解説

コモディティハードウェアを用いた並列処理技術

石川　裕／新情報処理開発機構

超並列コンピュータ開発熱が世界経済の縮小とともに冷め，コモディティハードウェア☆とギガビット級ネットワーク技術による並列コンピュータが注目されている．ATM-LAN等の数百bits/sec級ネットワークを用いた3，4年前のワークステーションクラスタは，poor men's並列コンピュータともいわれていた．しかし，この2，3年で，ギガビット級の高速ネットワークが市販されるようになり，また，高速通信ソフトウェア技術が開発され，既存の超並列コンピュータに匹敵する性能が実現できるようになった．

単に高速なネットワークにワークステーションをつなげても超並列コンピュータに匹敵する性能は出せない．一般的オペレーティングシステムを使った通信では，オペレーティングシステムカーネル（以降単にカーネルと呼ぶ）が介在するプロトコル処理や通信バッファのコピーに伴うオーバヘッドが生じる．これにより，通信ネットワークの持つデータ転送容量（通信バンド幅）を引き出せず，また通信遅延が大きくなり，並列処理に適さない．そこで最近では，カーネルを介在しないユーザレベル通信やプロセッサによるメモリコピーをなくすゼロコピー通信技術が開発されてきている．筆者らは，ギガビット級ネットワークであるMyrinetで結合されたPCクラスタ上で，ユーザレベルのゼロコピー通信を実現するPMと呼ぶ低レベル通信ライブラリを開発した．本稿では，これらの技術を中心にCompaq，Intel，Microsoftが策定したVI(Virtual Interface)アーキテクチャと呼ばれる規格についても触れる．

なぜコモディティハードウェアか？
―クラスタコンピューティングの夜明け―

1990年代前半の高性能並列計算機分野では，超並列計算機の開発が花盛りであり，Thinking Machines社CM-5，Intel社Paragon，Cray社T3-D等の製品が市場に出回った．これらの製品は，Sun社Sparc，Intel社i860，Dec社alpha等の既存のマイクロプロセッサと，独自開発の高速ネットワーク（**表-1**参照）による構成であった．これらの製品開発にはハードウェアのみならずオペレーティングシステムやプログラミング環境の開発に莫大な資金と年数を必要とした．そのため，64台プロセッサ規模で数億円という高価な計算機であった．

一方で，10Mbpsや100Mbps程度のローカルエリアネットワークに接続されたワークステーションを用いて並列環境を構築するという流れがあった．これが並列処理分野でワークステーションクラスタ，PCクラスタと呼ばれている形態の初期の頃のシステムである☆2．並列計算機と比べ，バンド幅は1/10以下，通信遅延100倍という低い性能のネットワークを用いているわけだが，アプリケーションによっては，効率よい並列実行が可能なものもある．また，2台からの並列環境を作ることができることから初期投資に超並列計算機のような莫大な費用を必要としない利点があった．さらに，超並列計算機のモデルチェンジの間隔が3年程度であったのに対し，ワークステーションは半年から1年程度の間隔でより高性能な新製品が市場に出荷されていたため，通信の性能よりも各ノードの計算性能が重要なアプリケーションでは，このようなワークステーションクラスタを利用する価値があった．

ワークステーションクラスタが注目されるようになったのは，コストだけの問題ではない．専用並列計算機上のシステムソフトウェアの完成度の低さ，あるいは，普段使いなれているワークステーション上での環境とまったく違う環境，に対するユーザの不満もかなりあったように思える．これに対し，ワークステーシ

☆　コモディティハードウェアとは一般に市販されているパーツであるハードウェアを指す．

☆2　クラスタという用語は，商用的には高い可用性（Availability）を実現するためのサーバシステムの構成方式を指すことが多い．

表-1　超並列計算機のネットワーク性能

製品名	メーカ	発表時期	リンク速度
CM-5	Thinking Machines	1992年	0.16 Gbps
Paragon	Intel	1992年	1.6 Gbps
T-3D	Cray	1993年	1.2 Gbps
T-3E	SGI	1995年	4.8 Gbps
SR 2201	日立	1996年	2.4 Gbps

表-2　ギガビット級ネットワーク

名前	規格／企業	リンク速度	物理層における通信の信頼性
Fibre Channel	ANSI X3 T11	1.06 Gbps	Reliable
SCI	ANSI/IEEE 1596	1.6 Gbps	Reliable
Memory Channel 2	DEC	1.06 Gbps	Reliable
Myrinet	Myricom	1.26 Gbps	Reliable
Gigabit ethernet	IEEE 802.3z	1.25 Gbps	Unreliable

ョンクラスタ型の並列計算では，ワークステーションのシステムソフトウェアがそのまま使えるという利点があった．

クラスタシステム上では，各計算機上にプロセスをそれぞれ生成し，これらのプロセスがネットワークを介してお互いに通信しあって並列処理が行われる．並列処理記述に必要な全体全通信，バリア同期，リダクション操作等の通信機能を支援している通信ライブラリとしてPVM[1]やMPI[2]が広く普及した．既存OSを使い，FortranやC，C++等の逐次処理記述用プログラミング言語とPVMやMPIを用いた並列アプリケーションが数多く開発された．

PVMやMPIは，TCP/IPプロトコルあるいはその他の通信プロトコルの上に実装される通信ライブラリである．PVMやMPI普及の背景には，フリーかつ完成度の高いソフトウェアが配布され，さまざまな計算機に移植されたことがある．特殊計算機上の特殊な通信ライブラリを使わずに，広く一般に使用可能なソフトウェアの組合せでアプリケーションが開発された．このようなアプリケーションはプログラムを変更することなく多くの並列計算機上およびクラスタ上で動かすことができるようになった☆3．しかし，このようなクラスタシステム上では，通信性能が支配的なアプリケーションで十分な性能を得ることはできなかった．これは主として，使用しているネットワークが10Mbpsや100Mbpsと専用並列計算機のネットワークの1/10以下であり，また，通信性能のチューニングが行われていなかったからである．

超並列計算機並みの通信性能が身近に
―高速ネットワークの出現―

近年，**表-2**に示すように，Fibre Channel，SCI，Myrinet，Memory Channel，Gigabit ethernet等のギガビット級ネットワークが出現している．Fibre Channelは133Mbpsから1Gbpsまでの転送レートが規格化されており，さらに2Gbpsおよび4Gbpsの規格化が進められている．Fibre ChannelはSCSIプロトコルやIPプロトコルを支援している．SCI（Scalable Coherent Interface）やMemory Channelでは，ネットワークにつながっている他の計算機上のメモリに直接データを書き込む機能が提供されている．Myricom社が開発したMyrinetは，3年前よりいち早くswitch構成でのギガビットネットワークを提供したシステムである[4]．最近製品が出始めているGigabit ethernetは上記ネットワークとは異なり，従来のethernet同様にパケット到着を保証していない．

一方，表-1に示した通り超並列計算機と呼ばれる並列計算機のネットワークの物理的性能は一世代前ではギガビット級であり，最近のものでも2～4倍程度の差となっている．最近の高速ネットワークは超並列計算機のネットワークと遜色のないレベルにまで到達しているといえるだろう．ここに大きな変革がおとずれたといっても過言ではない．すなわち，ワークステーションやPCにギガビット級ネットワークカードとスイッチを購入すれば，超並列計算機に匹敵する並列計算機環境が構築可能になったのである．

ネットワークだけ速くても高い性能は得られない
―ソフトウェアの問題点―

しかし，ネットワークの物理的性能が高いだけでは，高い並列処理性能を得ることはできない．従来通りのTCP/IPプロトコル等の通信レイヤの上にMPIやPVMを移植した作りでは，TCP/IPプロトコル処理のオーバヘッドが大きく，最近の高速なネットワークの物理的性能を引き出すことができない．この

☆3　なお，このような誰もが使えるソフトウェアによるクラスタ環境はBeowulf型と呼ばれることがある．Beowulfは米国JPLおよびCALTECHが進めてきたプロジェクトの名前である[3]．現在，米国では，beowulfをトレードマークにしようとしている．beowulf™を使うときは，コモディティハードウェア，フリーかつオープンなソースコードを使っていることが求められるようである．

表-3　TCP/IPプロトコルによるデータ転送のバンド幅

機種	OS	ネットワーク	バッファサイズ	バンド幅
Pentium PRO (200MHz)	NetBSD	100base-T	16,384	30Mbps
Pentium II (300MHz)	Linux	100base-T	65,535	88Mbps
DEC PWS600au (600MHz)	Digital	100base-T	32,768	98Mbps
DEC Alpha (500MHz)	WindowsNT	Gigabit ethernet	65,535	237Mbps *1
Penitum PRO (200MHz)	FreeBSD 2.2.2	Myrinet	—	309Mbps *2

*1 packet engines社のデータによる（http://www.packetengines.com/f_productsandsolutions.htm）
*2 myricom社のデータによる（http://www.myri.com/myrinet/performance/index.html）

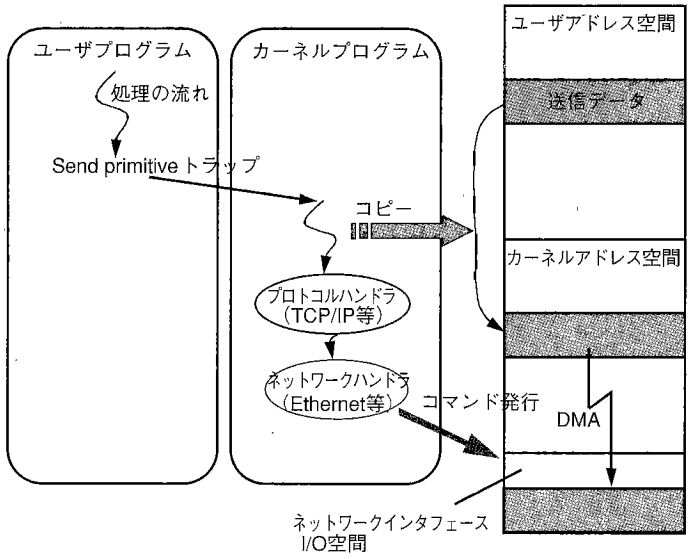


図-1　従来の通信プロトコルの実現モデル

事実を示すために，フリーソフトウェアのnetperf[5]を使用して計測したTCP/IPのバンド幅（point-to-point）を表-3にまとめた．バッファサイズとは，カーネル内の通信バッファの大きさを意味する．バンド幅は，1回の通信でバッファサイズ分を送った場合に，10秒間に何回送れたかを測定して求めている．表-3をみて分かる通り，最近のプロセッサを用いると100Mbpsネットワークでは，ネットワークの性能をほぼ使いきるだけの性能が出ている．しかし，ギガビット級ネットワークに対しては，2割から3割の性能しか出ていない．

なぜ，このように通信性能が低いのであろうか．図-1にTCP/IP等の通信プロトコルの従来型の実現イメージを示す．まず，ユーザプログラムは，データを送るためにsendカーネルプリミティブを呼ぶ．カーネルはユーザ領域にあるデータをカーネルが保持しているバッファ領域にコピーし，TCP/IPやUDPのようなプロトコルをハンドリングしているルーチンを呼ぶ．プロトコルハンドリングされた後に下位層のネットワーネットワークドライバが呼び出され，ネットワークインタフェースカード（以降NICと略す）に対してバッファ領域のアドレスを渡し，NICはデータをネットワーク上に送出する．あるいは，NICによってはTCP/IPハンドラがNIC上のバッファ領域にデータをコピーする必要がある場合もある．受信側は，この逆順で処理される．このようにデータを送るために，カーネル呼び出しと1回あるいは2回のデータコピーおよびプロトコル処理時間がかかるために，実際の通信性能が低くなっている．

このようにカーネルによってプロトコル処理を行ってきた理由は，従来のethernetのようにNICがデータリンク層のプロトコルのみを処理している場合，TCP/IPのような上位層のプロトコルを安全に実現するためにカーネルで実現する必要があったからである．

カーネル呼び出しのコスト，実際にはシステムコールのオーバヘッドは，プロセッサアーキテクチャに依存する．表-4に手元の計算機でのgetpidシステムコール（process idを取り出すだけのシステムコール）のオーバヘッドを示す．数年前に主流だったSun Sparc Station 20では5.5 マイクロ秒と大きい．最近のプロセッサのシステムコールのオーバヘッドはかなり小さくなっているが，それでも数百命令実行するのに等しいコストがかかる．後述のように，通信ソフトウェアを工夫することにより8バイトデータ通信が15マイクロ秒のラウンドトリップ時間で実現できるようになると，システムコールのオーバヘッドが問題となる．

表-5にCプログラムで呼び出すことができるbcopyライブラリを用いた主記憶上のデータコピーのコストを示す．高価なワークステーションでは，データコピーのバンド幅は高速ネットワークよりも優れているが，安価なPCではバンド幅が小さいのが分かる．近年の急速なマイクロプロセッサの性能向上により，バ

ンド幅の違いも年々小さくなってきているが，データコピーに伴うプロセッサの占有および通信遅延の増大の問題は変わることなく存在している．

このように既存のコンピュータを高速ネットワークで接続したシステムでハードウェア性能を生かした通信ライブラリを構築するためには，データコピーおよびシステムコールをなくすことが重要となる．

高い性能を目指して
ー並列処理のためのソフトウェア技術ー

本章では，高速ネットワークの性能を引き出し，かつ，マルチユーザ環境を実現するためのソフトウェア技術の事例として，新情報処理開発機構が研究開発を行ってきたクラスタシステムRWC PC Cluster 2号機（64台のIntel Pentium PRO（200MHz）をMyricom Myrinetネットワークでつなげている）[6]上のSCoreシステムソフトウェアで採用した技術を紹介する．また，関連する研究開発動向について簡単に述べる．

ユーザレベル通信

カーネルを介さずに通信を実現する手法として，ユーザプロセスが直接通信ハードウェアが使用しているメモリおよびコマンドレジスタをアクセスする方法がある．**図-2**に示す通り，NICが使用しているメモリ領域をユーザ仮想アドレスにマッピングして，直接，通信に必要なバッファアドレスやNICのメモリ領域に書き込むことによって実現する方法である．最近の高速ネットワークの実装では，プロセッサ搭載によるインテリジェント化が進み，NIC側でデータリンク層よりも上のプロトコル層をハンドリングすることも可能となってきている．たとえばMyricom社Myrinet[4]のNICには，Lanaiと呼ばれるプロセッサと1MBytesのメモリが搭載されている．

このようなNICで，たとえばUDPプロトコルにおけるポートのような機能を提供することができ，さらにポートの保護をプロセスごとに実現できるならば，複数のユーザが同時に直接アクセスできる環境を実現できる．別な言い方をすれば，NIC側でデータリンク層よりも上のプロトコル層のプロトコルハンドリングを行い，保護機能を実現すればユーザレベル通信が安全に実現できる．このとき，理想的にはNICをアクセスできるプロセスの数は制限されてはならない．しかし，多くのNICは，搭載されているメモリ領域の制限やプロトコルハンドリング用に搭載されているプロセッサ処理能力の制限から，NIC側で多数のポートを効率よく管理するのは難しい．

そこで，限られた資源内で可能なだけポートのようなものを提供し，同時にネットワークを使用できるアプリケーションの数を限定するか，限定されたポート数以上のアプリケーションがネットワークを利用したい場合には，何らかの形でポートを多重化する方法が用いられる．

表-4　getpidシステムコールのコスト

機種	OS	コスト
Pentium PRO (200MHz)	NetBSD	1.9 usec
Pentium II (300MHz)	Linux 2.0.32	1.6 usec
SUN Ultra 2 (300MHz)	Soralis 2.5.1	1.7 usec
DEC PWS600au (600MHz)	Linux 2.0.32	0.36 usec
DEC PWS600au (600MHz)	Digital Unix	0.43 usec

表-5　メモリコピーのコスト

機種	最小値 (MBytes/s)	最大値 (MBytes/s)
SUN Ultral 2 (300MHz)	175.9	209.2
DEC PWS600au (600MHz)	98.9	111.1
Pentium PRO (200MHz)	48.7	52.3

ユーザレベル通信の事例としてRWC PC Cluster2号機上に実装した通信ライブラリPMを取り上げる[8), 9)]．PMはチャネルという資源を提供している．チャネルはポートとは違って，各計算機からみえる同一番号のチャネル同士で通信を行うことができる．PMの現在の実装ではチャネルの数は4つである．チャネルを多重化することによって複数のアプリケーションがネットワークを利用できるようにする実現方法については後で紹介する．

PMが提供しているメッセージ通信では，カーネルが管理しているバッファ領域を用意している．バッファ領域からインタフェースへのデータコピーはMyrinetネットワークインタフェースが持っているDMA機能を用いて行われる．通信のためのカーネル呼び出しはないが，アプリケーションレベルにおける送信データコピーが必要となる．

図-3にPMのメッセージ通信性能を示す．図中，Msgは送信側のバッファ領域にあるデータを受信側のバッファ領域に書き込む性能である．Msg+copyは送信受信ともにユーザデータ領域からバッファ領域へ／からのコピーが入った場合の性能である．8Bytesメッセージ通信のラウンドトリップで15マイクロ秒を達成している．データサイズ8KBytesでコピーなしで119MBytes/s，コピーありで50MBytes/sのバンド幅が実現されている．次節では，プロセッサによるコピーを除去する手法について説明する．

ゼロコピー通信

プロセッサによるデータコピーが介在しない通信方式はゼロコピー通信と呼ばれている．ゼロコピー通信

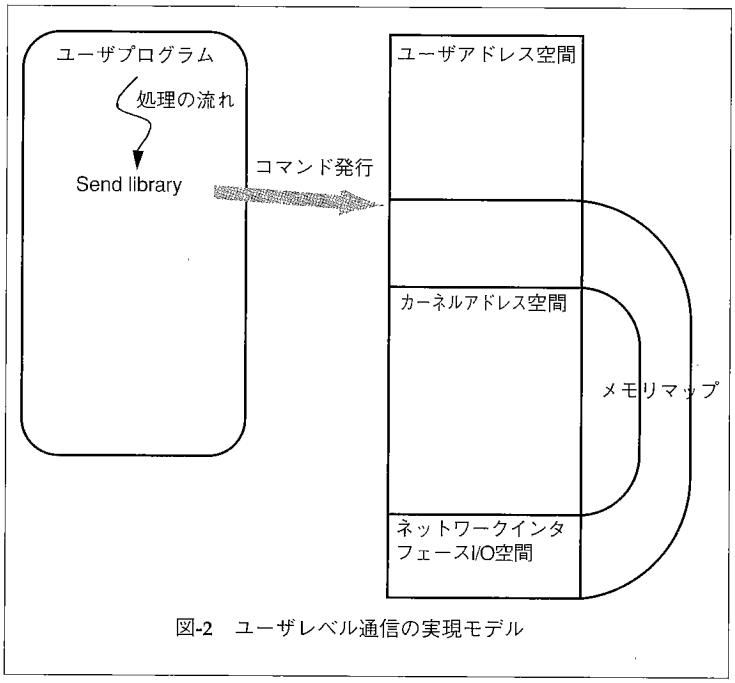


図-2　ユーザレベル通信の実現モデル

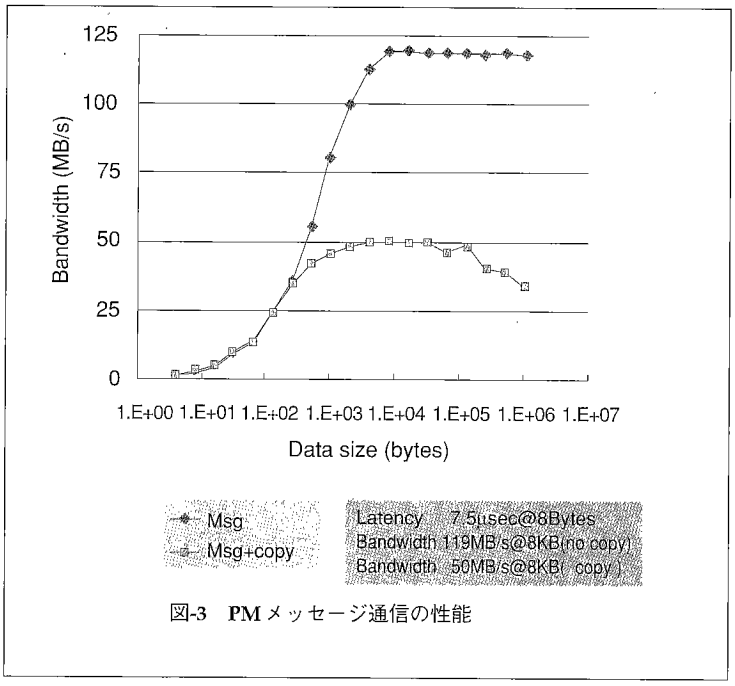


図-3　PMメッセージ通信の性能

では，最終的には，NICがホストメモリのデータをDMA転送することによりネットワークにデータを流すことになる．受信側のNICはデータをホストメモリ（ホストプロセッサのメインメモリ）上にDMA転送する．この意味ではデータはコピーされているのだが，送信側受信側共にホストプロセッサがデータをコピーしないのでゼロコピー通信と名づけられた．

ゼロコピー通信の動きについて，まず，送信側の動きについて説明する．ゼロコピー通信では，NIC側がホストメモリを直接アクセスする．このアクセスは，多くのハードウェアでは物理アドレスによって行われる．一方アプリケーションプログラムは仮想アドレス空間を参照している．したがってデータが格納されているメモリのアドレスを仮想アドレスから物理アドレスに変換してからNICに渡す必要がある．

また，メモリの物理アドレスがNICに渡された後，NICがそのデータを取り出してネットワークに送り出すまで，その物理アドレスに対象データが存在している必要がある．ところが，多くのオペレーティングシステムではページングシステムが動いており，任意のタイミングでページアウトが生じ，物理アドレス上の内容が別の仮想アドレスの内容に置き換わる可能性がある．このため，仮想アドレスと物理アドレスのマッピングを固定し，ページングされないようにするピンダウンと呼ばれる手法が用いられる．

受信側では，NICはネットワークからデータを受け取ると，ホストメモリ上にあらかじめ確保された領域に受信データを転送する．このとき，転送先のメモリがアプリケーションプロセス上で指定された領域であればゼロコピーが実現されていることになる．

送受信が同期して発行されているならば，受信側は受け取ったデータを，受信プリミティブが発行されたときに指定されたアドレスに書き込めばよい．しかし非同期通信において，受信側で受信プリミティブが発行される前に送信側がデータを送信すると，受信側は，受信プリミティブが発行されるまで受信データをバッファリングする必要がある．その後受信プリミティブが発行されると，バッファからアプリケーションにデータをコピーすることになりゼロコピー通信は達成されない．これを回避するために，次のような処理を必要とする．

(1) 受信側は受信アドレス領域をピンダウンして，アドレス情報を送信側に伝える．このとき，書き込まれるメモリ領域の仮想メモリアドレスと物理メモリアドレスの対応を受信側のNIC側が理解できるような仕組みが必要である．物理アドレスを直接送信するのは危険である．ユーザレベルでゼロコピー通信を実現するということは，ユーザプログラムによって間違えた物理アドレス情報が渡されシステム全体が不能になる可能性があるからである．ゼロコピー通信を安全に行う手法については，詳しくは文献9)を参照してほしい．

(2) 送信側は受信側からの受信アドレス情報を受け付けた後に，そのアドレス情報を付加して受信側にデータを送信する．送信メモリ領域は実際の送信を行う前までにピンダウンし，送信完了を待ってピンダウンを解除する．

(3) 受信側は，アドレス情報が正しいかどうかを検査した後に，受信データを書き込み，ピンダウン領域

を解除する．

上記処理は，送信側からみると，リモートメモリへの書き込みアドレスを指定してのデータ転送とみえるため，リモートメモリ書き込み機能を実現しているともいえる．

このようにユーザレベルのゼロコピー通信では，(1) 送信側および受信側双方でのメモリのピンダウン操作のためのカーネル呼び出し，(2) 送信側と受信側でネゴシエーションプロトコルを必要とする．

PM通信ライブラリでは，ピンダウン操作のためのカーネルコールを減少させるためにピンダウンキャッシュという技法を用いている[9]．PMのリモートメモリ書き込み機能では，(1) データ領域のピンダウン，(2) リモートメモリ書き込み，(3) データ領域のピンダウン解除，の3つのライブラリが提供されている．ピンダウンキャッシュとは，データ領域のピンダウン解除の際に，実際にはピンダウンを解除せずに，物理メモリが足らなくなった場合にのみピンダウンを解除する，というものである．ピンダウンが必要になったときに，ピンダウンされているかどうか調べて，過去のピンダウンが解除されていなければカーネル呼び出しを行わない．

図-4 にPMのリモートメモリ書き込みの性能を示す．ピンダウンキャッシュ率が100%とはピンダウンのためのカーネル呼び出しが発生しない状態を指す．このとき，1MBytesデータの送信バンド幅は108.8MBytes/sとなっている．常にピンダウンをしなければならない場合でも1 MBytesデータの送信バンド幅は78.8MBytes/sとなる．

参考のために，図-4にメッセージ通信の性能も示している．本図では分かりづらいが，1回当たりのデータ転送の大きさが8KBytes以下ではメッセージ通信の方がバンド幅が高く，その後はピンダウンキャッシュのヒット率に関係なくリモートメモリ書き込みの方が性能がよいことが分かる．

PMの持つメッセージ通信機能とリモートメモリ書き込み機能を効果的に利用したMPIライブラリであるMPICH-PM[10]を開発している．MPICH-PMは，フリーソフトウェアであるMPICH[11]を利用している．性能評価結果を**図-5**に示す．RWC PCクラスタ2号機上のMPICH-PMでは，8KBytesまではPMメッセージ通信を用い，それ以上のデータサイズにはリモートメモリ書き込み機能を使用している．

RWC PCクラスタ2号機上の性能を示すために並列ベンチマークで有名なNAS Parallel Benchmarksが定義している7つのベンチマークプログラムの結果を**図-6**に示す．PCクラスタ2号機以外の並列コンピュータの性能は，NASAがWWWで提供しているデータを掲載している．RWC PCクラスタ2号機の性能を

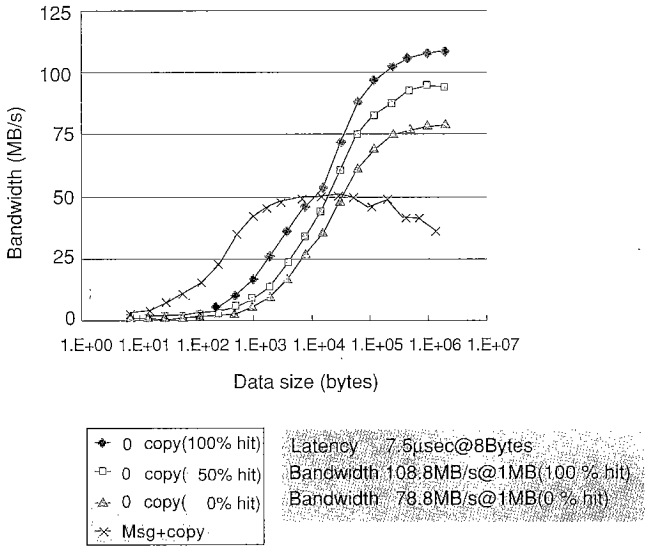


図-4 PMリモートメモリ書き込みの性能

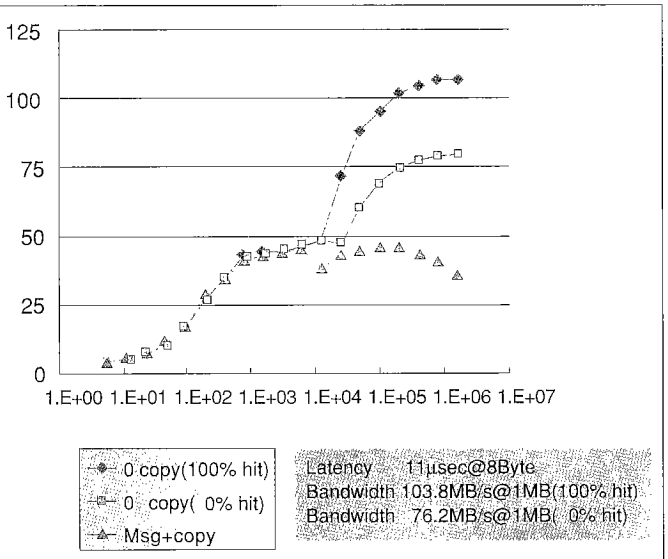
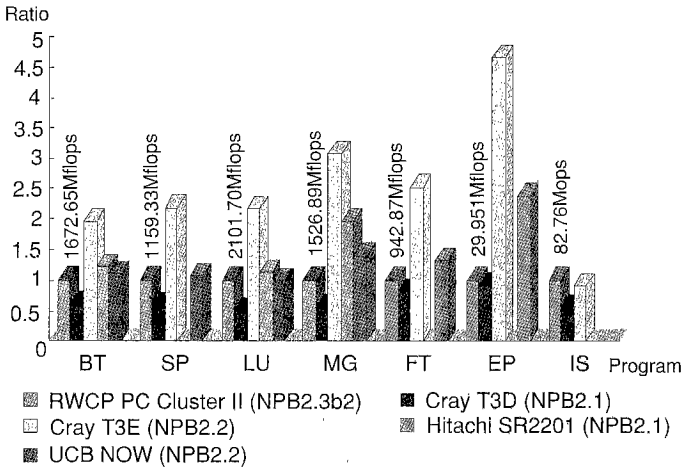


図-5 MPICH-PMの性能

1としたときに，他の計算機がどれだけ速いかを示した．BTおよびSPは計算流体力学を解くときに使われるアルゴリズムを基礎としたベンチマークである．LUは1次方程式を，MGは3次元ポイソン方程式を，FTはフーリエ変換を，ISは整数ソートをそれぞれ解くプログラムである．EPは浮動小数点演算の性能が支配的なベンチマークである．EPプログラムの性能がプロセッサ単体の性能といっても過言ではない．その意味では，単体性能（EPプログラム）では他のシステムはPCクラスタ2号機の2倍以上の性能を出しているが，並列処理プログラムでは逆に2倍あるいはそれ以下の差しかない．PCクラスタ2号機はCray社の一世代前のCray-T3Dより高性能を達成し，最新のCray-T3EにはISプログラムでより優れた性能を発揮



Mflops or Mops

	BT	SP	LU	MG	FT	EP	IS
RWCP PC Cluster II	1672.65	1159.33	2101.70	1526.89	942.87	29.95	82.76
Cray T3D	983.60	654 20	1055.30	839.00	772.10	25.20	47.40
Cray T3E-900	3225.40	2400.70	4529.40	4597.70	2420.00	129.20	76.60
Hitachi SR2201	1978.70		2341.30	2926.20			
UCB NOW	1816.00	1184.10	2079.20	2074.20	1264.30	65.90	

図-6　NAS Parallel Benchmarksの結果

していることが分かる．

スケジューリング

1つのクラスタシステム上で，複数の並列アプリケーションが同時に稼動している場合，それぞれの計算機上にはそれらの並列アプリケーションのための複数のプロセスが動いている．各計算機上でローカルなスケジューラがプロセス切替えを行うと，全体ではバラバラにプロセスが切り替わることになる．

これを，同時に全計算機がプロセスを切り替え，同じ時刻には同じアプリケーションに属するプロセスが実行されているようにするスケジューリング手法をギャングスケジューリングと呼ぶ．

ギャングスケジューリングを行うことにより，クラスタ上の計算機およびネットワークは1つのアプリケーション実行のために独占されていることになる．そこで，ユーザレベル通信の説明の時に述べたポート資源の多重化が可能となる．ギャングスケジューリングによるコンテキスト切替え時に，ネットワーク上にメッセージが浮遊していないことを確認し，各計算機上のNICが保持している状態を待避することによってポート資源を多重化することが可能となる．これにより，ユーザレベルによる高性能な通信機能を保持しながらマルチユーザ環境を実現することが可能となる．

また，ギャングスケジューリングを用いると，科学技術計算に多い複数のプロセスが一斉に計算，通信のフェーズを繰り返すアプリケーションにおいて，受信待ちの際に起きる余分なプロセス切替えを削減することが可能となる．

我々はカーネルを修正せずにデーモンプロセスによってギャングスケジューラを実現したシステムSCore-Dを開発してきている[13]．ギャングスケジューリングのオーバヘッドはタイムスライスを100ミリ秒としたときに64台規模のクラスタ環境でアプリケーション実行時間の4%以下であることが確認されている[12]．

だれでも使える高性能計算環境に向けて
―研究開発の動向と今後―

これまでに発表されているユーザレベル通信やゼロコピー通信を実現する高速通信ライブラリには，UCBのAM[15]，イリノイ大学のFM[16]，プリンストン大学のVMMC-II[14]，コーネル大学のU-NET[17]等がある．1997年12月，Compaq，Intel，Microsoftが主体となって規格化したVIアーキテクチャ（Virtual Interface Architecture，以降単にVIA） V1.0が公開された[18]．VIAはU-NETの研究成果を基としており，ユーザレベルゼロコピー通信を実現するインタフェースを制定している．VIAの特徴は次の通りである．

1. TCP/IPのようなコネクション型通信を支援しており，1対1通信機能を提供している．
2. Send/Receive型とリモートメモリ読み込み／書き込み型の通信を提供している．ただしリモートメモリ読み込みのサポートはオプションである．
3. カーネルを介さないユーザレベル通信を提供している．

VIAは，現在，Intel社からImplementation Guideのドラフト版が公開されている．4月に公開予定のConformance TestsおよびPerformance Testsは原稿執筆時にはまだ公開されていない．Sample Device Driver Codeは5月公開予定のようである．現在，VIA製品の発表が目白押しである．これらはWindows NTの小規模クラスタ向けで，並列データベース等のアプリケーションが最初のターゲットのようである．

VIAを推進している3社をみると，並列処理のための高速通信機能のインタフェースはVIAに決まったと思われるかもしれない．実際，VIAインタフェースを持ったギガビット級ネットワークの製品発表が相次いでいる．しかし，VIAは，コネクション型かつ1対1の通信機能しか提供していないため並列処理向けの通信機能を実現するには適していないだろう．今後，VIA上にMPIやPVMが実装されて初めて，並列処理におけるVIAの評価が決まるであろう．

ギガビット級のネットワークは，現在のところ，クラスタシステムのためのネットワークあるいは基幹ネットワークと考えられている．これがLANのように当たり前に使われ出すと，フロア内あるいは建物内に設置された計算機群全体が1つの並列処理を可能とするクラスタ計算機のようになるだろう．LAN上にはさまざまな種類の計算機が接続される．自ずと異機種環境における並列処理が現実的になってくると考えられる．

数十万円のパソコンと高速ネットワークで数年前の超並列計算機以上の性能を出す並列計算環境が手に入るようになった．これは個々人が自前のスパコンを持てる時代が到来したといっても過言ではない．しかし，現在の逐次処理言語とMPIやPVMを使ってプログラミングできる人の数は限られている．このために並列化コンパイラや，HPF，openMP等のコンパイラ指示文を付加することによって並列化する処理系の開発が行われている．逐次で記述されたプログラムを徐々に並列化していける，逐次処理から並列処理へのシームレスなプログラミング環境が望まれる．

コモディティハードウェアの組合せによるクラスタはいわばDIY（Do It Yourself）型の手作り並列計算機という色彩が現在は強い．計算科学を研究している人が，自ら計算機システム屋の作っているようなクラスタを真似して作るのはハードルが高すぎる．ハードウェアの組合せだけでなく，ソフトウェアをどう組み合わせたらよいかという問題もある．現在，Linuxオペレーティングシステムの CD-ROMを販売しているRed HatはBeowulfプロジェクト[3]の成果を元にしたLinux上で稼動するクラスタシステムのためのフリーソフトウェアを収録したExtreme Linuxを配布している．米国DOE（エネルギー省）傘下のLos Alamos国立研究所が中心になって，次のExtreme Linuxリリースを目指したプロジェクトが進行中である[7]．この中には，新情報処理開発機構が開発したSCoreシステムの一部も収録されることになっている．CD-ROM化によって，クラスタシステムが広く浸透していくことを期待している．

近年のハードウェアの性能向上のスピードは著しい．性能向上の早さは，マイクロプロセッサ，ネットワーク，メモリバス，I/Oバスの順ではないだろうか．このような状況でシステムソフトウェアをどう構築するかはトータルシステムとしてのバランスの上になり立つものである．最新の技術を見極めながらシステムソフトウェアを短期間に構築する体制が求められるだろう．

ギガビット級ネットワークの出現は，並列計算機の構成に大きなトレンドの変革をもたらしたといえよう．すなわち，今まで高価で個人レベルでは取得できなかった並列計算環境が，パソコンと高速ネットワークの組合せで実現可能となったことである．並列計算環境が身近になることにより，新たなる並列計算市場が生まれることを期待している．

参考文献

1) http://www.epm.ornl.gov/pvm/
2) http://www.mcs.anl.gov/Projects/mpi/index.html
3) http://cesdis.gsfc.nasa.gov/beowulf/beowulf.html
4) Boden, N. J., Cohen, D., Felderman, R. E., Kulawik, A. E., Seitz, C. L., Seizovic, J. N. and Su, Wen-King: Myrinet − A Gigabit-per-Second Local-Area Network, IEEE MICRO, 15(1), pp.29-36 (Feb. 1995). http://www.myri.com/
5) http://www.cup.hp.com/netperf/NetperfPage.html
6) http://www.rwcp.or.jp/lab/pdslab/clusters/
7) http://www.extremelinux.org/
8) Tezuka, H., Hori, A., Ishikawa, Y. and Sato, M.: PM: An Operating System Coordinated High Performance Communication Library, In Peter Sloot and Bob Hertzberger, editor, High-Performance Computing and Networking, volume 1225 of Lecture Notes in Computer Science, Springer-Verlag, pp.708-717 (Apr. 1997).
9) Tezuka, H., O'Carroll, F., Hori, A. and Ishikawa, Y.: Pin-down Cache: A Virtual Memory Management Technique for Zero-copy Communication, IPPS/SPDP'98, IEEE, pp. 308-314 (Apr. 1998).
10) O'Carroll, F., Tezukua, H., Hori, A. and Ishikawa, Y.: MPICH-PM: Design and Implementation of Zero Copy MPI for PM, To appear at ICS'98 (July 1998).
11) Gropp, W. and Lusk, E.: MPICH Working Note: Creating a New Mpich Device Using the Channel Interface, Technical Report, Argonne National Laboratory (1995).
12) 堀, 手塚, 石川: ギャングスケジューリングの高速化技法の提案, JSPP'98, 情報処理学会 (1998).
13) Hori, A., Tezuka, H., O'Carroll, F. and Ishikawa, Y.: Overhead Analysis of Preemptive Gang Scheduling, In D. G. Feitelson and L. Rudolph, editor, IPPS'98 Workshop on Job Scheduling Strategies for Parallel Processing, volume 1291 of Lecture Notes in Computer Science, Springer-Verlag, pp.262-276 (Apr. 1998).
14) Dubnicki, C., Bilas, A., Chen, Y., Damianakis, S. and Li, K.: VMMC-2: Efficient Support for Reliable, Connection-Oriented Communication, In HOT Interconnects V, pp. 37-46 (1997).
15) http://now.cs.berkeley.edu/AM/active_messages.html
16) http://www-csag.cs.uiuc.edu/projects/comm/fm.html
17) http://www2.cs.cornell.edu/U-Net/
18) http://www.intel.com/design/servers/vi/

（平成10年5月25日受付）