NILFS CPviewerの設計と実装

黒田大陽† 廣津登志夫††

1. はじめに

現在，Linux や FreeBSD といった OS では ext2, ext3，FFS といったファイルシステムが広く使われている．これらのファイルシステムでは，データの変更時にはそれまでの情報を上書きする形で更新を行う．そのため，障害時などにファイルシステムの一貫性を損なう可能性がある．ext3 のようなジャーナリングファイルシステムでは，ファイルの管理情報であるメタデータだけは，ジャーナルと呼ばれるログに追記的に記録することでファイルシステムの一貫性は向上するが，データブロック自体には上書きによる更新が行われる．これらのファイルシステムとは全く異なる発想の元に作られたファイルシステムに，ログ構造化ファイルシステムというものがある．ログ構造化ファイルシステムでは，変更はすべて追記的に行い，データブロックの上書きを行わない．そのため，障害時などにファイルシステムの一貫性を損なう危険性が少ない．データブロックを線形に配置するために，ランダムアクセス性能は少し低くなるが，書き込み性能が向上するという利点もある．

ログ構造化ファイルシステム自体は 1990 年代に考案されたものであるが[1)]，その後，広く一般的に使われる実装というものはなかなか出てこなかった．NILFS (New Implementation of a Log-structured File System)[2)3)] は，NTT サイバースペース研究所により開発・公開されている Linux 向けのログ構造化ファイルシステムの実装で，安定した動作と実用的な性能が実現されている．NILFS では，ファイルシステムの一貫性が取れた状態毎にチェックポイントが付けられており，連続的かつ自動的なスナップショット作成と同等である．このチェックポイントに対して，リードオンリーでマウントすることにより，過去の情報を参照することが可能となり，誤って消去してしまったファイルの復元などが容易に行える．しかし，現在の実装では操作を行うためには特権が必要になっており，一般ユーザーがこの機能を自由に使うことは困難である．そこで，NILFS における過去の変更履歴に容易にアクセスできるようなツールが望まれる．本研究では，チェックポイントの系列を追跡しながら，ディレクトリ構造の参照を行うことができるビューアー，NILFS CPviewer の実現を目指している．

† 豊橋技術科学大学 情報工学課程
†† 豊橋技術科学大学/JST CREST

2. NILFS CPviewer

本研究では，チェックポイントという時間軸方向の変更系列を持つ NILFS において，目的のチェックポイントを効率良く探索し，ファイル群に対する変更の履歴を容易に見出せるように支援するための NILFS CPviewer を作成している．

NILFS CPviewer を実現するために必要となる機能は，

- 過去（古いチェックポイント）のディレクトリ構造にアクセスする機能
- NILFS のチェックポイント情報を獲得する機能

の二つである．これらの機能を実現するためには次の三つのアプローチが考えられる．

(1) デバイスファイルのアクセス権限とマウントする権限のユーザーへの付与
(2) autofs などのシステムサービスを利用
(3) NILFS の情報を取得するためのインターフェースの拡張

最初のアプローチは問題を解決する一番容易な方法である．当然のことであるが，デバイスファイルをアクセス可能にすると，その計算機資源に対してどのような操作もできてしまい，セキュリティ上問題である．また，setuid などを用いて CPviewer のみにアクセス権を与える手段も考えられるが，全体を特権で動作させてしまうと，誰のファイルに対してもアクセスできてしまう．そのため，マウントとチェックポイントの取得という最低限の部分のみ特権で動くように注意深くプログラムを記述する必要が生じるため，ビューアーのような大きなプログラムに特権を与えるのは危険であると考えられる．

二つ目のアプローチである autofs のようなシステムサービスを用いることによって，一般ユーザーでもチェックポイントを動的にマウントすることが可能となる．しかし，autofs はディレクトリパスの一部として与えられた一つのキーに対して一回のマウントを実行するだけであるので，これだけでは全てのチェックポイントの間を網羅的に渡り歩くことは困難である．実際，NILFS には autofs を用いるための設定が用意されている．しかし，用意されている設定では時間をキーとしてマウントポイントが決定されるため，短い時間に多くの変更が行われていた場合や変更と変更の間が空いてしまっている場合には目的のチェックポイントを探し出すことは難しい．チェックポイントは離散的に存在するので，マウントするためのキーも連続的な時間よりチェックポイント番号であることが望ましい．autofs を用いて，チェックポイント番号をキーとしてマウントすることは容易であるが，チェックポイント番号のリストを得る機能が必要となる．

最後のアプローチはシステムコールインターフェースの拡張により，NILFS の管理情報を取得や，NILFS 特有のファイルシステム操作を実現するものである．例えば，チェックポイント情報の提供やファイルやディレクトリへのアクセスの際に，チェックポイントを指定可能にするという拡張が考えられる．これは，システムコール処理ルーチンで利用権限の確認が可能であるため，セキュリティー面では最も好ましいと考えられる．一方，システムに大幅な改造が入り，NILFS のモジュール性が失われることが欠点である．

本研究では，(2) と (3) を組み合わせる形で実現する．ここでは，チェックポイントを取得するためのシステムコールの拡張を行う．このインターフェースは，「ファイルパスを与えると，そのファイルが存在するファイルシステムが NILFS であった場合にチェックポイント情報が得られる」といった挙動のものを実現しようと考えている．チェックポイントが得られたら，autofs にチェックポイント番号をキーとしてマウントする設定を用意し，それらのマウントポイントにアクセスするビューアーを実装する．

CPviewer においてチェックポイントの履歴情報を提示する方法として以下の二つが挙げらる．

(1) 複数のチェックポイントのそれぞれのディレクトリ構造を平面的に並べて表示
(2) 各チェックポイントの表示を奥行き方向に並べた三次元的な表示

平面的な表示では，複数の時刻における状況を並べて比較しやすく，一覧性の点で利点がある．一方，三次元的な表示では，変更の時間間隔などの情報を奥行きの距離として表現するなど，多様な表現が可能なことに利点がある．現在は一覧性を重視し，実装が容易であることから，表示部分を二分割して二つ分のチェックポイントを表示する平面的な表示によって実装している．また，二つのチェックポイント間で変更があったファイルについては，異なる色で表示されるので一目で分かるようになっている（図 1）．inspect を用いて得られる情報は，チェックポイント番号と作成時間の情報のみであり，「特定のファイルやディレクトリについて変更があった時点」といった，ユーザーが目的としているチェックポイントを探すことは難しい．本研究で作成した CPviewer は，ユーザーの関心事であるファイルやディレクトリの空間で変更履歴を追うことができるため，従来の履歴のアクセス方法の弱点を補っているものである．

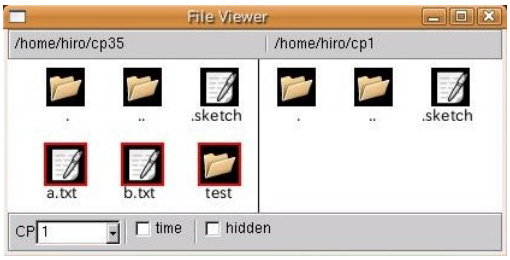

図 1 CPviewer の実行画像

3. 現状と今後の課題

現在はチェックポイントの情報を取得するシステムコールは実装中であるが，試作としてチェックポイント情報の取得部分だけ特権でデバイスファイルをアクセスする形で実装した．今後は実装中のシステムコールを利用するように改良すると共に，効果的な情報提示を行うための三次元的な表示方法や，よりモジュール性の高い実装について検討していきたい．

参 考 文 献

1) ユーレッシュ・ヴァハリア著, 徳田英幸, 中村明, 戸部義人, 津田悦幸訳: 最前線 UNIX カーネル, ピアソン・エデュケーション (2000).
2) 天海良治, 一二三尚, 小西隆介, 佐藤孝治, 木原誠司, 盛合敏: Linux 用ログ構造化ファイルシステム, *Linux Conference 2005* (2005).
3) 天海良治, 一二三尚, 小西隆介, 佐藤孝治, 木原誠司, 盛合敏: Linux 用ログ構造化ファイルシステム nilfs の設計と実装, 情報処理学会研究報告, 2005-OS-99, Vol. 2005, No. 48, pp. 61–68 (2005).