ストリーム処理システムにおける映像データ管理機構

高橋 翼† 川島 英之†,‡ 北川 博之†,‡

†筑波大学大学院システム情報工学研究科〒305-8573 茨城県つくば市天王台 1-1-1
‡筑波大学計算科学研究センター〒305-8573 茨城県つくば市天王台 1-1-1
E-mail: †violencello@kde.cs.tsukuba.ac.jp, ‡{kawashima, kitagawa}@cs.tsukuba.ac.jp

あらまし 近年, ネットワーク環境の発達やカメラの低価格化などにより, 映像ストリームが普及してきている. 映像ストリームの高度利用要求の 1 つにタプルストリームとの統合利用がある. 映像ストリームはバイナリデータであるため関係演算処理の適用が難しい. また, 映像データをタプルに格納するとサイズが原因となり通信ボトルネックが生じる. また, アプリケーションは映像データをタプル形式で扱うことは容易ではない. 本研究では, これらの問題を解消するために, 映像データ管理機構を提案する. 提案機構により, RTSP 通信を使った映像データ取得と映像データ特徴量に対する関係演算処理を実現する. 試作システムを実装し, 実験により有用性を評価する.

キーワード ストリームデータ処理, 映像ストリーム

A Media Manager for Stream Processing Systems

Tsubasa TAKAHASHI† Hideyuki KAWASHIMA†,‡ and Hiroyuki KITAGAWA†,‡

† Graduate School of Systems and Information Engineering, University of Tsukuba Tennoudai 1-1-1, Tsukuba City, Ibaraki, 305-8573 Japan
‡ Center for Computational Sciences, University of Tsukuba Tennoudai 1-1-1, Tsukuba City, Ibaraki, 305-8573 Japan
E-mail: †violencello@kde.cs.tsukuba.ac.jp, ‡{kawashima, kitagawa}@cs.tsukuba.ac.jp

1. はじめに

近年，ネットワーク環境の発達やカメラの低価格化などにより，映像ストリームの配信，取得，蓄積が容易となってきた．そのため，小規模なものではペットの留守番監視システムや家の防犯システム，大規模なものでは，ロンドンの監視カメラネットワークである Ring of Steel などが実際に利用されている．

映像ストリームの普及に伴い，その処理要求も複雑化している．例えば，あるカメラに特定の人物が映っている時のみ映像を配信してほしいといった要求がある．このように，監視システムにおいて利用者の求めている情報は膨大な映像データの一部分であることが多い．そのため，映像ストリームの配信，取得，蓄積は選択的に行われることが望まれる．映像ストリームを選択的に処理するためには映像の特徴量が必要となる．また，各種センサデータとの異種統合利用により更に高度な処理が可能となる．

近年，一般的なストリームデータ処理のための，ストリーム処理システムというミドルウェアの研究開発が盛んに行われている[1,2]．これはストリームデータを利用したアプリケーション開発のための基盤システムであり，これを用いることによってアプリケーション開発の負担を軽減できる．ストリーム処理システムは，利用者が処理対象の情報源を明記した問合せを登録することにより，明記された情報源から配信されるストリームデータに対して連続的に処理を行う．そして，アプリケーションプログラムはその処理結果を API 経由で利用することが出来る。処理体系は関係演算体系をベースとしており，利用者は一般的なデータベースで用いられる関係演算を利用し，問合せを記述することができる。また，関係演算処理を行うため，ストリームデータはタプル形式で表現される．多種多様なストリームデータをタプル形式で表現することにより，それらを統合的に扱うことが可能となる。

我々の研究室では StreamSpinner[3,4,5]というストリーム処理システムの研究開発を行ってきた．映像ストリームの統合に関しては予備的な研究を行っており，フレームのシーケンスである映像データを，一定枚数フレームのオブジェクトとしてまとめることによりタプル形式に変換している．しかし，この統合手法では MPEG 形式の様なフレーム間予測を用いた映像ストリームを扱う際に問題が生じる．

フレーム間予測とは，前方，または後方のフレームとの差分画像のみを符号化することにより圧縮率を高める符号化方式である．圧縮率が高いため，映像データの蓄積や通信に向いており，実際に多くの映像デー

タに用いられている．フレーム間予測のフレームの種類には前方向フレームの差分画像を符号化するP(Predicate)フレーム，前方向，後方向フレームを選択し，差分画像を符号化する B(Bi-directional Predicate)フレーム，フレーム間予測無しで展開可能なI(Intra-coded)フレームがある．P フレームは前方の I フレームまたは P フレームを参照し，B フレームは前方，後方の I フレームまたは P フレームを参照し，符号化される．そのため，P フレームと B フレームは，I フレームがなければ展開できない．また，I フレームは P フレーム，B フレームと比べて数が少なく，I フレームの数が少ないほど，圧縮率は上がる．

タプルの生成間隔が I フレームの生成間隔と合わない場合，展開不可能な P フレーム，B フレームがタプルに含まれてしまう問題が生じる．これは，タプルに一定枚数のフレームを格納する際，参照に必要な I フレームが必ずしも同じタプルにあるとは限らないためである．

I フレームの単位でタプルを生成した場合，映像データの展開は可能だが，タプルの生成間隔を自由に決めることができなくなる．例えば，I フレームが 1 秒間に 1 回しか来ないときに，1 秒間に 2 つ以上のタプルを生成すると，I フレームが入っていないタプルのフレームはすべて展開不可能となる．また，映像ストリームから I フレームを見つけ出す必要があるため，バイナリストリームに対しパターンマッチングを行う処理が必要となる．これは，バイナリストリームから I フレームのヘッダを見つけるためであり，多大な計算量がかかる．そして，これらの処理系はアプリケーション開発者が作る必要がある．

このように，映像ストリームをタプルストリームに変換する統合手法には問題点がある．そのため，本研究では映像ストリームとタプルストリームのそれぞれの特性にあった処理を行いつつ，相互に連携した利用を可能とする映像データ管理機構を提案する．提案機構を利用することにより，RTSP 通信を使った柔軟な映像データ取得を可能とし，映像データの特徴量のみをタプルに格納することにより，関係演算処理を実現する．

本論文の構成は次のようになる．まず 2.で先行研究である StreamSpinner について説明し，3.で映像データ管理機構を提案する．4.で提案機構と連携するシステムのプロトタイプについて述べ，5.で提案機構の検証実験について述べる．最後に 6.でまとめと今後の課題を述べる．

2. StreamSpinner

2.1. アーキテクチャ

StreamSpinner は我々の研究室で研究開発を行ってきたストリーム処理システムである．StreamSpinner は映像ストリームやセンサストリームなどの異種ストリーム情報源やリレーショナルデータベースなどの蓄積データの統合利用を可能としている．問合せは SQL をベースとした文法に基づいているため，SQL に慣れている利用者は容易に記述できる．

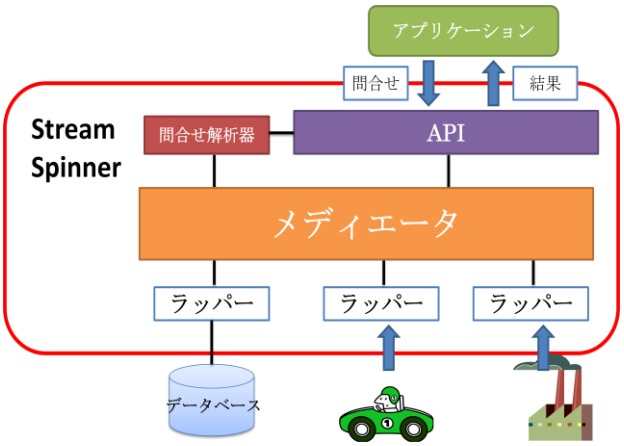

図 1　StreamSpinner のアーキテクチャ

StreamSpinner のアーキテクチャの概要を図 1 に示す．利用者から与えられた問合せは問合せ解析器によって内部形式に変換され実行プランが生成される．メディエータでは実行プランにしたがってストリームデータに対する処理を行う．

各情報源からのデータの統合にはラッパーを利用する．ラッパーは情報源固有のデータ形式をシステムで利用できるタプル形式に変換する機能を持つ．

2.2. 問合せ記述

問合せ記述例を図 2 に示す．StreamSpinner では SQL ライクな問合せ記述が可能である．MASTER 節で指定された情報源からデータが到着するたびに SELECT 節以下で書かれた問合せを実行させる．FROM 節に書かれた[1min]は，演算評価の際に処理対象のデータの決める時間的な幅であり，これをウインドウ幅と呼ぶ．ウインドウ幅には，時間ベースとタプルベースの 2 つが存在する．タプルベースウインドウ幅は無限に続くストリームを指定されたタプル数で区切り処理対象とする．時間ベースウインドウ幅は指定された時間幅でタプルを区切り処理対象とする．図 2 に示される[1min]は，時間ベースウインドウ幅であり，演算評価時刻から 1 分前までに到着したデータを処理対象としていることを表す．

2.3. 映像データの取り扱い

StreamSpinner では映像データをタプル形式に変換するために，映像専用のデータ型である Video 型を設けている．Video 型は一定枚数のフレームを保持する．

映像データはバイナリデータであるため，利用者が映像データの内容に基づいた問合せを記述することは難しい．そのため，問合せ記述を容易にするために，テキスト形式の特徴量が映像データからしばしば抽出される．特徴量の例には「誰が写っているか」,「何を行っているか」などが挙げられる．

```
MASTER  Sensor
SELECT  *
FROM  Sensor[Now], Camera[1min]
WHERE  Sensor.Temp>=30
```

図 2　問合せ記述例

3. 映像データ管理機構の提案

映像ストリームをタプルストリームに変換する統合手法では，MPEG-4 動画等の圧縮された動画データを扱う場合，展開不可能なフレームがタプルに格納される可能性がある．そのため，映像データをタプルの生成間隔に合わせて一定枚数のフレームに分割するのではなく，映像データに適した格納方式により保持されるべきである．

映像ストリームとタプルストリームを統合的に扱うためには，両ストリームに対して連携処理が行える必要がある．StreamSpinner では映像ストリームをタプルストリームに変換することにより連携処理を可能としていたが，タプルに映像データを格納すると展開不可能なフレームが生じる問題が発生する．そのため，映像ストリームをタプルストリームに変換せずに連携処理を可能とする新しい連携手法が求められる．

問合せ処理には映像データの特徴量が必要となる．特徴抽出に MPEG-4 動画等の圧縮された動画データを用いる場合，ユーザ定義関数では特徴抽出処理を行う前にフレームの抽出や展開処理等の前処理が必要となる．特徴抽出を行いやすい静止画を提供することで，このような前処理を省くことができ，特徴抽出が容易化され，ユーザ定義関数製作者の負担が軽減する．

StreamSpinner では映像ストリームをタプルストリームに変換して扱っていたため，結果の取得に SpinletAPI というタプルデータを取得するためのインターフェースを用いていた．映像ストリームをタプルストリームに変換せずに扱うためには，タプルデータを取得するためのインターフェースとは別に映像データを取得するためのインターフェースが必要となる．

また，ネットワーク環境の発達やカメラの低価格化などにより，映像ストリーム情報源が急速に増加しているため，多くの映像ストリームを扱える必要がある．

以上より，映像ストリームとタプルストリームの統合の際の要求事項を整理すると以下のようになる．(1)映像データに適した映像データの格納，(2)映像ストリームとタプルストリームの連携利用，(3)特徴抽出の容易性，(4)映像取得に適したインターフェースが機能的な要求として挙げられ，(5)分散環境で動作し，複数の映像ストリームを扱えることが性能的な要求として挙げられる．

本研究では，(1)から(5)の要求事項を満たす映像データ管理機構を提案する．(1)の要求を満たすために提案機構では取得した映像データを MPEG-4 形式のファイルとして一時保持する．(2)の要求を満たすため，提案機構とストリーム処理システムの間に，映像ストリームとタプルストリームを連携するためのラッパーを設ける．ラッパーでは提案機構から映像データを取得し，特徴抽出を行う．抽出した特徴量からタプルを生成し，ストリーム処理システムへ送る．このように，映像ストリームの特徴量のみをタプル化し，ストリーム処理システムへ送ることにより，連携処理が可能となる．(3)の要求を満たすため，提案機構はラッパーに JPEG 画像を提供する．JPEG 画像は OpenCV[8]などの画像処理ライブラリに対応しているため，利用者はライブラリを用いて簡単に特徴抽出を行うことができる．(4)の要求を満たすため，提案機構はアプリケーションへの映像配信に RTSP 通信を用いる．RTSP はメディアデータのストリーミング配信に用いられる標準的なプロトコルであり，アプリケーションは rtsp://から始まる URL にアクセスすることにより，簡単に動画を再生することができる．提案機構では MPEG-4 動画と JPEG 画像の 2 種類の映像データを扱うため，ストリーム情報源より 2 種類の映像ストリームを取得する．MPEG-4 動画の取得には RTSP 通信，JPEG 画像の取得には HTTP 通信を想定する．また，(5)の要求を満たすため，提案機構は分散環境で動作し，他ノードのアプリケーションやストリーム処理システムと協調動作が可能となっている．

3.1. アーキテクチャ

映像データ管理機構のアーキテクチャを図 3 に示す．映像データ管理機構はストリームマネジャ，ファイルマネジャ，ストリーミングサーバ，そしてメディアストレージから構成される．ストリームマネジャはクライアントプログラムから映像ストリーム格納要求を受け取り，映像ストリーム情報源と RTSP 通信を始め，MPEG 動画を取得し，メディアストレージに格納する．

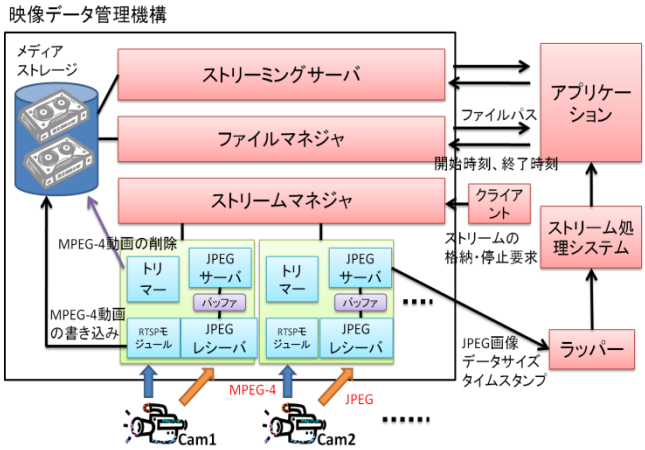

図 3　映像データ管理機構

それと並行して，映像ストリーム情報源から特徴抽出に必要な JPEG 画像の取得を開始する．ストリーム処理システムのラッパーは，この JPEG 画像を使い，特徴抽出を行い，タプルを生成する．このとき，MPEG 動画がメディアストレージに格納されているため，タプルに JPEG 画像を格納する必要はない．ストリーム処理システムのアプリケーションは，結果タプルを受けとった際に，結果に関する MPEG 動画を取得するために，MPEG 動画のファイルパスを取得する必要がある．このとき，ファイルマネジャに欲しい動画の情報を送ることによって，MPEG 動画のファイルパスが取得できる．最後に，アプリケーションがストリーミングサーバに RTSP 通信でアクセスすることにより，MPEG 動画を取得することができる．

次にストリームマネジャとファイルマネジャについて，詳しく説明する．

3.1.1. ストリームマネジャ

ストリームマネジャはクライアントプログラムから映像ストリーム格納要求を受け取り，映像格納プロセスを生成する．映像ストリーム格納要求の例を図 4 に示す．クライアントプログラムは，ストリーム名，ストリーム情報源のアドレス，情報源にアクセスするためのユーザ情報，JPEG サーバのポート番号，MPEG-4 動画取得に用いるポート番号，JPEG 画像のファイルパス，MPEG-4 動画のファイルパス， JPEG 画像の取得間隔(マイクロ秒)，MPEG-4 動画の量子化単位(秒)，蓄積動画の一時保持期間(秒)をストリームマネジャに与える必要がある．映像格納プロセスはこれらの情報をもとに RTSP モジュール，JPEG レシーバ，JPEG サーバ，トリマーの 4 つのスレッドを生成する．クライアントプログラムは映像ストリーム格納要求を複数送ることで，複数の映像格納プロセスが生成され，複数の

ストリーム名	camera1
情報源のアドレス	192.168.11.243
ユーザ情報	xxxx
JPEGサーバポート	12000
RTSP通信ポート	9000
JPEG画像のパス	/SnapshotJPEG?Resolution=320x240
MPEG-4動画のパス	/nphMpeg4/nil-640x480
MPEG-4動画量子化単位	15 sec
一時保持期間	300 sec
JPEG取得間隔	1000000μ sec

図 4　ストリーム格納要求の例

ストリーム情報源からデータを取得することができる．次に各スレッドについて説明する．

RTSP モジュールは情報源に RTSP 通信でアクセスする．そして，逐次 MPEG 動画を取得し，メディアストレージにファイルとして格納する．

JPEG レシーバは情報源に HTTP 通信でアクセスし，JPEG 画像を取得する．そして，JPEG 画像にタイムスタンプを付与し，バッファに格納する．JPEG サーバは，ラッパーから JPEG 画像取得要求が来たときに，バッファにある JPEG 画像とそのデータサイズ，タイムスタンプを返す．

トリマーはメディアストレージに格納されている一時保持期間を過ぎた MPEG 動画ファイルを定期的に削除する．

3.1.2. ファイルマネジャ

ファイルマネジャは入力としてストリーム名と動画の開始時刻，終了時刻を受け取り，出力として RTSP 通信に必要な動画ファイルパスとその動画の開始時刻を返す．アプリケーションは，このパスに RTSP 通信でアクセスすることにより，必要とする MPEG 動画を取得することができる．

ファイルマネジャはアプリケーションから必要な情報を受け取ると，メディアストレージにアクセスし，MPEG 動画ファイルを検索する．アプリケーションから与えられた開始時刻，終了時刻間の映像が 1 つの動画ファイルに含まれる場合，そのファイルパスを返す．もし，開始時刻，終了時刻間の映像が複数の動画ファイルに跨る場合，それらの動画ファイルをすべて結合し，新しい動画ファイルを生成する．そして，そのファイルパスを返す．このように，ファイルマネジャはアプリケーションから与えられた開始時刻と終了時刻の時間幅を包含する動画のファイルパスを返す．

4. プロトタイプシステムの実装

4.1. 映像データ管理機構

映像データ管理機構のプロトタイプシステムを実装した．開発言語は C++，開発プラットフォームは

図 5　ラッパープログラム

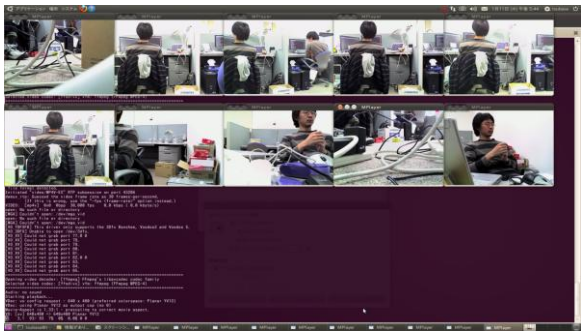

図 6　アプリケーションプログラム

Ubuntu10.04 を用いた．また，ストリーミングサーバに RTSP 通信に対応したオープンソースソフトウェアである Darwin Streaming Server[6]を用いた．

4.2. ラッパープログラム

本研究では評価実験のために 2 つのラッパープログラムを実装した．1 つ目は JPEG サーバより取得した JPEG 画像から顔を検出し，検出された顔の数を属性値として持つタプルを生成するラッパーである．本研究ではこれを顔検出処理ラッパーと呼ぶ．2 つ目はサーバより取得した JPEG 画像と，予め取得した背景画像との差分値を計算し，それを属性値として持つタプルを生成するラッパープログラムである．本研究ではこれを背景差分処理ラッパーと呼ぶ．本研究ではこれらの処理を OpenCV が提供するライブラリを用いて実装した．また，ラッパープログラムの処理間隔を自由に調整できるようにした．これにより，タプルの生成レートを制御できる．

顔検出処理ラッパーでは，cvHaarDetectObjects 関数を用いて顔検出を行った．また，背景差分処理ラッパーでは，cvAbsDiff 関数を用いて，最新画像と背景画像との差分画像を生成した．そして，cvNorm 関数を用いて差分画像のノルムを計算し，その値を差分値とした．

図 5 にラッパープログラムのスナップショットを示す．左が顔検出処理ラッパー，右が背景差分処理ラッパーとなる．評価実験においては，計算量削減のため，これらの画像表示は行っていない．

マシン 1，2

OS	Ubuntu 10.04
CPU	AMD Athlon™ 64 X2 Dual Processor 3800+
Memory	1.9GB

マシン 3

OS	Ubuntu 10.04
CPU	Intel(R) Core(TM) 2 Duo CPU E8600 @ 3.3GHz
Memory	2.0GB

マシン 4

OS	Ubuntu 10.04
CPU	Intel(R) Core(TM) 2 Quad CPU Q6700 @ 2.66GHz
Memory	3.3GB

図 7　実験用マシンのスペック

4.3. アプリケーションプログラム

実験評価のためにアプリケーションプログラムを実装した．アプリケーションプログラムはストリーム処理システムに連続的問合せを登録し，結果タプルを逐次受け取る．結果タプルを受け取った際に，ファイルマネジャにアクセスし，MPEG-4 動画のファイルパスを取得する．このとき，取得動画の開始時刻，終了時刻は結果タプルにあるタイムスタンプを利用した．こうすることにより，タプル結果に基づく MPEG-4 動画にアクセスすることができる．また，MPEG-4 動画ファイルはクライアントから与えられた量子化単位に基づき生成されるため，タプル結果を受け取った時点では動画ファイルが作られていないことがある．そのため，アプリケーションプログラムはタプル結果を受け取った後，量子化単位時間だけ待ち，ファイルマネジャにアクセスする．ファイルパス取得後は，RTSP 通信に対応したオープンソースソフトウェアのメディアプレイヤーである MPlayer[7]を用いて，MPEG-4 動画を再生する．

図 6 に分散環境で提案機構とストリーム処理システムを動かしたときのアプリケーションプログラムのスナップショットを示す．ここでは，10 の映像ストリームに対し，問合せを登録し，タプル結果に基づく MPEG-4 動画を再生している．

5. 評価実験

3.で述べた性能的要求を満たすか評価するため，プロトタイプシステムを用いて実験を行った．最初にストリームマネジャとラッパープログラムの処理性能を測定し，単一のマシンで扱える処理の限界について調査する．次に，分散環境で連携するシステムと協調して動かしたときの CPU 使用率を測定し，連携処理が正しく行われていることを確認する．また，複数マシンを使うことで，スケーラビリティが向上することを確

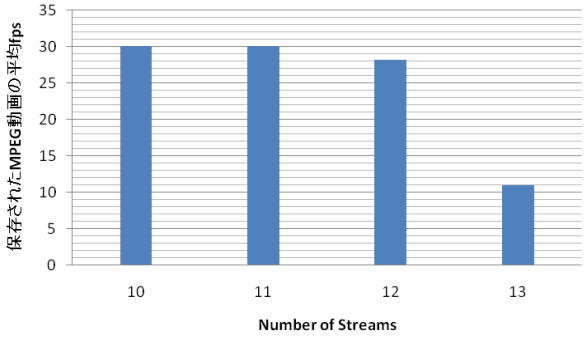

図 8 ストリームマネジャの実験結果

認する．

実験では全部で 4 台のマシンを用いた．各マシンのスペックを図 7 に示す，CPU 使用率の測定には vmstat コマンドを用いた．本研究ではそれぞれのマシンを，マシン 1，マシン 2，マシン 3，マシン 4 と呼ぶ．映像ストリーム情報源として Panasonic のネットワークカメラ BL-C131 を 11 台用いた．ストリームマネジャの実験に限り，予め録画しておいた 30fps の MPEG-4 動画ファイルを仮想カメラとして使用した．ネットワーク環境は，4 台のマシンと 11 台のカメラを繋いだローカルエリアネットワークを用いた．また，本研究室で研究開発を行っているストリーム処理システムのプロトタイプを分散実験に用いた．

5.1. 単一マシンでの実験

5.1.1. ストリームマネジャの映像取得性能

ストリームマネジャについて，映像ストリームの数を変化させたときの蓄積 MPEG 動画ファイルの fps の変化について調べた．このとき，映像の量子化単位は 15 秒，JPEG の取得間隔は 1 秒とした．ストリームマネジャを 1 分 30 秒動かし，メディアストレージに保存された MPEG 動画ファイルの平均 fps を調べた．実験結果を図 8 に示す．実験結果より，ストリーム情報源の数が 12 を超えてから，書き込まれた動画に欠損が出始めているのがわかる．これは，提案機構に届く MPEG 動画が減少していたため，ネットワーク帯域の圧迫によるパケットロスが発生していると思われる．

5.1.2. ラッパーの処理性能

単一マシンで扱えるストリーム数の限界を調べるために，単一マシンで提案機構とラッパープログラムを動かし，ラッパー数(ストリーム数)を変化させたときのラッパープログラムの処理性能の変化について調べた．このとき，MPEG-4 動画の量子化単位を 15 秒，JPEG 取得間隔を 0 秒とした．

初めに，顔検出処理ラッパーを動かしたときの実験

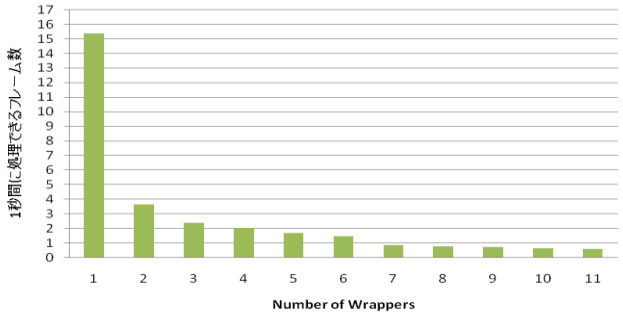

図 9 顔検出処理ラッパーの実験結果

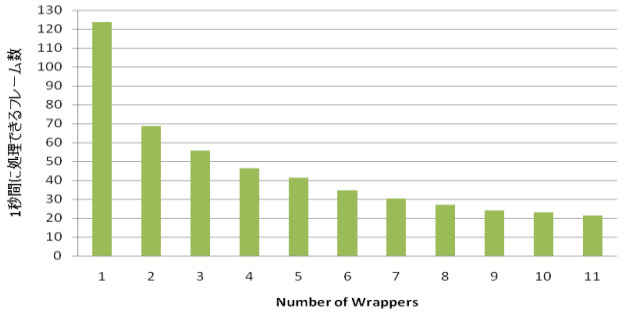

図 10 背景差分処理ラッパーの実験結果

結果について説明する．図 9 にラッパー処理間隔 0 秒のときの実験結果を示す．グラフより，ストリーム数 1 のとき毎秒 15 フレーム程度，ストリーム数 5，6 のとき毎秒 1 フレーム程度となり，ストリーム数 7 を超えると毎秒 1 フレームも処理が追いついていないことがわかる．

次に背景差分処理ラッパーを動かしたときの実験結果について説明する．図 10 にラッパー処理間隔 0 秒のときの実験結果を示す．グラフより，1 ストリームでは毎秒 120 フレーム，11 ストリームでも毎秒 20 フレーム程度処理できている．このように，特徴抽出処理が軽い場合，単一マシンでも多くのストリームを扱えることがわかる．

以上より，計算量の多い特徴抽出を行う際，単一マシンでは限界があることがわかった．また，計算量の少ない特徴抽出を行う際，単一マシンでも多くのストリームを扱えることがわかった．

単一マシンの実験より，複数の映像ストリーム情報源を扱うときは，単一マシンでは限界があるため，分散環境で提案機構を動かす必要があることがわかった．

5.2. 分散環境での実験

マシン 3 台を用いて映像データ管理機構，ストリーム処理システム，顔検出処理ラッパー，アプリケーションプログラムを 3 分間動かした．本研究では，これを分散実験 1 と呼ぶ．このとき，MPEG-4 動画の量子化単位 15 秒，JPEG 画像取得間隔 0 秒，ラッパー処理

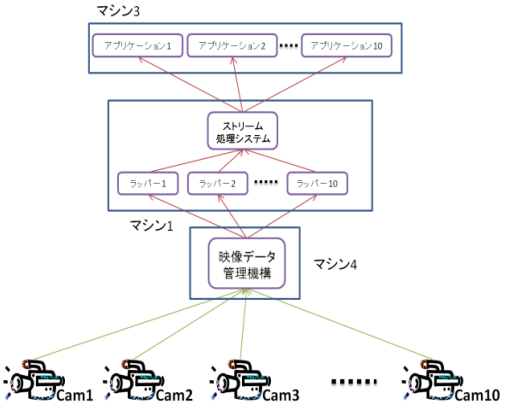

図 11　分散実験 1 の実験環境

間隔 0 秒とし，ストリーム情報源の数を 10 とした．各情報源のスキーマは int 型の 3 つの属性を持ち，それぞれタプル ID，タイムスタンプ，フレームから検出された顔の数を表す．問合せ処理は検出された顔の数でタプルのフィルタリングを行っており，ここでは，JPEG 画像から顔が 1 つ以上検出されたときに問合せ結果がアプリケーションプログラムに配信される．

実験環境を図 11 に示す．マシン 4 で映像データ管理機構を動かし，10 台のカメラから映像ストリームを取得する．そして，マシン 1 で 10 のラッパープログラムとストリーム処理システムを動かし，マシン 3 で 10 のアプリケーションプログラムを動かした．

映像ストリーム管理機構を動かしたマシン 3 の実験結果を図 12 に示す．CPU 使用率は 0%に近い値で正常に動作した．

アプリケーションプログラムを動かしたマシン 3 の実験結果を図 13 に示す．約 30 秒に 1 回結果タプルを受け取り，MPEG-4 動画をストリーミングサーバから取得しているため，グラフに 6 つの山が確認できる．動画再生中は CPU 使用率が最大 40%程度となっている．

顔検出処理ラッパーを動かしたマシン 1 の実験結果を図 14 に示す．CPU 使用率は 90%に達しているが，各ラッパーの処理フレーム数は毎秒 1 枚にも満たなかった．そのため，このような環境では，1 秒間に 1 回顔検出処理を行い，タプルを生成して欲しいといった要求に対応することができない．

次にラッパープログラムとストリーム処理システムの処理に 2 台のマシンを用いて同じ実験を行った．本研究では，これを分散実験 2 と呼ぶ．

実験環境を図 15 に示す．マシン 1，2 でラッパープログラムとストリーム処理システムを動かす．マシン 1 でカメラ 1 からカメラ 5 まで，マシン 2 でカメラ 6 からカメラ 10 までの問合せ処理をおこなった．

映像データ管理機構を動かしたマシン 4 とアプリケーションプログラムを動かしたマシン 3 の CPU 使用率は分散実験 1 と同程度の結果となった．また，ラッパープログラムとストリーム処理システムの処理を分散したマシン 1 とマシン 2 も，両マシンともに CPU 使用率は 90%程度と分散実験 1 と同様の結果になった．しかし，各ラッパープログラムが 1 秒間に処理できるフレーム数は 2 枚弱と増加した．各実験における平均処理フレーム数を図 16 に示す．このように，重い処理を複数マシンで分散処理することにより，多くのストリームに対応できる．

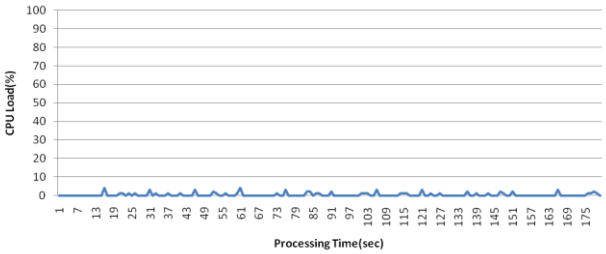

図 12　マシン 4 の実験結果

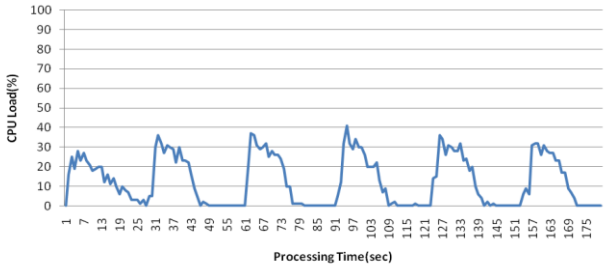

図 13　マシン 3 の実験結果

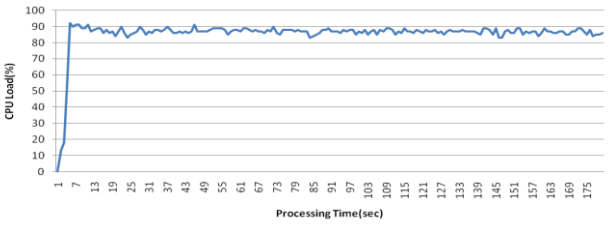

図 14　マシン 1 の実験結果

以上より，分散環境においても，映像データ管理機構とストリーム処理システムの連携処理は正常に動作することが確認できた．また，複数マシンを用いることによって，スケーラビリティが高まることが確認できた．

5.3. 映像データ管理方式に関する議論

本研究では，先行研究である StreamSpinner との比較実験は行わなかった．しかし，マルチメディア配信に適したプロトコルである RTSP 通信を用いた映像取得が行え，タプルの生成間隔と映像データの量子化単

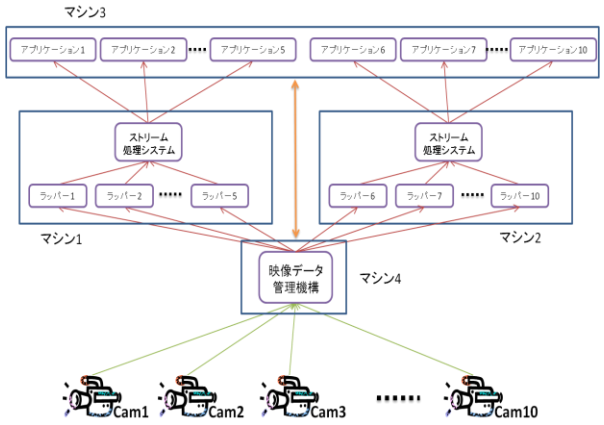

図 15　分散実験 2 の実験環境

位を独立に決めることができるという点で提案機構は優れている．

映像ストリームをタプルストリームに変換する場合，様々な問題が生じる．例えば，I フレームが 1 秒間に 1 枚含まれる映像ストリームに対して，毎秒 10 タプル生成したいという要求があったときに，1 秒間に生成される 10 タプルの内，I フレームを含むタプルは 1 つだけとなる．このとき，I フレームを含まない残りの 9 つのタプルの映像データはそのまま再生することができない．このように，映像データをタプル形式に変換すると，タプルの生成間隔が短いとき，アプリケーションプログラムが再生できない映像データを受け取る可能性が非常に高くなる．タプルの生成間隔を 1 秒に 1 回としても，I フレームの出現間隔とタプルの生成間隔が合わなければ，多くの展開不可能なフレームがタプルに含まれることになり，映像の再生に支障をきたす．また，映像データはタプル形式で配送されるため，映像を再生するにはタプルに格納された映像データを一度ファイルとして書き込み，動画プレイヤーで読みこむ処理や，映像データを展開し画像表示ライブラリなどを使って再生する処理などが必要になる．これらの処理は，アプリケーション開発者が実装する必要があるため，アプリケーション開発コストが増加する．

提案機構では，タプルの生成間隔を短くしても，映像データ量子化単位を I フレームの出現間隔に対して大きく設定できるため，必ず再生できる映像データを取得することができる．また，すべての映像データを一定期間保持することで，従来のストリーム処理システムではできなかった，過去のデータへのアクセスが可能となった．これにより，保持されている任意の映像データを RTSP 通信で簡単に取得することができる．このように，提案機構では，上記の問題点を解消することができる．

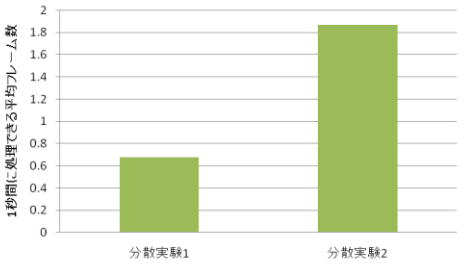

図 16　顔検出処理ラッパーの処理性能の変化

6. 終わりに

本研究では，ストリーム処理システムと連携する映像データ管理機構を提案した．提案機構を用いることにより，映像データをタプル変換することなく，結果タプルに基づく映像データを RTSP 通信で取得できた．また，映像の特徴量のみをタプル化することにより，映像データと独立したタプルの生成が可能となった．

プロトタイプシステムを実装し，提案機構の評価実験を行った．実験により，複数マシンを用いることでスケーラビリティが向上することがわかった．

今後の課題として，ストリーム処理システムと連携した映像データの格納，削除が挙げられる．

謝辞

本研究の一部は，科学研究費補助金基盤研究(A)(#21240005)，科学研究費補助金若手研究(B)(#22700090)による．

参　考　文　献

[1] A.Arasu, B.Babcock, S.Babu, J. Cieslewicz, M.Datar, K.Ito, R.Motwani, U.Srivastava, and J.Widom. STREAM: The Stanford Data Stream Management System. Technical Report. Stanford InfoLab. 2004.

[2] A.Arasu, S.Babu, and J.Widom. The CQL continuous query language: semantic foundations and query execution. In VLDB journal, 15(2).pp 121-142, 2006.

[3] 渡辺陽介，秋山亮，大喜恒甫，北川博之．“映像ストリームのための映像情報統合基盤システムの提案” 日本データベース学会 Letters, Vol. 6, No. 4, pp. 13-16, 2008 年 3 月．

[4] 大喜恒甫，渡辺陽介，北川博之，川島英之，「対象情報源を動的に選択可能なストリーム処理機能の実装と評価」情報処理学会論文誌：データベース, Vol.2, No.3, pp.1-17, 2009 年 9 月．

[5] Yousuke Watanabe, Hiroyuki Kitagawa, "Query Result Caching for Multiple Event-driven Continuous Queries", Information Systems 2009.

[6] Darwin Streaming Server. http://dss.macosforge.org/.

[7] MPlayer. http://www.mplayerhq.hu/design7/news.html./

[8] OpenCV.jp. http://opencv.jp/.