データベース・システム 73－9
オペレーティング・システム 44－9
（１９８９．９．８）

オブジェクト共有型分散オペレーティングシステムの構想

陣崎 明　樋口 昌宏　玉野 肇　加藤 光幾

㈱富士通研究所

分散型共有仮想記憶に基づくオブジェクト共有型分散オペレーティングシステムPathosの構想と実現性について述べる．Pathosは我々の提案した分散型共有仮想記憶システムであるネットワーク仮想記憶(NET-VMS) 方式に基づく分散処理システムPUMA上で動作する．Pathosでは単一階層記憶の概念に基づき，システム内の全ての共有資源を共有仮想記憶空間上のメモリオブジェクトに統合し，一元的に管理する．共有資源に対するアクセス制御はNET-VMS ハードウェアの共有メモリ管理機能を利用して効率的に実現できる．また，分散処理機能の実現例として，メッセージ通信の実現と予測性能について述べる．

A Distributed Operating System based on Object Sharing

Jinzaki Akira　Higuchi Masahiro　Tamano Hajime　Kato Koki

Fujitsu Laboratories Ltd.

In this paper, we describe a conception of distributed operating system "Pathos" based on shared object. Pathos operates on the distributed system PUMA which is designed based on NETworked Virtual Memory System(NET-VMS) we proposed. Pathos integrates the system resources as memory object under the idea of single level storage. On Pathos, the access control for shared resources is implemented efficiently using the NET-VMS functions of shared memory control. We also describe an implementation of message communication and its evaluation.

1. はじめに

分散システムを実現する上で最も重要な機能としてプロセス間通信（Inter-process communication: IPC）がある．IPC機構は大きく分けてネットワークハードウェア，通信制御ソフトウェア（オペレーティングシステム）から構成される．ネットワークハードウェアは光通信技術を始めとするデバイステクノロジの進歩により格段の進歩を見せており，100Mbpsのfddi[1]などが実用化されている．これに対してTCP/IPを代表とする通信制御ソフトウェアは10MbpsのEthernetにおいてさえ性能のボトルネックになっており[2] 100Mbps台の高速ネットワークを有効活用するためにはなんらかの改善が必要である．

通信制御オーバヘッドを軽減する試みは数多く提案されており[3][4][5]，いずれも通信制御の簡略化による通信性能の向上を目指しているが，例えばSprite[3]で700KB/秒(10Mbps Ethernet, 4KB 転送）程度の通信性能しか得られていない．

以上の問題を解決する一つの方法として，メモリイメージでコンピュータをネットワーク結合する方法がある．このアプローチの一つとして，我々は分散型共有仮想記憶システムであるネットワーク仮想記憶(NET-VMS) 方式[6]およびNET-VMSにおける共有メモリ制御方式である宣言的アクセス制御方式[7]を開発した．また試作システム上に分散オペレーティングシステムプロトタイプDAX-Iを開発し，並列マージソートの実験評価により方式の検証を行ってきた[8]．

本論文ではこの分散型共有仮想記憶で動作するオブジェクト共有型分散オペレーティングシステムPathosの構想について述べ，本システムにおけるメッセージ通信機構の実現とその性能予測を行う．

2. 分散型共有メモリ

分散型共有メモリの実現にあたっては，通信遅延の大きいネットワーク環境で共有メモリ制御（memory consistency制御，メモリアクセスに対する排他制御，プロセッサ間のアクセス同期制御）を効率良く実現する点が大きな課題となる．

この問題はネットワークシステムとメモリシステムをハードウェアによって連携させることによって，ある程度解決可能である．我々が開発したネットワーク仮想記憶(NET-VMS) 方式では分散したコンピュータ（Processor Node: PN）の仮想記憶システムをブロードキャストネットワークで結合し，全体として一個の共有仮想記憶を構成する（図１）．NET-VMS 方式では仮想記憶ページ単位の転送や共有メモリ制御のための通信をブロードキャスト（一斉放送）通信を用いて効率的に行うことができる．実験の結果，伝送路速度100Mbpsのネットワークを用いて4Kバイトの仮想記憶ページを転送するのに330 μs，512 バイトの場合50μsという性能を得ている（図２）[9]．

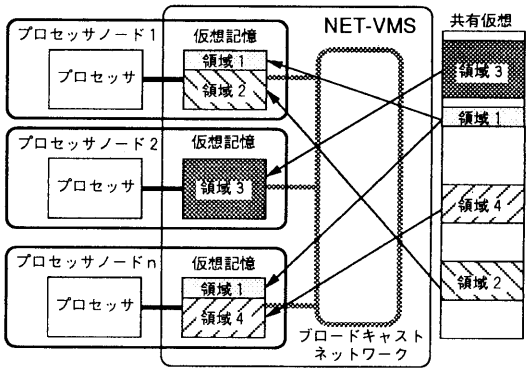


図１　ＮＥＴ－ＶＭＳの構成

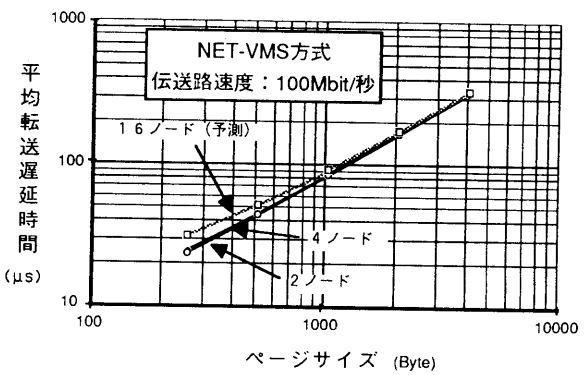


図２　ＮＥＴ－ＶＭＳの通信性能

また共有メモリ制御を行うため，共有仮想記憶ページ単位にメモリタグ（表１）を設ける．宣言的アクセス制御方式では仮想記憶ページに対するアクセス条件を，このメモリタグを設定することにより，あらかじめ仮想記憶システムに宣言しておく．実際にソフトウェアがページをアクセスした時，NET-VMS は宣言された条件と実際の仮想記憶ページの状態を比較検査する．条件と実際の状態が整合する場合はそのままアクセスさせるが，しない場合は整合するように必要な処理を行う．アクセス条件としては次のものがある．（VALID はアサート，VALID はネゲートを表す．）

(1) アクセス不可 (VALID)
(2) 読みだし可 (VALID, COPY, SYNC)
(3) 排他書き込み可 (VALID, LOCK, COPY, SYNC)
(4) 変更待ち (SYNC)

ここで変更待ちとは，該当ページを他のPNがアクセスするまで，自PNのアクセスを抑制するという条件で，二つのプロセスが一個のページを交互にアクセスする機能を提供する．

表１　メモリタグ

VALID	自PN中にデータが存在することを示す
LOCK	データに排他アクセス中であることを示す
COPY	データが複数のPNに存在することを示す
SYNC	他PNがアクセスするまで自プロセッサがアクセスできないことを示す
WAITER	排他アクセス中に他PNからアクセス要求があったことを示す
PRIVATE	他PNと共有しないローカルなページであることを示す

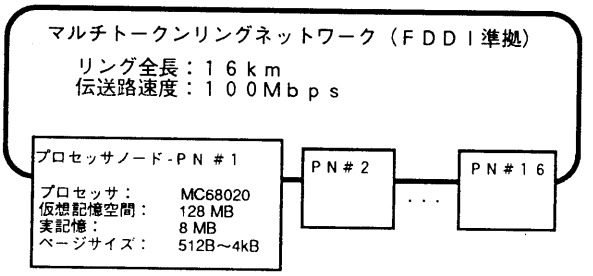


図３　ＰＵＭＡの構成

3. オブジェクト共有方式

NET-VMS 方式のようなハードウェアサポートを用いれば，ネットワーク結合により地理的に分散した大規模な分散システムで非常に大きな共有仮想記憶空間を実現可能である．また単一階層記憶 (single level storage) の概念を用い，システム内の資源を全て共有仮想記憶空間上のオブジェクトとして取り扱うようにすれば，システムに接続された全てのコンピュータがあらゆる資源に対して均一にアクセスできるような単一システムイメージの分散システムが構成できるはずである．

そこでシステム資源をメモリオブジェクトに統合し，メモリオブジェクト管理機構を提供するようなオペレーティングシステムPathosを考える．

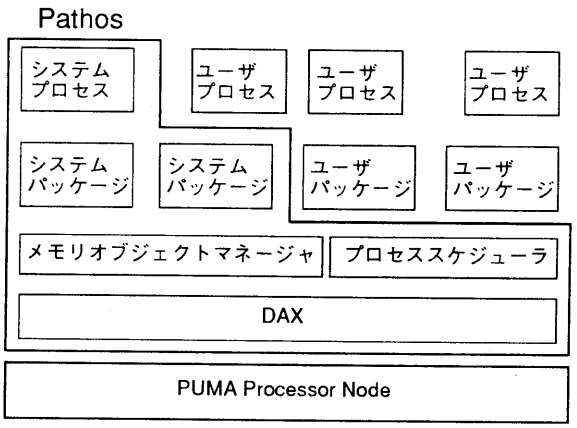


図４　Ｐａｔｈｏｓの構成

3.1 Pathosの構造

Pathosは現在開発中の分散システムPUMAで動作する．図３にPUMAの構成を，図４にPathosの構成を示す．Pathosは分散型共有仮想記憶の制御を行うDAX，オブジェクト管理を行うオブジェクトマネジャ，プロセススケジューラ，システムコールに相当するパッケージ群，デバイスハンドラなどを実現するシステムプロセスで構成する．

3.2 メモリオブジェクト

メモリオブジェクト(memory object: MOBJ) は識別番号 (memory object id: MOID) をもつ共有仮想記憶空間上のページの集まりとして定義する．MOIDは先頭ページの仮想アドレスである．メモリオブジェクトにはresource object (ROBJ)とdata object (DOBJ)の二種類がある．

ROBJはヘッダ部とデータ部からなる．ヘッダ部の形式は固定で，このROBJの内容，すなわちオブジェクトの仕様 (object specification: OSPEC) を記述する情報とオブジェクト管理情報を格納する．OSPEC はPathos固有のObject Specification Language (OSL) で記述する．オブジェクト管理情報はオブジェクトの所有権やアクセス条件などである．データ部はページのヘッダ部以外の部分で，任意のデータを格納する．

DOBJはヘッダ部のないROBJで，MOIDと長さで表される．DOBJのMOID，長さ，OSPEC はそれを参照しているROBJに格納される．

3.3 オブジェクトアクセス条件

アクセス条件はプロセスのオブジェクトに対するアクセスのしかたを規定する．すでに述べたようにPUMAでは仮想記憶ページ単位にアクセス条件の宣言ができる．PathosはROBJヘッダ部のオブジェクト管理情報を参照して，PUMAの仮想記憶システムに適切な設定を行う．

3.4 メモリオブジェクトの管理

MOBJを共有するためには，システム全体でMOIDの一貫性を保つ必要がある（これは共有メモリの割り付けを一元管理することに等しい）．そこで共有仮想記憶空間上に Global Object Table (GOT)を設け，MOBJの管理はGOT を参照して行う．このような集中管理ではGOT に対するアクセスがシステム性能のネックとなることが考えられる．GOT に対するアクセス集中を避けるためにPathosでは次の方法を用いる．

(1) GOT のページサイズを小さくし(PUMA/Pathosでは512 バイト)，ページ単位にアクセス管理を行う．

(2) 各PNがあらかじめダミーMOBJをいくつか獲得しておき，MOBJを生成する時はこのダミーMOBJを使用する．

3.5 メモリオブジェクトの共用

分散システムでは資源を最大限に共用することが重要である．例えばデータベースをアクセスする場合，もしそのデータベースがすでにメモリオブジェクトとしてシステム内のどこかに存在しているならば，そのメモリオブジェクトをアクセスした方が良い．

そこでPathosでは共有仮想記憶空間上にGlobal Object List (GOL)を設け，共用可能なMOBJを登録するようにする．GOL はMOIDを格納したハッシュ表であって，Pathosが提供するハッシュ関数によりOSPEC からハッシュアドレスを計算する．

OSPEC を記述するOSL については今後検討する予定であるが，例えばSQL [10]を包含することにより，データベースに対してSQL 処理を施した結果のオブジェクトを表現できるようにすることを考えている．

3.6 プロセス

プロセスはスケジューリングの対象となるプログラムでprocessObjで表す．processObjにはプログラムの実行環境が記述されている．Pathosでは共有仮想記憶空間上にGlobal Process Pool(GPP)を設け，GPP にprocessObjを登録しておく．各PNのプロセススケジューラはGPP から実行可能なprocessObjを取り出して実行する．

3.7 パッケージ

Pathosにおいて特定のオブジェクトに関する手続きを集めたものをパッケージと呼ぶ．パッケージは，それが扱うオブジェクトに対してのみアクセスできる．プロセスはパッケージに含まれる手続きを手続き呼び出しで実行する．プロセスとパッケージはprocessObjを生成する際，ダイナミックリンクで結合される．パッケージにはPathosが

提供するシステムパッケージとユーザが独自のオブジェクトを定義して使用するユーザパッケージがある．パッケージの例としてメッセージ通信パッケージを4 章に示す．

4. オブジェクト共有によるIPC

Pathos上でMach風のメッセージ通信機構[11]を実現し，性能予測と比較を行う．PathosではMachのポートとメッセージに対応するオブジェクトとしてportObj とmessageObjを使用する．Pathosのメッセージ通信機構はportObj，messageObj を扱う手続きを集めたパッケージとして実現する．プロセス，パッケージ，オブジェクト間の関係を図5に示す．

4.1 メッセージ通信パッケージ

メッセージ通信パッケージ (messagePack)は次の手続きからなる．

(1) portAlloc

portObj を生成し，GOL に登録する．生成したportObj のMOIDを返す．Machのport-allocate, env-set-portにそれぞれ相当する．

(2) messageAlloc

messageObjを生成し，MOIDを返す．

(3) portGet

OSPEC で指定したportObj のMOIDを返す．Machのenv-get-portに相当する．

(4) messageSend

指定したportObj に指定したmessageObjをキューイングする．Machのmsg-sendに相当する．

(5) messageReceive

指定したportObj にキューイングされたmessageObjを一個取り出しMOIDを返す．Machのmsg-receive に相当する．

4.2 メッセージ送受信

(1) 準備

受信側プロセスはportAlloc によりportObj を生成し，messageReceiveを行う．この時点でportObj に新しいメッセージが到着していなければ，受信側プロセスは変更待ちアクセス条件で停止する．

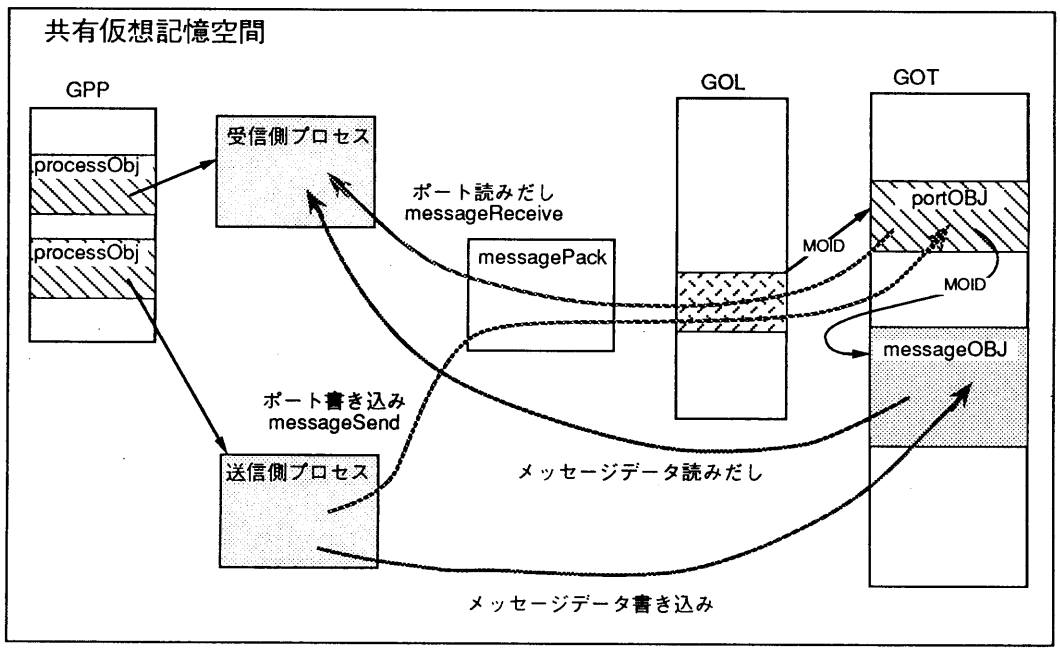


図5　Ｐａｔｈｏｓのオブジェクト（メッセージ通信の例）

送信側プロセスはまず受信側プロセスのOSPECを指定してportGet することにより，受信側プロセスのportObj のMOIDを得る．

(2) メッセージ送信

送信側プロセスは，messageAllocで生成し，データを設定したメッセージを受信側プロセスのportObj にmessageSend する．この時点でportObjは更新され，PUMAにより受信側プロセスのあるPNに転送され，変更待ちで停止していた受信側プロセスが実行を再開する．

(3) メッセージ受信

実行を再開した受信側プロセスはmessageReceiveが返したmessageObjにアクセスする．この時点でmessageObjの転送が行われる．

以上のシーケンスを図6に示す．portAlloc，portGetは最初に一回行うだけである．メッセージ送受信でネットワークを介して転送されるのはportObj (512B ページ) が2，messageObj (メッセージ長に依存) が1 である．

4.3 リモートプロシジャコール

リモートプロシジャコール (RPC)もメッセージ送受信とほぼ同じように実現する．Machではmsg-rpc を用いるが，Pathosでは応答メッセージに対するアクセス同期によってプロセスの同期をとるのでmessageSend を用いる．

(1) 準備

サーバ側はメッセージ受信の場合と同じようにしてportObj を生成し，メッセージ到着待ちとなる．

クライアント側はportGet によりサーバのportObj を得る．

(2) クライアント側のRPC 処理

クライアント側はRPC 要求用のmessageObj (re

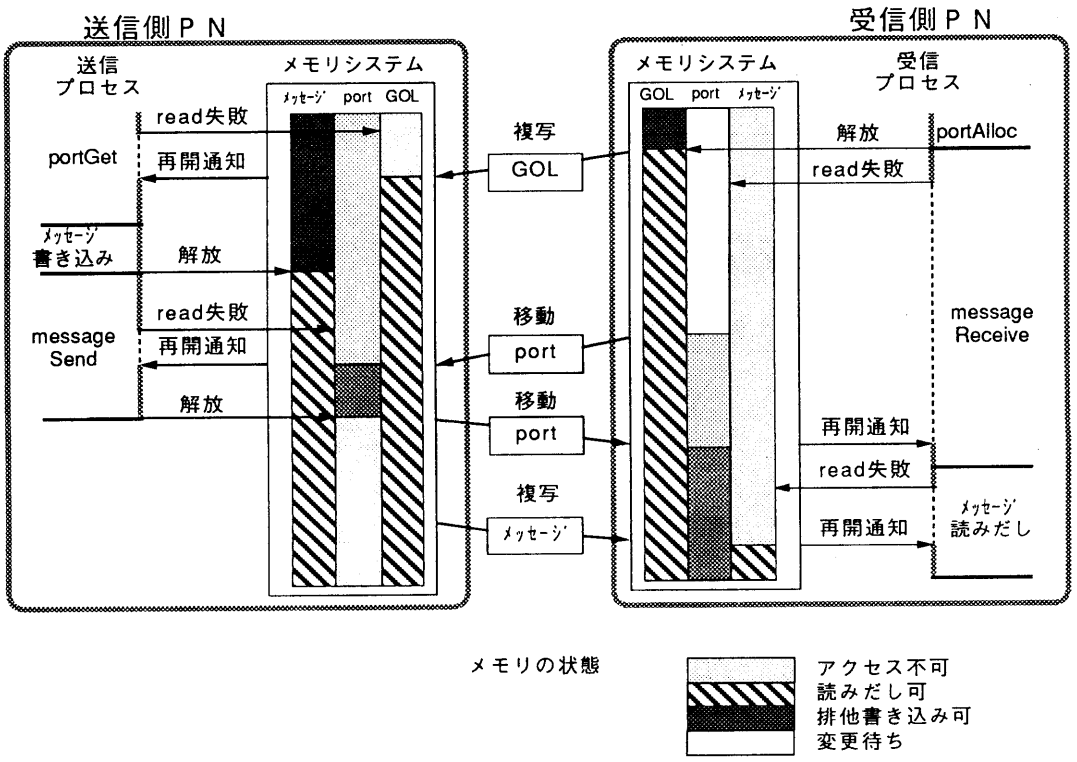


図6 メッセージ通信シーケンス

quest)とサーバが応答するためのmessagObj (response)を生成し，responseのMOIDをrequest に書き込んでmessageSend し，responseをリードアクセスする．この時点でresponseはデータが設定されていないので，クライアントは変更待ちアクセス条件で停止する．

(3) サーバ側のRPC 処理

サーバ側はrequest を受信すると処理を行い，responseに応答データを設定し，responseを解放する．この結果PUMAはresponseをクライアントのあるPNに転送する．サーバ側はresponseをmessageSend しなくてよい．

(4) クライアント側の応答受信

クライアント側は転送されたresponseをアクセスする．PUMA/Pathos ではプロセス間の同期を同一のオブジェクトに対するアクセス同期の形で実現できるので，RPC のように応答メッセージオブジェクトを受信側が知っている場合は特にメッセージ通信をしなくてよい．この場合のネットワーク通信回数はportObj (512B ページ) が2 ，messageObj (メッセージ長に依存) が2 である．

4.4 性能予測

4.2，4.3で示したPUMA/Pathos のメッセージ通信時間を，DAX-I での実験結果に基づいて予測した結果および比較対象としてVAX11/780 上で我々が実測したMachのメッセージ通信時間を表2に示す．Machの性能は単一プロセッサ上で動作しているプロセス間のメッセージ通信性能である．MachではRPC の動作 (図7) に示すようにユーザ空間とシステム空間の間でメッセージの複写を行っているのに対して，Pathosではユーザ空間の間で直接messageObjを移動する．この結果，PUMA/Pathos ではネットワークを介して，Machが単一プロセッサ上でオンメモリで行うよりも高速なメッセージ通信を期待できる．

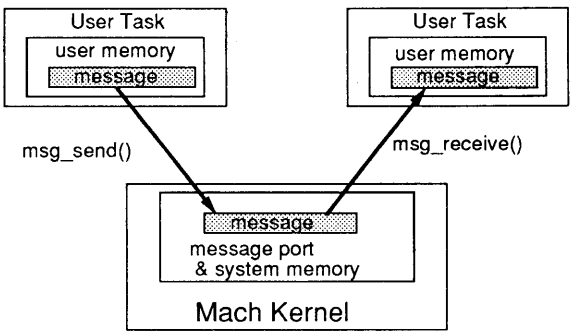
5. おわりに

ネットワーク仮想記憶方式および宣言的アクセス制御方式に基づくオブジェクト共有型分散オペ

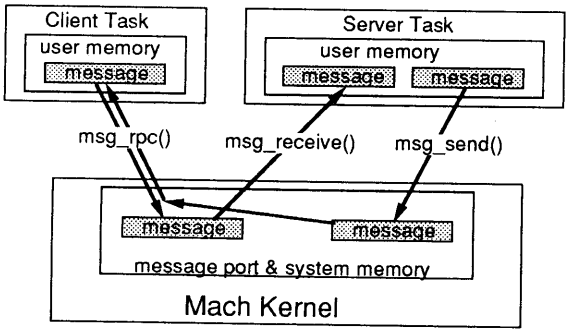
表2 メッセージ通信性能の比較

メッセージ長		512B	1kB	4kB
メッセージ送信	VAX/Mach	850	1080	2440
	PUMA/Pathos	350	350	350
メッセージ受信	VAX/Mach	900	1140	3000
	PUMA/Pathos	600	640	880
RPC	VAX/Mach	3990	4610	8360
	PUMA/Pathos	1200	1280	1760

(単位：μs)



(a) メッセージ送受信



(b) リモートプロシジャコール

図7 Machのメッセージ通信

レーティングシステムPathosの構想を述べた．Pathosは共有仮想記憶をもつマルチプロセッサシステムで動作し，メモリオブジェクトとして統合されたオブジェクト共有環境を提供する．Pathosにおけるメッセージ通信機構を検討，評価した結果，100Mbps ネットワークで結合した分散システムPUMAにおいて，Machの単一プロセッサ上のメッセージ通信よりも高速な性能が得られる見通しを得た．

今後，ネットワーク結合による分散システムはIPC の高速化を求めてメモリ結合の方向へ向かっていくと考えられる．そのようなシステムを構築する上で，分散した巨大な共有メモリ空間をいかに管理するかは，単に性能の問題だけにとどまらず，分散処理ソフトウェアを効率良く開発していく上でも重要なテーマと考えられる．今後はPathosの開発を進めると同時に，OSL，分散処理プログラミング言語，開発ツールについての検討も行いたいと考えている．

［参考文献］

(1) Floyd, E. R.: FDDI-a Tutorial, IEEE Communication, vol. 21, no. 2, pp. 23-36 (1986-05).

(2) Thon, B.: The Protocol Engine Project, UNIX REVIEW, pp. 70-77 (1987-09).

(3) Ousterhout, J., Cherenson, A., Douglis, F., Nelson, M. and Welch, B.: The Sprite Network Operating System, Computer, Vol. 21, No. 2, pp. 23-36 (1988-02).

(4) Chesson, G., Green, L.: XTP-Protocol Engine VLSI for Real-Time LANs, proceeding of the 6th European Fibre Optic Communi. and Local Area Networks Exposition, pp. 435-438 (1988-06).

(5) Cheriton, D. R., Williamson, C. L.: VMTP as the Transport Layer for High-Performance Distributed Systems, IEEE Communication, vol. 27, no. 6, pp. 37-44 (1989-06).

(6) 陣崎，八星：ブロードキャストネットワークによる分散型単一階層仮想記憶システム，信学技報，CPSY86-20 (1986-07).

(7) 陣崎，八星，樋口：ネットワーク仮想記憶システムNET-VMS (2)プロセス間通信方式，昭61後期情処全大，3T-8 (1986-10).

(8) 樋口，陣崎：ネットワーク仮想記憶システムにおける宣言的アクセス制御方式，並列処理シンポジウム JSPP'89, pp. 131-138 (1989-02).

(9) 陣崎，樋口，栗田：100Mb/秒トークンリングネットワークによる高速コンピュータ間通信システムの試作，信学技報，IN88-55 (1988-07)

(10) 芝野：データベース言語SQL，情報処理，vol. 29, no. 3, pp. 208-214 (1988-03).

(11) Young, M., Tevanian, A., Rashid, R., Golub, D., Eppinger, J., Chew, J., Bolosky, W., Black, D. and Baron, R.: The Duality of Memory and Communication in the Implementation of a Multiprocessor Operating System, Proc. of 11th Symp. on Operating Systems Principles, (1987).