス空間を持つ並列処理システム

工藤 知宏 † 山本 淳二 † 建部 修見 ‡ 佐藤 三久 † 西 宏章 *
天野 英晴* 石川 裕 †

† 新情報処理開発機構

‡ 電子技術総合研究所

* 慶應義塾大学 理工学部

我々は RHiNET と呼ぶ、PC を高速ネットワークで結合して効率の良い並列処理を行なうシステムの開発を行なっている。RHiNET は広バンド幅で低レイテンシな通信を実現すると同時に、様々な並列処理を支援する機能を持っている。本報告では、RHiNET がサポートする通信モデルを中心にシステム全体の概要を述べる。

Parallel processing system using network-connected PCs

Tomohiro Kudoh† Junji Yamamoto† Osamu Tatebe‡ Mitsuhisa Sato†
Hiroaki Nishi* Hideharu Amano* Yutaka Ishikawa†

†Real World Computing Partnership
‡Electrotechnical Laboratory
*Keio University

We are developing a parallel processing system called RHiNET which connects PCs using high performance network. RHiNET provides large-bandwidth low-latency communication as well as functions which support high performance parallel processing. In this report, we present overview of RHiNET focusing on the communcation model.

1 はじめに

近年、高性能 PC を多数用いて並列処理を行なういわゆるクラスタコンピューティングが注目されている。これらには、Myrinet に代表されるシステムエリアの低レイテンシネットワークを用いたものと [5]、100BaseT や Gigabit Ethernet などの汎用の高速ネットワークを用いたものがある [4]。しかし、前者は接続距離に強い制限があるため PC 同士を近接して設置しなくてはならず、またネットワークのトポロジに対する制約も大きい。後者は、多くの場合 TCP/IP が用いられるのでパケットの廃棄が起きる可能性があり、またレイテンシも大きい。

RHiNET[1] [1][2][3] では、1Gbps〜8Gbps の光インタコネクションとメモリ混載スイッチを用いてネットワークを構成することにより接続距離、トポロジの制約とレイテンシを同時に小さくしている。また、ネットワークインタフェースに様々な並列処理サポート機能を持つことにより、共有メモリモデルとメッセージパッシングモデルのいずれを用いたプログラムも容易に効率良く実行でき、全体として効率の良い並列処理を可能にする。

これまでハードウェアの試作および Myrinet[8] を用いたエミュレーションを行ないながら、通信プロ

[1] RHiNET は以前は MLC と呼ばれていた

トコルの策定とハードウェアにより実現する機能とソフトウェアによる機能の切りわけを行なってきた。本報告では通信モデルを中心にRHiNETの概要を述べる。

2 RHiNETの通信モデル

通信のレイテンシを削減し、バンド幅を最大にするために、RHiNETにおけるデータの転送の基本はゼロコピーのリモートメモリコピーである。

RHiNETのハードウェアは、ネットワークインタフェースと高速ネットワークから構成される。ネットワークインタフェースRHiNET/NIはPCIバスに装着される。PCIバスを用いることによる制約から、RHiNETの通信は以下のような特徴を持つ。

- PCIバスはI/Oバスであり、特にデータ転送の立ち上げに要する時間が大きい。ホストメモリとの間のDMAには、起動に1μsec程度の時間を要する。1Gbpsの転送容量を持つネットワークを想定すると、1μsecは1kbitの転送時間に相当する。このため、DMAを起動する回数はできるだけ少なくすることが望ましい。そこで、大きな領域をホストメモリとNIの間で転送しながら実際にはその一部だけの読み書きを行なう機能を用意する。これには、実際に転送する領域をリスト構造やビットベクタによって指定する手法と、TWINメモリを用いる手法(後述)がある。

- NIはCPUからホスト上のメインメモリに対するアクセスを監視することができない。したがって、メインメモリへの書き込み/読みだしによって通信を起動することはできない。通信は全てプログラム中で明示的に起動される。

- ホストCPUからNI上のメモリへアクセスするには大きなコストがかかる。アクセスに時間がかかるだけでなく、PCIバスを占有するので、NI上のメモリをポーリングするようなオペレーションは行なうべきでない。このため繰り返しアクセスするポーリングを行なう場合にはホストメモリ上で行なう必要がある。即ち、NIがホストメモリに書き込みを行なうことになる。

2.1 並列プロセスグループ

複数のノードに存在する協調して並列処理を行なうプロセスの集まりを並列プロセスグループ(parallel process group)と呼ぶ。同一並列プロセスグループに属するプロセス間では他のプロセスのアドレス空間に対して読み書きすることができる。通常の通信は同一プロセスグループに属するプロセス間で行なわれる。

2.2 通信エリアID

互いに通信するプロセス同士では、通信領域をおく仮想アドレスが異なる可能性がある。1対1の通信では通信相手のプロセスにおける仮想アドレスを用いて通信領域を指定することもできるが、マルチキャストなどの複数の通信相手を持つ通信では困難である。また、仮想アドレスそのものではなく、あるエリア中のオフセットによって指定する方が便利であることも多い。

そこで、通信に用いるエリアの先頭仮想アドレスをNIに登録しておき、通信時には通信エリアIDと先頭アドレスからのオフセットにより通信相手の仮想アドレスを指定する手段を提供する。通信エリアIDはプロセス毎に複数持つこともできる。通信エリアIDを介した通信に参加するプロセスは、通信エリアIDとエリアの先頭仮想アドレスをNIに登録する。各プロセスは複数のIDを登録することができる。これによって、通信エリアIDごとに仮想的に共有アドレス空間を持つことができる。

2.3 通信の起動とプロテクション

通信は全てプログラム中で明示的に起動される。通信のために必要な情報はホストCPUからのNIに対する書き込みによって伝えられる。通信の結果(終了通知など)は、起動時に結果を格納する仮想アドレスをNIに通知しておき、ここに対してNIが書き込むことにより通知される。通信の起動はユーザプロセスから直接NIへの書き込むことによって行なう。

プロテクションを実現するために、NI上の通信起動を受け付けるためのレジスタは複数の物理ページにマップされ、プロセス毎に異なるページを用いる(他のプロセスのための物理ページにはアクセスできない)。NIはレジスタがアクセスされた際の物理ページ番号によりどのプロセスがアクセスしたかを知ることができ、これによってプロテクションを実現する。

通信の相手は、ノードIDとプロセスIDによって指定される。ノードIDはNIごとに付けられる。SMPでは同一並列プロセスグループに属する複数のプロセスが同一ノードに存在することがあり、これらはプロセスIDによって区別される。

2.4 リモートメモリコピー

RHiNETの最も基本となる通信はリモートメモリコピーをである。リモートメモリライト(PUSH)は、起動側のプロセスから指定する通信相手側のプロセスへの1対1のメモリコピーである。コピーされる領域は、起動側は先頭の仮想アドレスとサイズにより規定される。受信側は、仮想アドレスを直接用いて指定する方式と、通信エリアIDとoffsetにより指定する方式がある。

リモートメモリリード(PULL)は、通信相手側から起動側への1対1のメモリコピーである。相手側の領域の指定法はPUSHと同様である。

2.5 ピンダウン

ゼロコピーのリモートメモリコピーでは、通信が実際に行なわれる際には送信側と受信側のいずれにおいても通信に用いられるユーザプロセスのメモリ領域がピンダウン(物理メモリに固定)されていなくてはならない。ピンダウンされた領域は、ページ単位で仮想アドレスから物理アドレスに変換するVP変換テーブルに登録される。VP変換テーブルはカーネル空間上に置かれる。

リモート側の通信に使用するメモリ領域のピンダウンを確認/要求するために、起動側はリモート側に対してピンダウンリクエストを行なうことができる。ピンダウンリクエストを受けたリモート側のNIは、VP変換テーブルをチェックしてピンダウンされていればその旨を起動側に通知する。されていなければ、ホストに割り込みをかけてピンダウンを行なった上で起動側に通知する。ピンダウンリクエストを行なったノードは、リモートノードでのピンダウンの必要がなくなったらピンダウンリリースを送ってピンダウンを解除してよい旨を伝える。

2.6 isend/ireceiveのサポート

MPIにおけるisend/ireceiveのような機能を実現するにはリモートメモリコピーだけでは不十分で、なんらかの待ち合わせ機構が必要になる。RHiNETでは、受信側に待ち合わせ機構を用意することによってこれを実現する。各ノードは、ペンディング送信バッファとペンディング受信バッファを持つ。これらのバッファでは、送受信の際に指定されるkeyによって送受信のマッチングがとられる。

受信側のNIは送信側から送信可能かどうかを調べるコントロールメッセージを受けとると、まずペンディング受信バッファ中に対応する受信が登録されているかどうかを調べる。登録されていれば受信側プロセスの通信領域の仮想アドレスを送信側に伝える。送信側はここでPUSH操作を実行することになる。登録されていなければ、ペンディング送信バッファに送信元の通信領域の仮想アドレスを登録する。図2に受信側における処理を示す。

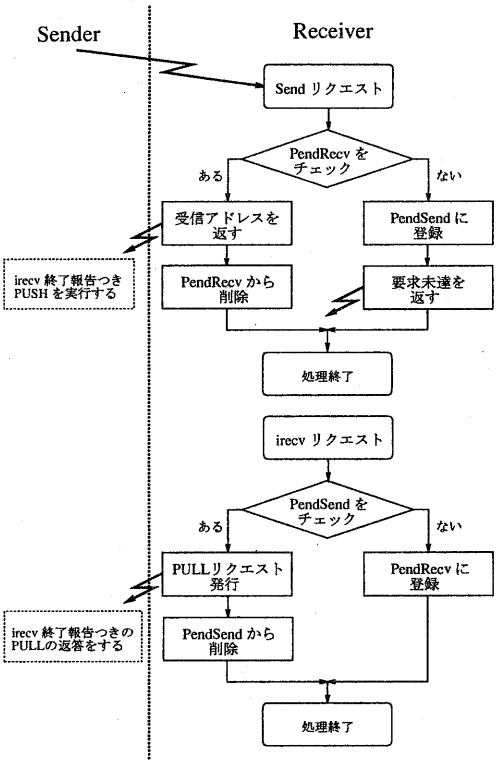

図 2: isend/ireceive

受信の際には、まずペンディング送信バッファを調べ、対応する送信があれば登録されている送信側の通信領域の仮想アドレスを用いてPULL操作を行なう。なければ、ペンディング受信バッファに登録する。

受信側で待ち合わせを管理することにより、wild-cardを用いた送信プロセスを特定しない受信が実現できる。

2.7 部分転送機能

RHiNETにおける通信は、PCIバスを介したDMAの起動に時間がかかるため、バンド幅に対して立ち上げに要するオーバヘッドが大きい。そこでRHiNETではできるだけDMAの回数を減らすため

```
push(my_addr,remid,global_id,offset,flag,status_ret,size)
  void *      my_addr;     /* 自プロセスでの転送するデータの先頭アドレス*/
  PID         remid;       /* 相手プロセスの ID: ノード ID およびノード内プロセス ID */
  GID         global_id;   /* グローバルアドレス ID */
  int         offset;      /* global_id で得られるアドレスからのオフセット*/
  FLAG        flag;
  STATUS *    status_ret;  /* 終了ステータスを受ける変数へのポインタ */
  size_t      size;        /* データのサイズ (byte) */
```

自プロセスの仮想アドレス my_addr から size バイトのデータを，remid, global_id, offset で決まる相手のメモリに転送する．

図 1: PUSH 基本形

に、一つの通信で大きな領域中のいくつかの部分を転送する部分転送機能を持つ。即ち、大きな領域を転送対象領域として指定しながら、さらにその中で実際に転送する領域を細かく指定することを可能にする。

実際に転送する領域を指定するには以下の 3 つのいずれかの手法を用いる。

TWIN/DIFF 方式: 他のノードから領域を PULL する際などにその領域の内容を TWIN メモリと呼ぶメモリにコピーしておき、PUSH などにより書き戻す際に、書き戻す内容を TWIN メモリの内容と比較して、変更されている部分のみを実際に転送する。これによって、multiple writer 機能 [7] をハードウェアのサポートにより大きなオーバヘッド無しに実現できる。

bitmap 方式: 書き戻す部分をビットマップによって指定する。領域中のどの部分を実際に転送する必要があるのかがプログラムにとって既知であり、それを示すビットマップを容易に作成できる場合に使用する。

stride 方式: 行列の特定の列を転送する際などに用いる。転送する部分の大きさと、間隔、個数を指定することにより一定間隔に並んだ領域を転送することができる。

2.8 マルチキャスト

同一のデータを複数の相手に送る場合、マルチキャスト機能を用いて一度の通信起動で送ることができる。マルチキャストには、ネットワークの機能を用いて実際に同時に送出を行なう同時マルチキャストと、NI が逐次に送出する逐次マルチキャストがある。

同時マルチキャスト: ネットワークの各ルータにあらかじめどの出力に対して送信するかのパターンを登録しておき、これにしたがってマルチキャストする。NI からネットワークへのパケットの送出は一回である。パターンは複数登録できるが、パターンを変更するにはネットワーク上の各経由値の設定を変えなくてはならないため、頻繁な変更はできない。並列プロセスグループ内のブロードキャストなどに用いる。

逐次マルチキャスト: ホストプロセッサが宛先のリストを NI に渡し、NI はこれにしたがって宛先ごとにパケットを送出する。

2.9 共有登録機能

部分更新機能を用いると、配列などを複数のプロセスで共有し、同時に更新することが可能である。この際、更新情報は配列を共有している全てのノードに伝達されなくてはならない。RHiNET では、どのノードがその領域を共有しているかを示すディレクトリを全てのノードが持ち、更新時にはこれにしたがってマルチキャストする。

このときの共有の単位をここでは領域と呼ぶ。領域の大きさは任意である。領域毎にホームノードをあらかじめ決めておき、新たに共有に参加するノードは、ホームに対して共有に参加したいという Scribe メッセージを送る。ホームは、共有に参加している全てのノードに対して、ディレクトリに新たに参加するノードを加えるよう要求する。この要求に対するアクノレッジを待って、新たに参加するノードにその領域のデータとディレクトリ情報を伝える。

また、共有をとりやめるノードは Unscribe メッセージをホームに対して送り、ホームを介して全共有ノードのディレクトリを更新する。

表 1: 部分転送に関係可能な通信機能の API

基本名	機能	基本	twin	diff	bitmap	stride
push	リモートメモリライト	○	—	○	○	○
pull	リモートメモリリード	○	○	—	○	○
mcastA	同時マルチキャスト	○	—	○	○	○
mcastB	逐次マルチキャスト	○	—	○	○	○
isend	isend	○	—	○	○	○
irecv	ireceive	○	○	—	○	○
subscribe	共有登録	○	○	—	—	—

表 2: その他の API

名称	機能
iprobe	メッセージ到着確認
unsubscribe	共有登録解除
lock	ロック
unlock	アンロック
barrier	バリア
pd_req	ピンダウン要求
pd_reply	ピンダウン要求応答
pd_release	ピンダウン解除可通知
sync	全要求完了待ち

Scribe 時には、新たに参加するノード、ホームノード、既に共有しているリモートノードの 3 者間でメッセージの行き違いによるレースコンディションが発生する可能性がある。即ち、新たに参加要求をしたノードがホームノードからデータを受けとる前に、リモートノードから部分更新要求を受けとる可能性がある。この場合には、部分更新要求を受けとらず再送要求を送って、データを得た後で受けとることにより、正しい処理を行なう。

2.10 ロックおよびバリア

ロックの要求はノードを跨いだ分散キューにより管理される．要求はまずマネージャに送られ，マネージャは自らが管理しているキューのポインタに従い要求をキューの最後尾のプロセスに送りつける．解放時にはこのキューの順に従い次のプロセスにロックは渡される．

RHiNET ではマネージャプロセスの管理による計数バリアを提供する．ツリーによるバリアなどはネットワークとポロジが一定しない RHiNET では使用しにくいため計数バリアを用いる．

2.11 シグナル

ほとんどの通信機能は、指定により受信時にホストプロセッサに割り込みをかける機能を持つ。

3 通信の API

図 1 に PUSH の基本形の API を示す。簡単のために実装上用いるいくつかの引数を省略している。

表 1 に API のうち、部分転送機能をサポートするものを示す。それ以外のものを表 2 に示す。

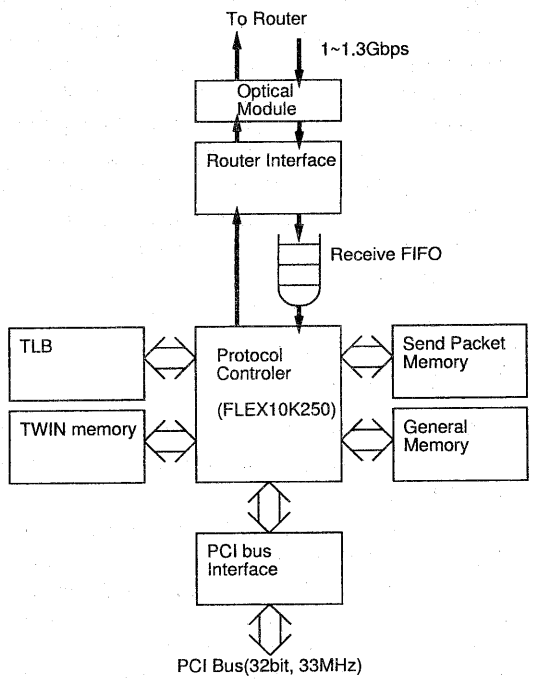

図 3: RHiNET-1/NI のブロック図

4 ハードウェアの構成

図 3 にネットワークインタフェース RHiNET-1/NI のブロック図を示す。NIC における処理は Altera 社の PLD(Programmable Logic Device) である FLEX10K250-2 を用いた Protocol Controler が行なう。

NI 上に VP 変換テーブルの TLB を持ち、エントリが TLB 上にあれば高速にアドレス変換を行なうことができる。TWIN メモリは TWIN をおくためのもので SDRAM を用いている。Send Packet Memory にはネットワークにエラーが発生した時に再送するために、送り出したパケットのコピーをおく。

このように、複数のメモリバンクを持ち、これらに同時に高速にアクセスすることができることが一般的なプロセッサを用いずに PLD を用いてプロトコルコントローラを作成する最大の利点である。

ネットワークスイッチは、Asynchronous wormhole routing によりレイテンシを押え、バーチャルチャネルを複数持つことによってトポロジの制約を小さくしながら、デッドロックを防ぐ。パケットの廃棄は行なわないので、物理的なエラーが起きない限り、送信されたパケットは宛先に確実に届き、通信の FIFO 性も保証される。

物理的なエラーが発生した場合、NI は自動的に再送を行なう。このために、送信パケットには宛先毎にシリアル番号をふり、受信側では番号順に到着したことを確認して acknowledge を返す。番号順に到着しなければ、エラーであるので、送信側に再送を要求する。また、送信側は一定時間の間に acknowledge または再送要求が返ってこなければタイムアウトとみなして再送を行なう。このために、NI は全てのノードについて送信シリアル番号と受信シリアル番号を管理する。

5 おわりに

通信モデルを中心に RHiNET の概要について述べた。現在、RHiNET-1 は NI 上にプロトコルを実装中である。

RHiNET-1 では 1.3Gbps の光インタコネクションもしくは GBIC(Gigabit Ethernet で用いられているインタコネクション) を用い、ASIC スイッチ [6] によってネットワークを構成する。システム全体としては 1999 年 8 月の稼働を予定している。今後、実機上で本報告で述べた通信モデルの有効性を確認していく予定である。

また、8Gbps クラスの光インタコネクションを用いた RHiNET-2 も開発中であり、2000 年秋の稼働を目指している。

謝辞

RHiNET に関する議論を通じて貴重なご意見を頂いた，新情報処理開発機構の手塚宏史氏、田邊 昇氏、原田 浩氏、シナジェテックの清水 敏行氏、NEC 情報システムズの宮脇 達郎氏、慶應義塾大学の横山知典氏に感謝致します.

参考文献

[1] Tomohiro Kudoh, Junji Yamamoto, Yutaka Ishikawa, Mitsuhisa Sato, Fukukyoh Sudoh, Hideharu Amano. Memory based light weight communication architecture for local area distributed computing. *IWIA*, 1997.

[2] 周東 福強, 山本 淳二, 西 宏章, 天野 英晴, 工藤 知宏. 軽量メモリベース通信のためのネットワークインタフェース. *ARC*, No. 128-18, pages 103-108, 1998.

[3] 工藤 知宏, 横山 知典, 周東 福強, 清水 敏行, 天野 英晴. Network based Parallel Computing のための Network Interface の評価. *CPSY*, No.98-60, pages 1-8, 1998.

[4] 益口 摩紀, 建部 修見, 関口 智嗣, 長嶋 雲兵, 佐藤 三久. アルファワークステーションクラスタ etlwiz の性能評価. *ARC*, No. 128-11, pages 61-66, 1998.

[5] 手塚 宏史, 堀 敦史, Fransic O'Caroll, 石川 裕. RWC PC Cluster II の構築と性能評価. *ARC*, No. 128-5, pages 25-30, 1998.

[6] 西 宏章, 多昌廣治, 工藤 知宏, 天野英晴. コモディティPC を用いた並列処理のためのネットワークルータアーキテクチャ. *CPSY*, Jan. 1999.

[7] P.Keleher, A.L.Cox, and W.Zwaenepoel. Lazy consistency for software distributed shared memory. *Proc. 19th ISCA*, pages 13-21, 1992

[8] http://www.myri.com/