L-023

放送時刻を使ってフレームデータにアクセス可能な分散ファイルシステム

Distributed file system for enabling to access frame based data using on-air time

金子 豊† 黄 民錫† 竹内 真也† 和泉 吉則†
Yutaka Kaneko Minsok Hwang Shinya Takeuchi Yoshinori Izumi

1．はじめに

放送コンテンツをインターネットなどの通信環境を使って提供するサービスが増えている．そのため，今後の放送システムでは，サービスに合わせて柔軟に放送コンテンツを提供できることが必要となる．また，様々なサービスに対応するため，字幕や副音声などの関連データを後から追加できるよう，様々なデータを放送コンテンツに関連付けて管理できることが要求される．

スケールアウト型の分散ファイルシステムは，必要に応じて容量を拡張でき，増え続ける大容量の放送コンテンツを永続的に保管するのに有効である．しかし，従来のディレクトリによるファイル管理だけでは，関連データを含めて増え続ける放送コンテンツを管理するには十分ではなく，新たな管理方法が必要である．

文献[1]では，分散ファイルシステムに保管したファイルを放送時刻などの連続する ID にマッピングすることで，仮想的な 1 つのファイルとして利用できる管理手法を提案した．本稿では，この提案手法を開発中の分散ファイルシステムに実装したのでその概要を述べる．また，放送番組ファイルを用いた性能評価結果について述べる．

2．分散ファイルシステムの概要

開発中の分散ファイルシステム（以下，分散 FS）は，複数のノードで構成され，ノードの参加・離脱を参加ノードで構成する構造型 P2P で管理している[2]．ノードはファイルを保管するストレージノードと，それ以外のタップノードに分類される．タップノードは，クライアントが分散 FS にアクセスするためのアクセスノード，ログを保存するためのログノードなど，ファイルの保管以外の個別の機能を実行するノードである．

ユーザが分散 FS に読み書きするファイルは，ストレージノードに保管される．内部ではすべて UMID[3]をファイル名とするユニークなファイル名で保管している．ファイルの保管先ノードは，ストレージノードで構成した分散ハッシュテーブルによるキーバリューストアで検索できる．

ユーザから見えるファイル名やディレクトリ名は，図 1 に示すように，i ノードファイルとディレクトリファイルにより分散管理している[4]．これらのファイルも内部的には，通常ファイルと同様に，UMID のファイル名を付けて保管されている．

分散 FS は Linux のアプリケーションとして実装しており，ユーザは分散 FS へのアクセス機能を持つクライアントを組み込んだ FUSE[5]を用いることで，分散 FS を通常のディスクとしてマウントできる．

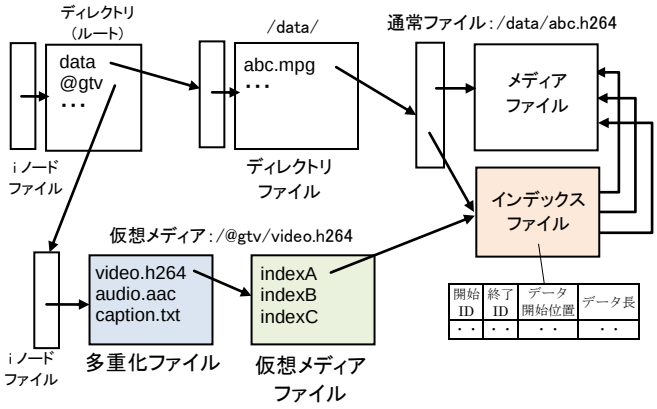

図 1　ディレクトリと仮想メディアの管理構造

†日本放送協会 放送技術研究所, NHK

3. 放送時刻 ID へのファイルのマッピング

3.1　仮想メディア

映像などのフレーム構造のデータを格納したファイルをここではメディアファイルと呼ぶ．字幕データなどのテキストデータもメディアファイルとして扱う．フレームをどのように定義するかは，あらかじめメディアファイルの種類毎にユーザが決める．提案手法の目的は，分散 FS に保管された複数のメディアファイルのフレームデータを連続する ID にマッピングすることで，仮想的な連続するファイルとして扱えるようにすることである．この仮想的なファイルを，ここでは仮想メディアと呼ぶ．マッピングとは，メディアファイル内の 1 フレーム分のデータを開始 ID から終了 ID までの範囲に対応付けることである．

3.2　仮想メディアの管理構造

仮想メディアは，図 1 に示すように，多重化ファイル，仮想メディアファイル，インデックスファイルの 3 層構造のファイルで管理する．多重化ファイルには同一の連続する ID にマッピングされた仮想メディアの構成を格納する．仮想メディアファイルには，1 つの仮想メディアに対してマッピングされたインデックスファイルのファイル名を格納する．インデックスファイルには，フレームデータのマッピング情報であるインデックス情報を格納する．インデックス情報は，マッピングの開始 ID，終了 ID，対応するメディアファイルの先頭からの位置，フレームデータ長からなる 32byte のデータである．仮想メディアを管理するこれらのファイルは，分散 FS 内では他のファイルと同様に UMID のファイル名を付けて保管する．

仮想メディアは，多重化ファイル毎のディレクトリ内のファイルのように，"/@ID 名/仮想メディア名"というファイルパス名で，ユーザからは通常のファイルとして見える．

3.3　放送時刻 ID

連続 ID としては，ここでは放送時刻 ID を用いる．放送時刻 ID は，日付を表す MJD(20bit)，タイムゾーン(6bit)，毎 0 時を 0 とした 90KHz のカウンタ値(33bit)を含む 64bit 値である．

表 1 pread(fd, buf, count, offset)システムコールの引数

引数	通常ファイルの場合	仮想メディアの場合
fd	ファイルディスクリプタ	← (同じ)
buf	バッファ	← (同じ)
count	読み出しサイズ	buf のサイズ
offset	ファイルの先頭からのオフセット	所望の放送時刻 ID

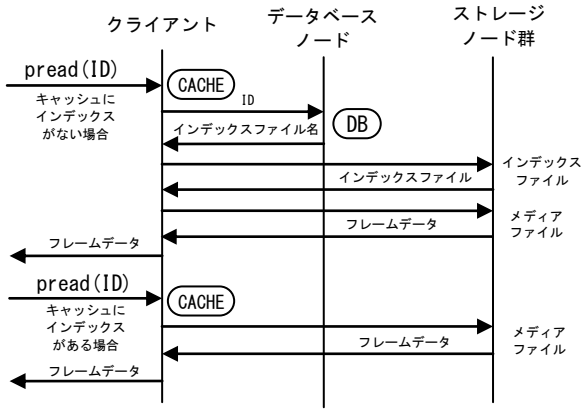

図 2 フレームデータの取得シーケンス

3.4 アクセス関数

仮想メディアの作成には専用の API を用いる．一方，仮想メディアの読み出しには，通常ファイルと同様 pread システムコール(または，lseek と read の併用)を用いる．但し，表 1 に示すように，仮想メディアへのアクセスの場合は引数の意味が異なる．仮想メディアを pread で読み出すと，放送時刻 ID に対応する 1 フレーム分の可変長データが buf に格納される．buf の先頭 32 バイトにはインデックス情報が格納される．この情報から，データ長，マッピングされている開始 ID と終了 ID が分かる．前のフレームで取得した終了 ID を，次の読み出しの ID とすることで，連続するフレームデータを取得することができる．

3.5 データベースノード

仮想メディアへのアクセスを高速化するため，タップノードとしてデータベースノードを実装した．データベースノードは仮想メディアの全インデックスファイルを読み込み，放送時刻 ID からインデックスファイル名を検索するためのデータベースを構築する．

図 2 にユーザが pread を呼び出したときの動作シーケンスを示す．クライアントはインデックスファイルをキャッシュし，連続した ID へアクセスされた場合には，データベースノードへの問い合わせをしないことで，フレームベースの高速なファイルアクセスを可能とした．

4. 実験結果と考察

19 台のストレージノードで構成した分散 FS に保存している番組ファイルを使って実験を行った．1 時間ごとのファイルとして保管している NHK 総合テレビの 2010 年 10 月から 2012 年 4 月の映像，音声，字幕のメディアファイルをそれぞれ，映像，音声，字幕の仮想メディアとしてマッピングした．

2012 年 4 月 20 日の 24 時間分の映像，音声，字幕の仮想メディアを同期して読み出したときの転送時間と速度を測定した．実験対象のメディアファイルの諸元を表 2 に，測定結果を表 3 に示す．測定結果は 10 回の測定の平均値で

表 2 24 時間分(2012/4/20)のメディアファイルの諸元

		映像	音声	字幕
フォーマット		H.264 ES (NAL)	AAC ES (ADTS)	テキスト (1 画面を 1 行)
フレームレート (frame/sec)		29.97 (30000/1001)	46.875 (48000/1024)	平均 7.5 frame/min
平均再生レート(bps)		2.3M	177K	85
フレーム数		2,589,339	4,049,818	10,840
フレームサイズ(byte)	最小	52	13	35
	最大	512214	1517	149

表 3 24 時間分の仮想メディアの読み込み速度

		映像	音声	字幕	合計
転送時間 (sec)	平均	389	100	69	559
	標準偏差	29	3	25	39
転送速度 (frame/sec)	平均	6682	40207	170	
	標準偏差	476	1287	38	
転送速度 (Mbps)	平均	506	152	0.115	
	標準偏差.	36	4	0.02	

表 4 ランダムアクセス時の 1 フレームの読み込み時間(秒)

	映像	音声	字幕
平均	0.299	0.449	0.067
標準偏差	0.584	1.000	0.126
最小	0.021	0.020	0.00005
最大	8.8	9.0	0.873

ある．24 時間の全フレームの読み出し合計時間は 559 秒であった．これはリアルタイムの再生時間の 154 倍の速度であり，リアルタイム再生の用途としては十分な速度といえる．字幕データの取得は，映像，音声に比較して遅いが，これは，字幕のフレームデータは，マッピング先の時刻 ID が連続していないことが多く，ID が連続していない場合，クライアントで毎回データベースノードへの問い合わせが発生するためである．

次に，仮想メディアをランダムアクセスしたときの速度性能を調べるため，2012 年 4 月の 1 ヶ月分の仮想メディアに対して，1 時間おきのフレームデータを取得したときの時間を測定した．結果を表 4 に示す．映像に比べ音声の方が悪い結果となった．これは，音声は映像よりもフレーム数が多くインデックスファイルも大きいため，クライアントがインデックスファイルを取得するまでの時間が長いためと考えられる．また，今回の実験では 1 フレームを取得するのに，映像，音声ともに 9 秒程度かかってしまう場合があった．この原因に関しては調査中である．

5. まとめ

放送システムで放送コンテンツを管理することを目的に，放送時刻からフレーム単位で放送コンテンツのデータを取得することが可能な分散ファイルシステムを提案し，実装による実験結果を示した．今後は，複数のユーザからアクセスした場合の性能評価を行う予定である．

参考文献

[1] 金子, 黄, 竹内, 和泉, "分散ファイルシステムにおける連続 ID を使ったファイル管理手法の提案", 映情学年大, 11-1, 2011
[2] 金子, 竹内, 南, 和泉, "OneHop-P2P 拡張方式の実装方法と性能評価", 信学技報, NS2008-52, pp.57-62, 2008
[3] SMPTE 330M-2004, “Unique Material Identifier(UMID)”
[4] 金子, 黄, 竹内, 和泉, "構造型 P2P を使った分散ファイルシステムにおける分散ディレクトリ管理手法", FIT2009, L-033, pp.213-216, 2009
[5] FUSE: Filesystem in Userspace, http://fuse.sourceforge.net/