NILFS: 世界を忘れないファイルシステム

久保 類　　田村 芳明　　天海 良治　　佐藤 孝治
小西 隆介　　木原 誠司　　盛合 敏

日本電信電話株式会社 NTT サイバースペース研究所

要 旨

ログ構造化ファイルシステム NILFS を使い，削除したファイルや編集前のファイルなど，過去のファイルを含めた検索システムを実現した．NILFS では過去のファイルを全て残すことが出来る特徴がある．しかし，全てを残すために，目的とする過去のファイルを探すことが非常に困難になる．そのため，NILFS と全文検索システムを組み合わせ，過去のファイルのキーワード検索を可能にした．併せて，過去のファイルを探すための手がかりになる情報を管理する，NILFS スケッチという機能を使った被験者実験を行い，システムの有用性や課題を抽出した．

NILFS: File system that never forgets the world

Rui Kubo　　Yoshiaki Tamura　　Yoshiji Amagai　　Koji Sato
Ryusuke Konishi　　Seiji Kihara　　Satoshi Moriai

NTT Cyber Space Laboratories, NTT Corporation

Abstract

In this paper, we describe our desktop search system which exploits the feature of NILFS. NILFS, which stands for the New Implementation of LFS, keeps all of the data written to the file system, and this feature enables us to find any files written in the past, including files deleted or modified. This feature, however, makes it difficult to find specific file in which one is interested because NILFS stores many versions of a file with a single name. In our desktop search system, we combined NILFS and a full text search system to solve this issue. Moreover, we introduced a new function called NILFS sketch, which recodes meta information, to provide a clue to find the searching file. Through out our experiments, we show the effectiveness of our system and future works to be done.

1　はじめに

1.1　背景

近年，多種多様で膨大な情報の蓄積や利用が出来るようになってきている．多種多様で膨大な情報とは，例えば，

- ソースファイルや論文原稿，プレゼン資料などのローカルファイル
- ニュースサイトや掲示板，ブログなどの時々刻々と書き換わる Web ページ
- 温度計やカメラなどから得られる実世界に関する情報

などである．

増え続ける情報に比べると人間の処理能力にはそれほど変化がない．そのため，人間と情報の間に立ち，人間（利用者）を積極的に支援する仕組みは非常に重要である．

1.2 ファイルシステム

我々は増え続ける情報に対しローカルファイルの観点から着目し，ファイルシステムの観点から課題に取り組んでいる．具体的には，Linux 用のローカルファイルシステムである NILFS[1] の開発を行っている [13, 11]．NILFS は，信頼性，可用性，復旧性，堅牢性などデータを保持するためにファイルシステムとして必要とされる機能に加え，近年の大容量ストレージを意識して，利用者を積極的に支援することを目指している．

NILFS は消さない（=忘れない）ファイルシステムである．削除したファイルや編集前のファイルなど，過去のファイルを全て残すことが出来る．これはデータを保持するという観点から見ると究極の手法であり，利用者の様々な負荷を軽減したり，ミスを救うことを可能にする．

究極の手法であるが故に，欠点もある．1 点目はストレージの容量は有限であるため，永久に消さないわけにはいかないこと，2 点目はファイル空間が時間軸方向に拡がるため，ファイル数が増大することである．1 点目については不要な過去ファイルを削除する，クリーナ機能の実装中である．本稿では 2 点目の，ファイル数の増大による過去ファイルを探すことの困難さを取り扱う．

増大する過去ファイルを容易に探すために，NILFS スケッチという時間軸を辿る手がかり情報を保持する機能を実装した．本稿では NILFS スケッチを使った過去のファイルを検索する仕組みと評価について述べる．

第 2 章で NILFS の概要と，NILFS を使った過去のファイルの検索システムについて述べる．次に第 3 章で NILFS スケッチについて述べ，第 4 章で NILFS スケッチを使った被験者実験について述べる．最後に第 5 章でまとめを行う．

[1] the New Implementation of a Log-Structured File System

2 NILFS

2.1 概要

ファイルシステムのさらなる信頼性，堅牢性の確保，そして，何より最新の大容量ストレージの恩恵を利用者の使いやすさやオペレータの操作負荷の軽減に活かすため，ログ構造化ファイルシステム (Log-Structured File System: LFS) に注目した．LFS はディスクブロックの書き込みを全てデータの追記とし，過去のデータを壊さない．

我々は，Linux 2.6 のカーネルモジュールとして，NILFS を開発し，第 1 版を GPL で公開している [4]．NILFS は，LFS の性質を活かしてディスク容量の全体を使用することで，過去に書き出したデータを最大限に保持し，あとから利用することを可能としている．このため，ハードウェア障害の発生や利用者が誤操作をした場合，事前のバックアップ取得やスナップショット取得指示をしていなくても，過去の首尾一貫したデータが多くの場合取り戻せる．

NILFS では，ディスク上のブロックを木構造 (B-Tree) で管理している．この木のルートは，スーパーブロックで管理する．データ書き出し処理は次のように進む．

1. 変更のあったファイルデータを集める
2. 変更のあったファイルブロック管理用のデータを集める
3. 集めたブロックに連続した新たなディスクブロック番号を割り当てる
4. この順でディスクに書き出す
5. スーパーブロックの木のルートを書き換える

ルートの書き換えが完了した段階で，ファイルシステム内容が首尾一貫した状態で更新される[2]．ブロックの書き出しの途中で電源切断などが発生しても，その前のルート書き換え時の状態は全てそのまま保持されている．ひとまとまりのブロックの書き出しをセグメント書き出しと呼ぶ．セグメントの先頭には，セグメントの内容表とデータのチェックサムがある．また，セグメントの最後のブロックには，木のルート情報が含まれる．これをチェックポイントブロックと呼び，このブロックが書き出されれば，ディスク上での首尾一貫性が維持出来る．

NILFS の公開版では，ディスクに保持している

[2] スーパーブロックは遅れてディスクに書き出される

過去のファイルシステムを読み込み専用でマウントし，その内容にアクセスする機能が実現されている．チェックポイントブロックでファイルシステムは首尾一貫しているので，過去は，このチェックポイントの単位でマウント出来る．なお，NILFS は現在の読み書き可能マウントと過去の複数の読み込み専用マウントを共存出来る．

過去の状態を全て残していると，ディスクの容量を使い切ったところで新たなデータを書き出すことは出来なくなる．よって，過去の状態のうち，利用者が指定したバージョンを残し，他の部分を再利用するためのクリーナが必要となる．利用者は必要なバージョンをチェックポイントの単位で指定出来る．残すよう指定されたチェックポイントを NILFS ではスナップショットと呼ぶ．クリーナは，現在，実装中である．

2.2 過去ファイルの検索

本稿ではファイルを検索するとは，利用者が目的とするファイルに辿り着くまでの過程を表す．後述する grep コマンドや全文検索システムは，利用者に対してファイルを絞り込む機能を提供するもので，検索の支援をする役割を果たす．

NILFS ではチェックポイント毎に全てのファイルを残しておくことが出来る．目的とする過去のファイルに辿り着くために，

1. チェックポイントのリストを表示する[3]

```
# listcp -l /dev/sda1
     1      8 Thu May 25 14:26:42 2006 ...
     9      4 Thu May 25 14:56:08 2006 ...
    13      8 Thu May 25 14:56:10 2006 ...
    21      8 Thu May 25 14:56:30 2006 ...
    29     10 Thu May 25 14:56:38 2006 ...
    ...
```

2. 時刻情報を元に，カラム一番左のチェックポイント番号を把握する
3. チェックポイント番号をパラメータにしてマウントする

```
# mount -t nilfs -r -o cp=13 \
  /dev/sda1 /mnt/nilfs-cp
```

4. マウント先に対して，grep や find コマンドなどを使って目的とするファイルを絞り込む

```
# grep -r 'keyword' /mnt/nilfs-cp
```

5. 絞り込んだ中から目的とするファイルを探す

というような手順を踏む必要がある．時刻（チェックポイント番号）が違うために必要なファイルが見つからない場合は，再度手順の頭に戻ってやり直す必要がある．

この方法は，目的とするファイルの時刻が明確でない場合に，ファイルを探す手間が大きいという問題がある．これを解決するために，NILFS と全文検索エンジンを組み合わせ，過去ファイル検索システムを実装した．既存のデスクトップ検索システム [1, 3] と同じようなユーザインタフェースで，過去のファイルを検索可能にする．

NILFS ではチェックポイント毎に同じパス名のファイルが複数存在しうる．パス名を基準にファイルを識別する既存の全文検索エンジンと組み合わせるために，次の 2 つの手法が考えられる．

1. 全文検索エンジンに手を加え，パス名とタイムスタンプでファイルを識別する
2. 全文検索エンジンに手を加えず，パス名にタイムスタンプを埋め込む

今回は実装を簡単にするために，後者の手法を採った．全文検索用のインデックス作成と検索結果の表示について詳細を説明する．

インデックスは次のように作成する．

1. 検索対象とするディレクトリへのファイルの追加や変更を監視する．Linux カーネルが提供するファイルシステムのイベント監視の仕組み (inotify) を使用した
2. 更新されたファイルのパス名とタイムスタンプを使って仮のパス名を生成する．例えば，次のファイルが

   `/mnt/nilfs/documents/prosym.tex`

   2006 年 11 月 20 日 11 時 22 分 33 秒に更新されたら，

   ```
   /mnt/nilfs-cp/2006.11.20.11.22.33/ \
   documents/prosym.tex
   ```

   という仮のパス名を生成する

[3] "listcp" は NILFS 付属のユーティリティプログラム

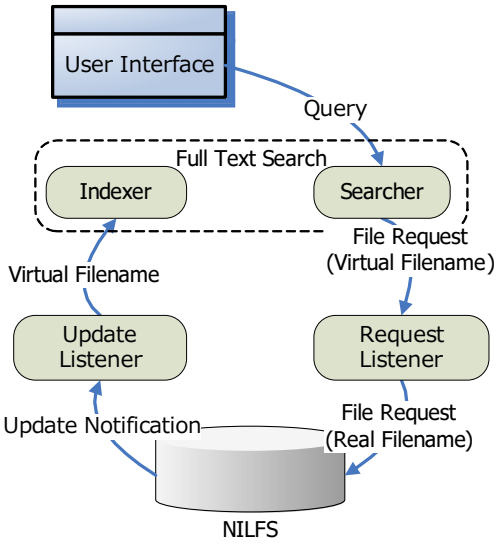

図 1: 過去ファイル検索システムの構成

3. 更新されたファイルのインデックスを作成する．その際，インデックスには仮のパス名を登録する

検索結果を表示する手順は次の通りである．

1. 仮のパス名へのアクセス要求を監視する
2. 仮のパス名から実際のファイル名とタイムスタンプを取り出す
3. タイムスタンプを元に対応するチェックポイントをマウントし，ファイルを取得する

アクセス要求の監視には，automount プログラムを使用した．automount のスクリプトで，仮のパス名からタイムスタンプを取り出し，チェックポイントをマウントする．

まとめると図 1 の通りになる．これらの仕組みにより，削除したファイルや編集前のファイルに含まれるキーワードを使った全文検索が可能になる．

2.3 過去ファイル検索の課題

過去ファイル検索システムにより，過去のファイルに辿り着く手間は軽減される．しかし，特に時間軸を辿るという観点から見ると，手間や困難さのさらなる軽減の余地があると考える．

膨大なファイルから目的とするファイルに辿り着くまでの過程を支援するために，全文検索エンジンのようなキーワード検索を使う以外に，次のようなアプローチがある．

- 対象とするファイルをソースコードなどに絞り込み，文法構造 [14] など対象に特化した知識を使う
- SpiralBrowser[12] や Time Machine[5] の様に 3 次元表示などを使って表示を工夫する
- CVS のコミットログ [7] など時刻に関するタグ情報を使う

時間軸を辿るという観点から，時刻に関するタグ情報を使うアプローチが有効であると考える．ファイルシステムでは任意のファイルの書き込みのタイミングを知ることが出来る．つまり，時刻に関連する何らかの付加情報をファイル書き込みに併せて保存することが可能で，この情報を過去のファイルに辿り着くための手がかりにすることが出来る．

我々は，このような時刻を特徴付ける情報を保持するための機能，NILFS スケッチを実装した．

3 NILFS スケッチ

3.1 概要

利用者の過去へのナビゲーションのため，時間軸へのタグ付け機能を実装した．NILFS スケッチと名付けたこの機能は，次の特徴がある．

1. チェックポイントに 4000 バイト程度のユーザデータ（スケッチデータ）の保持が可能
2. ファイルシステムは，スケッチデータの内容には関知しない
3. スケッチデータには，マウントポイントのトップディレクトリにあるファイル .sketch を通じて，通常のファイルの読み書きとしてアクセス出来る

ファイルシステムの状態が更新されセグメントが書き出される時，チェックポイントに .sketch の内容がコピーされ，.sketch の内容はクリアされる．.sketch へのデータの書き込みだけではセグメントの書き出しは発生しない．そのため，.sketch に複数回の書き出しを行った場合に，セグメントに書き出される .sketch のデータは最新のみである．

このような仕組みにより，利用者が時刻を特徴付けるデータを随時 .sketch に書き込むことで，各々のチェックポイントのタイミングの .sketch だけが書き出されることになる．利用者はチェックポイントが作成されるタイミングを意識する必要がない．

3.2 世界を忘れない

センサなどの様々なデバイスが身の回りに遍在しつつある [6, 9]．身の回りに関する情報は，時刻や記憶の想起と相性が良いと考えられ，NILFS スケッチに保存する情報として使わない手はない．例えば，次のような情報が有用ではないかと考えられる．

- マイク…利用者の叫び声や周辺の騒音
- カメラ…利用者の顔や服装
- 温度計…周辺の温度やラック内の温度
- 位置…利用者の座ったイスや机の場所
- 無線タグ…周辺にある物や人の一覧
- 生体センサ…利用者の健康状態
- インターネット…ニュースサイトのヘッドラインや天気の情報，Web の閲覧履歴，送受信したメールの件名

NILFS スケッチにこのような実世界に関する情報を保存することで，ローカルファイルだけでなく，世界を忘れないファイルシステムを実現する．

4 評価実験

4.1 実験条件

NILFS スケッチに書き込んだ情報について，目的とする過去のファイルに辿り着くための有効性を被験者実験を通して評価した．

具体的な実験手順は次の通りである．

1. NILFS スケッチにいくつかの情報を書き込む
2. NILFS スケッチを表示する機能を備えた過去ファイル検索システムを使い，下記の被験者グループ毎で，目的のファイルに辿り着く時間を比較する
   - NILFS スケッチを表示するグループ
   - NILFS スケッチを表示しないグループ

NILFS スケッチには次の情報を書き込んだ．

カメラ 部屋の天井四隅に備え付けのカメラで机やプロジェクタの投影先を映し，画像の輝度に変化があったタイミングの jpg 画像

マイク 机に置いたマイクから声のパワーが大きくなったタイミングを検出し，「うるさい」など机の周りの状況を表す文字列

傾きセンサ ペットボトルに無線センサ MOTE[8] を付与し，ペットボトルの傾きから，ペットボトルを飲んでいるタイミングを検出し [10]，「飲んでいる」などのペットボトルに関する文字列

NILFS スケッチを表示する機能を備えた過去ファイル検索システムの画面を図 2 に示す．検索結果に表示されるファイル毎のタイムスタンプに併せて，NILFS スケッチに保存されている画像や文字列を表示する．利用者は画像や文字列を参考に，目的とするファイルを探すことが出来る．例えば，次のような例が考えられる．

図 2: 過去ファイル検索システムの NILFS スケッチ表示例

- イスから立ち上がった時に保存したファイルに併せて，そのタイミングの画像が表示される
- 打ち合わせの最中に保存したファイルに併せて，打ち合わせ中の画像が表示される
- 打ち合わせ中の盛り上がりなどで，笑い声が起きた時に保存したファイルに併せて，「うるさい」と文字列が表示される
- 一服しようとペットボトルを飲んだ時に保存したファイルに併せて，「飲んでいる」と文字列が表示される

全文検索エンジンには FreyaSX[2] を使用した．

実験のために，部屋の中央の机で本原稿の執筆を 5 日間行った．期間中，執筆の隣で打ち合わせが 4 回行われた．打ち合わせ中は部屋の角で執筆を行った．期間中の詳細は表 1 のタイムテーブルの通りである．

20〜40 代の被験者 6 名で，次のタイミングで保存された原稿ファイルを探すタスクを設定した．

タスク 1 打ち合わせ B と打ち合わせ C の間の原稿ファイル

タスク 2 打ち合わせ B の始まる時の原稿ファイル

タスク 3 打ち合わせ A の終わった時の原稿ファイル

表 1: タイムテーブル

日付	時間	作業内容
11 月 15 日	15:00〜16:00	打ち合わせ A
	15:30〜16:00	原稿執筆
11 月 17 日	15:00〜19:30	原稿執筆
11 月 20 日	15:30〜19:00	原稿執筆
11 月 21 日	16:00〜20:00	原稿執筆
11 月 22 日	13:00〜14:30	打ち合わせ B
	14:30〜15:30	打ち合わせ C
	13:00〜15:30	原稿執筆
	15:30〜16:30	打ち合わせ D

なお，被験者には過去のファイル検索システムの使い方と，表 1 のタイムテーブルを事前に説明した．

4.2 実験結果

被験者毎の原稿ファイルに辿り着くまでの時間は図 3 の通りである．NILFS スケッチの表示にかかる時間は全体から見れば無視できる範囲である．そのため，NILFS スケッチの表示の有無で，原稿ファイルに辿り着くまでの時間に大きな差が無いことが分かる．これは，比較実験を行うために，NILFS スケッチの表示が無い場合であっても，タイムスタンプを手がかりにファイルを探せるタスクを設定した

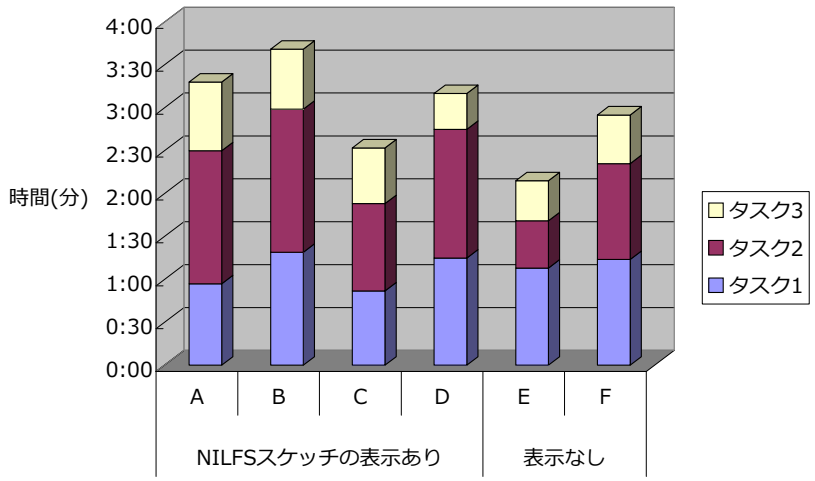

図 3: 被験者実験の結果

ことが原因であると考えられる．

今回の実験で NILFS スケッチへ書き込んだような，マイクやカメラからの情報が有用な状況もあると考える．例えば，タスク 3 では打ち合わせ終了時に更新された原稿ファイルを指示している．タイムテーブルでは打ち合わせは 16:00 に終了していることになっているが，NILFS スケッチに書き込まれている画像を確認することで，16:00 ちょうどに終了していないことが分かる．時刻情報では 16:00 としか把握できないが，画像では打ち合わせの終了のタイミングを正確に把握することが可能であった．

また，日付や午前・午後など大きな時間スケールで検索結果を絞り込む場合にも有用であると考えられる．タスクで指定される打ち合わせなどの作業内容から，タイムテーブルを参照することなく，部屋の画像を確認することで大きな時間スケールでの絞り込みが出来るためである．例えば，似た画像のクラスタリングや階層化を行い，時系列で並べて提示し，ファイル検索結果を絞り込むような機能が有用ではないかと考えられる．

このように，今回対象としたような時刻で探せるようなファイルの場合は，NILFS スケッチの表示方法についてさらなる検討の必要がある．

また，今回の実験ではタスク達成時間を評価の基準に用いたが，被験者にかかる精神的なストレスなども評価の基準として考えられる．

5　おわりに

5.1　まとめ

過去のファイルを全て残すことが出来る NILFS を使った，過去のファイルの検索システムについて述べた．過去のファイルを容易に探せることを目指し，ファイル書き込み時の時刻を特徴付ける情報を管理するための，NILFS スケッチについて述べた．過去ファイルの検索システムに，NILFS スケッチを表示する機能を実装し，被験者実験を通して，NILFS スケッチの表示方法について，さらなる検討の余地があることが分かった．

5.2　今後の予定

NILFS スケッチに様々な情報を書き込み，その有効性を検証したい．位置センサから得られるノートパソコンや利用者の場所や，脈拍など生体センサから得られる利用者の状態などを考えている．

また，今回の実験で NILFS スケッチへ保存した部屋や利用者の画像は，細かい時刻を探す場合よりも，全体を俯瞰するような場合に有効であると考えられ，適した表示方法についても検討する必要がある．

参考文献

[1] Beagle http://beagle-project.org/ .

[2] FreyaSX http://www.delegate.org/freyasx/ .

[3] Google Desktop http://desktop.google.com/ .

[4] NILFS http://www.nilfs.org/ .

[5] Time Machine http://www.apple.com/macosx/leopard/timemachine.html .

[6] Beigl, M., Krohn, A., Zimmer, T. and Decker, C.: Typical Sensors needed in Ubiquitous and Pervasive Computing, in *First International Workshop on Networked Sensing Systems*, pp. 153–158 (2004).

[7] Chen, A., Chou, E., Wong, J., Yao, A. Y., Zhang, Q., Zhang, S. and Michail, A.: CVSSearch: Searching through Source Code Using CVS Comments, in *International Conference on Software Maintenance*, pp. 364–373 (2001).

[8] Levis, P., Madden, S., Gay, D., Polastre, J., Szewczyk, R., Woo, A., Brewer, E. and Culler, D.: The Emergence of Networking Abstractions and Techniques in TinyOS, in *Proceedings of the First USENIX/ACM Symposium on Networked Systems Design and Implementation*, pp. 1–14 (2004).

[9] Schilit, B. N. and Sengupta, U.: Device Ensembles, *IEEE Computer*, Vol. 37, No. 12, pp. 56–64 (2004).

[10] 久保類, 真鍋義文, 盛合敏 : センサネットワーク環境における情報検索プラットフォームの提案, 情処学研報, 2006-DPS-126 (2006).

[11] 小西隆介, 佐藤孝治, 天海良治, 木原誠司, 盛合敏 : ログ構造化ファイルシステムの同期書き込み性能の最適化, 情処学研報, 2006-OS-103 (2006).

[12] 小池英樹, 赤井一章 : SpiralBrowser: 時間軸検索を支援するファイル検索インタフェースの開発, 日本ソフトウェア科学会第 14 回大会論文集, pp. 121–124 (1997).

[13] 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, 情処学研報, 2005-OS-099 (2005).

[14] 田原靖太, 松下誠, 井上克郎 : 既存ソフトウェアの変更履歴を利用したソースコード修正支援手法の提案, 情処学研報, 2001-SE-136 (2002).