オペレーティング・システム 43 － 6
（１９８９．６．９）

マルチプロセッサにおける分散ファイル管理方式

井村　佳弘　　谷口　秀夫　　遠城　秀和

ＮＴＴデータ通信（株）

マルチプロセッサ用オペレーティングシステムであるＤＩＲＯＳ（ＤＩｓｔｒｉｂｕｔｅｄ Ｒｅａｌ－ｔｉｍｅ Ｏｐｅｒａｔｉｎｇ Ｓｙｓｔｅｍ）の分散ファイル管理方式について述べる。ＤＩＲＯＳ核の内部は機能単位でモジュール化され、各モジュール間のインタフェースにリクエスト制御と呼ぶ方式を用いている。これを利用してリモートファイルアクセス機能とリモートデバイスアクセス機能を実現し、応用プログラムのさまざまな分散形態を可能にする分散ファイルシスムを構築できることを示す。

THE DISTRIBUTED FILE MANAGEMENT METHOD ON MULTI-PROCESSORS

Yoshihiro Imura Hideo Taniguchi Hidekazu Enjo

NTT Data Communications Systems Corporation

NTT Data Yokohama Nishi Bldg.,2-11-6 Kita Saiwai,
Nishi-ku,Yokohama-shi,Kanagawa 220,Japan

The distributed file management method on multi-processors' operating system DIROS (DIstributed Real-time Operating System) is described. The kernel of DIROS is devided into some facility modules, which are connected with "request control". Remote file access and remote device access are realized by using of request control. These facilities can construct distributed file system which has various form of distribution of application programs.

1　はじめに

近年の急速なＶＬＳＩ技術の進歩を背景として、複数のプロセッサを用いた価格性能比の優れた計算機や、複数の計算機をネットワーク化した分散処理システムの開発が盛んに進められている。このような分散環境を制御する分散処理ＯＳには、各プロセッサ間での資源の共用を可能にする分散ファイル管理が必要である。このような分散ファイル管理として、これまでにＲＦＳ[1]，ＮＦＳ[2]、ＤＰＥ[3]等が発表されている。しかし、これらの分散ファイル管理は主にファイルアクセス処理の応用プログラム（以下、ＡＰと略す）機能の分散化をねらったものであり、ＯＳ機能の分散化を行なうことはできない。そのため、サービス提供時にはその性能やメモリ効率がしばしば問題になる。

トランザクション処理用マルチプロセッサ（以降、ＣＳ：Control Stationと略す）のＯＳとして開発した、ＤＩＲＯＳ[4]（Distributed Real-time Operating System）は、OS機能を分割し、そのモジュール間のインタフェースとしてリクエスト制御と呼ぶ方式を用いることにより、OS機能の分散を可能にしている。

本稿では、OS機能の分散を利用して実現したリモートファイルアクセス機能とリモートデバイスアクセス機能を持つファイル管理について、その機能と処理方式を報告する。

2　ハードウエア構成

ＣＳのハードウエア構成例を図１に示し、その特徴を以下に述べる。

（１）複数のマイクロプロセッサをバスで結合している。

（２）共有メモリ（以降ＣＭと略す）と割込みを利用してプロセッサ間の通信を行なう。

（３）各プロセッサ（以降ＰＵと略す）の配下には、用途に合わせてディスクや通信回線

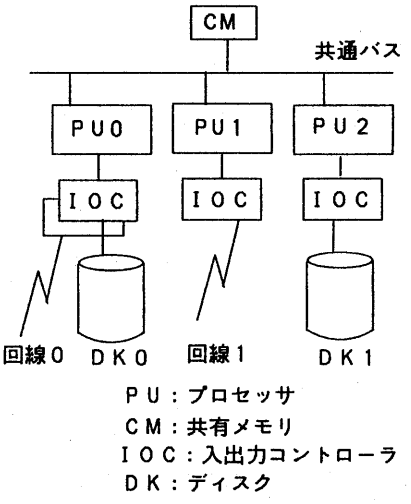


図１　ハードウエア構成例

などの周辺装置を接続する。従って、ディスクを持つＰＵと、持たないＰＵが混在する。

3　ＤＩＲＯＳ構造の特徴

3．1　構造上の特徴

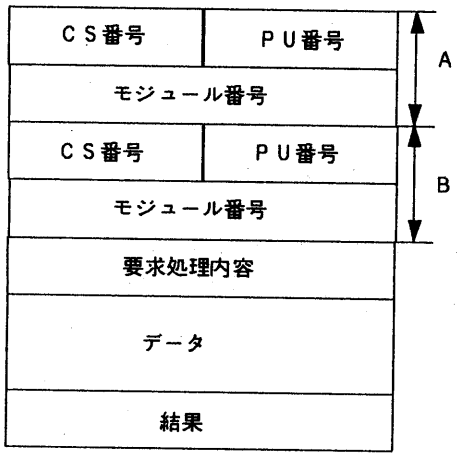
ＤＩＲＯＳのプログラム構造上の特徴を以下に述べる。

（１）各PU上に同様なＯＳを分散する。

（２）ＯＳ核の内部を機能単位でモジュール化しており、ＤＩＲＯＳ核の生成時に機能の組込みの有無を指定することができる。

（３）ＯＳ核の各モジュール間の呼び出し（以降、内部コールと略す）を「リクエスト制御」と呼ぶインタフェースで統一している。ここで、リクエスト制御とは、各モジュール間の処理の依頼、結果応答、結果取得、取消に関する情報の受け渡しを、「リクエストブロック」と呼ぶ制御ブロックを介して行なう制御方式である。リクエストブロックの構成を図２に示す。

（４）ＯＳが提供するＡＰインタフェース（以下、システムコールと呼ぶ）として、「非完了システムコール機能」を持つ。非完



A：処理依頼元情報
B：処理依頼先情報

図2　リクエストブロックの構成

了システムコール機能とは、ＯＳへの処理依頼、結果取得、処理取消を各々別システムコールとして提供する機能である。非完了システムコール機能を用いることにより、１つのＡＰプロセスからＯＳに対して複数の処理を依頼できるため、少ないプロセス数で多くの処理を行なうことが可能となり、プロセス切り替えに伴うOSの負担を少なくすることができる。

３．２　ファイル管理の特徴

分散ファイルシステムの構築に関係するＤＩＲＯＳファイル管理の特徴を以下に述べる。

（１）ディスク上のファイルを通常ファイル、入出力装置や通信回線などの周辺装置を特殊ファイルと呼び、ともに木構造で管理する。

（２）ファイルアクセスについてはキャッシュバッファ機能を持つ。APに対してキャシュバッファを経由するアクセスと経由しないアクセスの両方を提供する。

（３）通常ファイルや特殊ファイルに対する排他制御機能を持つ。

４　要求条件と実現方式

４．１　要求条件

どのPU上で走行するAPプロセスからも、すべてのPUにある資源が同じように見える分散ファイルシステムを構築するためには、以下の条件が要求される。

［条件１］リモートへのアクセスとローカルに対するアクセスを同じ形式でＡＰに提供すること。

［条件２］ファイル管理を持たず、APプロセスが走行しないＰＵにある資源に対しても、他PU上のAPプロセスからアクセスできること。

４．２　実現方式

4.1節で述べた要求条件を満たす分散ファイルシステム構築のため、次の機能を実現し、全てのPUの持つ通常ファイルや特殊ファイルを木構造で一元管理できるグローバルトリーを構成する。

（１）ファイルシステムを持つＰＵの通常ファイルや特殊ファイルに対するアクセスは、ファイル管理間のリクエスト制御による「リモートファイルアクセス」で実現する。

（２）ファイル管理を持たないＰＵの周辺装置に対するアクセスは、ファイル管理とドライバ間のリクエスト制御による「リモートデバイスアクセス」で実現する。

リモートファイルアクセスとリモートデバイスアクセスの具体的な処理方式は５章で述べる。

図１のハードウエア構成に対し、グローバルトリーによって分散ファイルシステムを構成した場合を図３に示す。例えば、ＰＵ０上のＡＰから，"／ＰＵ２／ｆ２" とファイルパス名を指定することにより、ＰＵ２上のファイル "／ｆ２" をアクセスできる（リモートファイルアクセス）。また、ＰＵ０上の

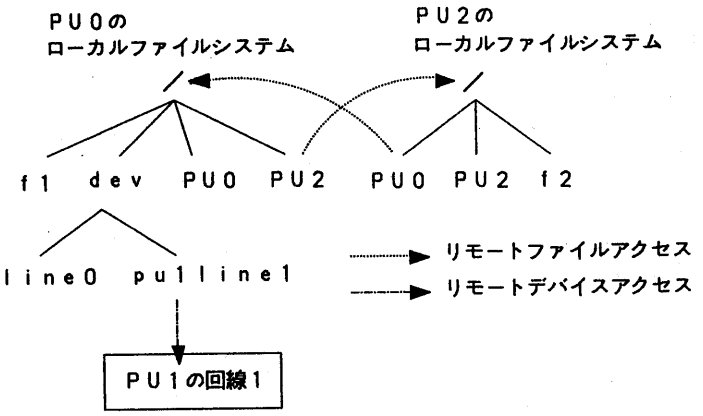


図3 グローバルトリー構成の例

APから，"／dev／pu1line1"とパス名を指定することにより、PU1の回線1をアクセスできる（リモートデバイスアクセス）。

5 処理方式

5．1 リモートファイルアクセス

リモートファイルアクセスの処理方式について、ローカル／リモートの識別方式、処理の流れを以下に述べる。

5．1．1 ローカル／リモート識別方式

以下の方法により、アクセス先がローカルかリモートかを識別する。

（1）ネットワークディレクトリは、ファイルシステムが存在するCSやPUの番号に関する情報を持つ。

（2）APから指定されたパス名の中にネットワークディレクトリが含まれ、かつ、そのネットワークディレクトリのCS番号、PU番号が自PUのそれと異なる時、リモート処理と判断する。

5．1．2 処理の流れ

リモートファイルアクセス処理の例として、PU0上のAPからPU2の通常ファイルへのリモートファイルアクセス処理の流れを図4に示し、以下に説明する。

①PU0上のファイル管理は、APから指定されたファイルパス名中のネットワークディレクトリの情報から、アクセス先がローカルかリモートかを識別する。

②PU0上のファイル管理は、処理依頼先情報としてPU2の代行プロセスを設定したリクエストブロックを作成し、その転送をリクエスト制御に依頼する。

③リクエスト制御からリクエストブロックを受取ったPU2上の代行プロセスは、その内容に基づき、依頼元のAPに代わってPU2のファイル管理に対しファイルアクセスの実行を依頼する。この時、依頼は非完了システムコール機能を利用する。

④PU2上の代行プロセスは、ファイル管理からアクセス結果を受取り、これをリクエストブロックに格納し、リクエストブロックのPU0への返送をリクエスト制御に依頼する。

⑤リクエスト制御からリクエストブロックをうけとったPU0のファイル管理は、リクエストブロック中のアクセス結果をAPに返却する。

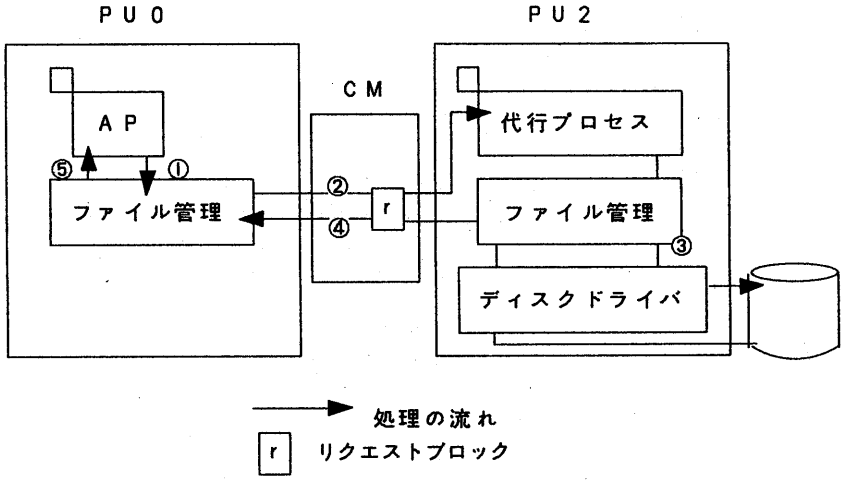


図4　リモートファイルアクセス処理の流れ

5．2　リモートデバイスアクセス

リモートデバイスアクセスの処理方式として、ローカル／リモートの識別方式、処理の流れを以下に述べる。

5．2．1　ローカル／リモート識別方式

以下の方法により、アクセス先がローカルかリモートかを識別する。

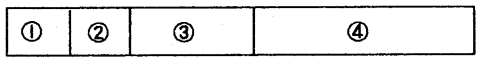
（1）特殊ファイルは、ドライバが存在するCSやPU、およびドライバ固有の情報を「ドライバ識別子」として持つ。ドライバ識別子の内容を図5に示す。

（2）APから指定されたファイルパス名の特殊ファイル中のドライバ識別子を参照し、そのCS番号、PU番号でローカル／リモートを識別する。

5．2．2　処理の流れ

リモートデバイスアクセスの例としてPU0上のAPからPU1の回線装置へのリモートデバイスアクセス処理の流れを図6に示し、以下に説明する。

①PU0上のファイル管理は、APから指定された特殊ファイルのドライバ識別子の情報からアクセス先がローカルかリモートかを識別する。



①　CS番号

②　PU番号

③　ドライバ番号

④　ドライバ固有情報

図5　ドライバ識別子の内容

②PU0上のファイル管理は、処理依頼先情報としてPU1の回線ドライバを設定したリクエストブロックを作成し、その転送をリクエスト制御に依頼する。

③リクエスト制御からリクエストブロックを受取ったPU1上の回線ドライバは、その内容に基づき、回線装置に対してでアクセス処理を実行する。

④PU1上の回線ドライバは、アクセス処理の結果をリクエストブロックに格納し、リクエストブロックのPU0への返送をリクエス

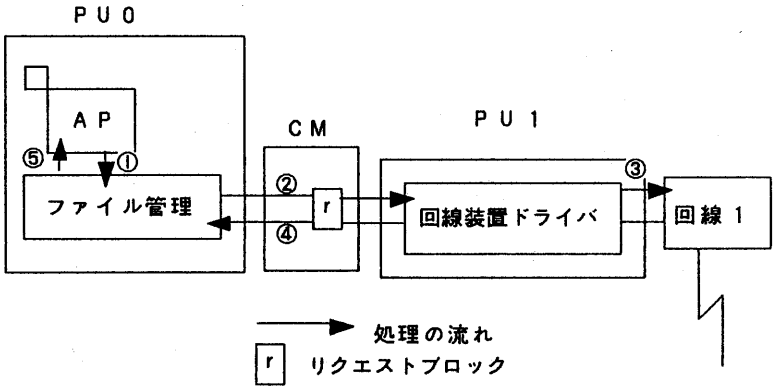


図6　リモートデバイスアクセス処理の流れ

ト制御に依頼する。
⑤リクエスト制御からリクエストブロックを受取ったPU0のファイル管理は、リクエストブロック中のアクセス結果をAPに返却する。

6　特徴

両機能の機能上の特徴を6.1に、性能上の特徴を6.2に述べる。また、両機能による分散ファイルシステムの形態を6.3に述べる

6.1　機能上の特徴

リモートファイルアクセスとリモートデバイスアクセスの機能上の特徴比較を表1に示す。表1からわかるように、リモートデバイスアクセスは、共用、排他制御でリモートファイルアクセスに比べ機能的に劣るが、アクセス先PUのOS構成を小さくできるため必要となるメモリ量を少なくでき、ディスクの有無にかかわらず全てのPUにAPを分散できるという利点がある[5]。

表1　リモートファイルアクセスとリモートデバイスアクセスの機能の比較

	リモートファイルアクセス	リモートデバイスアクセス
アクセス先PUのOS構成条件	ファイル管理を必要とする	ファイル管理を必要としない
資源の共用	任意のPUで走行するAP間で資源を同時に共用することが可能	ローカルAP、リモートAP間で同じ資源を同時に共有することは不可能
資源の排他制御	任意のPUで走行するAP間での資源の排他制御が可能	ローカルAP、リモートAP間では排他制御は不可能
ディスクを持たないPU上のAPからのアクセス	不可能	ディスクを持たないPUのローカルファイルシステムを別PUのディスク上に構成することにより、可能

6.2 性能上の特徴

リモートファイルアクセスとリモートデバイスアクセスで発生するPU間通信の回数と、システムコール及び内部コールの関係を図7に示す。図7からわかるように、リモートファイルアクセスで発生するPU間通信の回数はシステムコールの発行回数に等しいが、リモートデバイスアクセスで発生するPU間通信の回数は、ファイル管理とドライバ間の内部コールの発行回数と等しい。このため、両アクセス機能の有効な適応範囲は以下のように分かれる。

（1）通信回線などの特殊ファイルに対するアクセスでは、システムコールの発行回数と、ファイル管理とドライバ間の内部コールの発行回数が等しいため、リモートデバイスアクセスが速い。

（2）キャッシュバッファを経由した通常ファイルのアクセスでは、①キャッシュバッファへのヒット率が高いアクセス(例えば、小さなファイルを何度もアクセスするような場合)では、ファイル管理とドライバ間の内部コールの発行回数が減るため、リモートデバイスアクセスの方が速いが、②キャッシュバッファへのヒット率が低いアクセス(例えば、大きなファイルをシーケンシャルにアクセスするような場合)では、ファイル管理とドライバ間の内部コールの発行回数が増えるため、リモートファイルアクセスの方が速い。

6.3 分散ファイルシステムの形態

リモートファイルアクセスとリモートデバイスアクセスの2つの機能を持つ分散ファイルシステムにより、性能、経済性、APの作りやすさなど、目的に合わせ、柔軟にAPやOSを構成できる。分散形態の例として、2PU間での分散ファイルシステムを性能を重視して構成した場合を図8(a)に、経済性を重視して構成した場合を図8(b)に示す。

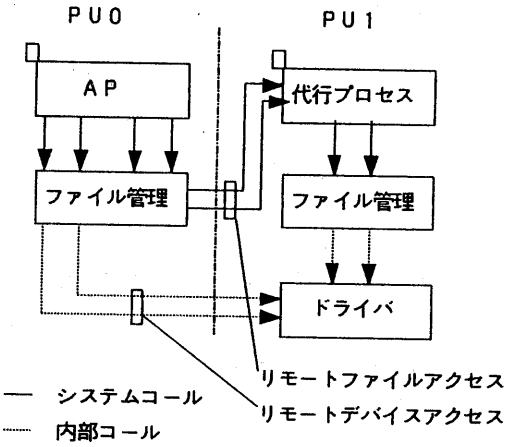
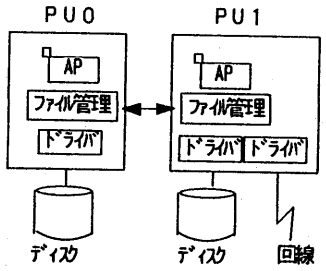
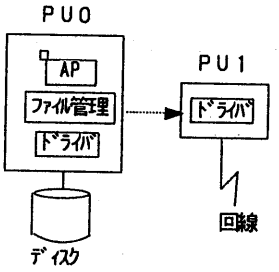


図7 PU間通信回数と
システムコール／内部コール回数の関係



（a）性能を重視した場合



（b）経済性を重視した場合

リモートファイルアクセス
リモートデバイスアクセス

図8 2PU間の分散形態の例

7 あとがき

非完了システムコール機能を持ち、OS機能分散を可能にしているDIROS上で実現した分散ファイル管理方式について報告した。

ファイル管理間のリクエスト制御によるリモートファイルアクセスと、ファイル管理－

ドライバ間のリクエスト制御によるリモートデバイスアクセスの２つの機能により、まざまな分散形態が実現できるな分散ファイルシステムが構成できる。

〈参考文献〉

[1]Rifkin,A.P.,et al.:RFS Architectual Overview,Proceedings of USENIX Summer Conference, USENIX Association, 1986

[2]Standberg,R.,et al.:Design and Implementation of the Sun Network Filesystem,Proceedings of USENIX Summer Conference, USENIX Association,1985

[3]谷口、鈴木、瀬々：ファイル管理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、pp56−63,1986

[4]谷口：OS機能の分散を可能にするOS構成法、電子情報通信学会論文誌D−I,vol.J72−D−I,No.3,pp168−174,1989

[5]井村、小林、谷口：マルチプロセッサにおけるマルチファイルシステムの構成、第38回情報処理学会全国大会、5N−2,1989