M-021

仮想フォルダ生成システム改善に向けたファイルアクセス履歴の詳細化
Acquiring precise file access history toward improvement of virtual folder system

向原 大貴[1)]　乃村 能成[2)]
Daiki Mukohara　Yoshinari Nomura

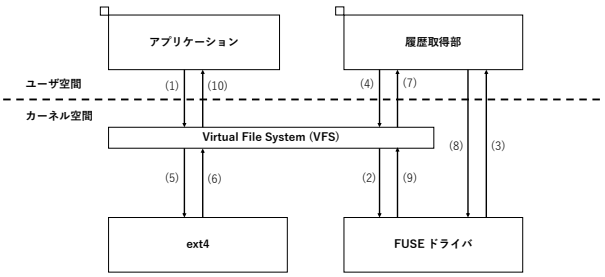

図 1　FUSE を用いた際のアプリケーションとファイルシステムの通信

1. はじめに

計算機のファイルアクセス履歴から計算機内のワーキングディレクトリ (以降，WD) を推定することで，利用すべきファイルの提示や作業に合わせたフォルダ構造の動的な変更を行う仮想フォルダ生成システム [1][2] が提案されている．当システムでは `fswatch`[3] を用いてファイルアクセス履歴を収集している．ここで，`fswatch` はファイルを更新履歴は取得可能だが，読込のみ (以降，参照) を行った履歴やミリ秒単位の履歴を取得できない．しかし，そのような詳細なファイルアクセス履歴を使用することで WD の推定精度向上が期待できる．本稿では，`FUSE` [4] を用いて詳細なファイルアクセス履歴を取得する手法を提案する．`FUSE` とは Unix 系 OS において，ファイルシステムをユーザ空間で作成する仕組みである．また，取得した詳細なファイルアクセス履歴を WD 推定に利用する方法を示す．

2. 詳細なファイルアクセス履歴の取得

2.1 詳細なファイルアクセス履歴の取得手法

本節では，`FUSE` を用いた詳細なファイルアクセス履歴の取得手法を示す．`FUSE` とは，Unix 系 OS においてユーザ空間で独自のファイルシステムを作成するための仕組みである. **図 1** に，`FUSE` を用いた際のアプリケーションとファイルシステムにおける通信の流れを示す．`FUSE` を用いない場合のアプリケーションとファイルシステムの通信の流れは，(1)(5)(6)(10) となる．また，図 1 において履歴取得部とは `FUSE` を用いて作成するファイルシステムである．

通常，`FUSE` で作成したファイルシステムは，指定のディレクトリにおける操作を，作成したファイルシステムをマウントした別のディレクトリ上で行うなどの用途で使用する．しかし，本手法ではそのような独自の機能を有したファイルシステムは作成しない．図 1 の (1)〜(3) において，アプリケーションがファイル操作を行った場合に `VFS` が履歴取得部にアクセスする．この際，

1）　岡山大学大学院自然科学研究科，Graduate School of Natural Science and Technology, Okayama University
2）　岡山大学学術研究院自然科学学域，Faculty of Natural Science and Technology, Okayama University

表 1　作成したファイルシステムが記録可能なファイル操作システムコール全 30 種類の内 5 種類とその処理内容

通番	システムコール名	処理内容
1	create	ファイルの作成とオープン
2	open	ファイルのオープン
3	read	オープンしているファイルからのデータ読み込み
4	release	オープンしているファイルの解放
5	write	オープンしているファイルへのデータ書き込み

履歴取得部は受け取ったファイル操作システムコールをファイルアクセス履歴として記録する．次に (4)〜(7) において，履歴取得部は元々マウントされているファイルシステム (図では EXT4) にシステムコールに対する処理を依頼し結果を受け取る．最後に (8)〜(10) において，履歴取得部は処理結果をアプリケーションへ返す．

2.2 ファイルアクセス履歴の形式

出力するファイルアクセス履歴の形式は CSV 形式とし，時刻，inode 番号，イベント，ファイルパスの 4 項目を持つ．リスト 1 にファイルアクセス履歴の例を示す．また，以下に各要素の説明を示す．

リスト 1　詳細なファイルアクセス履歴

```
2021-11-30T14:25:18.823560,171333,read,/home/
  mukohara.can/git/polaris/Pipfile
2021-11-30T14:25:18.823759,171333,getattr,/home/
  mukohara.can/git/polaris/Pipfile
2021-11-30T14:25:18.823977,171333,release,/home/
  mukohara.can/git/polaris/Pipfile
:
```

(1) 時刻
ファイル操作が行われた時刻を示し，フォーマットは ISO 8601 である．時刻の形式は 10^{-6} 秒とする．

(2) inode 番号
ファイル操作が行われたファイルの inode 番号を示す．inode 番号はファイルの同一性保持に利用する．

(3) イベント
ファイルシステムが受け取ったファイル操作システムコールを示す．

(4) ファイルパス
ファイル操作が行われたファイルのパスを示す．

表 1 に作成したファイルシステムが記録可能なファイル操作システムコールの一部を示す．表 1 に示したシステムコールは，3.2 節に示す手法において使用しているシステムコールである．記録可能なファイル操作システムコールは表 1 に示すものを含む全 30 種類存在する．

2.3　性能評価と考察

通常，ファイルシステムはカーネル空間に存在するため，処理はカーネル空間内で完結する．それに対して，作成したファイルシステムはユーザ空間に存在するた

表 2　評価に使用した計算機とディスクの詳細

項目	詳細
OS	Ubuntu 20.04.3 LTS (Focal Fossa)
CPU	AMD Ryzen 9 5950X 16-Core Processor
メモリ	64GB
SSD	CSSD-M2B1TPG3VND, M.2-2280 (NVMe)

表 3　SSD における 3 方式の処理速度の評価結果 (MB/s)

	seq read	seq write	rand read	rand write
bare	2610	1890	664	1729
fswatch	1308	1065	535	997
fuse	92	92	96	86

め，システムコールが発行される度にカーネル空間と通信を行う．したがって，作成したファイルシステムはカーネル空間で動作するファイルシステムと比較して処理速度が劣ると考えられる．そこで，作成したファイルシステムの処理速度を評価し，オーバヘッドの大きさを確認する．

評価方法としては，bare (作成したファイルシステムをマウントしない方式)，fswatch (`fswatch` を使用する方式)，fuse (作成したファイルシステムをマウントした方式) の 3 方式における sequential read，sequential write，random read，random write の性能を測定し比較を行うことで相対的に評価する．評価値は各項目 30 回測定を行った平均を結果の値とする．また本評価は**表 2** に示す環境で行う．

評価結果を**表 3** に示す．以下に評価結果を受けた考察を示す．

fswatch の各項目における処理速度が fuse に比べてより bare に近いことは，fswatch が `inotify`[5] を用いた実装であるためと考えられる．`inotify` はシステムコール群であるため，ファイル操作に際する処理はカーネル内のファイルシステムで完結する．また，fuse の処理速度はやはり他の方式と比べ非常に小さい．しかし，計算機上でのコーディングや資料作成などの作業においては特に差を感じなかった．これは，前述の作業などにおいて差を感じるほど大きなファイルを扱うことが無いためと考えられる．

3. 仮想フォルダ生成システムへの導入

3.1 詳細なファイルアクセス履歴の利用

WD の推定にはファイルの作成，更新の履歴を使用している．ここで，ファイルの参照は計算機を用いた作業において頻繁に行われる行為である．そのため，ファイルの参照履歴を用いて WD の推定を行いたい．しかし，仮想フォルダ生成システムにおける履歴取得部はファイルの参照履歴を取得できない．そこで，作成したファイルシステムを用いて詳細なファイルアクセス履歴を取得する．そして，この履歴からファイルの作成，更新および参照履歴を利用し，WD の推定を行う．

3.2 抽象度の高いファイルアクセス履歴の取得手法

取得可能な詳細なファイルアクセス履歴のイベントは，30 種類存在する．これらのイベントは，ファイル作成などの操作と一対一で対応していない．例えば，

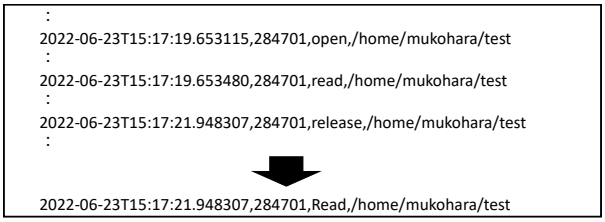

図 2　詳細なファイルアクセス履歴をもとにファイルの参照履歴を判別する流れ

`less` コマンドを用いてファイルを参照した場合発行されるシステムコールは複数存在するため，作成したファイルに関する履歴は複数出力される．したがって，3.1 節で示した履歴を取得したい場合は，出力される複数の履歴から行われたファイル操作を判別する必要がある．

ファイルの参照履歴を判別する流れを**図 2** に示す．ファイルの参照を，`less` コマンド，`cat` コマンド，`tail` コマンド，`vim`，`emacs` を用いて行った．その結果得られたそれぞれの履歴には，操作対象のファイルに対して `open`，`read`，`release` の順番で操作を行った履歴が出力されるという共通点が確認できた．したがって，あるファイルに対して，`open`，`read`，`release` の順番で操作を行った履歴が出力されると，そのファイルを参照したと判別できる．

抽象度の高いファイルアクセス履歴を得るためには，上記のような判別をする必要がある．しかしこれは，取得した詳細なファイルアクセス履歴を用いることで，ファイルの作成のようなファイル操作よりも具体的な操作の履歴を取得可能ということである．例えば，inode 番号 A のファイルをコピーして inode 番号 B のファイルを作成したという履歴の判別を考える．この操作によって出力される詳細なファイルアクセス履歴は，A を `open`，B を `create`，A を `read`，B を `write` という並びの履歴を含む．この様な履歴が出力されたら，A と B の ファイルサイズを確認し，等しければ A をコピーして B を作成したと判断可能と考える．この履歴を用いることで，ファイルの派生元を再帰的に辿ることができる．したがって，同じ派生元を持つ類似したファイルを集めた仮想フォルダの生成が可能と考える．

4. おわりに

`FUSE` を用いた詳細なファイルアクセス履歴の取得方法を示した．また，取得した履歴の仮想フォルダ生成システムにおける利用方法を示した．残された課題は，参照履歴を用いたより最適な WD 推定手法の提案，改変した仮想フォルダ生成システムの評価である．

参考文献

[1] nomlab: polaris, GitHub (online), available from ⟨https://github.com/nomlab/polaris⟩ (accessed 2022-06-23).
[2] 西　良太，乃村能成：作業を代表するフォルダの推定による仮想フォルダ生成システム，情報処理学会論文誌，Vol. 62, No. 2, pp. 1–11 (2021 年 2 月掲載).
[3] E.M.C., A. D.: fswatch, GitHub (online), available from ⟨https://github.com/emcrisostomo/fswatch⟩ (accessed 2022-06-23).
[4] Nikolaus Rath and Miklos Szeredi: libfuse, GitHub (online), available from ⟨https://github.com/libfuse/libfuse⟩ (accessed 2022-06-23).
[5] John McCutchan: INOTIFY, The Linux man-pages project (online), available from ⟨https://linuxjm.osdn.jp/html/LDP_man-pages/man7/inotify.7.html⟩ (accessed 2022-06-14).