7B－2

# 高速UNIXファイルシステムの開発における インタフェース仮想化方式の実現

山田 秀則　秋沢 充*1　山下 洋史*2　加藤 寛次*2　鬼頭 昭*3　牧 敏行
日立コンピュータエンジニアリング(株)　*1(株)日立製作所コンピュータ事業本部
*2(株)日立製作所 中央研究所　*3(株)日立製作所 ソフトウェア開発本部

## 1．はじめに

近年，UNIXワークステーションのCPU性能の向上は著しいが，ファイルアクセス性能の向上は十分ではない。筆者らは，ファイル・ストライピングを用いた高速UNIXファイルシステム“バーチャルアレイ・ファイルシステム(VAFS)”を提案し，ファイルアクセス性能の向上に取り組んでいる[1]。VAFSをより使い易いシステムにするには，アプリケーション・プログラム（AP）に対し従来のファイルアクセス・インタフェースを提供する必要がある。VAFSではファイル・ストライピングを行うシステムを1つのファイルシステムに仮想化して見せる“インタフェース仮想化方式”により従来のAPインタフェースを提供する。本稿ではVAFSのインタフェース仮想化方式の実現方式について報告する。

## 2．ファイル・ストライピング方式

ファイル・ストライピングでは複数ディスク装置に分割格納したデータの管理方法が課題である。VAFSでは各ディスク装置をUNIXファイルシステム（UFS）として管理し，分割したデータをUNIXファイルに格納し管理する。UFSを利用したのは，各ディスク装置の管理が容易に行える上，分割したデータの管理もファイル名で行えるからである。分割したデータを格納するファイルシステムとファイルを，それぞれサブファイルシステム，サブファイルと呼ぶ。図1にVAFSのファイル・ストライピング方式を示す。

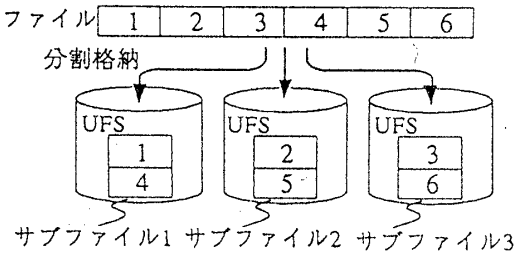


図1　ファイル・ストライピング方式

## 3．インタフェース仮想化方式

ファイル・ストライピングをAPが行なう場合には，分割されたデータを格納する複数のファイルをAPで管理する必要があり，困難である。VAFSでは“インタフェース仮想化方式”をUNIXオペレーティングシステムで実現し，APがファイル・ストライピングを意識することなく，UFSと同じインタフェースでVAFSを使用できるようにする。VAFSは複数のファイルシステムを用いてファイル・ストライピングを行うので，VAFSにおけるインタフェース仮想化方式では，複数のファイルシステムを1つのファイルシステムに仮想化する。仮想化したファイルシステムとファイルを，それぞれ仮想ファイルシステム，仮想ファイルと呼ぶ。図2にVAFSのファイルシステムの仮想化方式を示す。APは仮想ファイルシステムを介してVAFSにアクセスし，仮想ファイルと複数サブファイルの対応管理はVAFSが行う。APが仮想ファイルをオープンすると，VAFSは対応する複数サブファイルをオープンし，仮想ファイルのファイル記述子にサブファイルのiノードを対応付ける。

## 4．インタフェース仮想化方式の実現

### 4.1　実現に向けての課題

VAFSにおけるインタフェース仮想化方式を実現するための課題を以下に示す。

（1）仮想ファイルのパス名と，サブファイルのパス名の対応管理

（2）仮想ファイルのファイル記述子とサブファイルのiノードの対応管理

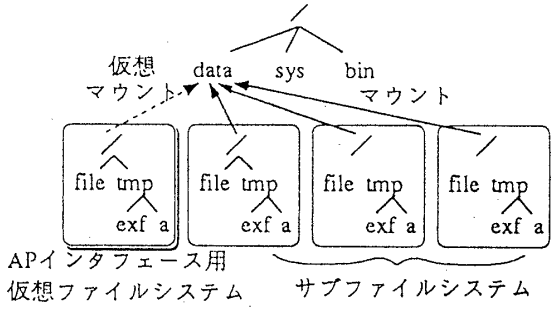


図2　ファイルシステム仮想化方式

---

An Implementation of Virtual Application Programming Interface in the Design of Performance Improved UNIX File System
Hidenori YAMADA, Mitsuru AKIZAWA*1,
Hirofumi YAMASHITA*2, Kanji KATO*2, Akira KITO*3,
Toshiyuki MAKI,
Hitachi Computer Engineering Co.,Ltd.
*1 Computer Group,Hitachi,Ltd.
*2 Central Research Laboratory,Hitachi,Ltd.
*3 Software Development Center,Hitachi,Ltd.

（1）はAPが仮想ファイルのオープン，作成，削除などを行う時の課題で，（2）はAPが仮想ファイルをオープンした後，ファイル記述子を用いて仮想ファイルにアクセスする時の課題である。これらの課題を解決するために，多重マウント方式と多重iノード管理方式を開発した。

## 4.2 多重マウント方式

多重マウント方式は，APが仮想ファイルシステムのマウント先として指定したディレクトリに，すべてのサブファイルシステムを多重にマウントする方式である。この方式の利点は，各サブファイルシステムのディレクトリ構造を同じにすれば，APが指定した仮想ファイルのパス名を，そのまま対応する複数サブファイルのパス名として使用できるため，パス名の対応管理が容易に行える点である。そのために，分割したデータは同じパス名のサブファイルに格納する。この多重マウント方式を実現するためにmlinks（mount table link structure）テーブルを新たに設ける。mlinksテーブルは各サブファイルシステムのmountテーブルを指すテーブルである。mountテーブルはマウントされたファイルシステムを管理するテーブルで，パス名の解析時に使用される。図3に標準UNIXファイルシステムをマウントした時と，多重マウント方式により仮想ファイルシステムをマウントした時のマウント管理の比較を示す。標準UNIXファイルシステムをマウントすると，mountテーブルは，そのファイルシステムのスーパーブロックとルートiノードを指し，マウントディレクトリのiノードからポイントされる。一方，仮想ファイルシステムをマウントすると，仮想ファイルシステムのmountテーブルはスーパーブロックの代わりに，mlinksテーブルを，ルートiノードの代わりにルートiノードのilinksテーブル（後述）を指し，mlinksテーブルがさらに各サブファイルシステムのmountテーブルを指す。mlinksテーブルは，仮想ファイルシステムのスーパーブロックに相当する。パス名解析の時，mlinksテーブルが指すmountテーブルを順次選択することにより，同じパス名で異なる複数サブファイルにアクセスすることが可能となる。

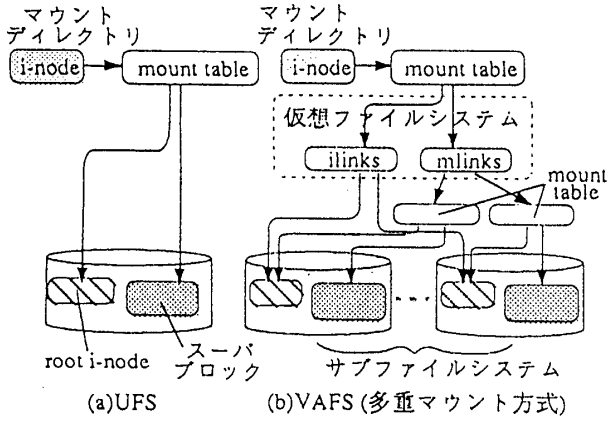


図3 マウント管理の比較

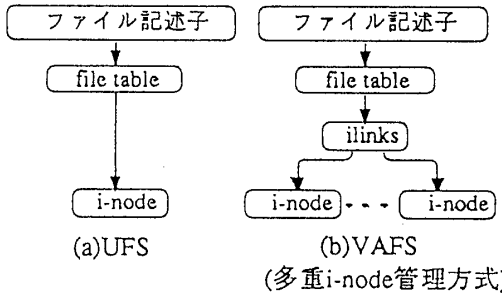


図4 iノード管理

## 4.3 多重iノード管理方式

多重iノード管理方式は，仮想ファイルのファイル記述子に，複数サブファイルのiノードを多重に対応付けることによって，オープン後の仮想ファイルと複数サブファイルの対応を管理する方式である。この方式を実現するために，サブファイルのiノードを指すilinks（i-node link structure）テーブルを新たに設ける。図4に標準UNIXファイルをオープンした時と，仮想ファイルをオープンした時のiノード管理の比較を示す。APが標準UNIXファイルをオープンすると，ファイル記述子に対応するfileテーブルが割り当てられ，fileテーブルはオープンしたファイルのiノードを指す。一方，仮想ファイルをオープンすると，fileテーブルは，オープンしたサブファイルのiノードを指すilinksテーブルを指す。この場合，ilinksテーブルは，仮想ファイルのiノードに相当する。仮想ファイルへのAPのアクセス要求に応じて，ilinksテーブルが指すiノードを選択することで，仮想ファイルからサブファイルへのアクセスの変換が可能となる。

## 5．おわりに

ファイルシステムレベルでのストライピングを行うVAFSのインタフェース仮想化方式を，多重マウント方式，多重iノード管理方式により実現した。これによりAPに対して従来インタフェースを提供することができた。

## 参考文献

[1]秋沢他5，「バーチャルアレイ・ファイルシステム（vafs）の基本構想」，情処全国大会講演論文集4－61，平4年10月

注)UNIXオペレーティングシステムはUNIX System Laboratories, Inc.が開発し，ライセンスしています。