データベースシステム 113－37
（１９９７．７．１５）

並列環境における投機的実行を用いたビデオデータベース格納可能容量の拡大

高山 毅, 弘中哲夫, 藤野清次
広島市立大学 情報科学部 情報工学科
〒731-31 広島市安佐南区大塚東3-4-1 Tel.:082-830-1561
E-mail: takayama@ce.hiroshima-cu.ac.jp

本稿では，映像編集システムの基盤となる映像データベースの議論をする．具体的には，MPEG-1 データをさらに gzip コマンドにより圧縮したデータをデータベースの格納データとし，そのために発生する映像再生時の伸長の負荷を，並列環境における投機的実行を用いて隠蔽または縮小しつつ，ビデオデータベースの格納可能容量を拡大することを提案する．映像データベースでは，データベースと呼ぶにふさわしい件数の映像を格納することが容易ではないことが問題である．本稿で行う評価によれば，本提案によりビデオデータベースの格納可能容量が 3.4 ～ 9.5 % 拡大し，従来よりも多くの件数の映像を格納することが可能になると期待できる．

キーワード：ビデオデータベース，ファイル圧縮，並列環境，投機的実行

Expansion of Video Database Capacity Using Speculative Execution Technique in Parallel Environments

Tsuyoshi Takayama, Tetsuo Hironaka, and Seiji Fujino
Department of Computer Engineering, Faculty of Information Science,
Hiroshima City University
3-4-1, Ozuka-Higashi, Asa-Minami-ku, Hiroshima, 731-31 Tel.: +81-82-830-1561
E-mail: takayama@ce.hiroshima-cu.ac.jp

This paper proposes to expand a video database capacity by adopting an MPEG-1 data compressed by "gzip" command as a storage data of the database. In here, we hide or reduce a load on decompression of the storage data by using "speculative execution" technique in parallel environments. Although a video database is effective for a base of a video editing system, it is not easy to store enough number of video data that will be worth called "database". An evaluation in this paper shows that we may expect to obtain from 3.4 to 9.5 % gain for a video database capacity which makes it possible to store more number of video data than conventional video databases.

Keywords: video database, file compression, parallel environments, speculative execution

1 はじめに

近年, 映像編集システムの基盤となるビデオデータベース (以降, データベースを DB と略す) に関して, さまざまな研究が進められている [2][5]. しかしながら映像データは比較的大容量であり, 単一のビデオ DB 内に, DB と呼ぶにふさわしいほどの件数の映像を格納することは容易ではなく, 問題である.

本研究では, 従来ビデオ DB 内で格納データとして取り扱われている MPEG-1 (以降単に MPEG と略す)[6] データをファイル圧縮コマンド gzip を用いてもう一段階圧縮し, これを DB の格納データとする. そして, 映像再生時の MPEG データへの伸長の負荷という副作用を, 並列環境における投機的実行 (speculative execution)[7] を用いて隠蔽または縮小する. 以上の方式を用いて, 目的とする『ビデオ DB の格納可能容量の拡大』を実現することを提案する.

本稿で行う, 本提案の実現可能性に関する第一次評価によれば, 3.4 ～ 9.5 % のビデオ DB 格納可能容量の拡大が期待でき, また前述した副作用も, 完全に隠蔽または許容可能なレベルへ縮小できると期待できる.

以降, 本稿は以下のように構成されている. まず次節で, 従来のビデオ DB 格納方式の問題点を述べた上で, 3 節で『並列環境における投機的実行を用いたビデオ DB 格納方式』を提案する. 4 節ではその実現可能性に関する第一次評価として, 実験指向の評価を行う. 最後に 5 節で結論と今後の展望を述べる.

2 従来のビデオ DB 格納方式の問題点

2.1 国際標準 MPEG

まず, 映像データの圧縮符合化の国際標準である MPEG について概括しておく [2][5][6]. MPEG では, 静止画列からなる映像データを, 以下に定義される I, P, B と呼ばれる三種類のフレームを用いて圧縮符合化する.

- **I フレーム：** 1 フレームの画像をその画像のみで一定のアルゴリズムに従って圧縮符合化するフレーム.
- **P フレーム：** 過去の I フレームまたは P フレームの画像からの変化分だけを順方向予測に基づき圧縮符合化するフレーム.
- **B フレーム：** 過去の I または P フレームと未来の I または P フレームの画像の動きの変化分を双方向予測に基づき圧縮符合化するフレーム.

以上三種類のフレームから, 以下に定義される GOP が構成される.

- **GOP(Group of Pictures)：** ランダムアクセスの入口を提供する映像の単位. ヘッダ部とデータ部からなる. 単一の GOP は, その中に I フレームを最低一つは含む.

GOP 内のフレームの混合方式, および 1GOP あたりのフレーム数を表現するパラメータとして, 以下の二つがある.

- **M ：** P フレームが現れる周期.
- **N ：** 1GOP 中に含まれるフレーム数.

2.2 格納データのサイズにおける問題点

まず, 映像を圧縮する場合の圧縮率 $c\_rate(\%)$ を,

- $before$ ：圧縮操作適用前のデータサイズ
- $after$ ：圧縮操作適用後のデータサイズ

として, 以下の式によって定義しておく.

$$c\_rate(\%) = \frac{before - after}{before} * 100$$

ビデオ・オン・デマンド (VOD) などのような, 映像データに対する逐次アクセスとは異なり, MPEG データを格納データとするビデオ DB は, 映像へのランダムアクセスが可能という点で, 映像編集システムの基盤として効果的である. 先行研究によれば, データ操作における操作単位としての適切さから [2], およびそれに加えてスワップ領域のサイズに起因する制限への考慮から [5], MPEG データは GOP 単位で DB の格納データとすることが適切であるとされている.

この方式の問題点は,

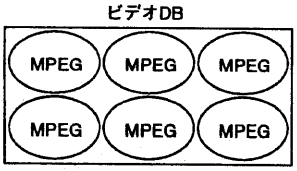
従来方式の問題点： 『圧縮率が, MPEG 圧縮を行う場合に採用したアルゴリズムに依存して制限され, DB の格納データのサイズも, その制限によって決定されてしまうこと』

である. 実際, 2 時間番組の映像の格納に約 1GB の容量が必要である [5] ことを考えると, 仮に DB のデータ格納用に 100GB のディスク領域が利用可能であるとしても, 2 時間番組 100 本で, DB は満杯になってしまう. 放送局が持つ過去の資産の量を考えれば, これでは不十分と言わざるを得ない. よって, 利用可能な一定量のディスク領域上により多くの番組を格納し, かつ DB として利用可能ならしめるための方法論を開発することが, ビデオ DB において不可欠な課題と言える.

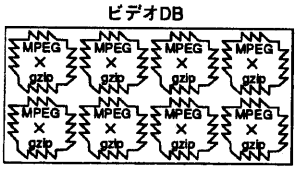
3 投機的実行を用いたビデオ DB 格納方式

3.1 本研究でのビデオ DB 格納方式

本研究では, 以下の提案を行う.



(a) 従来方式: MPEG データ.
映像再生コマンドを用いてそのまま再生可能.



(b) 提案方式: MPEG データを gzip コマンドを用いて更に圧縮したデータ.

(a) より多くの映像を格納できる. しかし, 映像再生コマンドを用いても, そのままでは再生不可能.

図 1: ビデオ DB における格納データ.

本研究でのビデオ DB 格納方式 (図 1): ビデオ DB において従来格納データとされている MPEG データを, ファイル圧縮コマンド gzip を用いて更に圧縮し, 各映像が占める格納容量を小さくすることによって, 一定量の格納可能容量のビデオ DB において, より多くの映像を格納することを目指す.

一般にファイル圧縮には, 情報の損失を伴うロッシー (lossy) 圧縮と, 伴わないロスレス (lossless) 圧縮がある. MPEG, gzip はそれぞれの代表的なものの一つである. gzip について, そのバージョン 1.2.4 のオンライン・マニュアルをもとに概括する. gzip は, (i) 『Lempel-Ziv Coding(LZ77) により圧縮を行う』もので, (ii) 『いかなるファイルに対しても適用可能』である. 以降, gzip を用いて圧縮されたファイルのことを単に, gzip ファイルと呼ぶ. (iii) 『ロスレス圧縮』であり, (iv) 『注意すべきこととして, 被圧縮ファイルによっては稀に, 圧縮率が 0 またはマイナス, すなわちファイルサイズを減少させないことがある. 』 (v) 『圧縮強度の設定は 9 段階で可能』であり, いずれも (vi) 『一般には他のファイル圧縮コマンド compress, pack での圧縮率を上まわる. 』 (vii) 『圧縮の逆操作である伸長は, gunzip コマンドにより可能』である.

本項冒頭の提案において, gzip を採用した理由は上記 (ii),(iii),(vi) である. 特に (iii) は, 映像の画質を劣化させないという点で有用である.

3.2 提案方式の副作用

提案方式では以下の二つの副作用の解決が必要である.

(副作用 1): MPEG ファイルは, 映像再生コマンドによりそのまま再生可能であるが, これを gzip 圧縮したものは, そのままでは再生できない.

(副作用 2): gzip ファイルの, MPEG ファイルへの伸長の負荷が, 無視できるものとは言えない. これが, 映像の再生時に, 従来にはなかった待ち時間を発生させる危険性がある.

伸長と再生を UNIX などでのパイプ (|) で結合する方法も考えられるが, これはフレームレートを低下させる. 映像編集はカットの貼りつけが中心であり, フレームレートの低下は時間長の正確な見積りを難化させるので, この方法は万全とは言えない.

3.3 投機的実行の導入

並列環境において, ジョブの応答時間を短縮するために有効な技術として, 投機的実行 [7] がある. ここで 3.1 項の提案に, 以下を加加する.

本研究でのビデオ DB 格納方式 (続き): gzip ファイルの伸長には, 投機的実行を導入する (図 2).

DB への投機的実行の導入は, 同時実行制御において文献 [8] で行われているが, 問合せ処理においては, 本年 3 月の著者らによる文献 [4] が初出である [3]. これは, 検索操作に対する応答性能を向上させることを目的としており, そのアルゴリズムは, 以下のようである.

step 1: ユーザによって指定される確率が相対的に高い複数の検索条件について, ユーザが検索条件を吟味している間に, 並列かつ投機的に各検索条件に見合う検索結果を計算し, 用意する.

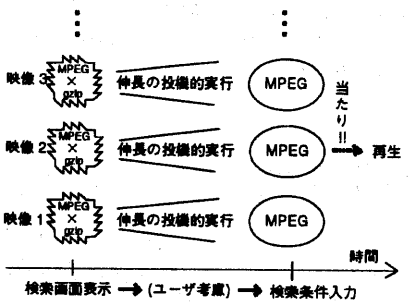


図 2: 提案方式における, 投機的実行を用いた映像の再生方式.
ユーザの考慮中に伸長の投機的実行を行う.

表 1: クライアント, サーバの仕様.

	クライアント	サーバ
CPU	MIPS R4600 Processor Chip Revision: 2.0	MIPS R10000 Processor Chip Revision: 2.5
OS	IRIX5.3	IRIX6.2
Memory	64 Mbytes	128 Mbytes
Swap Area	262888 blocks	787232 blocks

step 2: ユーザから実際に検索条件が入力されたら, あらかじめ用意しておいた検索結果のうち, 入力された検索条件に対する結果のみを出力する.

step1 ～ 2 の実現可能性および有効性について, 文献 [4] は, 評価プログラムを用いて確認している.

アプリケーションレベルでの投機的実行の導入に関する研究は活発化してきており, 本年 6 月に発表された文献 [1] では, ニュースシステムにおける応答性能の改善への投機的実行の導入が提案されている.

4 ビデオ DB 格納可能容量拡大の実現可能性に関する第一次評価

一般に, ビデオ DB の構築には, オブジェクト指向 DB の利用が効果的である [5]. 文献 [4] で行った評価によれば, 並列オブジェクト指向 DB の問合せ処理に投機的実行を導入した場合の, 各処理ノード間の通信オーバヘッドは, 問合せ処理に障害を生じさせるほどのものとはならない. そして, 応答時間の特性を決定するのは, 各処理ノードで実行される単一検索条件での投機的かつ逐次的な問合せ処理時間である. そこで本稿では, ビデオ DB 格納可能容量拡大の実現可能性に関する第一次評価として, 逐次環境を用いて, 以下の二つの実験を行う.

実験 1： MPEG データへ gzip コマンドを適用した場合, 3.1 項で述べたように, 圧縮率が 0 またはマイナスになることはないか. また, プラスの圧縮率で圧縮することが可能な場合, どの程度の圧縮率が達成できるか.

実験 2： gzip 圧縮された MPEG データを基の MPEG データへ伸長する負荷は, どの程度の時間的コストを生むのか.

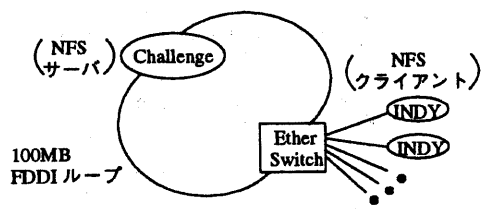


図 3: 実験で用いたマシン環境とネットワーク環境.

4.1 実験環境

4.1.1 マシン環境とネットワーク環境

後の評価において, ワークステーションクラスタによる並列環境を使用することを念頭に, 図 3 に示される SGI IRIS INDY R4600 (図中, および以降 INDY と略す) の一台に対する対話的操作を通じて評価を行う. この INDY は, SGI Challenge L(以降 Challenge と略す) の NFS クライアントになっている. そこで以降では, INDY, Challenge をそれぞれ, クライアント, サーバとも呼ぶ. クライアント, サーバの仕様は, 表 1 の通りである. また, サーバ-クライアント間は 100MB FDDI ループ, および 3 Com LANplex 2500 の Ether Switch により接続されている (図 3).

4.1.2 INDY 上での MPEG 圧縮

INDY 上での MPEG ファイル (拡張子 .mpv), 静止画列からなる映像ファイル (拡張子 .mv) を, 以降それぞれ, MPV ファイル, ムービーファイルと呼ぶ. ムービーファイルから MPV ファイルへの圧縮は, IRIX のファイル変換ツール mediaconvert で行う. この場合, 生成される各 GOP 中に含まれる I フレームの数は常に 1 となる.

mediaconvert の特長は, パラメータ N, M の指定が可能なことである. デフォルト値は`N=16, M=4` であり, そのフレーム構成を 1GOP について具体的に書くと, 「IBBBPBBBPBBBPBBB」である. 今回は N, M の組合せとして, `(i)N=16, P=1, (ii)N=16, P=4, (iii)N=16, P=8,(iv)N=24, P=8, (v)N=24,`

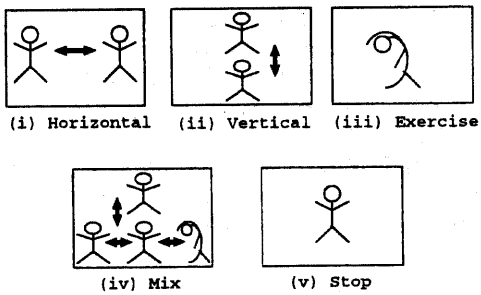


図 4: 実験で用いた五種類の映像.

P=12 の五通りを用いる.

4.1.3 実験映像

人間が (i)Horizontal: 水平方向に往復運動する, (ii) Vertical: 鉛直方向に往復運動する, (iii)Exercise: ラジオ体操をする, (iv)Mix:(i) ～ (iii) の運動をミックスして行う, (v)Stop: 静止している, 五種類の映像を用いる (図 4). フレームレートは 30(frame/sec), 画像サイズは 160 × 121 (dot), また, 各映像の開始から終了までの時間長は, いずれも約 30(sec) となるように VHS テープからキャプチャする.

今回の実験では簡単のために以下の方針をとる. すなわち, ビデオ DB において, 映像は GOP 単位で保存すべきであるが [2][5], 時間長が約 30(sec) と, 文献 [5] で指摘されているキャッシュ領域の問題が生じる範囲に達していないので, 簡単のために映像を GOP には分割せず, 30(sec) 全体をそのまま格納の単位とする.

4.2 実験方法と実験結果

4.2.1 予備実験

まず予備実験として, ムービーファイルから MPV ファイルへの MPEG 圧縮が, mediaconvert を用いて実現されていることを確認する.

表 2 に示した圧縮結果より, 以下の二点が言える.

1. Horizontal で 95.8%, それ以外の映像で 96.2 ～ 3% の圧縮率が得られている.
2. N, M の違いによる MPV ファイルのサイズの差異は 1% 程度である.

これらの MPV ファイルを映像再生コマンド movieplayer を用いて再生したところ, 基のムービーファイルの再生と比較して, 画質の劣化を知覚しない程度で再生できた. よって, mediaconvert による MPEG ファイルの作成は, 成功していると見做せる.

表 2: mediaconvert による圧縮結果.

映像種	N	M	File Size (Bytes)	圧縮率 (%)
Horizontal	(ムービー)		47996936	
	16	1	2029804	95.8
	16	4	2019228	95.8
	16	8	2030192	95.8
	24	8	2031052	95.8
	24	12	2028364	95.8
Vertical	(ムービー)		58131532	
	16	1	2177100	96.3
	16	4	2182928	96.2
	16	8	2183772	96.2
	24	8	2183988	96.2
	24	12	2201948	96.2
Stop	(ムービー)		55556492	
	16	1	2087768	96.2
	16	4	2093188	96.2
	16	8	2090340	96.2
	24	8	2086016	96.2
	24	12	2087052	96.2
Exercise	(ムービー)		55685244	
	16	1	2086192	96.3
	16	4	2089408	96.2
	16	8	2089788	96.2
	24	8	2089820	96.2
	24	12	2088340	96.2
Mix	(ムービー)		51050172	
	16	1	1929268	96.2
	16	4	1914048	96.3
	16	8	1914224	96.3
	24	8	1915536	96.2
	24	12	1913300	96.3

4.2.2 実験 1

gzip による MPV ファイルの圧縮率を,

```
% gzip -v -c -9 (MPV ファイル名).mpv
  > (MPV ファイル名).mpv.gz
```

により評価する. 各オプションの意味は, 以下の通りである.

- -v : 圧縮率を表示する.
- -c : 基の MPV ファイルを消去せずに, gzip ファイルを生成する.
- -9 : 最大の圧縮強度で圧縮する.

表 3 に示した実験結果より, 以下の二点が言える.

1. 予備実験の MPEG 圧縮の場合よりも, 映像の種類による圧縮率のバラつきが大きい.
2. Horizontal 以外の 四種の映像について, いずれも N, M が大きくなるに比例して, 圧縮率が上昇している. N=24, M=12 の場合が最大で, 3.4 ～ 9.5% の圧縮率が得られている.

いま, x% の圧縮率が得られたとすると, その x% 分の領域は他の映像の格納に利用可能となる. 換言すれば, 格納可能容量が x% 拡大したと言える.

4.2.3 実験 2

伸長時間を,

```
% timex gunzip -c (MPV ファイル名).mpv.gz
  > /tmp/(MPV ファイル名).mpv
```

により評価する. timex コマンドは, それに続くコマンドの実行時間を, ターンアラウンド (real), ユーザ時間 (user), システム時間 (sys) の三つで報告する. なお, 出力を /tmp 下に行うのは, 相対的に高速な書込みが可能であるとともに, 『問合せ処理の投機的実行を行った結果は処理ノードに局所的に置く』 [4] ためである.

表 3: 実験 1. の結果.

映像種	N	M	圧縮率 (%)
Mix	16	1	3.2
	16	4	6.5
	16	8	8.7
	24	8	8.8
	24	12	9.5
Vertical	16	1	3.1
	16	4	5.3
	16	8	7.2
	24	8	7.5
	24	12	9.4
Horizontal	16	1	3.0
	16	4	5.3
	16	8	3.4
	24	8	3.4
	24	12	3.4
Stop	16	1	2.4
	16	4	3.2
	16	8	4.7
	24	8	7.2
	24	12	7.8
Exercise	16	1	2.8
	16	4	3.5
	16	8	4.2
	24	8	4.2
	24	12	4.6

表 4: 実験 2 の結果.

映像種	N	M	伸長時間 (sec)		
			real	user	sys
Mix	16	1	4.09	1.27	0.31
	16	4	3.53	1.23	0.29
	16	8	3.74	1.21	0.28
	24	8	3.48	1.21	0.30
	24	12	3.38	1.21	0.30
Vertical	16	1	4.29	1.44	0.33
	16	4	4.27	1.42	0.34
	16	8	4.89	1.39	0.33
	24	8	4.24	1.39	0.34
	24	12	4.21	1.38	0.32
Horizontal	16	1	4.27	1.34	0.33
	16	4	4.29	1.31	0.31
	16	8	3.88	1.34	0.31
	24	8	3.85	1.34	0.32
	24	12	4.03	1.34	0.32
Stop	16	1	4.23	1.38	0.33
	16	4	4.10	1.37	0.33
	16	8	3.95	1.36	0.33
	24	8	3.88	1.33	0.32
	24	12	3.89	1.32	0.32
Exercise	16	1	4.08	1.38	0.32
	16	4	3.87	1.37	0.33
	16	8	3.98	1.37	0.32
	24	8	3.98	1.37	0.34
	24	12	3.93	1.37	0.33

表 4 に示す実験結果より, ターンアラウンドで見て 3.4 ～ 4.9(sec) で伸長が可能である. これはリアルタイムとは言えないので, 投機的実行を行う必然性があり, かつまた, 投機的実行によって隠蔽または縮小が可能なレベルである [4].

5 結論と今後の展望

前節で得られた第一次評価の結果はいずれも, 本研究の目的に対してポジティブな結果を提供している. すなわち, 並列環境における投機的実行を用いることによって, ビデオ DB の格納可能容量を拡大することは, 実現可能であると期待できる.

今後, 以下について検討を進めていく予定である. (i) GOP に切り出しての評価, (ii) 並列環境で実際に投機的実行を行っての評価, (iii) 種々の時間長, 特に 1 ～ 2 時間といった長時間番組に相当する時間長の映像の場合の, プロセッサ割り当てのスケジューリングに関する検討.

参考文献

[1] 池口祐子, 村山和宏, 上田和紀：『見込み計算を用いたニュースリーダの応答性改善法』, 情報処理学会論文誌 6 月号, Vol.38, No.6, pp.1235-1244, 1997.

[2] 小川政行, 石川佳治, 植村俊亮：『圧縮映像データベースシステムにおける映像演算と実現手法』, 情報処理学会データベースシステム研究会研究報告, DBS-106-1, pp.1-8, 1996.

[3] 私信, 東京大学喜連川優助教授, 於第 8 回データ工学ワークショップ, 1997 年 3 月 19 日.

[4] 高山 毅, 弘中哲夫, 藤野清次：『投機的問合せ処理：データベースのためのもう一つの並列処理』, 第 8 回データ工学ワークショップ, pp.263-268, 1997.

[5] 野中和明, 増永良文：『MPEG-1 動画像データベースシステムのプロトタイプ構築』, アドバンストデータベースシンポジウム'96, pp.107-114, 1996.

[6] 藤原 洋：最新 MPEG 教科書, アスキー出版局, 294pp., 1994.

[7] 山名早人ほか：『投機的実行研究の最新動向とタスク間投機的実行の有効性』, 第 51 回情報処理学会全国大会講演論文集 (6), pp.75-76, 1995.

[8] Bestavros, A., Braoudakis, S.: "Value-cognizant Speculative Concurrency Control", *Proc. 21th VLDB*, pp.122-133, 1995.