Vol.2011-AVM-72 No.3
2011/3/11

仮想ファイル化による大容量符号化映像のマルチフォーマット参照方式の検討

北村匡彦[†1]　白井大介[†1]　藤井竜也[†1]

4K をはじめとする高解像度映像を利用する場合，膨大な計算機資源が必要になる．また，その映像符号方式も多様であり，映像素材として利用する場合には，その相互接続性が求められる．本稿では，これらの課題を克服するために，符号化映像に含まれる映像フレームを仮想ファイル化することにより，ユーザーの利用する品質要求に適したファイルフォーマットで，高効率かつ高速に参照する手段を提案する．また，実際に JPEG2000 符号化映像を利用する場合に要求される計算機資源について，本提案手段と従来のトランスコード手段との比較を行う．

On Approaches to Decode Compressed Video Images to Multi-formatted files and its Impremention with Virtual File System

Masahiko Kitamura ,[†1] Daisuke Shirai [†1]
and Tatsuya Fujii [†1]

Vast computational resources including memory, storage and CPU power are required when one uses higher resolutional video image for secondary use. Besides, there are very wide variety of coding scheme for the video compression. Thus, the video image codec system should have a compatibility about several image file formats. To combat these issues, we propose a system that provides users to get the target part of video image frame with the required file format by the user. This system is implemented with the virtual file system that virtualizes the coded video frame as an image files of general video file format such as "TIFF". This paper also describes about the computatoional resources compared with conventional decoding methods.

1. はじめに

近年，われわれの身の回りのさまざまな場面で，映像が使われている．携帯電話におけるモバイル放送の視聴から，映画館における 4K パブリックビューイングに至るまでさまざまな利用用途が実現されている[1)]．これらは，液晶ディスプレイをはじめとした大型化表示デバイスの普及，計算機処理能力の向上，記憶媒体の低価格化によるディジタル処理技術の進化などの恩恵であるが，一方では，従来になかった新たな問題を生んでいる．例えば，米映画制作現場では，ディジタルワークフローの出現以前と比べて，扱う映像情報の量が圧倒的に巨大化し，映像制作にかかるコストが増加してしまう問題，ディジタルアーカイブによる保存データの安全保証性等の問題が「ディジタルディレンマ」として報告されている[2)]．

また，われわれの身近な映像技術の例として地上デジタル放送などで用いられている HD 解像度の映像[⋆1]を扱う場合を考えてみると，復号後の映像の情報量は 1 秒間に約 1.8Gbyte であり，これだけでも相当のフレームバッファが必要になる．空間解像度，時間解像度という観点からも，高解像度化の試みがなされており[3)]，今後もこの傾向は続くと思われる．

これらの増大する映像情報に対して，映像圧縮符号処理を施すことで，伝送や保存に対する情報量を圧縮する手段[4)] が一般化に用いられており，世界的に標準化が進んでいるが，その圧縮符号方式の種類は増える傾向にある．

実際に，映像を利用する場合には，元の映像符号方式に対応する処理系を用意するか，もしくは，映像符号方式を処理系に適応する形式にトランスコード[⋆2]する必要がある．前者は，デジタル放送や DVD-Video をはじめとする，コンシューマ向け映像配信で用いられる．一方で，映像編集や 2 次利用などの場合，基本的に復号した映像を用いる場合が多く，処理作業後には，適切な映像形式に符号化することが一般的である．この場合，非圧縮映像データが中間ファイルとして生成する過程を経るため，復号処理コストとフレームバッファサイズを考慮する必要が出てくる．

特に，高解像度，高フレームレートの映像，すなわち大容量映像を扱う場合，こうした計

†1 日本電信電話株式会社　NTT 未来ねっと研究所
Nippon Telegraph and Telephone Corp., NTT Network Innovation Labs.

⋆1 解像度 1920x1080 画素，フレームレート 30fps，色空間 YCbCr:422，色深度 8 ビットの映像信号

⋆2 なお，トランスコードとは，符号方式の変換を意味する用語であるが，符号化映像を一度復号した後に別の符号形式に再符号化する処理を指す場合，符号化映像を復号せずに符号情報の切り捨て (truncation) による解像度や符号化率の変換を指す場合がある．そこで，本稿では前者の広義的な意味においてトランスコードという用語を用いることにする．

算機リソースに関わる問題の他に，その映像用途の多様化に対応する必要がある．例えば，大容量映像がある場合，ユーザーによっては低解像の映像を用いたり，高解像度の映像の一部分のみを利用するというように，場面によってその利用方法が大きく異なる．こうした背景も含め，大容量映像の処理系においては，高速ネットワークを用いて計算機同士を強調することでリソース分散させて高速処理を実現するアプローチ[3] が理にかなうと考えられる．

本研究では，こうした高速ネットワークの環境下において，大容量映像の多様な利用用途に適応し，処理に係るコストを最小化させて機能する大容量映像処理 (メディアエクスチェンジ) システムの実現を目指す．本稿では，その第一段階として，高品質かつ大容量な符号化映像のトランスコードにおいて，符号化映像情報を映像フレーム単位で仮想ファイル化することで，ユーザーが求める形式，品質で統一的に，かつ，効率的に符号化映像を参照することを可能にするシステムを提案する．

2. 映像方式変換の要件

本節では，まず，一般的な符号化映像の利用場面を想定し，映像の復号やトランスコード処理をする場合の特性要素を検討する．実際の映像の利用シーンを考察し分類すると，(i)DVDVideo の視聴などの符号化映像の単純復号，(ii) 地上デジタル放送の IP 再送信などにみられるリアルタイムトランスコード，(iii)MPEG ファイル形式から H.264 形式への映像形式変換などのオフラインイントランスコード，(iv) 映像編集，(v) 映像検索などの映像認識の 5 つの利用用途を考えることができる．ここで，トランスコードにおける主な動作特性の差異を考えると，トランスコードのターゲットとなるフレームデータへのランダムアクセス性，および，トランスコード過程において生成される非圧縮フレームデータを一時蓄積するためのフレームバッファ量の 2 点が主なコスト指標となる．

この尺度から考察すると，(i)—(iii) の用途ではあまり考慮する必要がなく，従来の逐次的な復号処理を施すことが最適な方式であるといえる．一方で，(iv),(v) の用途では，両指標への要求が厳しいため，トランスコード処理において，実行タイミングや処理方式を考慮する必要が出てくる．

例えば，動画編集においては，複数ある素材となる符号化映像情報の中から，各映像データの特定部分を抜き出して，最終的にそれらを合わせた映像を作り出すことが主な処理系になる．この場合に，すべての素材映像をトランスコードするのは，メモリコストの点から現実的ではないため，編集者 (ユーザー，アプリケーション) が指定した時点でトランスコードをはじめ，その結果を即座に返すことが求められる．また，(iv),(v) の用途では，ユーザの参照ごとに解像度，レート，色ビット深度などが変わることもあるため，要求されるトランスコード映像の品質が異なる．さらに，ユーザーとの映像のやり取りには，交換用のデータ形式を定める必要があり，これもユーザーの要求に依存して変化する．

以上より，トランスコード処理において，ランダムアクセス性，フレームバッファ量が特に求められる動画編集や映像認識の用途では，

- (a) ユーザーが映像フレームを指定した時点でトランスコード処理を開始し即時に処理結果を返すこと
- (b) ユーザの求める映像品質でトランスコードすること
- (c) トランスコード結果をユーザーの指定するデータ形式で渡すこと

の 3 点がシステムの動作要件として求められる．

3. 映像のファイル仮想化によるトランスコードシステム

本節では，前節で明らかにした要件，すなわち，符号化映像へのランダムアクセス，ユーザの要求品質，形式での復号処理，復号後の映像を扱う映像処理システム間のインタフェース互換性の 3 要件を満たすためのシステムとして，ファイル仮想化によるマルチファイル参照システムの提案を行う．

3.1 符号化映像フレームのファイル化

映像に関わる処理系では，映像のフレーム単位で扱うことが基本になる．ただし，テレビ放送映像をはじめとする一般的な映像情報では，単位時間当たりのフレーム数 (フレームレート) は 30 フレーム/秒 (fps) から 60fps 程度である．このため，ユーザーが映像フレームをそのまま再生して利用するような場合，例えば映像配信のような場合には，映像データをフレーム毎に扱おうとすると，操作するデータ数が巨大し扱いにくくなるため，基本的にはある程度フレームをまとめたものを一つの映像データとして扱うのが普通である．特に，

表 1 箇条書環境の使用箇所（表の例）
Table 1 Use cases of usage of compressed video image and its property in terms of decodeing/trancoding processes

種類	ランダムアクセス性	フレームバッファ量	利用例
単純逐次復号	低い	少ない	DVD ビデオの再生視聴
リアルタイムトランスコード	低い	少ない	放送の IP 再送信
オフライントランスコード	低い	多い	WMV から MPEG2 への変換
動画編集	高い	多い	映像素材を基にした映像制作
映像認識	高い	多い	映像認識，映像要約，映像検索

映像の符号化を考えると，複数のフレームをまとめて圧縮するほうが効率が良いため，符号化する時点でフレームをまとめて処理を行っている．

一方で，映像編集に代表されるような，映像フレームごとに処理を変えるような場合，上述のようにフレームをまとめることができないため，基本的に映像フレーム毎に独立した処理を施すことになる．このため，膨大な映像フレーム群に対して，効率的にランダムアクセスし，かつ，統一的に操作する必要が出てくる．

そこで，本稿では，単位映像フレームを単位ファイルとして扱うことで，この条件を満たすトランスコードシステムを提案する．ファイルという概念は，計算機上では，基本的データの単位として扱われ，また，ディレクトリ (フォルダともいわれるが，本稿ではディレクトリという用語を用いる) 構造の概念によってファイル構造の階層化が可能である．これは，映像情報の階層符号の概念，トランスコードのユーザー要求パラメータ，映像フレームのグループなどをディレクトリ/ファイル構造に対応付けることで，直感的な映像フレームの操作が可能であると考えられる．以降では，映像フレームをディレクトリ/ファイルにどのように対応付けるべきか，その方式について検討していく．

3.2 ファイルシステムをインタフェースにするトランスコードプロセス

映像フレームのディレクトリ/ファイル構造への対応付けを議論する前に，ユーザー要求するした映像フレームに必要な要素，すなわち，ディレクトリ/ファイルに対応付けすべき情報 (パラメータ) を検討する必要がある．

そこで，実際に符号化映像の特定のフレームをトランスコードして利用する場面を考えてみる (図 2)．はじめにユーザーはトランスコードシステムに対して，映像フレーム番号，出力の解像度，ファイルデータ形式などを指定する (図中の破線)．これを本稿ではトランスコードパラメータと呼ぶことにする．そして，その要求に応じてトランスコードシステムが適切な処理を施した映像フレーム結果をユーザーに返す (図中の実線)．これを本稿ではトランスコードデータと呼ぶことにする．

以上の考察により，映像フレームのトランスコードパラメータをディレクトリ/ファイル構造に対応付け，かつ，トランスコードデータをファイルの実体データと対応付けることで，上述の 2 つの情報の交換をファイル操作によって実現できる．つまり，ユーザがトランスコードパラメータを指すディレクトリ/ファイルにアクセスし，それに基づいて，トランスコードシステムがトランスコードデータをファイルの実体データとしてユーザに渡すことで，トランスコード処理が実現される．それゆえ，実際にユーザーがファイルの内容を参照するまではトランスコード処理する必要がなく，この時点においてフレームバッファを消費することはない．よって，この提案手法によって，第 2 節で求めたトランスコードシステムにおける動作要件 (a)–(c) を満たすことができる．

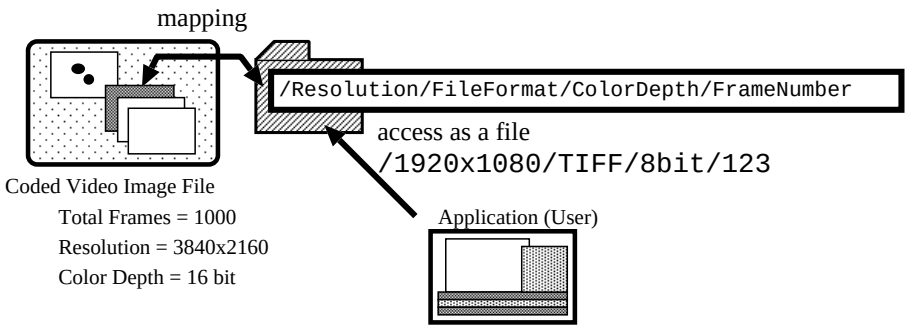

図 1 ディレクトリ/ファイル構造とトランスコードパラメータ，トランスコードデータとの対応付け
Fig. 1 Mapping between file/directory structure and transcoding data

例えば，トランスコードシステムとの取り決めによって，ディレクトリ階層とトランスコードパラメータが “/Resolution(解像度)/FileFormat(データ形式)/Depth(色深度ビット)/FramNumber(フレーム番号)” という対応付けがなされているとする．ここに，総フレーム数 1000，解像度 3840x2160 画素 (4K)，色深度 16bit の符号化映像があり，これに対して，ユーザーが 123 番フレームを 1920x1080(HD)，TIFF 形式，色深度 8bit の品質で要求する場合を考える．このとき，ユーザーは “/1920x1080/TIFF/8bit/123” というファイルを参照することで該当するトランスコード結果を得ることができる．

以上のトランスコードパラメータとディレクトリ/ファイルとの対応付けは一例であり，必要なトランスコードパラメータの種類によってこの対応付けは様々に可能である．換言すれば，ディレクトリ構造とパラメータの順序の対応付けは任意であり，例えば，映像フレームをいくつかにグループ化して符号化映像を扱いたい場合には，そのための “グループ” を識別するためのパラメータを対応付けるディレクトリを追加すればよい．

このように，ユーザーのニーズやとシステムの環境に合わせて両者間の対応付けを適合することで柔軟なトランスコードシステムの構成が可能になる．

4. FUSE を用いた提案手法トランスコードシステムの実装

これまでの議論では，ディレクトリ/ファイル構造をトランスコードシステムにトランス

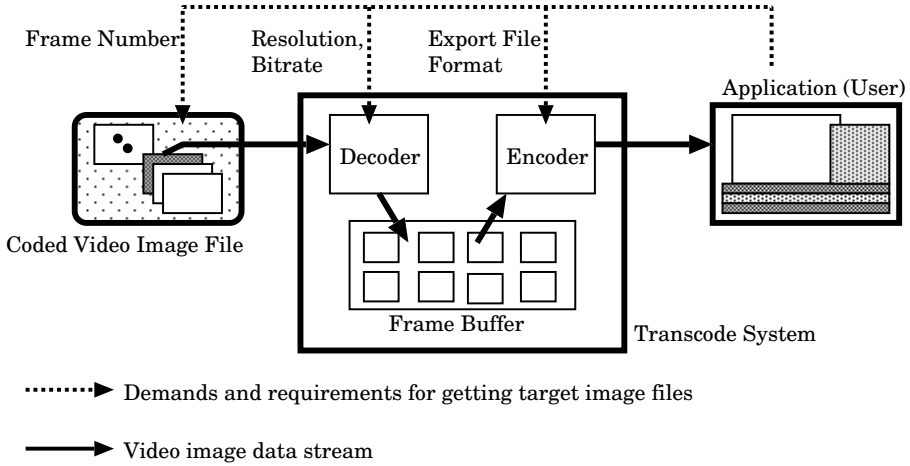

図 2 ユーザーアプリケーションとトランスコードシステムの処理系と情報の流れ
Fig. 2 Conversion framework and data stream between user's application and transcoding system.

コードパラメータとして渡せることを前提に進めてきた．本稿では，この提案概念を実装する手段として FUSE(Filesystem in UserSpace) を用いて OS のファイルシステム経由でトランスコードシステムとユーザーアプリケーションとの情報交換を実現するシステムを提案する．

4.1 FUSE フレームワーク

FUSE とは，ユーザー権限のみで OS の提供するファイルシステムを実装を可能にする技術であり，UNIX 系 OS のライブラリとして提供されている[7]．具体的には，ユーザーがファイル処理系のイベントハンドラとして用意した関数を FUSE ライブラリが提供する関数経由でコールバック登録する．その後で，ユーザーが OS の提供する仮想ファイルシステム上でファイル操作した時に，このコールバック関数が呼び出され，処理結果をユーザーに返す仕組みになっている．ここで，コールバック関数はユーザー側で任意に実装することが可能であるため，ユーザ権限でファイルシステムを実装することが可能になる．

4.2 FUSE を用いた JPEG2000 映像のトランスコードシステムの実現

本稿では，以上で提案した手法の有効性を示すことを目的として，FUSE を用いた JPEG2000 符号化映像[5] のトランスコーディングシステムを構築した．

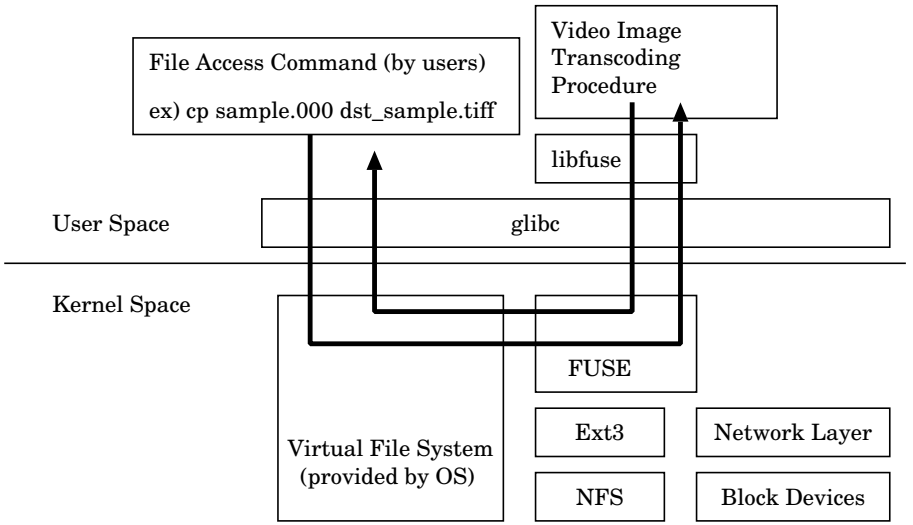

図 3 FUSE ライブラリを用いた符号化映像のフレームの仮想ファイル化
Fig. 3 Virtualization of compressed video frame as a file and its implementation with FUSE library.

まず，トランスコードパラメータとして，トランスコードの解像度，出力ファイル形式，色深度，映像フレームを対象とする．ただし，映像フレームを表すファイルを一つのディレクトリ内で扱う場合，ファイルシステム内でファイルエントリー数が膨大になり，実際の処理系で大きい遅延が発生してしまう．このため，本実装では映像フレームを 1000 フレーム単位でまとめるための，映像フレーム群をトランスコードパラメータに含めることにした．

以上を考慮して，本システムでは，ファイルパスに対して，“/解像度/出力映像フレームデータ形式/色深度/フレーム番号群 (1000 フレーム単位)/フレーム番号” という形式でトランスコードパラメータを定義する．それぞれのディレクトリ構成を以下に説明する．

第 1 階層: 解像度 解像度は元の符号化映像の解像度を基準にして，JPEG2000 符号のウェーブレットレベルに合わせて低解像度のディレクトリエントリを決定する．たとえば符号化映像の解像度が 3840x2160 画素であり，そのウェーブレットレベル $W = 5$ である場合，解像度のエントリは，3840x2160，1920x1080，960x540，480x270，240x135，120x68 の種類になる．

第 2 階層: 出力映像フレームデータ形式 本システムでは，汎用的に用いられる静止画

フォーマットとして TIFF 形式 (Tagged Image File Format, 非圧縮) および PNM 形式 (Portable aNyMap) を出力データ形式として採用する．

第 3 階層: 色深度　符号化映像の色深度によらず，16 ビットと 8 ビットの 2 種類を定義する．SDI(Serial Digital Interface) 等の映像信号では 10 ビットなどの色深度も使われるが，計算機上のファイルとして映像を扱う場合，そのメモリ構造から 16 ビットまたは 8 ビットで使用されることが一般的であるため，本システムではこの 2 種類に限定した．

第 4 階層: フレーム番号群　先にも説明したとおり，一つのディレクトリ内で膨大な数のフレーム=ファイル数を扱う場合，ファイルシステムの処理に大きな遅延を要する．この問題を回避するため，本稿では 1000 フレーム単位で扱うための中間的なトランスコーディングパラメータとしてフレーム番号群を定義した．

第 5 階層: フレーム番号　フレーム番号をファイル名とする．ファイルシステム上においては，ファイル名のみ存在し，そのファイル実体はなにもない．ユーザーがファイルシステムから実際にこのファイルの読み出しを行うときに初めてトランスコードされた映像フレームデータが生成される．

次に，このディレクトリ/ファイル構造に基づいて，トランスコードを行う処理系について説明する．この処理動作は，実際にはコールバック関数として定義され，FUSE ライブラリを介して OS が提供する仮想ファイルシステムから呼ばれる．処理が実行されるタイミングは，対象とする符号化映像をマウントポイントにマウントするとき (mount)，マウント後にディレクトリ内のファイル一覧を参照するとき (readdir)，ファイルの読み出しを行うとき (read) の 3 種類に分類される．それぞれにおいて，具体的な動作を以下に述べる．

マウント処理 (mount)　対象となる符号化映像 (ここでは JPEG2000 符号化映像) の符号パラメータ，すなわち，映像の解像度，ウェーブレットレベル，色深度を取得する．これらの符号パラメータとトランスコードパラメータ (解像度，出力映像データ形式，色深度，フレーム番号) からディレクトリ構造を構築する．また，解像度ディレクトリにおいては，JPEG2000 符号の解像度スケーラビリティ性を利用するため，ウェーブレットレベルに応じた解像度を対象にしている．

ディレクトリ参照処理 (readdir 関数)　マウントポイントからディレクトリ階層数を調べ，それぞれの階層に合わせたディレクトリ，または，ファイル名のリストを返す．例えば，マウントポイントから 1 階層目のディレクトリ内で readdir 関数が呼ばれた場合，JPEG2000 符号化映像の解像度とウェーブレットレベル数から，各レイヤの解像度のディレクトリ名を返す．

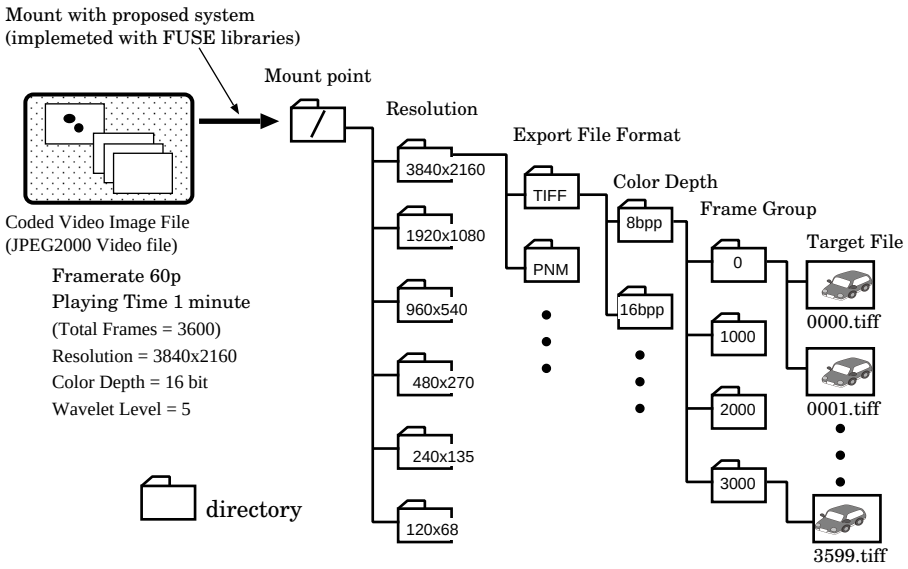

図 4　提案手法に基づく JPEG2000 符号化映像のファイル仮想化
Fig. 4　A implementation with FUSE library based on proposed concept.

ファイル読み込み処理 (read 関数)　与えられたファイルパスに基づき，トランスコードパラメータを抽出する．その後，JPEG2000 デコードプロセスを起動し，非圧縮の対象映像フレームを生成し，指定されたデータ形式で OS の仮想ファイルシステムに返す．この過程で生成される映像フレームの非圧縮データは，この処理の終了時に解放する．

4.3　JPEG2000 符号を用いた映像ファイルと実際の計測

前節で実装したトランスコーディングシステムでは，OS のファイルすシステムを介してトランスコードプロセスを起動させている．このため，単純にトランスコードプロセスを動作させる場合と比べてオーバーヘッドが生じる．そこで実際に計算機上で両社の比較を行った．計算機は，CPU として Intel 社 Core i5 M450/2.40GHz，OS として Linux Kernel2.6.32-26 のものを用いた．解像度 3840x2160 画素，ウェーブレットレベル 5 の JPEG2000 符号映像を用いて，3 種類の解像度でトランスコーディングした場合の計算時間比較 (単位フレームあたりの処理時間) を行った．両者でのトランスコード処理部分は全く同じものを用いてお

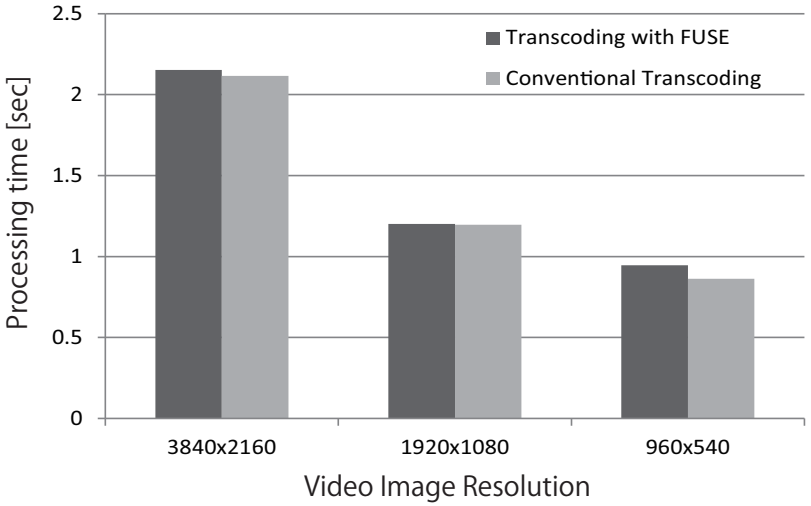

図 5 本提案システムで発生するオーバヘッドの評価
Fig. 5 Evaluation of overhead that stems from proposed method.

り，処理系全体としては，ファイルシステムを介して動作させたのか，直接コマンドラインから動作させたかの違いしかない．この結果を図 5 に示す．これより，FUSE を用いたことによるオーバーヘッドはトランスコード処理に対してほぼ無視できると考えられる．

4.4 今後の課題

本研究は，大容量映像の多様な利用用途に適応し，処理に係るコストを最小化させて機能する大容量映像処理 (メディアエクスチェンジ) システムの実現を目的とする．本稿では，その第一段階として，単独計算機内で実現されるトランスコーディングシステムを検討したが，いくつかの課題が見えている．

第 1 に，本稿の提案手法の実装では，動作処理の単純化のため，ユーザーがファイルシステムを指定する毎にトランスコード処理を施す設計にした．しかし，映像フレームへのランダムアクセス性が高い場合においても，現在参照しているフレームの前後のフレームを次点で参照する確率はある程度高いと考えらる．このような映像フレームへの参照局所性を考慮に入れることで，映像フレームへの参照要求がくる前に，予測的にトランスコード処理を実行させ，ユーザーへの応答性を向上させることが可能であると考えられる．

第 2 に，今後の課題として，ネットワーク環境における動作の最適化を検討する必要があると考えられる．先にも述べたとおり，映像の大容量化に伴ってネットワーク環境下でのシステム構成が前提になってくる．したがって，符号化映像自身の保存位置と利用ユーザーの処理系がネットワークを介して分散化されることになる．本稿で提案した実装は，単独計算機上での動作を前提にしているため，トランスコーディング処理の中間で生成される映像フレームの容量については考慮する必要がなかったが，ネットワーク環境下では利用可能な伝送帯域が限られているため考慮する必要がでてくる．

今後はこれらの課題を克服する方式を検討していく予定である．

5. ま と め

本稿では，符号化映像を仮想ファイル化することで，ユーザーの要求に適した形式で映像を参照する方式の提案を行った．また実際に，映像フレームをファイルとして扱い，トランスコードに関係する処理を OS の提供するファイルシステムを介して実行するシステムの提案および実装を行った．今後は，高速ネットワークにおいて多様なユーザー環境に適合して機能する大容量映像交換 (メディアエクスチェンジ) の仕組みを検討していきたい．

参 考 文 献

1) 青木孝文, 特集　映像情報メディア年報 -2008 年 4 月から 2010 年 3 月の進展-, 映像情報メディア学会誌, vol. 64, No. 8, pp. 1126–1135 (2010).
2) Randall Davis, The digital dilemma, Communications of the ACM, v.44 n.2, p.77-83, Feb. (2001).
3) Larry L. Smarr, Andrew A. Chien, Tom DeFanti, Jason Leigh, and Philip M. Papadopoulos. 2003. The OptIPuter. Commun. ACM 46, 11, pp. 58–67 (2003).
4) Vasudev Bhaskaran, Konstantinos Konstantinides, Image and Video Compression Standards: Algorithms and Architectures, Springer; 2nd edition (1997).
5) Christopoulos C., Skodras A., Ebrahimi T., The JPEG2000 still image coding system: an overview, Consumer Electronics, IEEE Transactions, Volume 46, Issue 4, pp. 1103–1127 (2000).
6) Schwarz H., Marpe D., Wiegand T., Overview of the Scalable Video Coding Extension of the H.264/AVC Standard, Circuits and Systems for Video Technology, IEEE Transactions, Volume 17, Issue 9, pp. 1103–1120 (2007).
7) FUSE: Filesystem in Userspace, http://fuse.sourceforge.net/