社団法人　電子情報通信学会
THE INSTITUTE OF ELECTRONICS,
INFORMATION AND COMMUNICATION ENGINEERS

信学技報
TECHNICAL REPORT OF IEICE.

リアルタイム動画像解析アプリケーションフレームワークにおけるセンサ・クラウド間負荷分散の性能評価

黒崎　裕子† 　竹房　あつ子†† 　中田　秀基†† 　小口　正人†

† お茶の水女子大学　〒 112–0012 東京都文京区大塚 2–1–1
†† 産業技術総合研究所　〒 305–8560 茨城県つくば市梅園 1–1–1 中央第一

あらまし　一般家庭やオフィスビルでカメラや各種センサの情報を取得してクラウドに転送し，解析処理を行うライフログ解析アプリケーションが数多く開発されている．しかし，センサとクラウド間のネットワーク帯域やクラウド側の資源の制限により，動画像解析のようなデータ量，計算量の多い処理をクラウドでリアルタイムに行うことは困難である．我々は，センサ・クラウド間で分散処理を行う動画像解析アプリケーションフレームワークを提案しているが，センサからクラウドまでの実行環境を考慮しつつ，特徴量抽出等の前処理から解析までの各処理に対して適切な並列度を設定する必要がある．本研究では，センサとクラウドを模擬したクラスタ環境を構築し，動画像解析アプリケーションの性能を調査する．実験から，処理スレッド数を増やすと処理性能が向上すること，センサ・クラウド間が低帯域の場合はセンサ・クラウド間の分散処理が有効であることが示された．

キーワード　Apache Storm, 負荷分散，ストリーム処理

Performance Evaluation of Load Balancing between Sensors and a Cloud for a Real Time Video Streaming Analysis Application Framework

Yuko KUROSAKI†, Atsuko TAKEFUSA††, Hidemoto NAKADA††, and Masato OGUCHI†

† Ochanomizu University　2–1–1 Otsuka，Bunkyo，Tokyo 112–8610，Japan
†† National Institute of Advanced Industrial Science and Technology (AIST)　1–1–1 Umezono，Tsukuba-shi，Ibaraki 305–8560，Japan

1. はじめに

一般家庭やオフィスビルでカメラや各種センサの情報を取得して，防犯対策やセキュリティ，お年寄りや子供のための安全サービスを目的としたライフログ解析アプリケーションが数多く開発されている．それらのアプリケーションを一般家庭で採用する場合，サーバやストレージを設置して解析までを行うことは難しいため，クラウドでの処理が必要となる．しかし，動画像を含む多数のセンサデータをクラウドに送信してクラウド側で処理すると，各センサとクラウド間のネットワーク帯域やクラウドの資源の制限により，リアルタイムに処理できない可能性がある．また，動画像のような大容量ストリームデータに対して，継続的に特徴量抽出から解析までの高負荷な処理を行う必要がある．

クラウド側で全ての処理を行う方式に対し，センサ側，すなわちクライアント側にある程度の機能を持たせて，クラウド側の処理を軽減する，ファットクライアントと呼ばれるパラダイムが注目されている．また，ユーザまたは各種センサと物理的に近い場所に配置された小規模なエッジサーバと連携させクラウドの負荷を軽減しつつサービスの応答遅延を減らすことを目指したエッジコンピューティング [1] や，クラウドとデバイスの間に分散処理環境を置くことにより，クラウドへの一極集中を防ぐフォグコンピューティング [2] が提案されている．

我々はセンサ・クラウド間で分散処理を行う動画像データ解析アプリケーションフレームワークを提案している [3]．本フレームワークでは，分散ストリームコンピューティング基盤を提供する Apache Storm(以降，Storm と呼ぶ) を用いて開発しており，動画像データ解析における高負荷な処理を分散実行することができる．しかしながら，リアルタイムに動画像データ解析を行うためには，センサからクラウドまでの実行環境を考慮しつつ，各処理に対して適切な並列度を設定する必要がある．

本研究では，センサとクラウドを模擬したクラスタ環境を構築して動画像解析アプリケーションの性能を調査する．また，センサ・クラウド間のネットワーク帯域を考慮した実験を行い，

センサ・クラウド間での負荷分散の有効性を示す．実験から，Bolt スレッド数が増加すると各スレッドがデータの取得に要する時間が長くなるものの，全体として処理性能が向上することがわかった．低帯域環境では，センサ・クラウド間負荷分散を行うことで，一定経過時間に処理できる総画像数が増加し，ファットクライアントとクラウドを連携させた分散処理が有用であることが示された．

2. 関 連 技 術

我々が開発している動画像解析アプリケーションでは，センサ側とクラウド側の負荷分散の基盤として Storm を，解析にはオンライン機械学習フレームワーク Jubatus を使用する．以下に各ソフトウェアの概要を述べる．

2.1 Apache Storm

Storm は，Back Type 社によって開発された分散型のリアルタイム計算システムである [4]．現在は Twitter 社に買収されており，Twitter でのつぶやきのリアルタイム解析に用いられている．Apache Hadoop はバッチで大規模処理を行うのに対し，Storm はリアルタイムでの分散処理に特化している．また，同じストリーム処理フレームワークに Apache Spark [5] がある．Spark は，ストリーム処理を 1 秒間に受信したイベント群に対するバッチ処理の連鎖としてバッチ処理の性質を保ったまま実行するため，スループットは向上するものの，遅延は 0.5〜2.0 秒程発生して純粋なストリーム処理より遅くなる [6]．

Storm は，図 1 のような Spout と Bolt からなる Topology と呼ばれるネットワーク構造を，Storm クラスタに配備して処理を行う．Spout とはストリームを流し始める処理の起点となるプロセスで，Bolt は流れてくるデータに対する変換処理を行うプロセスを指す．ストリームとは，連続して送られる Tuple であり，標準データ型やシリアライズ・コードを追加したユーザ定義型を用いることができる．

また，Storm には Local mode と Distributed mode の 2 種類が存在する．Local mode とは，Storm クラスタを 1 台の計算機の中で実行可能なモードであり，Distributed mode は，Topology で定義した処理を，Storm クラスタ上の複数ノードに動的に負荷分散して実行するモードであり，本研究では，Distributed mode を用いる．Distributed mode では，Storm クラスタ全体を Nimbus で，個々の Slave ノードを Supervisor で管理し，Nimbus と Supervisor 間の協調を Zookeeper が行っている．Nimbus に対してコマンドを用いて Topology を流し，Supervisor 上に起動した Worker 上で Spout または Bolt の処理を実行する．Worker 数は Supervisor を起動する際に定義でき，Topology 内の定義で指定したスレッド数が Worker に分配されるようになっている．

2.2 オンライン機械学習 Jubatus

Jubatus は，Preferred Infrastructure と NTT SIC により共同開発された，オンライン機械学習フレームワークである．本来トレードオフの関係であった「ストリーム (オンライン) 処理」「並列分散処理」「深い解析」の要素を満たすオンライン機械学習フレームワークである [7]．バッチ機械学習ではモデルを更新する度に同期を行っていたが，更新頻度が少ないため，処理性能に大きく影響することはなかった．しかしオンライン機械学習をそのまま分散処理すると，学習モデルの更新頻度が多くなるため，その分同期コストが大きくなり，サーバ処理性能が劣化する問題が生じる．そのため，上記の 3 つの要素を同時に満たすフレームワークは開発されていなかった．通常，UPDATE はサーバ間でデータを共有して学習モデルを更新するが，Jubatus では，UPDATE でデータ自体は共有せず，MIX という処理の中で各サーバの学習後の解析モデルのみを共有し，リアルタイムかつ大規模な並列分散処理を可能にしている．

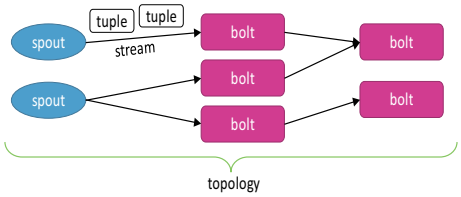


図 1　Topology の流れ

Jubatus は解析時に Datum という key-value データ形式を用いる．Datum の value には，文字列，数値，バイナリデータの 3 種類のデータが利用できる．バイナリデータには画像や音声などのマルチメディアデータなど，任意のバイナリデータを入れることが可能である．このデータから，機械学習を行う際に必要となる特徴量を Jubatus のデータ変換モジュールが抽出する．また，Jubatus の特徴ベクトル変換器は，この特徴抽出処理を JSON 形式ファイルでカスタマイズすることが可能であり，特徴抽出器にはプラグインを利用することができる．プラグインは動的ライブラリファイル (.so ファイル) からなり，JSON 形式ファイルでパスを通すことによって利用可能である．

3. 動画像データ解析フレームワーク

本節では我々が提案する動画像データ解析アプリケーションフレームワークの設計概要について説明する．動画像データ解析は，基本的に (1) 画像の取得，(2) 特徴量抽出，(3) 機械学習処理の 3 つのステップからなる．クラウド側で全てのセンサデータを収集して処理を行う場合は，センサ側で (1) の処理を行った後，動画像を含むセンサデータをクラウドに送信してクラウド側で (2)，(3) の処理を行う．一方，センサ・クラウド間負荷分散を行う場合は，クラウド側処理の負荷の軽減とセンサとクラウド間のデータ転送量削減のため，(2) の処理を一部センサ側に分散させる．

(2) の処理を全てクラウド側で行う場合は，クラウドへの通信量が大きくなり，そのオーバーヘッドによる性能劣化が懸念される．(2) の処理をセンサ側とクラウド側の両方で行う場合は，クラウドへの通信量は少なくなるものの，センサ側の特徴抽出部分の処理による負荷が大きくなってしまう．

3.1 動画像データ解析フレームワークの設計

実装したアプリケーションは図 2 に示すように次の 3 つのプロセスに分かれている．

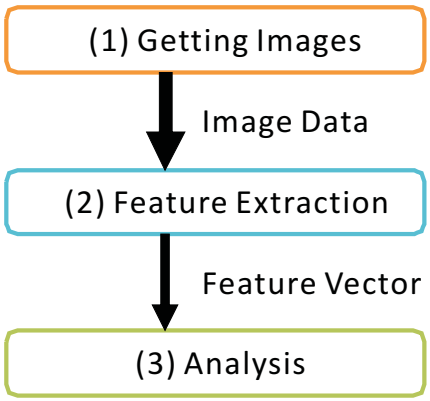


図 2　動画像データ解析アプリケーションの流れ

（1） WEB カメラからのキャプチャ
（2） Bag-of-Features を用いた特徴抽出
（3） Jubatus での画像分類処理

以下に各プロセスの詳細について述べる．

3.1.1　WEB カメラからのキャプチャ

まず，データストリームの起点となるセンサ側に，WEB カメラからの画像キャプチャ部分を実装する．OpenCV [8] を用いて WEB カメラから画像を取得し，取得した画像を構造体に格納し，(2)Bag-of-Features を用いた特徴抽出処理を行うプロセスに，構造体を渡す．

3.1.2　Bag-of-Features を用いた特徴抽出

次に，OpenCV を用いた特徴量の抽出と Bag-of-Features [9] による画像データのベクトル化である．ベクトル化したデータを (3)Jubatus での画像分類プロセスに転送する．

Bag-of-Features とは，あらかじめ複数画像の特徴量データから K-means 手法を用いて特徴量をクラスタリングした辞書データをもとに，画像から得られた局所特徴量の集合から，各グループに属する特徴量をもつ特徴点の数をヒストグラム化する手法である．

本研究で行った Bag-of-Features の抽出手順は以下のようになる．まず，OpenCV を用いて画像から局所特徴量を抽出する．局所特徴量にはいくつか種類があり，中でも有名な SIFT は，照明変化や回転，拡大縮小に不変な頑強な特徴量である．その SIFT より認識精度は少し落ちるが，特徴点検出の処理を軽量化，高速化したものが SURF である．本研究では，局所特徴量に SURF 特徴量を用いた．SURF では keypoint と呼ばれる画像中の特徴的な点をいくつか抽出する．各 keypoint は 128 次元の特徴ベクトルとなるが，抽出される keypoint の数は画像によって異なるため，画像全体の特徴ベクトルとして機械学習でそのまま使用するのは困難である．局所特徴量をクラスタリングして各クラスタの中心ベクトルを Visual Word と呼ばれる特徴的なパターンとし，辞書を作成する．この辞書を使ってある画像から抽出された特徴ベクトル群を Visual Word にマッチさせ，画像全体の特徴パターンの頻度をヒストグラムで表現する．一度辞書を作成してしまうと，Visual Word 数は一定であるため，各画像の特徴量を同じサイズのベクトルデータに変換することができる．次に，生成したベクトルデータを Jubatus で扱う形式に変換する．2.2 節で述べたように，Jubatus は解析時に Datum というデータ形式を使用する．よって，作成したヒストグラムを Datum に変換する．ヒストグラムの各値を 1 つずつリスト形式で Datum に格納し，Jubatus へ転送可能なデータに変換する．最後に，Jubatus サーバとコネクションを確立し，画像から取得した特徴ベクトルの格納された Datum を Jubatus サーバへ転送する．

3.1.3　Jubatus での画像分類処理

Jubatus では分類，推薦，線形回帰などいろいろな解析が可能であり，今回は Classifier API を用いて学習と分類を行う．ライフログ解析を行う前に train API を利用し，予め教師あり学習を行う．その学習結果を用いて分類するには，classify API を利用する．Jubatus サーバに送られてきた Datum はフィルタ，特徴抽出という 2 段階のデータ変換を経て解析される．フィルタ処理では，学習に不要なものを取り除き，特徴抽出では，フィルタされたデータから特徴を抽出する．フィルタ処理でデータからどのような要素を取り除くか，特徴抽出でどのアルゴリズムを使用し，どのように重み付けをするかは，サーバ起動時に指定する JSON ファイルで設定することが可能である．今回はフィルタはデフォルトを利用し，特徴抽出では与えられた数値をそのまま重みに利用する．分類に使用するアルゴリズムには Adaptive Regularization of Weight vectors(AROW) を選択した．学習に対する感度パラメータは 1.0 に設定する．Jubatus サーバで解析された結果は，ユーザへ送信される．

3.2　Storm を用いたフレームワークの実装

動画像データ解析アプリケーションはリアルタイム処理であることから，負荷の高い処理を適宜並行実行して高速化を図る必要があるため，Storm を用いた．3.1 節で説明した (1) は Spout，(2) および (3) は Bolt の処理として実装し，センサ側の計算機とクラウド側の計算機群で 1 つの Storm クラスタを構築してセンサおよびクラウドを用いた分散処理を実現した．本研究で用いた画像データ解析アプリケーションでは，(1)，(2) 間のデータ転送量が多く，(3) の処理の負荷が相対的に高いため，図 3 に示すように，(1) はセンサ側，(2) はセンサ側またはクラウド側で処理し，(3) はクラウド側で行われるようにした．

4.　評 価 実 験

本研究は，動画像データ解析フレームワークの特徴抽出部分をセンサ側およびクラウド側で負荷分散した場合の性能を調査するため，以下の 2 つの実験を行う．

実験 1)　Bolt スレッド数の変化に伴う性能評価実験
実験 2)　ネットワーク帯域を考慮したセンサ・クラウド間負荷分散の評価実験

また，各実験では経過時間あたりに完了したジョブ数，1 つ目の実験では処理スレッド数の変化に伴う 1 ストリームデータあたりの平均処理時間，Bolt スレッド上での Tuple 受取間隔を比較する．

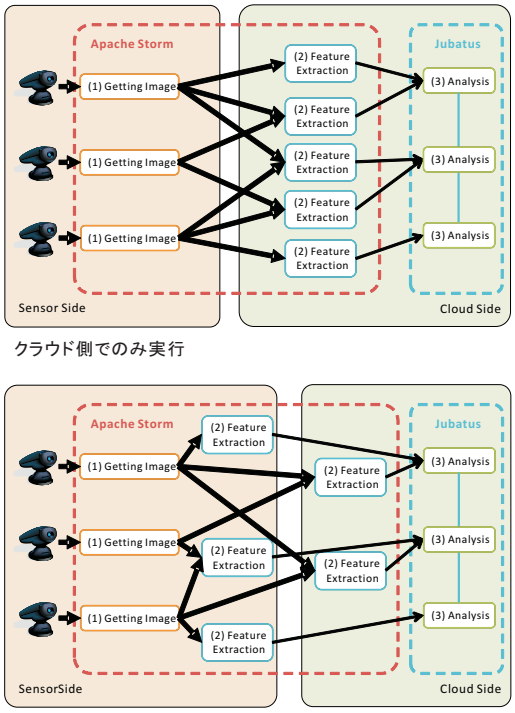


図 3　提案するフレームワーク

表 1　実験 1 に用いたパラメータ

Number of Spout Threads	2
Number of Bolt Threads	8, 16, 24, 32
Inter-arrival Time of Image Data	10[ms/tuple]
Network Bandwidth	1000[Mbps]

表 2　実験 2 に用いたパラメータ

Number of Spout Threads	2
Number of Bolt Threads	16
Inter-arrival Time of Image Data	10[ms/tuple]
Network Bandwidth	100, 1000[Mbps]

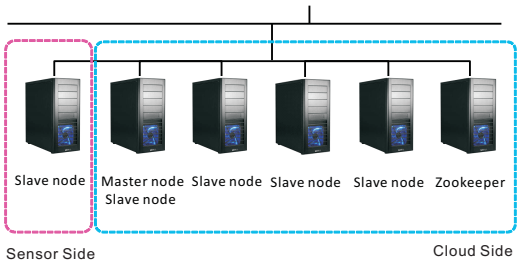


図 4　大規模クラスタの構成

4.1　実験概要

実験では，実装したフレームワークを用いて 2 種類の人の行動を判別する．予め 320 × 240 ピクセルの画像を 100 枚を用いて「ドアを開けた」状態と「イスに座った」状態を学習させる．次に，画像データを Spout で定義するストリーム生成時間毎にランダムに選出し，選出された画像データの特徴量抽出を行う．特徴量抽出における Visual Words 数は 100 に設定した．実験 1 では，表 1 に示すように，Spout スレッド数を 2，ストリームデータの生成速度は 10ms/tuple とし，Bolt スレッド数を 8，16，24，32 と変化させた．また実験 2 では，表 2 に示すように，Spout スレッド数を 2，Bolt スレッド数を 16，ストリームデータの生成速度を 10ms/tuple とし，ネットワーク帯域を 100Mbps，1Gbps と変化させて実験を行った．帯域制御には，PSPacer [10] を用いた．

構築した Storm クラスタは図 4 のような構成をとる．センサ側計算機，クラウド側計算機ともに，Intel Xeon W5590((3.33GHz，4 コア) × 2 ソケット) を使用した．Supervisor ノードを 5 台用意し，1 台がセンサ側，4 台がクラウド側計算機であると想定する．Supervisor ノードには worker をコア数に合わせて 8 つずつもたせ，Supervisor ノードの 5 台のうち 1 台は Nimbus ノードとしても機能させ，クラウド側に配置する．また，クラウド側に Zookeeper を稼働させた計算機を 1 台用意した．

4.2　実験結果

4.2.1　Bolt スレッド数の変化に伴う Storm の性能評価実験

Bolt スレッド数の変化に伴う性能評価を行った．経過時間あたりに処理できるジョブ数の計測結果は図 5 に示す．縦軸は処理したジョブ数の合計を表し，横軸は経過時間を秒で表している．Bolt スレッド数が増加するにつれて，経過時間あたりに処理できるジョブ数は増加することが分かった．しかし，スレッド数が 16 に達すると処理できるジョブ数は大きく増加しなくなった．これは生成されるデータ量を処理するのに十分な Bolt スレッド数に達したからと考えられる．また，ストリーム量に対して過剰な Bolt スレッドを用意しても性能劣化は見られなかった．

この時の 1tuple あたりの処理時間と Bolt スレッド上で Tuple を受け取る間隔を図 6 に示す．縦軸は画像 1 枚あたりの平均処理時間と Bolt スレッド上で Tuple を受け取る間隔をミリ秒で表している．横軸は Bolt スレッド数を表す．Bolt スレッド数が増加することによって，処理時間に変化はないが，Tuple を受け取ってから次の Tuple を受け取るまでの間隔が大きくなった．これは，Bolt スレッド数が増えたことによって Spout スレッドから Bolt スレッドへの Tuple の割り振りの時間がより多くかかったと考えられる．

4.2.2　ネットワーク帯域を考慮したセンサ・クラウド間負荷分散の評価実験

センサ・クラウド間の負荷分散の有無による性能比較を，ネットワーク帯域を考慮して行った．図 5 より，10ms/tuple の環境において，Bolt スレッド数が 16 以上ではジョブ数の急激な増加が見られなかったため，この実験では Bolt スレッド数を 16 と設定した．計測結果は図 7 に示す．縦軸は処理したジョブ数の合計を表し，横軸は経過時間を秒で表し，処理スレッドの配置は表 3 のようにした．センサ・クラウド間の帯域が 1Gbps と 100Mbps の場合の処理ジョブ数を比較すると，帯域を制限することによってジョブ数の大幅な減少がみられた．一方，セ

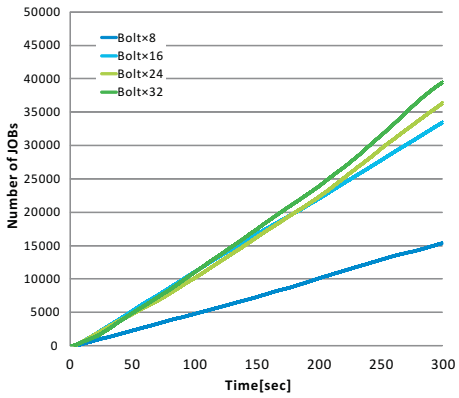


図 5　経過時間あたりの処理ジョブ数の比較

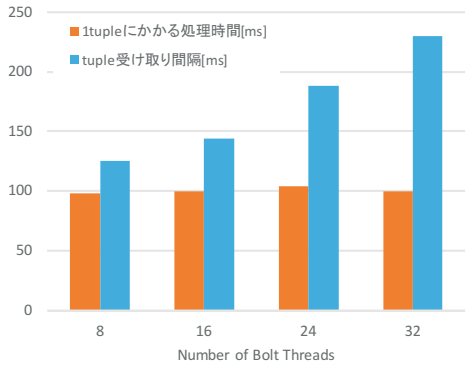


図 6　処理スレッド数の変化に伴う 1 ストリームデータあたりの平均処理時間と Bolt スレッド上での Tuple 受取間隔

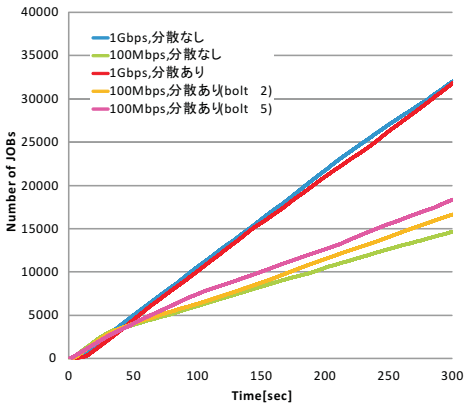


図 7　帯域を考慮した経過時間あたりの処理ジョブ数の比較

ンサ・クラウド間での分散処理の有無の比較では，1Gbps ではセンサ・クラウド間での分散処理の有無による性能差は見られなかったのに対し，100Mbps では，センサ・クラウド間での分散処理を行うことによる処理できるジョブ数の増加がみられた．よってネットワーク帯域が低い場合，センサ・クラウド間での負荷分散処理が効果的であることが示された．

5. 関 連 研 究

クラウド技術を用いた動画像データ解析は，近年数多く研究されているが，クラウド上での効率的な動画像データの内容検索や，クラウドプラットフォームを使用した動画像データのオンデマンドシステムにおけるロードバランスに焦点が当てられていた．既にストリーム動画像データの検出，追跡，解析を行う内蔵システム [11] や，人々の異常行動の追跡や活動の検出する自動監視システム [12] が開発されているが，これらは 1 ノード上で動作するシステムとして開発されており，スケーラビリティについて考慮されていない．

表 3　クライアント・クラウド間で分散処理した場合のスレッド配置

Number of Bolt Threads in Sensor Side	Slave Node (Sensor side)	Slave Node1 (Cloud side)	Slave Node2 (Cloud side)
0	Spout × 2, Bolt × 0	Bolt × 8	Bolt × 8
2	Spout × 2, Bolt × 2	Bolt × 7	Bolt × 7
5	Spout × 2, Bolt × 5	Bolt × 5	Bolt × 6

Abdullah ら [13] は，クラウド上で動画像データの取得，処理，解析を行うストリーム処理フレームワークを構築している．時間のかかる高画質の画像処理を GPU を用いることで高速化し，クラスド上でのスケーラブルなフレームワークを提案している．大量の動画像データを扱い，スケーラビリティを考慮したフレームワークの設計は本研究と近いが，全ての処理をクラウド上で行っているのに対し，本研究はファットクライアントモデルを採用している点で異なる．

6. まとめと今後の課題

本研究では，動画像データ解析アプリケーションのリアルタイム処理を実現させるため，通信量と処理の負荷を考慮して，動画像データ解析の前処理をセンサ側とクラウド側で負荷分散させることで高速化を図った．Storm を導入し，センサ側およびクラウド側で負荷分散処理する動画像データ解析フレームワークを設計，実装した．また，大規模クラスタ環境において，動画像解析アプリケーションの基本性能の調査とセンサ・クラウド間のネットワーク帯域を考慮した実験を行い，センサ・クラウド間での負荷分散の有効性を示した．

本研究では，一定速度のストリームデータを対象に実験を行ってきたが，実環境では，動体検知した時に動画像解析を行うため，ストリームデータ量が変動することが考えられる．今後は，このような大きく変動するストリームデータに対応できるよう，適切な負荷分散手法を開発する．

文　　献

[1] Milan Patel, Brian Naughton, Caroline Chan, Nurit Sprecher, Sadayuki Abeta, Adrian Neal, “Mobile-edge computing”, ETSI, Tech. Rep., 09 2014.
[2] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli, “Fog computing and its role in the internet of things,” in Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, ser. MCC ’12. ACM,2012, pp. 13–16.
[3] 黒崎裕子，竹房あつ子，中田秀基，小口正人，“ファットクライアント型リアルタイム動画像解析アプリケーションフレームワークの提案”，DICOMO2015，5C–4，2015 年 7 月．
[4] Apache Storm, https://storm.apache.org/
[5] Apache Spark, https://spark.apache.org/
[6] Matei Zaharia, Tathagata Das, Haoyuan Li, Timothy Hunter, Scott Shenker, Ion Stoica, “Discretized Streams:

A Fault-Tolerant Model for Scalable Stream Processing”, Electrical Engineering and Computer Sciences, UCB/EECS-2012-259，2012.
[7] オンライン機械学習向け分散処理フレームワーク Jubatus，http://jubat.us/ja/
[8] 画像ライブラリ OpenCV，http://opencv.org/
[9] Tomoyuki Nagahashi, Hironobu Fujiyoshi,“Object Category Recognition by Bag-of-Features using Co-occurrence Representation by Foreground and Background Information”, Machine Vision Applications，pp.413， 2011.
[10] Ryousei Takano，Tomohiro Kudoh，Yuetsu Kodama，and Fumihiro Okazaki,“High-resolution Timer-based Packet Pacing Mechanism on the Linux Operating System”，IEICE Transactions on Communications，Vol.E94.B，No.8，pp.2199-2207，2011.
[11] B. S. System,“IVA 5.60 intelligent video analysis”，Bosh Security System, Tech. Rep.，2014.
[12] Project BESAFE，http://imagelab.ing.unimore.it/besafe/.
[13] Tariq Abdullah，M Fahim Tariq，Yusuf Baltaci and Nick Antonopoulos,“Traffic Monitoring Using Video Analytics in CloudsTraffic Monitoring Using Video Analytics in Clouds”，7th International Conference on Utility and Cloud Computing，pp.39-48，2014.