バージョン管理機能を持つファイルシステムの設計

平井 貴浩 （最所研究室）

あらまし

ソフトウェア開発やデータ管理においてはバージョンの管理は重要な課題の一つである。バージョン管理機能をOS に実装することによりアプリケーションでのバージョン管理が必要でなくなる。本研究では、バージョン管理機能を持つファイルシステムの設計を行った。

1 はじめに

近年の計算機の普及により、多くのソフトウェアやデータが生成されている。これらは常に変化しており、作成者などの働きによって形を変えている。このため一つの情報ですら数多くのバージョンを持つことになり、これらの管理の重要性は大きくなっていく。バージョン管理を手軽に行う方法として今日でも、CVS[1], Elephant File System[2], 世界 OS[3] といったバージョン管理機能を持つシステムが利用されているが、速度、機能という両面から見ると十分とは言えない。そこで本研究ではバージョン管理機能を持つファイルシステムの研究を行なう。提案するシステムは、利用者の負担も軽く、動作も十分に高速なバージョン管理機能を実現する。

2 バージョン管理機能の設計

2.1 概要

本システムではファイルがアプリケーション等により内容が変更された場合、変更後の内容だけではなく、変更前の状態も保存することで、必要に応じて過去の状態の参照を可能にすることを目的とする。ただし、バージョン管理機能特有の操作を行わなかった場合は、従来のファイルシステムと同じ振る舞いをする。

バージョン管理を行った場合、1ファイルが多くのバージョンを持つことになる。従来のファイルシステムが path によりファイルを識別するのに対して、本システムでは、 path に加えてバージョン番号を利用しファイルとそのファイルのバージョンを特定する。バージョン管理機能を実現するために、本システムではバージョン管理領域と削除ファイルテーブルなどを追加した。

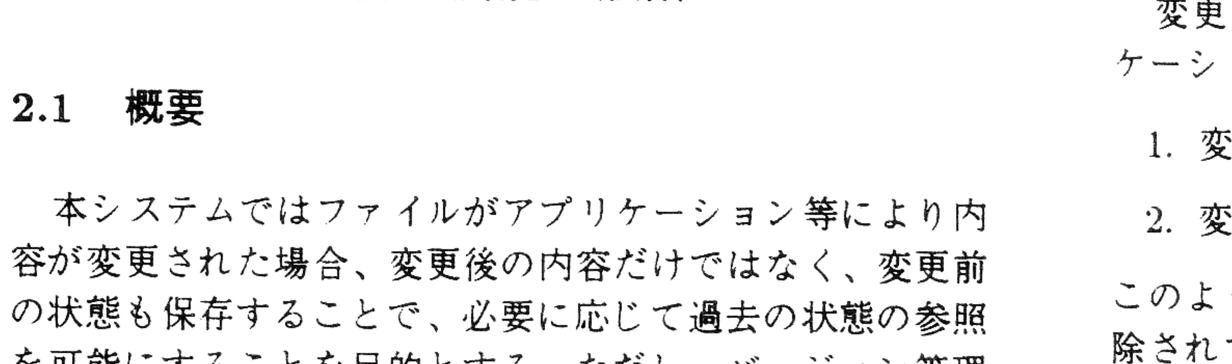

図 1: 従来のファイルシステムと本システムの違い

2.2 バージョン管理領域

バージョン管理領域は、1 つのファイルが複数のバージョンを持つために、追加した領域である。図 1 に示すように、従来のファイルシステムと異なりディレクトリエントリには、ファイルの i-node 番号 を登録するのではなくバージョン管理領域の i-node 番号を登録する。バージョン管理領域にはファイルの内容を保存しているバージョンの i-node 番号を登録する。

バージョン管理領域は、図 2 の表で示すようなデータ構造をとることで複数のバージョンを保持することを可能にしている。また、変更前のバージョン番号を親バージョン番号として保持しているため容易に変更前の状態に戻すことができる。

2.3 削除ファイルテーブル

変更したファイルを保存する際、 Emacs などのアプリケーションでは、以下の 2 つの操作を行う。

1. 変更するファイルの削除
2. 変更するファイルの新規作成

このような処理が行われた場合、バージョン管理領域が削除されてしまうため保持していた過去のバージョンも削除されてしまう。削除ファイルテーブルは削除したファイルを一時的に保護するために利用され、ディレクトリエントリから削除されたファイルを登録する。削除ファイルテーブルに登録することで、ファイルの実体が削除されるのを防ぐ。

Version No.	Parent VersionNo.	i-node No.
1	NULL	101
2	1	102
3	1	103
4	2	104

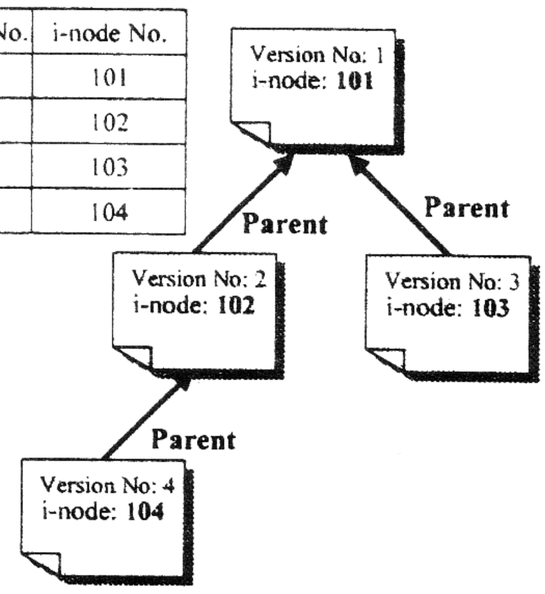

図 2: バージョン間の親子関係

新規にファイルを作成する場合は、ファイル削除テーブルを検索し、以前にファイルが削除されていないかを調べる。削除ファイルテーブルに存在した場合は、以下の 3 つの操作を行う。

1. ディレクトリエントリに登録
2. 削除ファイルテーブルから削除
3. 新規バージョンの作成

削除ファイルテーブルはファイルシステムマウント時にメモリ上に作成される。削除されたファイルが実際に削除されるのは、以下の 2 つの場合がある。

- 削除後一定時間が経過した場合
- ファイルシステムがアンマウントされる場合

2.4 データブロックの共有

バージョン間で、ファイルの内容を保持しているデータブロックの共有を行なう。例えば、図 3 で示すように既存のバージョンを親バージョンとして新たなバージョンを作成する場合、新バージョン用の i-node を確保しデータブロックは共有する。このことにより、バージョン作成時の処理を高速化するとともにディスクの利用効率を上げる。

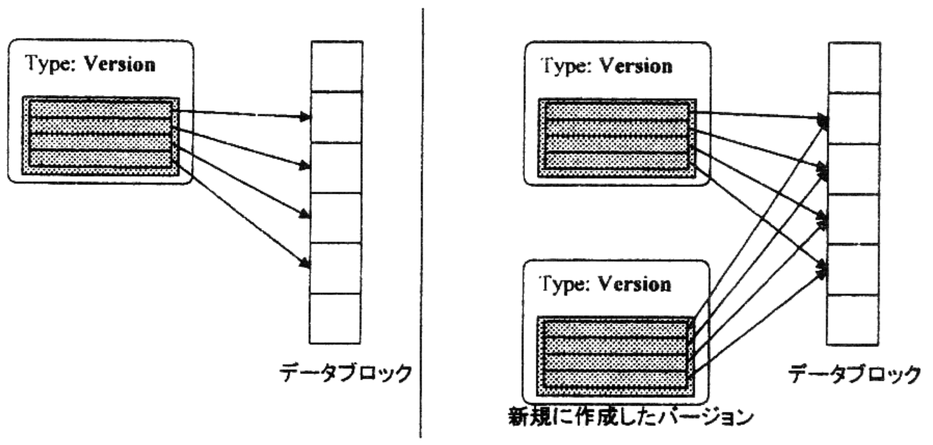

図 3: バージョン作成時のデータブロックの共有

共有しているデータブロックに対し書込みを行なう場合は、親バージョンが変更されないようにデータブロックの共有を中止する必要がある。図 4 に、示すように、書込み対象となったデータブロックの共有を解除し新規にデータブロックを確保する。

また、書込み範囲の先頭のブロックと最後のブロックは、データブロック全体が書込まれない場合があるため、これらのブロックは以前共有していたブロックの内容をコピーする。

2.5 バージョン管理特有のシステムコール

本システムでは、open, write などのシステムコールだけでは、バージョン管理機能のすべてを利用できない。そのため、バージョン管理特有のシステムコールを追加する。

version_open
: ファイルのパスとバージョン番号を指定することで、任意のバージョンをオープンするシステムコールである。このシステムコールを使うことで任意のバージョンの内容を変更することができる。

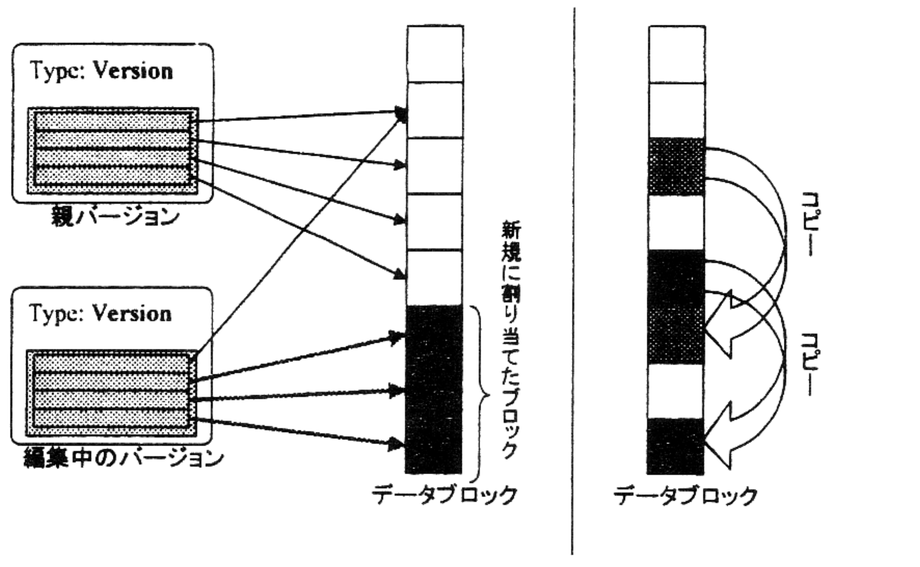

図 4: 書込み時のデータブロックの割当てと割当て後のデータブロックのコピー

version_close
: version_open() でオープンしたバージョンをクローズするシステムコールである。通常の close() システムコールとの違いは、バージョン管理機能特有の処理を行なわない点である。従来のファイルシステムにおける close() システムコールとほぼ同等である。

version_create
: 指定したバージョンを親バージョンとするバージョンを新しく作成するシステムコールである。

version_delete
: 任意のバージョンを削除するシステムコールである。削除するバージョンを親バージョンとして保存している場合は、その親バージョンを親バージョンの親バージョンに置き換える。また、削除するバージョンのみが使用していたデータブロックがあれば削除する。

read_version_list
: 指定したファイルのバージョン管理領域を読み取りバージョンの一覧を返す。

set_current_version
: open() システムコールを利用してファイルをオープンする際に利用するバージョン番号を保持する Current Version No. を指定する。

set_version_policy
: バージョンを自動生成する際のポリシーを設定する。

3 まとめ

バージョン管理機能を持つファイルシステムについて述べた。今後は実装を完了し評価を行なう予定である。

参考文献

[1] http://www.cvshome.org/

[2] Santry, D., Feelley, M., Hutchinson, N., and Veitch, A., "Elephant: The File System that Never Forgets", In *Proceedings of the IEEE Workshop on Hot Topics in Operating Systems*, March 1999.

[3] 新城, 孫: 並列世界モデルに基づくオペレーティングシステム, 情報処理学会コンピュータシステム・シンポジウム, 1997