DEIM Forum 2015 F2-5

ファットクライアントを利用した 動画像データ解析アプリケーションフレームワークの提案

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　〒 112–8610 東京都文京区大塚 2–1–1
†† 産業技術総合研究所　〒 305–8568 茨城県つくば市梅園 1–1–11
E-mail: †yuko-k@ogl.is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp, †††oguchi@computer.org

あらまし　カメラやセンサ等が手軽に利用できるようになり，一般家庭でライフログを取得して，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は，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画像の特徴抽出等の前処理から解析まで，全ての工程をクラウド側で行うと，各センサとクラウド間のネットワーク帯域やクラウド側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，動画像はデータ量が大きく，連続的であるため，大容量ストリームデータに対する解析処理が必要となる．本研究では，ファットクライアントの概念を取り入れ，クライアント側で画像から特徴抽出し，前処理済みの小さいデータをクラウドに収集して解析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を提案する．また，提案するフレームワークを Apache Storm を用いて実装する．

キーワード　センサデータ解析，ストリーム処理，ファットクライアント

Proposal of a Video Stream Analysis Application Framework based on the Fat Client Computing Paradigm

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

† Ochanomizu University　Otsuka 2–1–1,Bunkyo-Ku, Tokyo 112–8610, Japan
†† National Institute of Advanced Industrial Science and Technology (AIST)　1-1-1 Umezono, Tsukuba, Ibaraki 305–8568, Japan
E-mail: †yuko-k@ogl.is.ocha.ac.jp, ††{atsuko.takefusa,hide-nakada}@aist.go.jp, †††oguchi@computer.org

1. はじめに

近年ではカメラやセンサ等を手軽に利用できるようになり，一般家庭でライフログを取得して，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，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，クラウドに動画像を含む多数のセンサデータを送信し，特徴抽出等の前処理から解析までの全ての工程をクラウド側でリアルタイムに行うことは困難である．

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，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，ユーザと物理的に近く配置された小規模なエッジサーバと連携して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．

本研究ではファットクライアントモデルを取り入れた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する．動画像データはデータ量が大きく，連続的であるため，センサ側で画像から特徴抽出等の前処理を行い，前処理済みの小さいデー

タをクラウドに収集して解析する．

既発表研究 [3] により，画像処理では前処理に多くの時間がかかることが分かっている．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ため，負荷分散機能をもつ Apache Storm(以降，Storm と呼ぶ) を導入し，画像データ解析フレームワークを実装した．予備実験として，提案するフレームワークに Storm を用いた場合と Storm を用いなかった場合の処理速度を測定し，Storm のオーバヘッドを調査した．実験より，Storm を用いたことによるオーバーヘッドは無視できることが確認できた．

2. 関 連 技 術

今回提案するフレームワークには，センサ側のリアルタイム処理基盤の部分に Storm を，クラウド側の解析部分に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．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のが Spark の特徴であり，スループットは向上するものの，遅延は 0.5〜2.0 秒程発生するため純粋なストリーム処理より遅くなってしまう [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 クラスタで指定することによって処理を行う．Spout とは Steam 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の変換処理を行うプロセスを指す．Stream とは，Tuple の無限の連続であり，標準データ型を表したり，シリアライズ・コードを追加したユーザ定義型を表したりすることができる構造体のようなものである．

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，Storm クラスタを複数ノードに分散して実行するモードである．本研究では，まずはローカルモードで，画像データ解析フレームワークを実装する．

2.2 オンライン機械学習 Jubatus

Jubatus とは，NTT SIC と Preferred Infrastructure 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]．オンライン機械学習をそのまま分散処理すると同期コストが大きくなるという問題が生じるが，Jubatus では，UPDATE の処理ではデータ自体は共有せず，MIX という処理の中で学習後のモデルのみを緩やかに共有することで，並列分散処理を可能にしている．

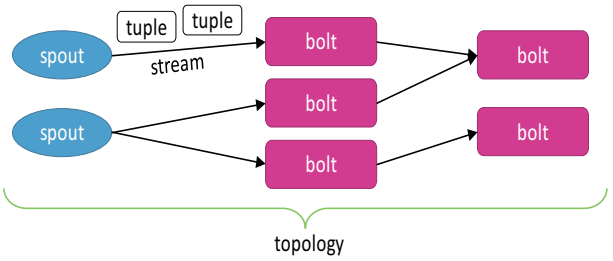

図 1 Topology の流れ

Jubatus は解析の時に Datum という key-value データ形式を用いる．Datum には 3 つのタイプの key-value が存在する．value が文字列の文字列データ，value が数値の数値データ，最後に value が任意のバイナリデータがあり，key は 3 タイプとも文字列である．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 3 つのデータから，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. Apache Storm と Jubatus を用いた動画像データ解析フレームワークの設計

本節では提案する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 (前処理)，(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，図 2 に示すようにセンサ側で (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，多数センサを扱う場合にクラウド側処理の負荷を下げるため，本研究では図 3 に示すように，(1)，(2) をセンサ側で行い，特徴量データのみをクラウド側に送信して (3) の処理を行うアプリケーションフレームワークを提案する．

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，分散型リアルタイム計算システムの Storm を導入した．センサ側端末上で Strom による分散処理で OpenCV [8] を用いた特徴抽出等の前処理を行い，クラウド側計算機で画像の分類処理を Jubatus を用いて行う．

今回は図 4 に示すようにセンサ側で特徴抽出を行うフレームワークを実装した．実装したアプリケーションは次の 3 つのプロセスに分かれている．

（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 での解析

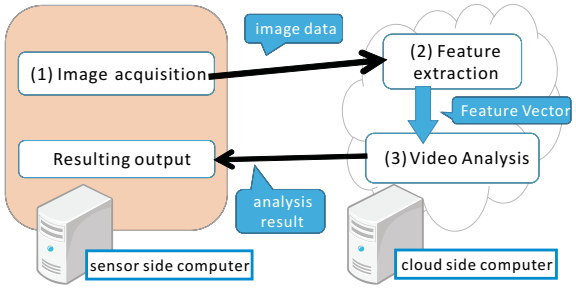

図 2　クラウド側で特徴抽出

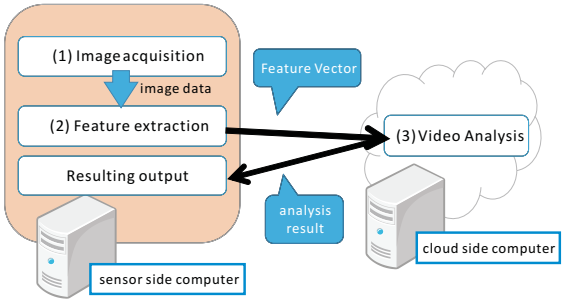

図 3　センサ側で特徴抽出

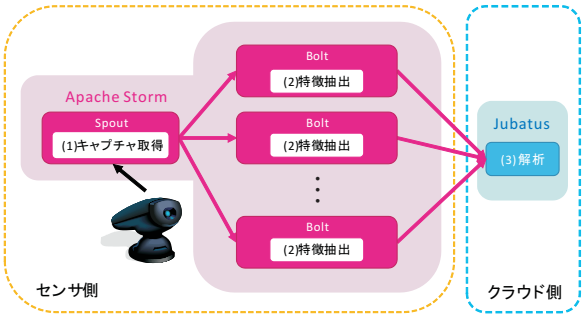

図 4　実装する Apache Storm を用いたフレームワークの概要

図 4 の最初の 2 つの (1)，(2) は Storm 上で行い，Storm 内での特徴抽出処理が終わると，ネットワークを介してクラウド上に特徴ベクトルが転送され，(3) が行われるように設計する．以下に各プロセスの詳細について述べる．

3.1　WEB カメラからのキャプチャ

まず Storm のストリームの起点となる Spout の中に，WEB カメラからのキャプチャ部分を実装する．OpenCV 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となる Bolt に，構造体を渡す．

3.2　Bag-of-Features を用いた特徴抽出

Storm 上で行う 2 つ目のプロセスが，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，ネットワークを介して，(3)Jubatus での解析プロセスに転送する．

Bag-of-Features とは，画像から得られた局所特徴量の集合から，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，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．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．このヒストグラムを Bag-of-Features と呼ぶ．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 節より，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．

3.3　Jubatus での解析

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 trainAPI 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，classifyAPI を利用する．Jubatus サーバに送られてきた Datum はフィルター，特徴抽出という 2 段階のデータ変換を経て解析される．フィルター処理では，学習に不要なものを取り除き，特徴抽出では，フィルターされたデータから特徴を抽出する．フィルター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．今回はフィルター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 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 2 段階のデータ変換を終えた後，解析され，解析結果が送信される．

4.　予 備 実 験

本研究は，3 節で述べた画像データ解析フレームワークを実装し，Storm を用いて実行した場合と Storm を用いずに実行した場合で処理速度を比較する．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．今回は分散処理

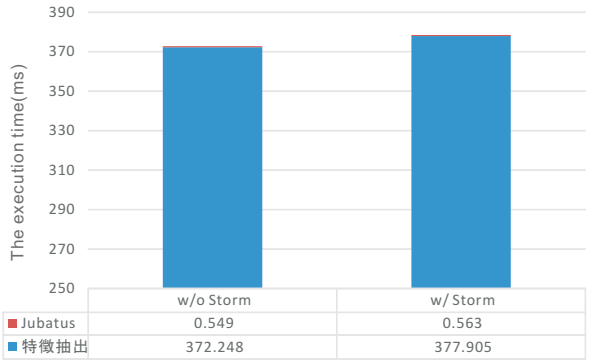

図 5　実 験 結 果

は行わず，1 台の計算機を用いてローカルモードでの実験を行い，Storm のオーバーヘッドを調査する．

クラウド側計算機は，Intel Xeon CPU 3.10GHz を使用し，OS には Linux 2.6.32-5-amd64 Debian GNU/Linux 6.0.4 を用いた．センサ側計算機は，Intel Core i7-3537U CPU @ 2.00GHz を使用し，OS には Ubuntu 14.04LTS を用いた．また，Jubatus v. 0.6.6，OpenCV v. 2.4.9 を使用した．

実験では，予め 100 枚の画像を用いて「ドアを開けた」状態と「イスに座った」状態を学習させた後，画像サイズ 640 × 480 の画像データを毎秒 1 枚ランダムに選出する．選出された画像データの特徴抽出を行い，生成されたベクトル値を Datum に格納して Jubatus に転送した．特徴抽出における Visual Words 数は 100 に設定した．Storm の設定は，Topology は Spout を 1 つ，Bolt を 1 つ用意し，Spout，Bolt のスレッド数はそれぞれ 1 とする．

実行結果を図 5 に示す．縦軸は画像 1 枚あたりの平均処理時間をミリ秒で表している．横軸は Storm の利用の有無を指す．グラフから，Storm を用いた場合の実行時間が約 5 ミリ秒増加したが，全実行時間に対して Storm のオーバーヘッドは無視できることが確認できた．

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 [10] や，人々の異常行動を追跡や活動の検出する自動監視システム [11] が開発されているが，これらは 1 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．

Abdullah ら [12] 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実装した．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. まとめと今後の予定

本研究では，ファットクライアントの概念を取り入れ，センサ側計算機で前処理を行い，クラウド側処理の負荷を軽減する画像データ解析フレームワークを新たに検討した．リアルタイム処理を実現するため，Storm を用いて前処理を行い，Jubatus で解析を行う画像データ解析フレームワークを設計，実装した．予備実験より，Storm を導入したアプリケーションフレームワークのオーバーヘッドは無視できることが確認できた．Strom は負荷分散可能で，スケールアウトできることから，今後 Storm の分散モードを用いることで処理時間が高速化することが期待できる．

今後は，Raspberry Pi [13] のような小型コンピュータを複数台用意し，提案するフレームワークを実装し，その性能特性を調査する．

文　　献

[1] Milan Patel, et al.,“Mobile-edge computing,” ETSI, Tech. Rep., 09 2014.
[2] F. Bonomi, R. Milito, J. Zhu, and S. 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:”クラウド上でのライフログ解析のためのオンラインフレームワーク Jubatus を用いたアプリケーション実装”, DICOMO 2014, pp.254–257, 2014 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] T. Nagahashi, H. Fujiyoshi,“Object Category Recognition by Bag-of-Features using Co-occurrence Representation by Foreground and Background Information”, Machine Vision Applications，pp.413， 2011.
[10] B. S. System,“IVA 5.60 intelligent video analysis”, Bosh Security System, Tech. Rep.，2014.
[11] Project BESAFE，http://imagelab.ing.unimore.it/besafe/.
[12] Tariq Abdullah，M Fahim Tariq，Yusuf Baltaci and Nick Antonopoulos,“Traffic Monitoring Using Video Analytics in Clouds”, 7th International Conference on Utility and Cloud Computing，pp.39-48，2014.
[13] Raspberry Pi，http://www.raspberrypi.org/