2E-2

放送時刻でアクセス可能な放送コンテンツのアーカイブシステムの試作

金子　豊　　黄　民錫　　竹内　真也　　砂崎　俊二

日本放送協会　放送技術研究所

1. はじめに

放送メディアは編成というタイムテーブルに基づき，一定量の放送コンテンツを視聴者に提供する特徴がある．毎日，毎週の決まった時刻に放送される番組が，生活習慣に密着している視聴者も多い．放送時刻は番組名と共に放送番組の特定に重要な役割を持つ．最近では放送後の番組は，VOD(Video On Demand)などの通信環境を使ったサービスで視聴されるが，現在の通信系サービスでは，番組単位での視聴が基本であり，放送時刻を意識して視聴されることは少ない．今後，通信系でタイムシフト視聴サービスが可能になれば，番組単位の視聴だけでなく，放送時刻方向にザッピングしながらの視聴[1]などが可能になる．また，放送時刻を意識して過去の番組を視聴することで，今後放送される番組への視聴につなげることができるものと考えられる．

本稿ではタイムシフトサービスを行う放送システムでの利用を想定し，放送コンテンツを時系列に管理，閲覧できるアーカイブシステムを試作したので報告する．

2. システム要件

試作したアーカイブシステムの主な要件を以下に示す．

① 永続的に放送コンテンツ（映像，音声，字幕など）を蓄積できる．
② 放送した放送コンテンツを時系列に閲覧できる．
③ 複数の放送コンテンツを同期して利用できる．

3. システム構成

試作したアーカイブシステムの構成を図１に示す．本システムでは，放送コンテンツのファイルを分散ファイルシステムに保管する．保管したファイル内のデータは，時刻情報でアクセスできる仮想的なファイル（以下，仮想メディア）にマッピングすることができる．

3.1　分散ファイルシステム

分散ファイルシステムは複数のノードから構成され，互いに参加・離脱を監視・通知する[2]．ノードには，ファイルを保管するストレージノード，分散ファイルシステムにアクセスするためのアクセスノード，仮想メディアへのマッピング情報を管理するデータベースノードなどがある．利用者の作成したファイルやマッピング情報は，ストレージノードのいずれかに保管し，その保管場所はストレージノードで構成する DHT(Distributed Hash Table)による KVS(Key Value Store)で分散管理する[3]．ストレージノードを必要に応じて追加することで，分散ファイルシステムの総容量を増やすことができる．

3.2　仮想メディア

仮想メディアは，日時情報を含むタイムスタンプ(以下TST(Time Stamp on Timeline))の時間軸上にマッピングされたスパースな仮想的なファイルである．TST は 90KHz の時間精度を持つ．図１に示したように，実ファイル内のデータ（ここではフレームと呼ぶ）が，仮想メディアの開始 TST から終了 TST にマッピングされる．

3.3　アクセスポイント

試作システムに保管されたファイルには，図１に示した３方法(AP1～AP3)でアクセスすることができる．

AP1：分散ファイルシステムをディスクとしてマウントする方法．AP1 では通常のディスクと同等にファイルの読み書きができる．仮想メディアへのアクセスは，通常のファイルと同じ pread システムコールを使う．引数として与えるオフセット値には TST を用い，その時刻へマッピングされている可変長のフレームデータが読み出される[4]．フレームサイズ，マッピングされている時刻範囲は，読み出されたデータの先頭に書き込まれている．実装には FUSE[5]を用いた．

AP2：WebSocket Protocol(RFC6455)を使って仮想メディアへアクセスする方法．AP2 が提供するインタフェースは，仮想メディアを読み出すために必要なシステムコール(open, close, pread, opendir, closedir, readdir)をほぼそのまま提供する構造とした．クライアントはサーバと WebSocket のコネクションを張り，AP1 と同様，１フレーム毎に TST を要求し，フレームデータを取得する．サーバの実装には，Node.js と RFC6455 に準拠するように改修した node-websocket-server[6]を用いた．

AP3：１番組分などの番組コンテンツをまとめてファイルとして HTTP サーバを介して取得する方法．クライアント側からメディア名，開始日時，継続時間を要求すると，サーバは該当時刻の映像および音声の仮想メディアを AP1 を使って読み出し，MP4 ファイルを作成する．

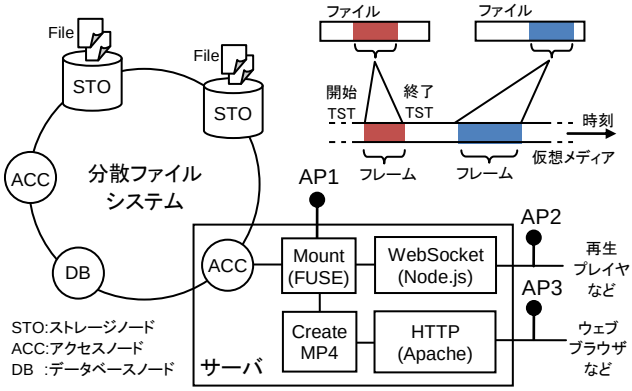

図１　アーカイブシステムの構成

4. 試作システム

分散ファイルシステムの実装には Linux，各ノード間の接続には 1Gbps のイーサネットを用いた．ファイルの保管対象を放送番組とし，映像，音声，字幕データは１

An Experimental Broadcasting Contents Archive System Enabling to Access using On-air Time
Yutaka Kaneko, Minsok Hwang, Shinya Takeuchi and Shunji Sunasaki
Japan Broadcasting Corporation (NHK)

時間毎，番組情報は１日毎のファイルとして保管した．映像は放送時の MPEG2 を H.264 に変換した．保管したファイルはそれぞれ仮想メディアにフレーム単位にマッピングする．映像，音声ファイルの諸元を表１に示す．

分散ファイルシステムの全容量と使用量の２年間の経過を図２に示す．必要に応じてストレージノードを追加し容量を増やした．2012 年 12 月末では，19 台のストレージノードで約 367TB の容量，利用率は約 70%である．

AP2 に接続するクライアントとして，保管した放送番組を閲覧するための再生プレイヤを試作した．映像，音声，字幕，番組情報の各仮想メディアを選択し，それらを同期再生できる．図３にその GUI を示す．ジャンプボタン（30 秒，１時間，１日，１週間，１カ月，１年）やスライドバーを使用し，放送時刻でシームレスに移動しながら過去の放送コンテンツを閲覧できる．

表１ 保管ファイルの諸元

	映像	音声
フォーマット	H.264 ES (NAL)	AAC ES (ADTS)
フレームレート	29.97	46.875
平均再生レート	2.3Mbps(VBR)	177Kbps(VBR)
平均フレームサイズ	9465 バイト	472 バイト

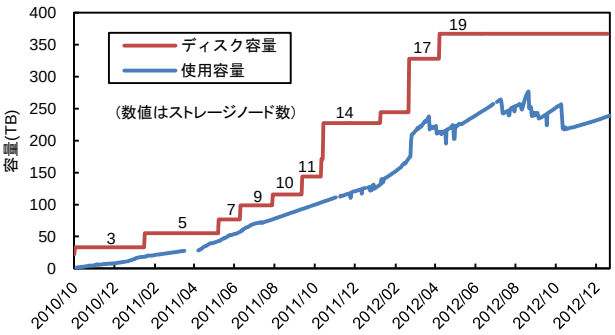

図２ 分散ファイルシステムの使用容量

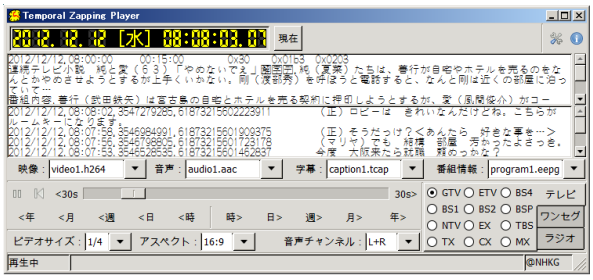

図３ 閲覧用再生プレイヤの GUI

5. アクセス性能評価

仮想メディアへのアクセス速度を測定した．サーバには Xeon L5420(2.5GHz, 4Corex2), 4GB メモリを用いた．

AP１および AP2 に対して 1～20 の同時アクセスした場合の，１コネクションあたりの平均フレームレートを図４に示す． AP2 は AP1 に比べて遅くなるものの，20 コネクションの同時接続の場合でも，リアルタイム再生可能なフレームレートを確保できた．測定結果を指数関数で回帰(図の曲線)し，リアルタイム再生が可能な範囲のフレーム速度となるコネクション数を求めた．映像と音声の同期再生では，AP1 では 44，AP2 では 28 となり，１サーバでこの程度の同時アクセスの収容が期待できることが分かった．

次に，AP3 を介して MP4 ファイルを取得する時間を測定した．結果を図５に示す．要求する放送コンテンツ長１分当たり１秒強程度の速度が得られた．転送時間に比較して，ファイル作成時間が大きく影響することが分かる．この速度は，別のサーバによる実験から，サーバの性能を変えることで改善できることを確認している．

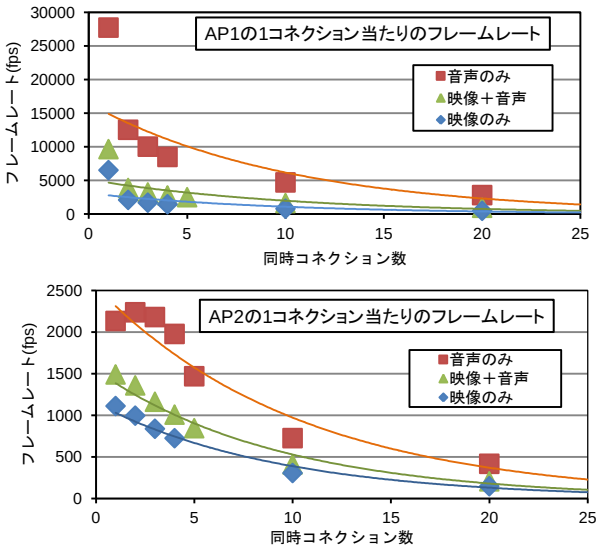

図４ 同時アクセス時の平均フレームレート

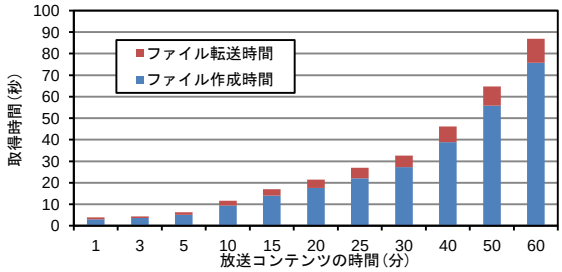

図５ MP4 ファイルの取得時間

6. まとめ

試作した放送コンテンツのアーカイブシステムの概要を述べた．試作システムは，永続的に蓄積した放送コンテンツを，放送時刻を使って連続したファイルとしてアクセスできる．従来のアーカイブシステムは，ファイルの保管とデータベースが分離しており，番組単位の検索・閲覧には便利である反面，時系列に過去の放送を閲覧するには使いづらい点もある．試作したアーカイブシステムは，時刻管理のデータベースを分散ファイルシステムに内包することで，容易に過去の放送番組を時系列に閲覧できる．本システムは，長期間の放送番組を解析するなど，放送技術や放送文化の発展のための研究用途にも有効なシステムであると考える．

参考文献

[1] 竹内，黄，金子，和泉，"タイムシフトザッピング視聴サービスのためのキャッシング制御手法"，信学総大，B-6-9，2012
[2] 金子，黄，竹内，和泉，"OneHop-P2P 拡張方式の実装と性能評価"，通学技報，NS2008-52，pp. 57-62，2008
[3] 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ディレクトリ管理手法"，FIT2009，L-033，2009
[4] 金子，黄，竹内，和泉，"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"，FIT2012，L-023，2012
[5] FUSE: Filesystem in Userspace, http://fuse.sourceforge.net/
[6] miksago/node-websocket-server: https://github.com/miksago/node-websocket-server