計算機アーキテクチャ 132−20
システムソフトウェアとオペレーティング・システム 80−20
ハイパフォーマンスコンピューティング 75−20
（1999. 3. 5）

コモディティPCを用いた並列処理のための通信機構について

山本 淳二[†1]　工藤 知宏[†1]　宮脇 達朗[†2]　坂 光彦[†3]　清水 敏行[†4]
横山 知典[†5]　天野 英晴[†5]

[†1] 新情報処理開発機構　[†2] (株)NEC 情報システムズ　[†3] (株) エー・イー・エス
[†4] (株) シナジェテック　[†5] 慶應義塾大学理工学部

本報告ではユーザプログラム，OS，ネットワークインタフェース (NI) 間で通信処理の分担を変えた場合のデータ転送性能を Myrinet と我々が開発しているハードウェアによる NI である RHiNET-1/NI を用いて測定した結果について述べる。ホストメモリから NI へのデータの転送にはホストプロセッサによる書き込み (PIO) と NI による DMA の 2 つの方法がある。Myrinet での測定から，極小さなデータを転送する場合を除き DMA を使用した方がよいことが分かった．ユーザレベル通信とシステムコールを用いた通信を比較すると、Myrinet においても若干ユーザレベル通信の方が性能が良い．これは，ホストプロセッサを用いたことによる処理の高速化よりもシステムコールを呼び出すオーバヘッドの方が大きい事を示している．RHiNET-1/NI ではプロトコル処理は完全なハードウェアによって行われるため，ユーザレベル通信の優位性がさらに大きくなる。

A study of communication mechanism for parallel processing with commodity PC

Junji Yamamoto[†1]　Tomohiro Kudoh[†1]　Tatsuaki Miyawaki[†2]
Mitsuhiko Saka[†3]　Toshiyuki Shimizu[†4]　Tomonori Yokoyama[†5]
Hideharu Amano[†5]

[†1] Real World Computing Partnership　[†2] NEC Informatec Systems
[†3] Advanced Engineering Services　[†4] Synergetech　[†5] Keio University

In this report, we measured performance of some alternative implementations of communication using Myrinet and a hardware network interface RHiNET-1/NI. To transfer data between the host memory and network interface(NI), there are two alternatives: data write operations by the host processor (PIO) and DMA by the NI. The result of the measurement using Myrinet shows DMA yields better performance except in case the size of the transferred data is very small. User level communication yields a little better performance than communication using system call. This result shows that the overhead of system call is larger than the performance improvement by using host processor. Since processing of RHiNET-1/NI is faster than Myrinet, user level communication will yield much better performance over communication using system calls.

1 はじめに

現在，一般的なPCをネットワークで接続した環境上での並列処理が盛んに研究されている．このシステムの性能は個々の要素の性能により左右される事はもちろんの事，それぞれの処理をどの要素で行うかによっても変化する．

そこで，本報告では一般的なPCとネットワークインタフェースを用いて，ライブラリ，OS，ネットワークインタフェースの処理の内容によりどのように性能が変化するかの報告を行う．

ネットワークインタフェースとして，MyrinetとRHiNET-1/NIを用いて測定する．

以下，ネットワークインタフェースをNIと略す．

2 通信処理

ネットワーク環境での並列処理では通信が欠かせない．通信の方法にはメッセージパッシングやリモートメモリアクセスなどがあるが，もっともプリミティブな処理はホストマシン上に用意されたデータを相手ノードに転送することである．今回はそのプリミティブな動作についていくつか実装し，比較する．

一般的にホスト上のデータを送信するための手順は次のようになると考えられる．

1. 送信相手先，データのサイズなどをNIに伝える
2. データをNIに伝える
3. パケットを組み立てネットワークに送出する

ここで，それぞれの処理を誰が行うかでいくつかの選択が存在する．

3 実装上の選択肢

ここでは前節で述べたもっともプリミティブな通信についての選択を更に検討する．

まず，ホストプロセッサとNIの分担についての選択について示す．

通信先等の情報の伝達: 通信に必要な情報は構造体(指示構造体)として準備し，これをNIに伝える．この伝達手段としていくつかの実装が考えられるが，次のデータの送信と同じであるため，そこでまとめて述べる．

データの送信: パケット本体となるデータをネットワークに送出するための方法はいくつか考えられる．

- ホストプロセッサがNI上のメモリにデータをコピーし，その後通信を起動する
- ホストプロセッサはNIにデータのあるアドレス(とサイズ)を伝え，NIがホストメモリにアクセスし通信を行う

表 1: 実装の種類

タイプ名	指示構造体	データ
User	V.Addr	V.Addr
User:ST	Copy	V.Addr
User:ST,DT	Copy	Copy
Syscall	P.Addr	P.Addr
Syscall:ST	Copy	P.Addr
Syscall:ST,DT	Copy	Copy

V.Addr: 仮想アドレスをNIに伝える
P.Addr: 実アドレスをNIに伝える
Copy: 実体をNIにコピーする

ホストプロセッサが処理を行う場合，それをユーザプログラムが行うか，OSなどのシステムが行うかによりさらに分ける事ができる．

一般に，NIがホストメモリにアクセスする場合には実アドレスを必要とする．一方，ユーザプログラムが使用するのは仮想アドレスである．そのため，アドレスをNIに伝える場合，どこかの段階で仮想アドレスから実アドレスへの変換作業が必要となる．

ユーザプログラムが通信を直接起動するユーザレベル通信を用いる場合は，NIがその変換を行う必要がある．一方，途中にシステムが介在する場合，そのシステムソフトウェアはアドレス変換テーブルにアクセスできるため，その段階でアドレス変換を行う事も可能である．

以上の選択肢の組合せにより考えられる実装形態は表1のようになる．

4 実装環境

実装に使用したホストマシンの主な仕様を表2に示す．このホストマシンにMyrinet[1]もしくはRHiNET-1/NI[2]をNIとして追加し測定した．

測定に用いたNIの一つであるMyrinetの主な仕様を表3に示す．

Myrinetは処理ユニットとしてLANaiプロセッサを持つ．LANaiはネットワークインタフェースカード(以下NIC)上のSRAM上のプログラムを実行する．このSRAMはホストプロセッサからもアクセスが可能であるため，プログラムを変更する事で自由にプロトコルを変更する事が可能である．

また，SRAMとネットワークおよびホストメモリ

表 2: ホストマシンの仕様

プロセッサ	Intel Pentium II 300MHz
チップセット	Intel 440LX
OS	RedHat Linux 4.2
カーネル	Linux 2.0.31

表 3: Myrinet の仕様

プロセッサ	LANai 4.1
システムロック	33MHz
ネットワークバンド幅	160MB/s
メモリバンド幅	264MB/s

間では DMA を使用する事ができる.

LANai プロセッサはホストメモリにアクセスできないため, ホストメモリにあるデータをアクセスする場合は DMA を用いる必要がある.

次に, もうひとつの NI である RHiNET-1/NI の主な仕様を表 4 に示す. RHiNET-1/NI は完全にハードウェアにより通信を処理する. ただし, RHiNET 自体のプロトコルが現在開発中である事もあり, コアのプロトコル処理ユニットは内部の構造を変化させる事ができる Altera 社の FLEX シリーズを用いている.

全処理をハードウェアで行うため, Myrinet では不可能な部分についての並列処理も可能である.

5 実装

5.1 ユーザプログラム

ユーザプログラムは次のように構成される.

1. データの用意
2. 指示構造体の用意
3. 指示構造体のコピー
4. データのコピー
5. 通信の起動
6. 通信の終了待ち

表 4: RHiNET-1/NI の仕様

プロトコル処理ユニット	Altera FLEX10K
システムクロック	16MHz

1 と 2 は計算により, もしくは予め用意されるとし, 今回の計測には含めない. 3 および 4 は表 1 中に Copy と示した実装でのみ行う.

通信の起動方法はユーザレベル通信かシステムコールを利用した通信かにより異なる. ユーザレベル通信では NI に直接アクセスし通信を起動する. これは Myrinet と RHiNET-1/NI では少々異なる.

Myrinet: 構造体のアドレスまたは通信終了を通知するアドレスを NIC 上の SRAM の特定の場所に格納する. その後, LANai に直接接続されている信号をアサートし通信を起動する.

これは, 特定の信号が LANai のレジスタに直結しており, メモリをポーリングするより高速にポーリングできるためである.

RHiNET-1/NI: 構造体のアドレスまたは通信終了を通知するアドレスを NI 上のレジスタ (SAR) に書き込む事で同時に通信を起動する.

これは, プロトコル処理ユニットがレジスタに対しての書き込みを検出できるためである.

システムコールを使用した通信の場合は, 指示構造体の先頭アドレスを引数としたシステムコールを利用する. 詳細については次節で述べる.

5.2 システムソフトウェア

今回, NI を操作するシステムソフトウェアは全てドライバ内に記述し, カーネル本体には一切手を加えていない.

ドライバでは基本機能として次の機能を提供する.

mmap: NI 上のレジスタおよびメモリをユーザ空間にマップする機能.

ピンダウン: メモリがスワップアウトされないことを保証する操作.

NI のメモリのマップやデータ領域のピンダウンなどは予め行われているとし, 計測に含めない.

表 1 の後半 3 つはシステムコールを利用した通信の実装である. ドライバ内にこれらの処理を実装し, ユーザプログラムは `ioctl` システムコールによりこの処理ルーチンを呼び出す.

`ioctl` で呼び出されたドライバ内の処理ルーチンでは以下の処理を行う.

1. 指示構造体のカーネル空間へのコピー
2. 指示構造体を NI 上のメモリにコピーする
3(a) データ領域の実アドレスのリストを作成し, NI 上のメモリに書き込む

(b) データを NI 上のメモリにコピーする
4. 通信を起動する

通信の種類により行うべき処理の内容が異なる．そのため，カーネル空間に指示構造体を取り込む必要がある．

表 1 の指示構造体の欄が Copy となっている場合に限り，2 での処理が実行される．P.Addr の場合は，通信を起動する際に指示構造体のおかれている場所の実アドレスを伝える．

ホストプロセッサがデータをコピーする (表 1 のデータ欄が Copy) 場合には，3b の処理を行う．データがコピーされた場合，NI がホストメモリにアクセスする必要はなくなる．

一方，ホストではアドレスのリストを用意する場合 (同 P.Addr) では，3a の処理を行う．こちらでは，NI はホストメモリにアクセスを行いデータを取り込む必要がある．しかし，ユーザレベルでの通信と異なり，必要な実アドレスがシステムソフトウェアで用意されているため NI が仮想アドレスから実アドレスへの変換を行う必要はない．

5.3 ネットワークインタフェース

NI では通信が起動されると次のような処理を行う．

1. 指示構造体のアドレスを得る
2. 指示構造体をホストメモリから取り込む
3. パケットヘッダの作成
4. パケットヘッダの送信
5. ホストメモリからのデータの取り込み
6. パケットボディ(データ) の送信
7. 終了通知

ホスト側のプログラムであるユーザプログラムもしくはシステムソフトウェアが指示構造体もしくはデータをコピーする場合には，2 および 5 の処理はそれぞれ不要である．

上記の一覧では省いているが，ユーザレベル通信では NI に与えられるアドレスが仮想アドレスであるため，メモリアクセスを行う際に仮想アドレスから実アドレスへの変換が必要である．一方，システムコールを用いている場合，システムソフトウェアがアドレス変換を行い，必要なアドレスの一覧を用意するため，NI でのアドレス変換は必要無い．

NI として Myrinet を使用した場合，LANai プロセッサにより処理されるため，ほとんどは逐次に処理される．ただし，DMA 実行中にはプロセッサは別の処理を行える．ホストメモリ上のデータをパケットボディとして送信する際には，メモリ間 DMA と送信 DMA を同時に使用してパイプライン的にデータを送り出すよう実装をする事で 2 つの DMA を並列に動作させている．

一方，RHiNET-1/NI では，完全にハードウェアで実装されているため，依存関係の無い処理は全て並列に処理される．たとえば，構造体を読み込んでいる途中でパケットヘッダの作成と送信が開始される．これは，構造体を途中まで読めばパケットヘッダに必要な情報が得られ，また，パケットの生成と同時に送信が行えるためである．

6 測定結果

6.1 Myrinet での測定

図 1 に Myrinet を使用した場合の転送バンド幅を示す．ここでの転送バンド幅はユーザプログラムから見て送信が終了した時点までの時間を基準としている．そのため，ネットワークそのものの性質などは反映されていない．

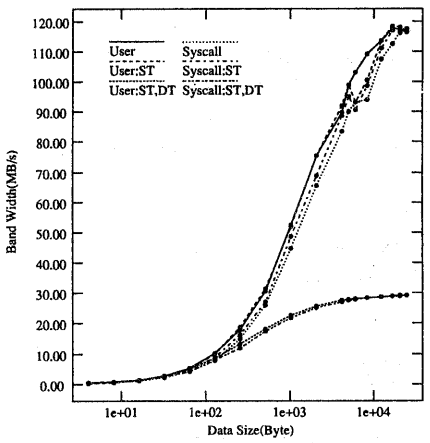

図 1: 転送バンド幅 (Myrinet)

ホストプロセッサがデータを書き込む User:ST,DT, Syscall:ST,DT では，64byte 程の転送まではそれ以外と同等か少々性能が良い．しかし，データサイズが大きくなると非常に性能が悪く，10Kbyte あたりでは他の物に比べ 1/4 程度の性能しか得られない．

Myrinet では LANai がホストメモリのデータにアクセスするためには DMA を必要とする．データが少ない場合，データを読み込む時間に比べ DMA のセットアップにかかる時間が短くない．そのため，プロセッサが直接データを書き込むほうが高速に処理できると思われる．

一方，データが多くなると，データの書き込みとNIの送信動作が逐次に行われることによるオーバヘッドが目立つようになる．図2にシステムコールを使用した通信のカーネル内での処理の内訳を示す．図からデータのコピーを行うSyscall:ST,DTはデータをコピーする時間が処理の大半を占めている事が分かる．データのコピーを行わないSyscallおよびSyscall:STではmisc.で示したその他の処理が時間の大半を占めている．この処理は主にシステムコールを呼び出すためのコストや与えられた引数の整合性の確認などである．

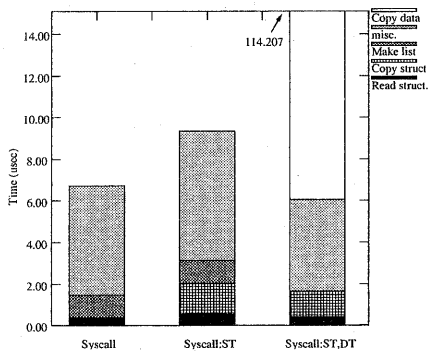

図 2: ホスト処理内訳 (カーネル内, 4096byte)

図3にそれぞれの実装におけるLANaiでの処理時間の内訳を示す．この図から分かる通り，もっとも時間のかかっているのはパケットの送信処理である．

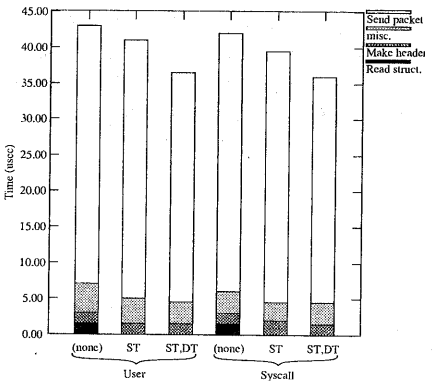

図 3: NI 処理内訳 (Myrinet, 4096byte)

測定結果からネットワークに対しての実効バンド幅を求めた所、どの実装においても最大130〜133MB/sであり、Myrinetのリンクバンド幅160MB/sを下回っている。

ところで，Myrinet上のSRAMは33MHzのクロックで動作し，1クロックにつき2回アクセスを行い，計264MB/sのバンド幅がある．しかし，そのうち1回はLANaiプロセッサのインストラクションフェッチに使用されるため，DMA他で使用できるのは半分の133MB/sである．これが転送性能を律速していると考えられる．

Myrinetではパケットに関するレジスタもプロセッサからアクセスできるため，全ての処理をプロセッサで行う事も可能である．比較のため，全処理をプロセッサで行った場合の結果を表6に示す．データの非常に少ない場合 (4byteもしくは8byte) の場合は，これまでの実装に比べてわずかに速いが，大きなデータの送信では全く性能が出ない事が分かる．

Data size*	バンド幅**	U1†	U2‡
4	0.414	0.340	0.355
8	0.762	0.671	0.711
16	1.324	1.340	1.419
4096	4.713	89.661	27.474
8192	4.738	109.042	28.441
24576	4.754	116.288	29.140

* 単位: byte
** 単位: Mbyte/sec
† User (参考値)
‡ User:ST,DT (参考値)

表 6: 全処理をプロセッサで行った場合

6.2 RHiNET-1/NIでの測定

RHiNET-1/NIを用いた場合のRHiNET-1/NI内での処理の内訳を表7に示す．表7中，段を下げて表記した処理内容は送信処理のクリティカルパス上には無い，すなわち全体の処理時間に影響を与えていない処理である．

RHiNET-1/NIでは依存性の無い処理は完全に並列に実行できる．しかし，表4で示したようにRHiNET-1/NIは現在も開発中のためPCIクロックの半分である16MHzで動作している．さらに，バースト転送が16byte単位でしか行えないため，ホストメモリからの転送性能が最大44Mbyte/sしか得られない．そのため，4096byteの転送に102.056μsecかかる．これはプログラム処理の必要なMyrinetに比べて2倍強の時間である．

今後，33MHz動作を行うようにし，バースト転送も8Mbyteまで連続して行えるよう改良する予定

表 5: ホスト処理内訳 (ユーザレベル通信)
(4096byte 転送時, 単位:μsec)

	構造体コピー	データコピー	通信起動	終了待ち
User	—	—	1.787	44.693
User:ST	1.693	—	1.550	42.587
User:ST,DT	1.697	108.437	1.403	38.150

である．その際には，36.06μsec 程度になると予測されている．

図 4 に現在の転送バンド幅と改良を加えた場合のバンド幅を示す．参考までに，Myrinet でユーザレベル通信を行った場合のバンド幅も掲載した．

表 7: NI 処理内訳 (RHiNET-1/NI)
(4096byte 転送時, 単位:μsec)

処理内容	時間
Read SAR	0.452
DMA setup	1.048
DMA Delay	1.062
Read struct.	0.690
Send setup	1.216
Send header	0.528
Delay	0.200
DMA setup	1.048
Delay	0.936
Send delay	0.216
Read data	95.940
Setup FIFO	0.464
Write end of packet	0.680
合計	102.056

段の下がった項目は
クリティカルパス上に無い処理

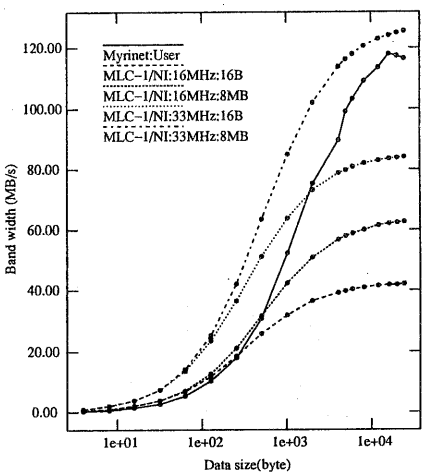

図 4: 転送バンド幅 (RHiNET-1/NI)

7 まとめ

ユーザプログラム，システムコールおよび NI 全般を使用した複数の実装により通信性能がどのように変化するか測定を行った．ホストプロセッサは NI によるデータコピーは処理が逐次的になる事もあり性能はそれほど良い物ではなかった．現在の実装ではハードウェア処理を行う RHiNET-1/NI もそれほど高速ではないが，改良が加えられればこれを用いたユーザレベル通信がもっとも高速に処理される事となろう．

参考文献

[1] Myricom, Inc.: *http://www.myri.com/*.

[2] 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, 電子情報処理学会技報, Vol. 98, No. 234, pp. 1–8 (1998).