B-024

木構造ファイルシステムの問題
Problems of Tree Structured File Systems

瀧本 栄二 † **門脇 直人** † **小花 貞夫** †
Eiji Takimoto Naoto Kadowaki Sadao Obana

1. はじめに

近年，2 次記憶装置の記憶容量の増大化や，複数計算機間でのファイル共有が一般的になっている．これに伴い，ファイルシステムは肥大化しつつある．ファイルシステムが大きくなるにつれ，ファイルの管理が困難になるという問題が顕著になってきている．ファイル管理を難しくする要因として，ファイルシステムがユーザに提供するビューと，ファイルに割り当てられるパスが共に 1 つであることが考えられる．

この問題を解決するために，属性を用いたファイル管理などの研究 [1][2] [3] が行われてきた．しかし，従来の研究は，ファイルシステムの上位で木構造と異なる抽象化を行っているなど，木構造ファイルシステムが持つ問題を根本的に解決しているとはいえない．

我々は，従来の解決手法では根本的な解決が困難であると考えている．そこで，本稿では，木構造ファイルシステムが孕む問題と，従来の研究のアプローチを整理する．さらに，木構造ファイルシステムの問題を解決するためには，どの様なアプローチが必要であるかについて述べる．

2. 木構造ファイルシステム

木構造ファイルシステムが持つ欠点は，大別して木構造というモデルに関するものと，システムに関するものがある．以下，本章では，これらの詳細について述べる．

2.1 木構造モデルの欠点

木構造ファイルシステムは，根となるディレクトリを始点として，階層構造をとる．この構造は，各葉へのパスが 1 つのみ存在する有向木と見なすことができる．したがって，ディレクトリ間の移動やファイルの指定には，有向木の制約を受けることになる．ディレクトリは，有向木の節に相当し，ファイルを分類する役割を持つ．その観点から考えると，ファイルのパスに含まれるディレクトリは，ファイルを特徴づける属性としての意味を持つといえる．木構造モデルでは，ファイルとディレクトリは自身が属する上位のディレクトリに依存する．

これらにより，柔軟で分かりやすい名前空間の形成が困難になるという問題が生じる．ディレクトリは属性を表すため，適切にディレクトリを扱うことでファイル管理を容易にすることができる．しかし，粒度の細かい属性付けは，名前空間を複雑にするために限界がある．また，各節であるディレクトリには順序関係がある．例えば，パス/a/b/c と/b/a/c では，異なる意味を持つ．このことは，最適な名前空間の構成を困難にし，結果として複雑な名前空間の原因となる．また，木構造ファイルシステムでは，縦方向の参照しか許されない．したがって，横断的な参照ができず，ファイル検索が非効率になるなどの問題がある．

2.2 システム構成の問題

ファイルシステムは，ファイルの永続性を実現するために，ファイルを 2 次記憶装置に記録する．従来の木構造ファイルシステムは，木構造モデルに基づく方式で 2 次記憶装置への記録を行っている．

例えば，ext2 などの Unix 系オペレーティングシステムで利用されているファイルシステムでは，inode を用いてファイル管理を行う．ファイルへのアクセスは，最初にルートディレクトリを指す inode を見つけ，指定されたデータブロックを参照し，次に参照すべきディレクトリを指す inode 番号を調べる．調べた inode 番号を基に inode を参照し，次のディレクトリを指す inode 番号を取得する．このような inode を辿っていく処理を，目的のファイルが見つかるまで繰り返す．

このように，抽象化されたディレクトリ構造を 2 次記憶上でも実現するため，ディレクトリ構造と 2 次記憶上の物理的構造が依存しあう．したがって，複数のビューを用意しユーザに最適なビューを提供することが困難である．ビューに関する問題は，ファイル共有時においてより大きな影響を及ぼす．2 次記憶装置とファイルシステムの関係上，ファイル共有を行う単位は副木となる．すなわち，あるディレクトリ以下が共有の対象となる．共有されている副木に対するビューは，すべてのユーザ間で統一されている必要がある．したがって，あるユーザが共有空間に対して行う操作は，共有している全ユーザのビューに影響を与えることになる．また，共有空間のディレクトリ構成のポリシが周知されていなければ，管理者以外のユーザはファイルの検索や配置を行うことが困難である．

3. 関連研究と研究アプローチ

3.1 関連研究

Semantic File System(SFS)[1] は，ファイルに対して属性を与え，その属性に基づいたファイル管理を実現している．ファイルのパスとは別に属性を用いることで，より詳細なファイル特性の記述が容易となり，柔軟なファイルの検索などが実現できる．属性は，$< attribute\_name : value >$ の形で表される．ファイルは，複数の属性を持つことができる．ある属性に基づく検索結果は，仮想的なディレクトリを用いて提供される．また，トランスデューサと呼ばれる属性を抽出するプログラムを持ち，これを用いることで内容に応じた属性をファイルに自動的に与えることができる．

Virtual System[4] は，大規模分散ステムを対象とした名前空間の提供を行う．システムモデルとして名前解決オートマトンを定義し，filter，union link，synonim，

†株式会社 国際電気通信基礎技術研究所

closure の概念により，ユーザが自由に名前空間を定義し，かつ複数の名前空間を扱うことを可能にしている．

文献 [3] では，ファイルシステムのデータモデルとして，属性とファイルの関係を表にしたものと，従来の木構造を組み合わせた方式を提案している．提案方式では，属性とその関係を木構造で表現し，表によりファイル名を与えている．このように，木構造と表形式で管理する内容を分けることで，互いの方式の欠点を補う柔軟なシステムの構築が可能であると述べられている．

Apple 社が開発した Spotlight[5] は，ファイルに与えた属性からインデックスを作成し，属性に基づくファイル検索するツールである．Spotlight は，ファイルシステムと連動することで，ディレクトリ構成の変化に即座に対応できる．さらに，複数の属性によるクエリを基に仮想的なディレクトリを提供している．

上で挙げた研究では，ファイルにパスとは異なる属性を与え，それらを用いることで，効率的で柔軟なファイル管理を実現している．多くの研究では，Spotlight と同様に，属性とファイルのインデックスを作成し，属性を単位とした検索クエリを提供している．Spotlight や SFS など既存の研究の多くは，実装アプローチとして既存の木構造ファイルシステム上に，ツールやライブラリの形で機能を提供している．このため，2.2 節で述べた木構造ファイルシステムの問題は根本的に解決されていない．これは，既に木構造ファイルシステムが一般的であるため，互換性等を考慮したためである．

3.2 研究アプローチ

我々は，木構造ファイルシステムの問題を解決するためには，ファイルシステムのモデルと 2 次記憶装置への格納方法を明確に分離する必要があると考える．データベースでは，外部スキーマ，概念スキーマ，内部スキーマからなる 3 層スキーマの考え方がある．従来のファイルシステムは，これらのスキーマが区別されていないといえる．特に，概念スキーマと内部スキーマは，互いに依存し合っている．

3 層スキーマの概念に基づいてファイルシステムを定義することで，データ独立を実現することができる．さらに，従来の木構造ファイルシステムもそのスキーマに取り組むことができるため，既存システムとの互換性を実現することができると考えられる．

ファイルシステムの概念モデルには，木構造モデルとは異なるモデルを採用すべきである．既存研究と同様，属性情報に基づく構造を採用することで，木構造の持つ有向木特性を解消し自由なファイルパスの形成を実現することができると考えられる．すなわち，属性の順序に捕われないファイルアクセスが可能になる．また，このようなモデルは，外部スキーマとの間の依存性がないため，複数のビューをユーザに提供したり，様々なビューの形を実現することが容易である．これにより，ファイル共有時に従来のディレクトリ単位ではなくファイル単位での共有なども容易に実現することができると考えられる．

以上のように，ファイルシステムを 3 層スキーマの概念から構成し直すことで，柔軟で扱いやすいシステムをユーザに提供できる．しかし，これらを実現するためには，属性の扱いについて考慮する必要がある．既存の研究 [1][2] では，属性とその値を自動で与えたりユーザが与えるなどの方法があるが，適切な属性の与え方ができなければユーザの混乱を招くことになる．また，ファイル共有時では，自分と他人が考える属性が異なるという問題もある．このように，属性を用いたファイル管理は，柔軟ではあるが最適な構成を取りづらい．したがって，属性に関するポリシを明確にすることが重要である．

4. おわりに

本稿では，最も一般的な木構造ファイルシステムに関する問題をあげ，新しいモデルに基づくファイルシステムの重要性について述べた．また，既存研究の傾向から，属性情報を用いたモデルが効果的であることと，3 層スキーマに基づく明確なモデルと実装の分離が必要であることについて述べた．

今後は，実際に新しいファイルシステムのモデルを考案し，そのモデルに基づくファイルシステムの構築を行っていく予定である．

参考文献

[1] David K. Gifford, Pierre Jouvelot, M. A. S. and Jr., J. W. O.: Semantic File Systems, *Proceedings of the 13th ACM Symposium on Operating Systems Principles*, pp. 16–25 (1991).

[2] Marsden, G. and Cairns, D. E.: Improving the usability of the hierarchical file system, *SAICSIT '03: Proceedings of the 2003 annual research conference of the South African institute of computer scientists and information technologists on Enablement through technology*, pp. 122–129 (2003).

[3] Sechrest, S. and McClennen, M.: Blending Hierarchical and Attribute-Based File Naming, *Proceedings of the 12th International Conference on Distributed Computing Systems*, pp. 572–680 (1992).

[4] Neuman, B. C.: *The Virtual System Model:A Scalable Approach to Organizing Large Systems*, PhD Thesis, University of Washington (1992).

[5] Apple Inc.: *Spotlight*, http://www.apple.com/jp/macosx/features/spotlight/.