

分散オペレーティングシステム Solelc における ファイルシステムの構築

水口 孝夫[†] 芝 公仁[†] 大久保 英嗣^{††}

[†]立命館大学大学院理工学研究科 ^{††}立命館大学理工学部

現在、我々は、分散オペレーティングシステム Solelc を開発している。Solelc では、単一のオペレーティングシステムでネットワーク上の複数の計算機を管理する。各計算機の抽象化機構が協調動作することにより、計算機資源の位置透過性を実現するため、システムは複数の計算機を1台の計算機のようにユーザプロセスに見せることが可能である。このような環境において、システム全体で1つの大きなディレクトリツリーを構成するようなファイルシステムを構築することにより、ユーザプロセスは、自身の動作する計算機の位置に依らず、システムにおけるディレクトリあるいはファイルを常に同じ方法で参照することが可能となる。本稿では、Solelc における分散ファイルシステムの構築手法について述べる。

The Design and Implementation of File System on Distributed Operating System Solelc

Takao Mizuguchi[†] Masahito Shiba[†] Eiji Okubo^{††}

[†]Graduate School of Science and Engineering, Ritsumeikan University

^{††}Faculty of Science and Engineering, Ritsumeikan University

We have been developing a distributed operating system Solelc. Plural computers which Solelc works on are managed by an operating system. With cooperative work of the abstraction mechanism on each computer, the location transparency of computer resources be realized. Therefore, user processes can consider plural computers as a single one. The distributed file system which has a log-structure, such as Unix file system, has been constructed in this environment. This file system makes it possible for user processes in the system to access to a directory or a file in a same way wherever they work. In this paper, a construction method of the distributed file system on Solelc is described.

1 はじめに

近年、計算機の低価格化と高速ネットワークの普及に伴い、多くの計算機がネットワークに接続された形態で利用されるようになってきている。しかし、従来のオペレーティングシステム（以下 OS と記す）は、自身の動作する計算機上に存在する資源のみを管理するため、その OS 上で動作するプロセスは他の計算機上に存在する資源を利用することが困難である。また、これらの計算機は環境に差異があり、それぞれ処理能力や接続されるデバイスが異なる。そこで、我々は、これらの様々な環境を持つ計算機群を統一して管理する OS の構築を考えている [1]。

我々が開発している分散 OS Solelc では、従来の OS と異なり、単一の OS でネットワーク上の複数の計算機を管理する。分散仮想記憶と呼ばれる共有アドレス空間上で OS を動作させることにより、任意の計算機で動作している機能がすべてのユーザプロセスから利用可能となる。したがって、すべての計算機に同一の環境が実現され、ユーザプロセスは任意の計算機で他のすべての計算機と同様の機能を利用することが可能となる。また、各計算機の抽象化機構が協調動作することにより、計算機資源の位置透過性を実現しているため、システムは複数の計算機を 1 台の計算機のようにユーザプロセスに見せることが可能である。

我々は、以上の Solelc における基本的な考え方を検証するために、Solelc へのファイルシステムの実装を行っている。本ファイルシステムは、UNIX 系 OS と同様のディレクトリツリー構造を持つ。位置透過な環境を利用し、システム全体で 1 つの大きなディレクトリツリーを構成することにより、ユーザプロセスは、自身の動作する計算機の位置に依らず、システムにおけるディレクトリあるいはファイルを常に同じ方法で参照することが可能となる。また、抽象化機構により位置透過な環境が実現されるため、ファイル管理モジュールはネットワーク上の任意の計算機から利用可能となる。ファイルを管理する機構が 1 つで済むことから、ファイルシステムの構成が簡素化され、実現も容易になる。

以下、本稿では、2 章で Solelc の概要を述べ、3 章で分散ファイルシステム、4 章で位置透過なディスクアクセスについて述べる。次に、5 章で評価を行い、6 章で関連研究について述べ、最後に 7 章で本

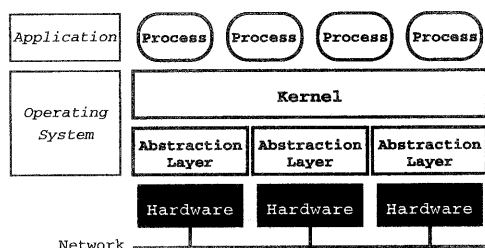


図 1 システムの全体構成

稿のまとめと今後の予定について述べる。

2 Solelc の概要

我々が開発している分散 OS Solelc では、従来の OS と異なり、単一の OS でネットワーク上の複数の計算機を管理する。各計算機の抽象化機構がネットワークを介して協調動作を行うことにより、計算機資源の位置透過性を実現する。これにより、システムは、ユーザプロセスに対し複数の計算機を単一の計算機のように見せる。

Solelc では、分散仮想記憶と呼ばれる共有アドレス空間を利用している。分散仮想記憶上で OS を動作させることにより、任意の計算機で動作している機能がすべてのユーザプロセスから利用可能となる。すなわち、システムは、OS 管理下にあるすべての計算機に同一の環境を実現しているといえる。

Solelc が管理するシステムの全体構成を図 1 に示す。すべての計算機上では、計算機資源の抽象化を行う抽象化層が動作する。各計算機がネットワークに接続され、各抽象化層が他の計算機上で動作する抽象化層と協調することにより、上位層に対してシステム上のすべての計算機資源を位置透過に提供することが可能となる。

カーネルは、各計算機の抽象化層により抽象化された環境上で動作する。そのため、カーネルは、任意の計算機上ですべての計算機資源を管理することが可能である。カーネル内の各モジュールあるいはスレッドを、それぞれ異なる計算機上で動作させることも可能である。また、カーネルは、ユーザプロセスの実行環境を実現する。抽象化された環境において、ユーザプロセスは、すべての計算機上で同一の機能を利用することが可能である。

このような構成のため、例えば、ユーザプロセスから OS へのサービス要求時において、次のような

手順を経る必要がある。

- (1) ユーザプロセスは、要求するサービスに対するシステムコールを発行する。システムコールは、そのプロセスの動作する計算機の抽象化層に伝えられる。
- (2) 抽象化層は、要求されたサービスを提供するカーネルモジュールを探し、そのモジュールの動作する計算機へ要求を送信する。
- (3) 要求を受信した計算機の抽象化層は、カーネルモジュールに要求を伝える。

すべてのユーザプロセスおよびカーネルモジュールとのデータ送受信は、このように各計算機の抽象化層を介することで実現される。このシステム構成に従い、ユーザプロセスにファイルサービスを提供するシステムを構築することで、位置透過性を持つ分散ファイルシステムを実現することが可能となる。次章では、Solelcにおける分散ファイルシステムの構築手法について述べる。

3 分散ファイルシステム

今回、UNIX系OSのファイルシステムと同様のディレクトリツリー構造を持つファイルシステムを構築した。本ファイルシステムは、位置透過な環境を利用し、Solelcにより管理されるシステム全体で1つの大きなディレクトリツリーを構成する。したがって、すべてのパス名は、計算機の位置に依らず、システムにおけるディレクトリあるいはファイルを一意に決定する。

本ファイルシステムは、次の機構により構成される。

- ファイル管理モジュール
カーネルモジュールの1つであり、ユーザプロセスからの要求を受け、ディレクトリツリー全体を対象としたファイル管理を行う。
- 要求管理機構
抽象化層に含まれ、ユーザプロセスからの要求や、その応答などの情報を管理する。また、各計算機固有の情報を管理し、自身の計算機上の適切な機構にアクセスする。
- 通信機構
抽象化層に含まれ、ファイルサービスを行う際に必要となる通信部分を担う。要求管理機構

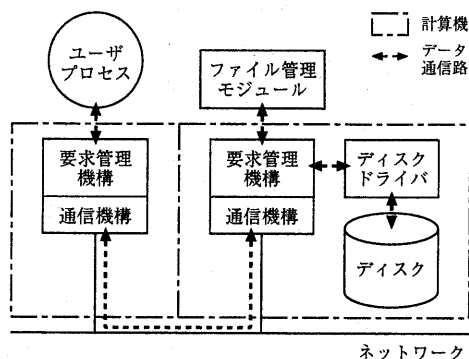


図2 ファイルサービス提供時のデータ通信路

の処理において、他の計算機との通信が必要となるときにアクセスされる。

● ディスクドライバ

抽象化層の最下位で動作し、ディスクに対するデータの入出力を管理する。ディスクドライバにより、ディスクデバイスに依存した部分が吸収されるため、ハードウェア独立な環境を実現することが可能となる。

このような構成をとることにより、サービス提供時のデータの通信路を図2のように制限する。ユーザプロセス、ファイル管理モジュール、ディスクドライバとのデータ入出力は、要求管理機構のみによって行われる。計算機間の通信は、通信機構がネットワークを利用して行う。計算機資源の位置情報が要求管理機構により隠蔽され、ユーザプロセス、ファイル管理モジュール、ディスクドライバにとって、他のあらゆる機構とのデータ送受信のためには、自身の計算機の要求管理機構にアクセスすることが必要十分となる。

以下、ファイル管理モジュール、要求管理機構、通信機構の詳細について述べる。

3.1 ファイル管理モジュール

ファイル管理モジュールは、ユーザプロセスからの要求に従い、ファイルサービスを提供する機構である。ファイル管理モジュールは、そのふるまいにより、Solelcにおけるファイルシステムの構造を決定する。Solelcにおいて、カーネルは抽象化された環境で動作するため、ファイル管理モジュールは、任意の計算機で動作可能である。また、抽象化層により、位置透過な環境が実現されているため、単一

表 1 ファイルサービス要求レコード

フィールド	意味
command	サービスの種類
fd	ファイルディスクリプタ
buf	バッファアドレス
size	要求データサイズ
host	要求元計算機 ID
ret	要求の返却値

のファイル管理モジュールの動作する計算機の位置に依らず、任意の計算機からサービスが利用可能となる。

本ファイルシステムは、システム全体で 1 つのディレクトリツリーを構成する。このツリーは、1 つのディスクがルートディレクトリの情報を持ち、他の複数のディスクが部分木にあたる子ディレクトリツリーの情報を持つことで実現される。ツリーを辿るにあたり、パス名の解釈はファイル管理モジュールが行う。ファイル管理モジュールは、計算機資源の位置情報を持たないため、単独でパス名から目的のディスクの存在する計算機を判別することは不可能である。そこで、抽象化層により、システム上に存在するディスクを一意に識別する整数値として、ディスク ID を定義する。ファイル管理モジュールは、パス名とディスク ID の対応情報をマウントテーブルとして保持する。ディスク ID と計算機の対応を要求管理機構に任せることにより、システム全体としてパス名と計算機の対応づけを可能としている。

ユーザプロセスは、ファイルサービスを要求するために、システムコールを発行する。要求は、抽象化層の要求管理機構に伝えられ、ファイル管理モジュールの動作する計算機のリクエストキューに蓄積される。要求は、要求管理機構により、複数のフィールドを組にしたレコードとして管理され、それをファイル管理モジュールが解釈し、処理を行う。要求レコードのフィールドを表 1 に示す。ここで、計算機 ID とは、Solelc の管理する計算機を一意に識別する整数値であり、これを指定することでデータの送信先を決定する。ファイル管理モジュールは、要求レコードの buf フィールドを参照することにより、ユーザプロセスが指定した分散仮想記憶上のバッファにアクセスすることが可能となる。

ユーザプロセスからの要求によりオープンされ

表 2 ファイルディスクリプタテーブルレコード

フィールド	意味
state	レコード使用状況
inode	i ノードデータ
offset	読み書きを行ったバイト数
disk	データの存在するディスク ID
inum	i ノード番号

たファイルは、ファイル管理モジュールにより、ファイルディスクリプタテーブルに登録され、管理される。ファイルディスクリプタテーブルのレコードは、表 2 に示すフィールドで構成される。ファイルをオープンしたときに i ノードデータが読み出され、それに含まれるブロック番号などのデータをもとにディスクにアクセスすることで、ファイルの読み書きを行う。

3.2 要求管理機構

要求管理機構は、ファイルサービスに伴う要求を管理する機構である。計算機資源の位置情報を管理し、通信機構の機能を利用して、要求を適切な計算機へ送信する。要求管理機構は、すべてのユーザプロセス、ファイル管理モジュール、ディスクドライバとの直接的なデータ入出力の役割を担うことで、すべてのユーザプロセスに対して位置透過なファイルサービス環境を提供する。

ファイル管理モジュールによるユーザプロセスへのサービス提供の過程において、さまざまな要求が発生する。また、すべての要求は、結果の応答を伴う。要求管理機構は、これらの要求およびその応答を適切な機構に伝える。また、要求を伝えてから応答が返るまでの間、要求元モジュールあるいはスレッドを停止させておき、応答が返ると実行可能状態に戻す。

ファイルサービスに伴う要求には、大きく次の 2 種類がある。

- ユーザプロセスによるファイルサービス要求
- ファイル管理モジュールによるディスクブロック要求

要求管理機構は、それぞれの要求毎に要求レコードを生成する。前者に対する要求レコードは、前節で述べたファイルサービス要求レコードであり、フィールドは表 1 の通りである。後者に対しては、

表 3 ブロック要求レコード

フィールド	意味
disk	要求先ディスク ID
block	開始ブロック番号
cnt	連続ブロック数
buf	バッファアドレス
host	要求元計算機 ID

表 3 に示すフィールドを持つブロック要求レコードを生成する。要求レコード生成後は、要求の伝達にその要求レコードを用いる。

ユーザプロセスからのファイルサービス要求は、ファイル管理モジュールの動作する計算機の抽象化層に集められる。抽象化層の要求管理機構は、それらの要求をリクエストキューに入れ、ファイル管理モジュールからの取出し要求を待つ。ファイル管理モジュールからの指示により、要求管理機構は、リクエストキューから順次ファイルサービス要求を取り出し、ファイル管理モジュールに渡す。

要求管理機構は、ディスクテーブルにより、ディスク ID と計算機 ID の対応情報を保持する。ファイル管理モジュールからのディスクブロック取得要求において、目的のディスクがファイル管理モジュールによりディスク ID で指定される。要求管理機構は、ディスクテーブルを参照することで目的の計算機を知ることが可能となる。要求を受信した計算機の要求管理機構は、ディスクドライバを介して目的のディスクブロックにアクセスし、要求された処理を行う。

3.3 通信機構

通信機構は、ファイルサービスの過程でネットワークを利用するための機能を提供する機構である。この機構により、各計算機の抽象化層に含まれる要求管理機構間のデータ送受信が可能となる。

ネットワークを利用してデータの送受信を行うには、送信側と受信側の両方の処理が必要である。送信側において、通信機構は、要求管理機構からの要求に応じてメッセージを生成し、抽象化層に含まれる通信機能を利用して送信する。受信側において、通信機構は、メッセージの種類を判別し、要求管理機構の適切な機能呼び出す。

通信機構が生成するメッセージのフォーマットを図 3 に示す。メッセージの先頭 1 バイトの領域

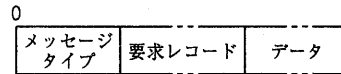


図 3 メッセージフォーマット

は、メッセージの種類を表す。メッセージの種類には、次に挙げるものがある。

- ファイルサービス要求
- ファイルサービス要求の応答
- ディスクブロック読出し要求
- 読出し要求の応答
- ディスクブロック書込み要求
- 書込み要求の応答

最初の 2 つは、ユーザプロセスによるファイルサービス要求に関するものであり、要求レコード領域には、ファイルサービス要求レコードが格納される。残りの 4 つは、ファイル管理モジュールによるディスクブロック要求に関するものであり、要求レコード領域には、ブロック要求レコードが格納される。データ領域は、ディスクブロック要求の処理においてブロックデータの格納に使用される。データ領域のサイズは、ブロック要求レコードの cnt フィールドを参照することで算出可能である。

4 ディスクブロックの参照

ファイル管理モジュールは、サービスをユーザプロセスに提供するために、頻繁にディスクにアクセスする。ファイル管理モジュールがディスクブロック読出し要求を発行し、目的のディスクブロックを取得するまでに、次の手順で処理が行われる。

- (1) ファイル管理モジュールは、ディスク ID、ブロック番号、ブロック数、バッファアドレスをパラメータとして、抽象化層の要求管理機構にディスクブロック読出し要求を発行する。
- (2) 要求管理機構は、ディスクテーブルを参照し、指定されたディスクの存在する計算機を知る。要求管理機構は、計算機の位置によって次の処理を行う。

- 自身の計算機上に目的のディスクが存在するとき、ディスクドライバを介して目

的のディスクブロックを取得し、要求処理を終了する。

- 他の計算機上に目的のディスクが存在するとき、ファイル管理モジュールからの要求と自身の計算機 ID をもとに要求レコードを生成する。要求レコードと目的の計算機 ID を通信機構に渡し、以降の処理を続ける。

- (3) 通信機構は、要求レコードからディスクブロック読出し要求のメッセージを生成し、計算機 ID で指定された計算機に対してメッセージを送信する。要求管理機構に制御が返り、要求管理機構は、ファイル管理モジュールを停止させる。
- (4) メッセージを受信した計算機の通信機構は、メッセージがディスクブロック読出し要求であることを知り、ブロック要求レコードを取り出す。
- (5) 通信機構は、ブロック読出しバッファ領域を確保し、ディスクブロックを指定するパラメータとともに要求管理機構に渡す。
- (6) 要求管理機構は、ディスクドライバを介して目的のディスクブロックを読出しバッファ領域に取得し、制御を通信機構に返す。
- (7) 通信機構は、ブロック読出し要求の応答メッセージを生成し、要求元計算機にメッセージを返信する。
- (8) 要求元計算機の通信機構は、受信したメッセージがブロック読出し要求の応答であることを知り、メッセージから要求レコードを取り出し、要求レコードのバッファアドレスにメッセージのデータ領域の内容をコピーする。
- (9) 通信機構は、要求レコードを要求管理機構に渡す。
- (10) 要求管理機構は、ファイル管理モジュールを実行可能状態にする。

ファイル管理モジュールがディレクトリツリーを辿る際、複数のディスクを参照しなければならないことがある。例として、図 4 に示すような環境を想定する。ここで、ディスク 1 が “/” ディレクトリ、ディスク 2 が “/usr” ディレクトリを持つとする。また、各ディスク内のディレクトリツリーについて、最上位ディレクトリの i ノード番号は 0 であり、その i ノードを含むブロック番号は予めファイル管理モジュールが知っているものとする。

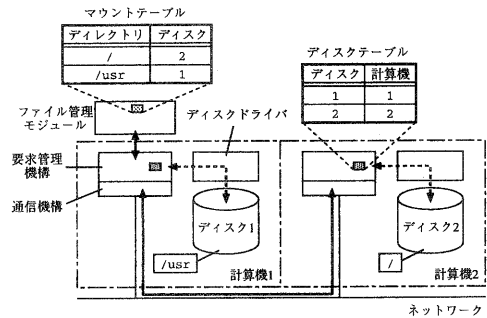


図 4 ディスクブロックの参照

ファイル管理モジュールがパス名 “/usr” を解釈するとき、以下の手順で処理が行われる。

- (1) ファイル管理モジュールは、マウントテーブルを参照して “/” ディレクトリがディスク 2 に存在することを知る。そこで、[ディスク 2] と [i ノード 0 のブロック番号] を指定して抽象化層の要求管理機構にディスクブロック読出し要求を発行する。
- (2) 先に述べた手順により、要求管理機構がディスクテーブルを参照して計算機 2 とのデータの送受信を行うことで、ファイル管理モジュールは、“/” i ノードを取得する。
- (3) ファイル管理モジュールは、取得した i ノードからアクセス許可情報を取得、確認し、“/” データを含むブロック番号を取得する。
- (4) (1) および (2) と同様に、ファイル管理モジュールは “/” データを含むブロックを取得する。
- (5) ファイル管理モジュールは、“/” データから “/usr” ディレクトリが存在することを確認する。次に、マウントテーブルから “/usr” ディレクトリがディスク 1 に存在することを知り、[ディスク 1] と [i ノード 0 のブロック番号] を指定して要求管理機構にディスクブロック読出し要求を発行する。
- (6) 要求管理機構は、ディスクテーブルからディスク 1 が自身の計算機に存在することを知り、ディスクドライブを介してディスクブロックを指定されたアドレスに読み出す。

以上の手順により、ファイル管理モジュールが “/usr” ディレクトリを参照することが可能となる。

表 4 位置関係とオーバーヘッド

位置関係	ネットワーク	ページ転送
(1)	—	—
(2)	—	レ
(3)	レ	—
(4)	レ	レ

レ：オーバーヘッドあり —：オーバーヘッドなし

5 評価

Solelc における分散ファイルシステムでは、ファイルサービスに関わるユーザプロセス、ファイル管理モジュール、ディスクの位置関係によって、そのサービスの所要時間が異なる。本章では、これら 3 要素の位置関係によるサービスの比較および考察を行う。

ユーザプロセス、ファイル管理モジュール、ディスクの位置関係として、次の 4 つの場合について考える。

- (1) すべてが同一計算機上に存在する場合
- (2) ユーザプロセスのみ異なる場合
- (3) ディスクのみ異なる場合
- (4) それぞれが異なる場合

それぞれの位置関係について、ファイルサービスに伴うネットワークのオーバーヘッドを表 4 に示す。

(3) または (4) の場合、ディスクがファイル管理モジュールと異なる計算機上に存在するため、ファイル管理モジュールがネットワークを介してディスクにアクセスする必要がある。また、(2) または (4) の場合、ユーザプロセスとファイル管理モジュールが異なる計算機上に存在するため、ユーザプロセス上のメモリ領域を参照するときに分散仮想記憶によるページ転送が発生する。

それぞれの位置関係において、ユーザプロセスが read システムコールを発行してから応答を受けるまでに要した時間を図 5 に示す。また、得られた数値から導いたサービス所要時間の大きな内訳を図 6 に示す。ここで、それぞれの計算機は、Pentium 200MHz の PC であり、100Mbps のイーサネット で接続されている。また、通信プロトコルには UDP/IP を用いている。

(1) の場合、ファイルサービスに伴う一連の処理が単一の計算機上で完結するため、サービスの所要

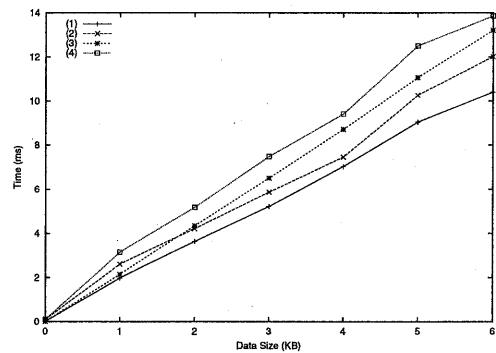


図 5 ファイルサービスの所要時間

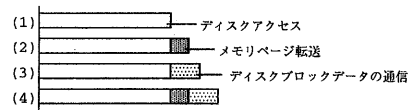


図 6 時間の内訳

時間は、あらゆる位置関係の中で最短となる。このとき、サービス所要時間は、ユーザプロセスが要求したデータサイズにほぼ比例する。

(2) の場合、ユーザプロセスが指定したバッファアドレスにファイル管理モジュールがアクセスしようとするとき、分散仮想記憶によるメモリページ転送が生じる。ページサイズが 4 キロバイトであるため、データサイズが 4 キロバイト増加する毎に転送ページ数も増加し、そのための時間を要する。

(3) の場合、(1) と同様に、サービス所要時間がデータサイズにほぼ比例する。ネットワークのオーバーヘッドのため、単位データサイズあたりの所要時間の増加量が (1) よりも大きくなる。

(4) の場合、すべての要求および応答においてネットワークを利用するため、(2) および (3) の両方のオーバーヘッドを伴う。ネットワークのオーバーヘッドが (1) の 40% 程度となり、サービスに最も時間を要する。

データサイズが大きくなるに従い、(3) および (4) の場合に見られるネットワークを介したディスクアクセスのオーバーヘッドが無視できなくなる。この問題は、ファイル管理モジュールが 1 回のディスクブロック取得要求で複数のブロックを要求し、通信回数を削減することで解決可能である。このとき、(3) および (4) は、それぞれ (1) および (2) に近

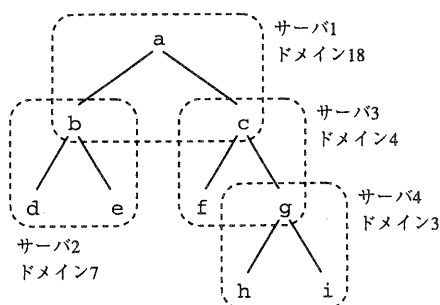


図7 Sprite のツリー構造

づく。また、ディスクドライバにブロックのキャッシュ機構を持つことで、いずれの位置関係においてもサービスの所要時間を短縮することが可能である。

6 関連研究

システム全体で1つのディレクトリツリーを構成することで、ファイルおよびディレクトリは、単一のパス名によりシステム上のすべての計算機からアクセス可能となる。したがって、このようなファイルシステムは、完全な位置透過性を実現しているといえる。

システム全体で1つのディレクトリツリーを構成する分散ファイルシステムとして、Sprite ファイルシステム [2] がある。Sprite において、部分木にあたるツリー構造をドメインと呼ぶ。ファイルシステムは、ドメインを接続することで1つのディレクトリツリーを構成する。サーバが1つあるいは複数のドメインを管理し、複数のサーバによりディレクトリツリー全体を管理する。Sprite ファイルシステムにおけるディレクトリツリー構造を図7に示す。ここで、各サーバは、自身の管理するドメインの接続情報を記憶しておらず、それら接続情報を持つプレフィックステーブルを参照することにより、パス名の解釈を行う。このテーブルは、ファイルシステムの変化に応じて動的に変更することが可能である。

このように、Sprite ファイルシステムでは、複数のサーバとプレフィックステーブルによりディレクトリツリーを管理している。Solelc において、ファイルシステムによるディレクトリツリーの管理は、Sprite においてすべてのドメインを単一のサーバ

で管理することに相当する。したがって、Solelc におけるファイルシステムは、Sprite のそれと類似した性質を持つと考えられる。また、Solelc では、単一のサーバによりディレクトリツリー全体を管理することで、Sprite において生じる複数サーバ間の通信を削減している。このことは、ファイルシステムの構成が簡素化し、実現が容易になることを意味する。

7 おわりに

本稿では、分散 OS Solelc の概要と、Solelc における分散ファイルシステムの構築手法について述べた。Solelc では、単一の OS でネットワーク上の複数の計算機を管理する。各計算機の抽象化層が協調して動作することにより計算機資源の位置透過性が実現されるため、システムは、複数の計算機を単一の計算機のようにユーザプロセスに見せることが可能である。

Solelc における分散ファイルシステムは、Solelc により管理されるシステム全体で1つの大きなディレクトリツリーを構成する。すべてのパス名は、計算機の位置に依らず、システムにおけるディレクトリあるいはファイルを一意に決定する。これにより、ユーザプロセスは、位置透過なファイルサービスを利用することが可能となる。

今後は、ファイルサービスを利用するためのシステムコールの拡充、ディスクブロックのキャッシュ機構の実現方法についての検討を行う予定である。

参考文献

- [1] 芝公仁, 大久保英嗣: “位置透過な資源管理のための計算機資源の抽象化手法,” 情報処理学会研究報告 2000-OS-83, pp.67-72 (2000).
- [2] Brent B. Welch and John K. Ousterhout: “Pseudo-File-Systems,” <ftp://ftp.cs.berkeley.edu/ucb/sprite/papers/pfs.ps> (1989).