Vol. 36 No. 8　　情　報　処　理　　Aug. 1995

解　説

日本におけるオペレーティングシステム研究の動向

3.3 分散型リアルタイムOS：DIROS†

箱守　聰††　谷口秀夫†††

1. はじめに

ハードウェア技術の進歩にともない，マイクロプロセッサを用いた計算機は，メインフレームに匹敵する価格性能比を実現している．また，計算機間を高速な通信路で結ぶことも可能になっている．分散型リアルタイムオペレーティングシステム（DIstributed Real-time Operating System：DIROS）は，複数のマイクロプロセッサを搭載した計算機を高速な通信路で結んだ分散環境において，トランザクション処理を可能にするオペレーティングシステム（OS）である[1]．応用プログラム（AP）としては，端末制御処理や中継処理を想定したものである．ここでは，DIROSの設計方針，ハードウェア環境，および特徴について述べる．

2. 方　針

DIROSの設計は，主に以下の3つの方針で進めた．

（1） 分散環境指向のAPインタフェース

分散環境でない場合，1つのプロセッサで行われるOSの処理は，同じ内容であれば処理時間もほぼ同じである．そのため，APはOSに依頼した処理の時間を予測することができる．一方，複数のプロセッサで行われる分散環境でのOSの処理は，これと異なる．OSの処理時間は，プロセッサ間の通信時間も含む．通信時間は，トラフィック量に大きく左右される．したがって，同じ内容であってもOSの処理時間が異なることがある．このような分散環境においても，APが処理を進めやすいことが大切である．そこで，OSの処理時間の予測が難しくても，APの処理をうまく実行できるAPインタフェースを提供する．

（2） ネットワーク透過性

分散環境では，セマフォ，メールボックス，およびファイルなどの資源が分散している．APと資源の距離により資源操作のインタフェースが異なるとAPの可搬性が悪くなり，APを各計算機の上で自由に実行することができない．そこで，APに資源の分散を意識させない資源操作のインタフェースを提供する．これは，資源に関するネットワーク透過性と呼ばれている．さらに，物理的な資源配置と論理的なものを分離する．論理的な資源配置をAPに提供することにより，APの動作環境条件を大幅に緩められる．

（3） ヘテロ環境の支援

計算機の特化が進み，用途別に最適な計算機が登場している．したがって，これらの計算機を結んで適切な機能分散を図り，より効率的なシステム構築を可能にする必要がある．そこで，いろいろな計算機と通信できる機能を提供する．具体的には，様々な通信手順を実現できるプログラム構造とする．また，既存の有効なAPの利用を可能にし，かつ開発のプログラム量の削減のため，既存のOSインタフェースを提供する．

OSの処理時間によらないAPインタフェースの例として，SunのAsynchronous I/O[8]がある．これはローカルな資源に対して処理の依頼と結果の取得を非同期に行うことができるが，分散された資源には対応していない．また，ファイルI/Oについて実現しており，メッセージ通信等のプロセス間通信では実現されていない．

一方，ネットワーク透過性を実現しているシステムとして，Mach[9]，CHORUS[10]，Sprite[11]が

† Distributed Real-time OS: DIROS by Satoshi HAKOMORI (Research and Development Headquarters, NTT DATA) and Hideo TANIGUCHI (Department of Computer Science and Communication Engineering, Kyushu University).

†† NTTデータ通信（株）技術開発本部
††† 九州大学工学部情報工学科

ある．MachおよびCHORUSは，OSが扱う資源にネットワーク内で一意な識別子を付与することによりネットワーク透過性を実現している．Spriteは，ネットワークで一意なファイルシステムを構成しており，ファイルに対するネットワーク透過性を提供する．

ヘテロ環境に対応するシステムには，Sunら[12]，藤長ら[13]の研究がある．Sunらは異なるOS間で共通なインタフェースを規定している．この場合，実現可能な機能が制限される．また，加藤らはRemote Procedure Call (RPC) において一方のデータ形式を他方に合わせることにより異なるシステム間の接続を実現する．この方式では，同種のシステム間を接続する場合に比べてデータ形式の変換のために性能が低下する問題がある．

3. ハードウェア環境

DIROSが想定する分散環境の例を**図-1**に示し，以下に説明する．

プロセッサは，内部バスにより，ローカルメモリ，および周辺装置や通信回線との入出力を制御するI/Oコントローラと接続されている．各プロセッサは共通バスで接続され，共有メモリによりプロセッサ間の通信が可能になっている．DIROSが走行する計算機間は，SCSIまたはLANで結ばれている．LANには，UNIXが走行する計算機も接続されている．

4. 特　　徴

ここでは，DIROSの基本的なプログラム構造と，特徴的な機能について述べる．

4.1 プログラム構造

プログラム構造の特徴は，「処理モジュール識別子」と「リクエスト制御」に代表される[2]．

OSのプログラムを処理内容に基づき分割した．分割した各々のプログラムを区別するために「処理モジュール識別子」を導入した．処理モジュール識別子の内容を**図-2**に示す．その内容は，当該プログラムが走行する計算機の番号（処理装置番号）とプロセッサの番号（プロセッサ番号），およびプログラムの識別番号（モジュール番号）からなる．

「リクエスト制御」とは，分割したプログラム間の制御の移行を定めたプログラム間インタフェース規定である．リクエスト制御の処理流れを**図-3**に示す．処理を依頼する処理モジュールと依頼を受ける処理モジュールの間に，2種類のプログラム部がある．ユニットと呼ぶプログラム部は，処理モジュールとのインタフェースを司る部分である．依頼側の処理モジュールと接したユニットは，処理モジュールとの間に3つのインタフェース（ジョブの依頼，結果取得，および取消）を有している．このユニットは，4つのプログラム部（レスポンス，リザルト，クリーン，およびキャンセルと呼ぶ）によりインタフェースを実現している．また，被依頼側の処理モジュールと接

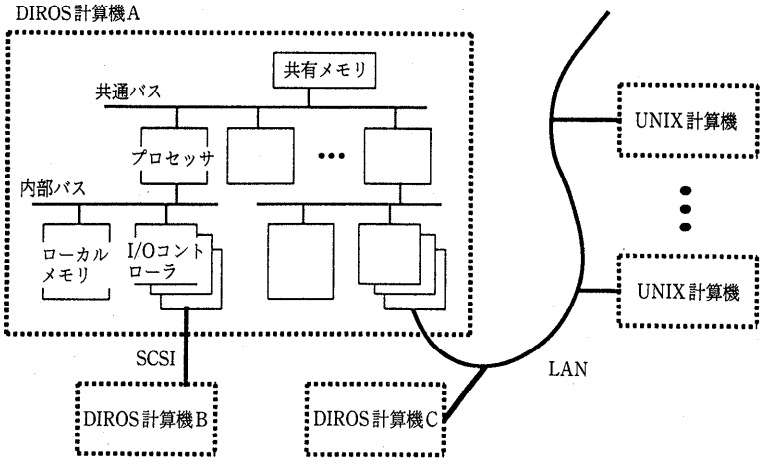


図-1 ハードウェア環境

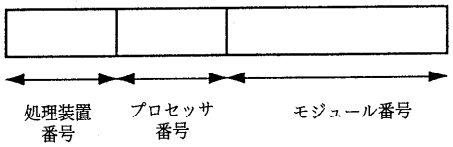


図-2 処理モジュール識別子

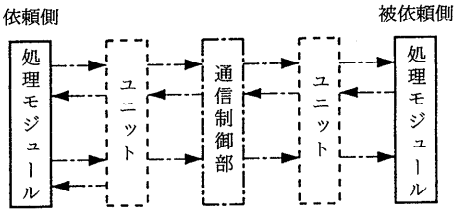


図-3 リクエスト制御の処理流れ

したユニットは，処理モジュールとの間に1つのインタフェース（結果応答）を有している．このユニットは，2つのプログラム部（アクションとキャンセルアクションと呼ぶ）によりインタフェースを実現している．合計4つのインタフェースは，リクエストブロックと呼ばれる情報表の授受で実現される．2つのユニット間にある通信制御部は，通信を制御するプログラム部である．共有メモリ，SCSI，およびLANの各通信手順を実現し，同じインタフェースでユニットに提供している．

ここに示した処理モジュール識別子やリクエスト制御が，以降で述べる機能の実現を容易にしている．

4.2 非完了システムコール機能

分散環境指向のAPインタフェースとして，非完了システムコール機能を実現した．非完了システムコールは，OSがAPに提供する機能を3つのインタフェースで提供するものである．1つ目は，処理受付であり，処理の実行を待たずして制御をAPに返す．このときの返却値をリクエスト識別子と呼び，受け付けた処理を識別するものである．2つ目は，結果返却であり，リクエスト識別子に対応する処理の結果を返却する．基本的な処理は，この2つのシステムコールで実行できる．最後の1つは，実行取消であり，リクエスト識別子に対応する処理を取り消す．

1つの処理に関するOSインタフェースを2つのシステムコール（処理受付と結果返却）に分割しているため，APは複数の処理を同時にOSへ依頼でき，かつ依頼時にAPは他の処理を実行できる．したがって，OSの処理時間が予測できない分散環境においても，APは処理を中断なく進めることができる．

さらに，この機能は，次の2つのことを可能にし，プロセス数の増加を防いでOSオーバヘッドを抑える．1つは，1つのプロセスが複数種のサービスを実行できることである．もう1つは，1つのプログラムが入力の要求をOSに依頼したままで他の処理を実行できることである．

4.3 ネットワーク透過性のための機能

ネットワーク透過な資源操作を実現した．資源は，処理モジュール識別子とその資源の通番によりネットワーク一意に指定できる．特に，ファイルやプリンタなどの資源は，文字列での指定を可能にし，資源配置を論理化した．また，資源の操作は，システムコールや内部コールの内容をリクエスト制御により転送することで，ネットワークを意識しない操作インタフェースを実現した．ここで，内部コールとは，処理モジュール識別子で指定できるプログラム部が提供している機能の呼出形式である．

したがって，他の計算機にあるファイル，プリンタなどの装置，プロセス，およびメールボックスなどの資源について，自分の計算機にある資源と同様な形式で操作できる．具体的には，リモートファイルアクセス機能，リモートデバイスアクセス機能，リモートプロセス生成機能，およびリモートプロセス間通信機能である[3),4)]．

4.4 ヘテロ環境を支援する機能

ヘテロ環境を支援する機能として，大きく2つの機能を実現した．1つは，UNIXのOSインタフェースの実現である．もう1つは，UNIX計算機との間のファイル共有機能の実現である[5)]．

UNIXのOSインタフェースは，次の3つの要求を満足して実現した．1つ目は，APがDIROSとUNIXの両システムコールを同時に利用できることである．これにより，たとえば，両OSの特徴を両取りしたAPの作成が可能になる．2つ目は，APが1つのファイルを2つのOSインタフェースで操作できることである．これにより，たとえば，既存のUNIXのAPとDIROSのAPによる高度なサービス実現が可能になる．3つ目は，DIROS計算機と同種のプロ

セッサを持つUNIX計算機で走行するプログラムが，そのまま実行できることである．これにより，より多くの既存のAPを利用できる．これらの要求を満足するため，OSカーネル内にUNIX機能を実現した．このような環境においては，異なる性質のプロセスが同時に走行する．そのため，プロセスの属性としてDIROS属性またはUNIX属性を与え，走行を制御する機能を実現した．UNIX属性のプロセスは，資源操作の権限や走行優先度を低位に制限した．また，UNIX属性のプロセスは，特定された利用者識別子を持つもののみがDIROSのシステムコールを利用できるとした．

UNIX計算機とのファイル共有は，UNIX計算機間の分散ファイルシステム[6]とNFS[7]を検討した．2種類の分散ファイルシステムを検討することにより，ファイルを共有しようとする計算機に合わせて複数の分散ファイルシステムを共存制御できることを確認した．

5. おわりに

DIROSの設計方針，ハードウェア環境，および特徴について述べた．以上の記述は，第一段階の開発に関するものである．第一段階は，分散処理OSの基本的機能を実現したものである．上位にトランザクションを管理するプログラムやサービスプログラムを載せて走行させ，満足な性能を得ることを確認できた．

現在，第二段階の開発が進んでいる．通信機能の充実や保守性の向上などを中心に行っている．

ハードウェア技術の進歩により，制御可能なプロセッサの数が増加している．しかし，制御方式の検討が遅れている．特殊なサービスだけではなく，より一般的なサービスにおいても多数のプロセッサを利用できるようにする必要がある．この際には，高性能であるとともに信頼性や保守性が高いことが必要であろう．

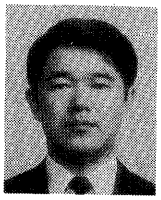
参考文献

1) 谷口, 遠城, 井村, 境：分散型リアルタイムオペレーティングシステム：DIROS, 情報処理学会研究報告, 89-OS-45-9 (1989).
2) 谷口：OS機能の分散を可能にするOS構成法, 信学論, Vol. J 72-D-I, No. 3, pp. 168-174 (1989).
3) 井村, 谷口, 遠城：マルチプロセッサにおける分散ファイル管理方式, 情報処理学会研究報告, 89-OS-43-6 (1989).
4) 谷口, 境：プロセス生成の高速化と並列化に関する検討, 信学技報, CPSY 89-28, pp. 81-86 (1989).
5) 谷口, 遠城, 箱守：異種OSを結ぶ分散ファイルシステム, 情報処理学会研究報告, 89-OS-43-7 (1989).
6) 谷口, 鈴木, 瀬々：ファイル管理機能のネットワーク化による分散処理OSの構成法, 情報処理学会論文誌, Vol. 27, No. 1, pp. 56-63 (1986).
7) Walsh, D. et al.: Overview of the Sun Network File System, Proc. of USENIX Winter Conf., pp. 117-124 (1985).
8) Yoo, H. et al.: UNIX Kernel Support for OLTP Performance, Proc. of USENIX Winter Conf., pp. 241-247 (1993).
9) Accetta, H. et al.: Mach: A New Kernel Foundation For UNIX Development, Proc. of USENIX Summer Conf., pp. 93-112 (1986).
10) Rozier, M. et al.: CHORUS Distributed Operating Systems, Computing Systems, Vol. 1, No. 4, pp. 305-370 (1988).
11) Ousterhout, J. et al.: The Sprite Network Operating System, IEEE Computer, Vol. 21, No. 2, pp. 23-36 (1988).
12) Sun, Z. et al.: Developing a Heterogeneous Distributed Computing System, ACM SIGOPS, Vol. 22, No. 2, pp. 24-31.
13) 藤長, 加藤, 鈴木, 浦野：RPCに基づく分散処理システムの相互接続に関する検討, 1990年代の分散処理シンポジウム論文集, pp. 21-30 (1990).

(平成6年3月16日受付)

箱守 聰（正会員）

1986年名古屋大学工学部電気工学科卒業．1988年同大学院工学研究科修士課程修了．同年NTTデータ通信(株)入社．現在技術開発本部勤務．分散処理に興味を持つ．ソフトウェア科学会，ACM，IEEE各会員．



谷口 秀夫（正会員）

1978年九州大学工学部電子工学科卒業．1980年同大学院工学研究科修士課程修了．同年日本電信電話公社電気通信研究所入所．1988年NTTデータ通信(株)開発本部移籍．1993年九州大学工学部助教授．工学博士．オペレーティングシステム，分散処理に興味をもつ．著書「オペレーティングシステム」(昭晃堂)．電子情報通信学会，ACM各会員．