オペレーティング・システム　55－5
（１９９２．６．８）

クラスタシステムのための通信方式

田胡　和哉　　根岸　康　　干場　美佳子
日本ＩＢＭ　東京基礎研究所

光ファイバを用いた、きわめて広帯域の通信媒体が開発されている。しかしながら、既存の通信ソフトウェアがこれを十分活用しているとは言い難い。特に、通信の実行過程で行なわれる、通信データのソフトウェアによるコピーにより、全体のスループットが制限されてしまうことが起きている。このコピーをすべてなくし、通信媒体の性能を有効に利用することを目的とした、通信機構の実現方式について述べる。さらに、提案方式をクラスタシステムの実現に利用することを目的とし、通信機構およびこれを利用した分散ファイルシステムを、専用ハードウェアで結合された複数台の個人用計算機上に実現した。実現法、および、得られた知見について述べる。

COMMUNICATION MECHANISM FOR CLUSTER SYSTEMS

Kazuya Tago　Yasushi Negishi　Mikako Hoshiba

IBM Research, Tokyo Research Laboratory

5 - 18 Sanbancho, Chiyoda-Ku, Tokyo 102, Japan

High performance communication media using optical link are becoming available. However, communication software which is currently used can not make full use of their advantage. Data copies which are performed by the software limit the total throughput of the communication system. We propose an approach to improve communication system performance by eliminating these data copies. We try to apply the approach for implementation of cluster systems. A prototype system has been developed by implementing the communication system and distributed file system using the approach. The design and performance evaluation of the system is stated in this paper.

1．まえがき

光ファイバや半導体レーザの製造技術の発展により、１Ｇｂｐｓを越える、きわめて広帯域の通信媒体が低価格で利用できるようになっている。これを利用することにより、単に従来の通信システムの性能が改善できるだけでなく、新たな形態の計算機システムが実現できる可能性が生じてきた。たとえば、既存の計算機を効率よく接続する機構として利用することにより、スケーラブルな性能を持つシステムが構築できる。

このように、通信機構の高機能化は新たな可能性をもたらすものであるが、従来のシステムソフトウエアやアーキテクチャをそのまま利用してこれを実現することは、困難である。たとえば、ＤＲＡＭのサイクルタイムは通信媒体の性能ほど飛躍的に向上しないために、プロセッサによる主記憶領域間でのデータコピーの実行速度は、通信媒体上でのデータの転送速度より低くなりつつある。このため、通信機構全体のスループットは、プロトコル階層を実現するために通信機構内部で行われる、データのコピーによって制限されてしまう。このような極端な条件は、従来の、たとえばIP等の通信プロトコルや、ソケット等の通信機構の実現には考慮されていない。

本稿では、通信媒体の性能向上に見合った通信機構の実現方式について提案する。さらに、これを、高性能な通信機構の適用分野として最も有望なものの一つと考えられる、クラスタシスタム[1]の実現に適用する。クラスタシステムは、計算機や周辺機器を閉鎖型の高性能通信機構を用いて結合することにより、数値計算、データベース処理等の分野における大規模なサーバシステムを実現することを目的としている。複数の構成要素を結合して性能を向上することが目的であるから、その通信機構は、スループット、レスポンスの双方に関してきわめて高い水準の性能を実現することが要請される。

通信性能を改善するうえでの基本方針として、通信過程で行われるデータのコピーをまったくなくしてしまうことに目標をおく。すなわち、仮想空間上にあるユーザプログラムのデータを、１回もコピーすることなく他の計算機上のユーザプログラムの仮想空間、あるいは、入出力装置に転送する。このためには、新たな通信プロトコルおよび制御ソフトウエアを設計することのみならず、ユーザプログラムに提供する機能のインタフェース（ＡＰＩ）、および、通信ハードウエアの機能についても詳細に検討する必要がある。これらについて述べる。また、提案した方式にもとづいた通信機構、および、これを利用したファイルシステムを実現したので、実現方法およびその性能について述べる。

2．通信方式の検討

2．1　クラスタシステム

クラスタシステムでは、図1に示すように、クラスタを実

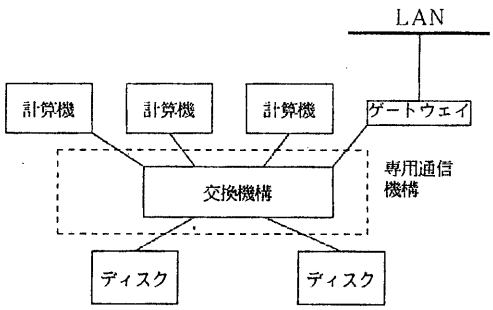

図1．クラスタシステム

現するためだけに利用される専用の通信機構を実現し、これを用いて、計算機、および、ディスク等の周辺機器を接続することにより構成される。すなわち、この通信機構は、計算機間の接続、および、周辺機器の接続の双方に利用できるように設計される。以下、結合される単位をノードとよぶ。通信機構は、光ファイバを用いた通信媒体、および、クロスバスイッチ、多段結合ネットワーク等から構成される交換機構を用いて実現される。１００Ｍｂｐｓから１Ｇｂｐｓ程度の媒体が開発されている。

クラスタシステムでは、既存の、単一プロセッサ向きに開発されたアプリケーションプログラムの処理スループットを向上すること、および、並列処理によりデータベース処理や数値計算処理の処理時間を短縮することを目的としている。

クラスタの通信機構は、ＬＡＮの通信機構に比べて、より簡素化し、性能に重点をおいた設計が可能である。たとえば、この通信機構は、ＬＡＮ、ＷＡＮ等の開放型ネットワークとはゲートウエイを用いて接続され、未知の計算機との接続に直接利用されることはない。すなわち、信頼できるノード(trusted node)間の通信のみに利用され、保護機能を省略できる。また、通常、ただ一つの通信媒体が用いられ、多用な通信媒体に対応する必要がない。

2．2　ＬＡＮ用通信機構

現在、計算機間の通信機構として、最も広く用いられ、かつ、技術が確立されているものの一つとして、ＴＣＰ／ＩＰ等のプロトコルに基づくＬＡＮ用システムがあげられる。クラスタ用通信にもこの技術を適用することができれば、開発は最も容易である。適用が可能であるか否か検討してみる。

２．２．１　ＬＡＮ用通信機構の実現

ＬＡＮ用通信機構は、ソケットを経由してユーザプログラムから直接利用される場合と、分散ファイルシステム等のオペレーティングシステムの機能の実現のために核内で利用される場合がある。典型的な通信機構の構造を、図２に示す。

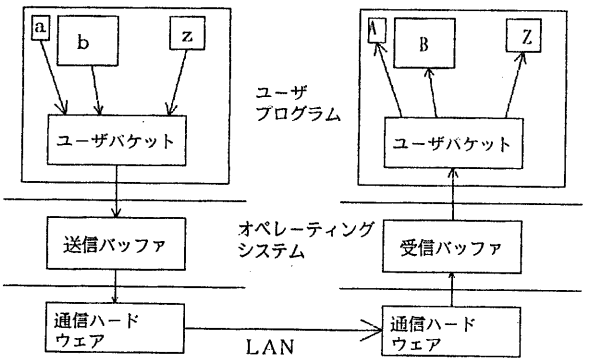

図２　通信機構の構造

ここでは、送信側のユーザプログラム中のデータａ－ｚを、受信側のユーザプログラム中の領域Ａ－Ｚに転送している。呼び出し側のユーザプログラムは、ａ－ｚをユーザバケットに集め、システムコールによって、その送信をオペレーティングシステムに依頼する。システムコールの仕様は、ソケット形式が広く用いられている。データの送信は、ファイルへの書き出しと同様に、

write(fd, buffer, size)

fd: ファイルデスクリプタ

buffer: ユーザバケットのアドレス

size: ユーザバケットのサイズ

によって行われる。送信先は、ファイルデスクリプタによって間接的に指定される。

このシステムコールにより、オペレーティングシステム核内にある通信機構は、送信のためのバッファを割り当て、データをコピーする。これに、ｔｃｐ，ｉｐ等の通信プロトコルによって定められたヘッダを付加し、通信ハードウェアによって相手先に送信する。受信側では、受信バッファを割り当て、データを受信する。受け手のユーザプログラムは、ユーザバケットを割り当て、readシステムコールを発行する。これにより、受信バッファからユーザバケットへのデータ転送が行われる。ユーザプログラムは、このバケットを分解して、ユーザデータＡ－Ｚにデータを配分する。これにより、送信側のユーザデータａ－ｚが受信側のユーザデータＡ－Ｚにそれぞれ転送されたことになる。

ＮＦＳ+)の実現について述べる。ＵＮＩＸシステムでは、ファイルシステムは、核内に設けられたファイルキャッシュを基礎として実現されている。キャッシュは、８ｋバイト程度のファイルページを単位として構成されている。異なるノード間でファイルキャッシュページを転送する機構を実現することにより、遠隔ファイルアクセスが実現される。ＮＦＳの実現では、ユーザバケットを作成することなく、ファイルキャッシュページから直接送信バッファを作成し、受信バッファから直接データをファイルキャッシュに転送することによりオーバヘッドの削減に努めている。

２．２．２　クラスタへの適用可能性の検討

分散システムの実現において最も基本的な機能の一つである、分散ファイルシステムの性能について検討してみる。図３に、ＩＢＭ社のＰＳ／５５++)計算機上に実装されたＮＦＳに関して、８ｋバイトのファイルキャッシュページの転送に要する時間の内訳を示す。通信媒体として１０Ｍｂｐｓのイーサネット、オペレーティングシステムとしてＯＳＦ／１+++

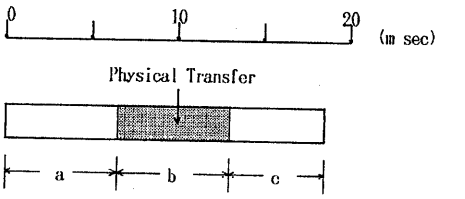

図３　キャッシュページ転送時間の内訳

+)　ＮＦＳは，Sun Microsystemsの登録商標です。
++)　ＰＳ／５５は，International Business Machinesの登録商標です。
+++)　ＯＳＦ／１は，Open Software Foundationの登録商標です。

)を用いている。PS/55は、INTEL社の80386プロセッサを実装している。

図3aの部分は、送信側のソフトウェア処理に費やされた部分である。これは、おもに、送信バッファの作成と相手先アドレスの決定に費やされている。送受信バッファの実現には、mbufと名付けられた、チェーニングが容易に行える構造体を用いる。送信時には、キャッシュページ、UDPプロトコルヘッダ、および、IPプロトコルヘッダの各々に対応するデータ領域を割り当て、mbufを用いて結合することにより送信バッファを作成する。これを、通信媒体の仕様に対応したサイズのパケットに分割して送出する。この例に用いた通信ハードウェアでは、DMAを用いておらず、送信バッファからハードウェアの持つバッファ領域にコピーを行うことによって実際の送信が行われる。

図3cの部分は、受信側のソフトウェア部分に費やされた部分である。分割して送信された複数のパケットの各々にバッファ領域を割り当て、相互に結合して受信バッファを構成する。内容を分析し、UDP、および、IPパケットヘッダを、mbufチェインの構造を変更することにより削除する。これらの処理により、図3に示したオーバヘッドが生じている。

図3から、10Mbpsの媒体の場合でも、プロトコル階層の処理に要する時間が、実際のデータ転送に要する時間より大きいことがわかる。このため、通信媒体の転送速度を向上しても、通信機構全体の性能向上が少ないことがわかる。すなわち、この例では、より広帯域の通信媒体の導入は、ほとんど意味がない。ここでは、3MIPS程度のプロセッサを用いていた。この10倍程度高速のプロセッサも利用されているが、100Mbpsの媒体を用いれば図3と同様の比率になると予想される。より高速の媒体を利用するためには、ここでみたソフトウェアのオーバヘッドを無視しうる程度まで削減する必要がある。

2.3 通信方式の提案

2.3.1 オーバヘッド要因の考察[2]

2.2において見たように、既存の通信機構の実現をそのままクラスタに適用することは困難であると考えられる。クラスタ向きの通信機構を新たに設計するにあたり、通信システムの構造をより一般的に議論してみる。

ここでは、特に、通信システムの内部で行われるデータのコピーに注目して構造を論ずる。たとえば、図2に立ちかえって考えてみると、少なくともプロセッサによるデータコピーが4回行われている。最も高性能なワークステーションにおいても、主記憶装置を構成するDRAMのサイクルタイムは100n sec（10M Hz）前後であり、データバス幅は128ビット程度である。そこでは、主記憶領域間でのデータ転送は、

10 ※ 128 / 2 = 640 M bps

程度しか得られない。ここで、2で割っているのは、読みだしと書き込みを考慮している。そこで、1回のデータコピーでも無視しえない影響があり、通信システム全体の性能が、内部で行われるコピーの回数によって制限されてしまうことになる。このように、大量のデータコピーでは、その実行速度は、MIPS値ではなく、主記憶の記憶素子のサイクルタイムによって制限されてしまう。この結果、プロセッサの処理速度ほど急速に改善されない。

データコピーに注目した議論とは別に、プロトコル階層に注目した議論の方法がある。しかしながら、特にクラスタでは、通信媒体の性質に依存した特殊なプロトコルを利用することが可能であり、一般的な議論がしにくいので、ここでは議論は行わない。

図2において行われている4回のデータコピーは、ユーザプログラムがユーザパケットを作成、分解するために行うものと、ユーザパケットと核内のバッファ間の転送のふたつに分類できる。

ユーザプログラムがパケットを作成せずに通信を行えるようにすることは、一般にはむつかしい。限定的な方法として、たとえば、Amoebaシステム[3]における通信のAPIでは、制御情報と、一般のデータを分割して指定できるようにしている。制御情報はデータ量が少ないことが普通であるので、ユーザパケットを作成して転送する。より多量な、一般のデータは、1個にかぎり、パケットを作成せず直接アドレスを指定してシステムコールが発行できる。

核内の送受信バッファに伴うコピーについて検討する。送信すべきデータは、誤りによる再転送が必要になる可能性があるため、通信の完了が確認されるまで送信側で保持している必要がある。送信側のデータバッファは、送信者が通信全体が完了するまで長期間停止する危険を避けるために設けられている。受信側に転送されたデータはユーザプログラムが受信要求を行うまでは送り先が確定しないために、一時データを蓄えておくための受信バッファが必要である。

受信バッファを省略する方法として、同期メッセージの交

換による方法が用いられてきた。送信において、送信要求があった事実のみを同期メッセージとして先送りし、後に、送り先のノードで受信要求が発生しデータの送り先が確定した時点で同期メッセージを受信側から送信側に送り返すことにより、受信バッファを省くことができる。

受信バッファを設けたことによるデータコピーを省略する今一つの方法として、仮想記憶機構を用いる方法がある。通信データが記憶管理機構のページサイズより大きければ、大半のデータはコピーではなく、受信バッファに割り当てられたページをユーザ空間に再マッピングすることにより効率良くユーザプログラムが利用できる。

2．3．2　方式の提案

2．3．1で議論したデータコピーを、すべて無くす方法について検討する。

前述のように、送信バッファは送信者が長期間停止するのを防止する目的を持っている。クラスタの実現では、広帯域の通信媒体のみが用いられること、転送の誤りの確率がきわめて低いこと、等の理由により、送信バッファは省略してもさしつかえないものと考えられる。

ユーザプログラムが、送信したいデータの各々（図2　a－z）のアドレスとサイズを個別に指定できるＡＰＩを設定すれば、ユーザパケットの作成も省略できる。すなわち、送信側の通信ハードウェアが、ユーザの論理アドレス空間から直接データを集めてデータ転送を行うようにする。

受信側のデータコピーを省略するためには、より根本的な変更が必要になる。従来の、ユーザパケットのアドレスを受信者が指定し、そこへデータを転送する方式では、受信要求が生ずるまでデータ転送が行えないために、原理的に少なくとも一つのバッファが不可欠である。言い替えれば、ユーザデータ図2　a－zを、ユーザプログラムが指定した領域Ａ－Ｚに転送することを、中間バッファ無しに行うことは、このままでは原理的に不可能である。2．3．1で述べた、同期パケットを交換する方法や、仮想記憶システムを利用することにより中間バッファを省略する方法が、改善方法として利用されてきた。しかしながら、前者では同期パケットの交換によるオーバヘッド、後者ではページの再マッピングのオーバヘッドが新たに生じ、適用分野が限定されている。

ここでは、通信機構の仕様自体を変更してデータコピーを無くすことを考える。受信側では、受信したデータを保持する領域をユーザプログラムが指定するのではなく、受信時に、通信機構が領域を割り当て、そのアドレスをユーザプログラムに通知することにする。これは、受信バッファを、ユーザプログラムの論理アドレス空間におくことにより、実現できる。データの受信に際し、通信ハードウェアは、ユーザプログラムの論理アドレス空間中にある受信バッファの領域を割り当て、そこへデータを直接転送する。この受信バッファは、ユーザプログラムが、コピーせずそのまま利用する。

このとき、通信システムが単一のユーザパケットを転送するのではなく、構造を持ったデータを転送できるようにすることにより、受信側で行われる、ユーザパケットの分解処理に伴うデータコピーも無くすことができる。提案方式では、図4に示すように、送信側のユーザデータａ－ｚには、受信側でそれぞれ別個のメモリ領域が通信ごとに新たに割り当てられる。ユーザプログラムは、通信終了後もそれらを内部データとして利用し続ける

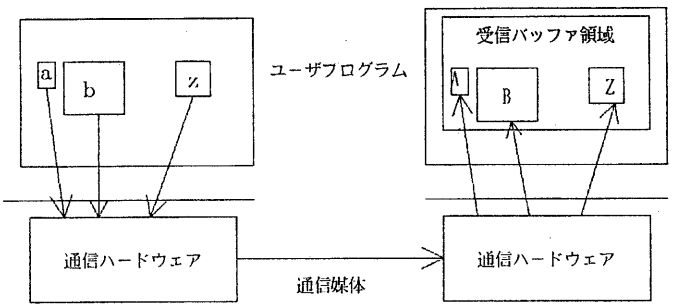

図4　提案方式による通信

2．3．3　仕様設計

提案方式は、一つのＲＰＣの仕様としてまとめ、利用しやすい形式にまとめることができる。これを、ＡＰＩとして用いる。設計したＲＰＣは、同期型であり、ユーザプログラムや分散ファイルシステムは、プリミティブを呼び出すことにより利用できる。cpi_msg_rpcﾌﾟﾘﾐﾃｨﾌﾞにより呼出し、cpi_msg_receiveにより受付、cpi_msg_replyにより受け付けたＲＰＣの返答を行う。呼出を行った者は、cpi_msg_replyが実行されるまで停止する。

cpi_msg_rpcの呼出形式は、

cpi_msg_rpc (port,

addr, size, tag --- 第１引数のアドレスとサイズ
addr, size, tag --- 第２引数のアドレスとサイズ

);

となる。tagによって、引き数ごとのデータ転送に関する細かな制御を行う。たとえば、呼出時に呼出手から受け手に転送されるデータか、終了時に逆方向に転送されるデータかを識別するビットを含む。

ＲＰＣの受付においては、ユーザプログラムは、その領域内に十分な大きさを持つ受信バッファを登録しておく必要がある。はじめて通信を行う前に、

cpi_mem_reg(size, addr, mode)
size ---------- 受信バッファ全体のサイズ
addr ---------- 開始アドレス
mode ---------- 割り当てのモード

を発行する必要がある。これにより、領域の割り当て、および、種々の初期化が行われる。実際の受信は、

p = cpi_msg_receive(port)

により行われる。受信されたデータは、すべて、cpi_mem_regによって指定された領域内に割り当てられる。受信データは、全体として、図５に示す構造を持つ。構造情報中に、受信されたデータのサイズおよびアドレスが記述されている。cpi_msg_receiveの返り値ｐに構造情報の先頭アドレスが返される。受信データは、構造情報中のアドレスを利用する事により、参照できる。構造情報は、受信側のユーザプログラム中では、構造体としてあらかじめ宣言しておく。

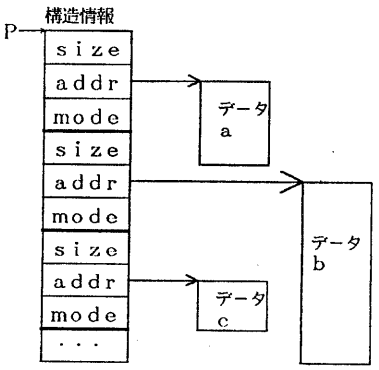

図5　受信データの構造

受信側でＲＰＣの実行が終了すると、

cpi_msg_reply(p)

によって戻り値を返す。受け手の通信ハードウェアは、構造情報を参照しながら戻り値を送信し、送り手の通信ハードウェアはmpi_msg_rpcの引数によって指定された領域に値を返す。受信時に割り当てられた領域は、指定のないかぎり、自動的に開放される。

２．３．４　利用法

このＲＰＣは、分散ファイルシステムを実現するために核内から利用される。また、ＤＣＥ　ＲＰＣ等の既存のＲＰＣを効率よく実現するために、システムコールを経由してユーザプログラムから直接用いられる。分散ファイルシステムの実現、および、ＤＣＥ　ＲＰＣの実現の概要を示す。

（１）　分散ファイルシステム

遠隔ノード上にあるファイルに対する書込について述べる。ユーザプログラムからシステム核に渡された書込データは、ファイルキャッシュに一旦保持され、その後、ファイルが存在する遠隔ノードのファイルキャッシュにＲＰＣによって転送される。ＲＰＣの引数として、各制御情報の他に、書き込むべきデータが保持されているファイルキャッシュのページを直接指定し、ファイルが存在するノード上のファイルサーバを呼び出す。ファイルサーバには、それぞれのデータに別個の領域が割り当てられて渡されるので、ファイルサーバは、ファイルページを受信するために割り当てられた領域をそのままファイルキャッシュページとして転用することができる。このようにして、コピーなしにファイルページを転送することができる。

（２）　ＤＣＥ　ＲＰＣ

ＤＣＥ　ＲＰＣでは、呼出側と受付側のスタブを用いることにより、他のノードにある手続きを呼び出すことを、同一ノード上での手続き呼出と同一の手順で行えるようにしている。ソケットによる通信を用いる呼出スタブは、手続きの引数からユーザパケットを作成する。受付スタブは、ユーザパケットから手続きの引数を取り出して実際の手続き呼出を行う。

提案方式では、呼出側のスタブは、cpi_msg_rpcプリミティブによって手続きの引数をそのまま転送する。たとえば、ある引数が配列である場合にも、その先頭アドレスとサイズを指定すれば転送できる。受付側のスタブでは、cpi_msg_receiveが返す構造情報から引数のアドレスを得て、手続きを呼び出す。たとえば、配列が引数である場合には、構造情報には、そのアドレスとサイズが保持されている。使用している記述言語がＣである場合には、そのアドレスのみを取り出して手

続き呼出の引数とすればよい。呼び出すべき手続きの引数の構造についてはインタフェース記述言語によってあらかじめ記述されているので、このようなスタブは容易に生成することができる。この方式によれば、配列自体は、送信側でも受信側でもプログラムによってコピーする必要がない。

3．実現方式の検討

提案方式は、通信ハードウエアの機能に強く依存している。このようなハードウェアの実現方式を中心として、設計したＲＰＣの実現方式について述べる。

cpi_msg_rpcプリミティブは、通常の手続きとして実現されている。呼出時に、その引数は、スタック上に連続的に積まれている。このスタックの先頭アドレスを通信ハードウエアに与えるのみで、ソフトウェアによる送信処理は終了する。

通信ハードウェアは、このスタックにアクセスし、送信すべきデータのアドレスおよびサイズを把握し、パケットを作成しながらパイプライン処理によって送信を行う。受信側のハードウェアは、論理アドレス空間ごとに、受信バッファの空き領域を管理する表を保持している。これを用いて、受信したデータに対してパイプライン処理によって空き領域を割り付け、そこへデータを書き込む。

このとき、最も重要な点は、通信ハードウェアが論理アドレス空間に直接アクセスすることにある。そこで、通信ハードウェアは、アドレス変換、および、ページフォールトの処理を行う必要がある。

アドレス変換を行う際、マルチプロセッサにおけるＴＬＢのコヒーレンシ維持と同様の処理の必要が生じる。たとえば、ページ枠取り上げ処理によりアドレス変換表の書き換えを行っている最中に、そのページに対して通信ハードウェアがデータを書き込むと、そのデータがそのまま捨てられてしまうおそれが生ずる。通信ハードウェアはアドレス変換表を参照するのみで、書換を行わないことを利用し、通信を実行していない期間に限ってページ枠のとりあげを行うことにより、同期によるオーバヘッドを小さくする方法をとる。

通信ハードウェアが論理アドレス空間から読出を行っている最中にページフォールトが発見された場合には、オペレーティングシステムにページインを依頼し、転送を中断し、ページイン終了後に転送を再開する必要がある。これによる転送の中断に対応できるようにプロトコルを設計する。論理アドレス空間への書き込みを行っている際にページフォールトが発見されたら、別の、すでに確保してあるページ枠にデータを書き込む。これにより、転送を中断せずにページフォールトに対応できる。このページ枠はすべての転送が終了した後にこの論理アドレス空間にオペレーティングシステムがマッピングする。

4．実現

提案方式による通信機構の一部を実現し、それを利用したクラスタシステムを実現した。

実現したシステムは、４台のＰＳ／５５ワークステーションをノードとして結合することにより構成されている。ＯＳＦ／１、および、ＯＳ／２オペレーティングシステム用ドライバを作成した。通信機構は、実際にアダプタカードを作成するのではなく、別個のＰＳ／５５システムを用いてエミュレーションを行った。これを通信エミュレータと名付ける。通信エミュレータから多芯ケーブルを経由して各ノード計算機の主記憶にアクセスできるハードウェアを作成し、これを、実際の通信媒体の代わりとした。約２３Ｍ　ｂｐｓの転送速度を持つ。

実現したＲＰＣは、現状では、オペレーティングシステム核内からのみ利用でき、ページフォールトの処理機能は実現されていない。ＮＦＳを改造することにより、分散ファイルシステムを実現した。

ＯＳＦ／１によって制御されるノードをファイルサーバとして用い、ＯＳ／２によって制御されるノード上でトランザクション処理を実行する、サーバを作成した。ファイルサーバを二重化することにより、フォールトトレランシを実現している。

5．考察

試作機を実現することにより、ソフトウェアによるデータコピーを行わない通信システムが実現できることが確認できた。

方式の有効性について考える。ＮＦＳシステムの性能の比較を図６に示す。通信媒体の転送速度は、条件をそろえるために、双方とも１０Ｍ　ｂｐｓにしてある。従来の方式に比較して小さなオーバヘッドで通信が行えることがわかる。しかしながら、試作機ではハードウェアサポートによりデータコピー以外の部分のオーバヘッドも削減されおり、正確な比較にはなっていない。今、仮に、データコピーおよび中間バッファの割当以外の部分の実行時間が両者ですべて同一であると仮定した場合の、通信機構の性能の比較を図７に示す。

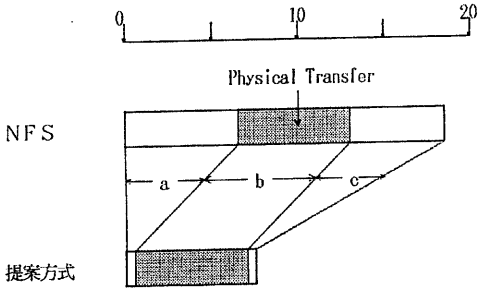

図6　キャッシュページ転送時間の比較

縦軸が、８ｋ　バイトを単位として通信を繰り返し実行した場合の、転送されるデータのスループットを示す。横軸が、通信媒体の速度を示す。スループットの上限が４倍以上改善されている。

提案方式を導入したことによる、アプリケーションプログラムやシステムソフトウェアに与える影響について検討してみる。分散ファイルシステムは、従来と同一のインタフェースを維持できるので、アプリケーションプログラムに対する影響はほとんどない。ＤＣＥ　ＲＰＣに関しては、従来とスタブの構造が異なるので、再コンパイルが必要になる。

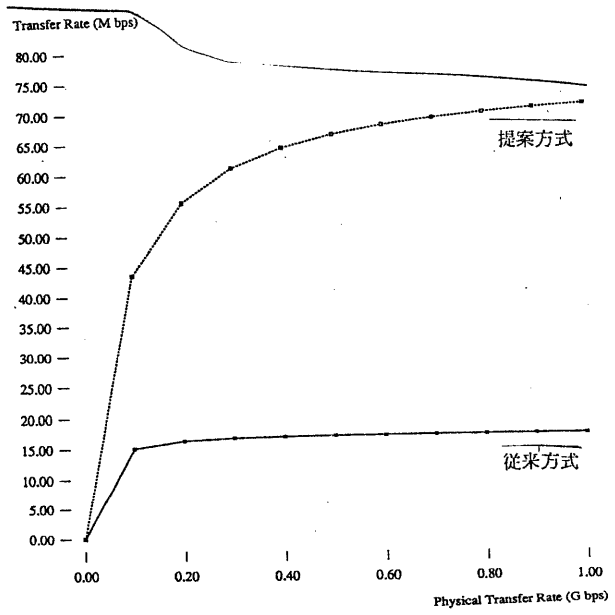

図7　スループットの比較

オペレーティングシステム核内の、ファイルシステムの変更が必要になる。また、通信ハードウェアが仮想記憶機構と関連して動作するために、仮想記憶機構の変更が必要になる。しかしながら、通信機構の大半は、通信ハードウェアによって実現されているので、従来のシステムソフトウェアに対する変更点は大きくないものと期待される。

６．むすび

光ファイバの急速な発展によって可能になった、きわめて広帯域の転送媒体を有効に利用することを目的とした通信システムの実現方式について提案した。このような、１Ｇｂｐｓを越える転送速度の通信媒体の利用に、従来の通信ソフトウェアをそのまま適用すると、大きな無駄が生ずることを示した。この問題を軽減するためには、通信ソフトウェアの内部構造を変更するのみならず、その外部仕様、および、ハードウェアにおよぶ、広い範囲の検討が必要であると考え、その一つの方法を示した。性能の面のみからみれば、比較的大きな改善を見た。

通信システムの性能向上はさしせまった問題であるとは言え、このような大きな変更が導入に見合うものであることを示すためには、具体的な適用分野を定めてされに慎重に検討を進めることが重要である。トランザクション処理の分野において試験的に導入した範囲では有効な結果を得ている。さらに検討を進めたうえで、より総合的な評価を試みたい。

参考文献

1) N.P.Kronengerg, H.M.Levy and W.D. Strecker : VAXclusters: A Closely-Coupled Distributed System, ACM Transaction on Computer System, Vol.4, No.2, PP.130-149 (1989).

2)前川守，所真理雄，清水謙多郎：分散オペレーティングシステム，共立出版（１９９１）．

3)R.Van Renesse, H. Van Stavern and A. S.Tanenbaum:Performance of the World's Fastest Distributed Operating System, Oper. Syst. Rev., Vol.22, No.4, pp.25-34 (1988).