Apache Storm を用いた リアルタイムストリームデータ解析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 ‡産業技術総合研究所

1. はじめに

カメラやセンサ等により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量や課金等の制限により，クラウドに動画像を含む大量のセンサデータを送信し，特徴量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．我々は，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，動画像解析処理全体の高速化を図る．負荷分散機能をもつ Apache Storm(以降，Storm と呼ぶ)[1] を導入し，動画像データ解析フレームワークを実装した．既発表研究 [2] では，大規模クラスタ環境においてネットワーク帯域を考慮した実験を行い，低帯域環境においてセンサ・クラウド間の負荷分散が有効であることがわかった．本研究では，提案フレームワークの性能について，使用している Storm に焦点を当ててより詳しく解析する．解析結果より，スレッド数が多い環境において，スレッドのポーリング間隔を調整することで，性能が向上することがわかった．

2. ライフログ解析アプリケーションフレームワークの設計

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処理は，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．(1) では，OpenCV[4] を用いて WEB カメラから画像を取得，構造体に格納し，(2) のプロセスに構造体を渡す．(2) では，(1) で取得した画像を OpenCV を用いて特徴抽出し，Bag-of-Features[5] による画像データのベクトル化を行い，ベクトル化したデータを (3) のプロセスに転送する．(3) では，(2) から受け取ったベクトルデータをもとにオンライン機械学習フレームワーク Jubatus[3] を用いて解析を行う．

動画像データ解析をリアルタイムに処理するには，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．既発表研究 [2] では，センサ側およびクラウド側で前処理を負荷分散処理するフレームワークを実装した．本節で説明した (2) の処理を全てセンサ側で行う場合は，クラウドへの通信量は少なくなるものの，前処理の負荷が大きくなる．(2) の処理を全てクラウド側で行う場合，前処理の負荷がクラウド側で分散できるものの，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．よって，(1) はセンサ側，(2) はセンサ側とクラウド側で処理し，(3) の処理はクラウド側で行われるようにした．

3. 提案フレームワークの性能解析

提案フレームワークの性能をより向上させるため，以下の 2 点に関して性能解析を行った．

1. Spout 数の変化に伴う性能解析
2. スレッドのポーリング間隔の変化に伴う性能解析

Storm では，Spout と Bolt からなる Topology でアプリケーションを表現する．Spout はストリームデータ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．各実験では経過時間あたりに完了した総ジョブ数を比較する．

3.1 実験概要

実験では，2 節で説明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特徴ベクトル長である Visual Words 数は 100 に設定した．

構築した Storm クラスタは図 1 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．Spout や Bolt の処理が行われる Worker を管理する Supervisor ノードを 4 台用意し，1 台がセンサ側，3 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 つずつもたせる．各 Worker で Spou および Bolt のスレッドが複数実行される Supervisor ノードの 4 台のうち 1 台はマスターとなる Nimbus

---

Proposal of camera image data analysis framework using fat client and cloud

Yuko Kurosaki† , Atsuko Takefusa‡ , Nakada Hidemoto‡ and Masato Oguchi†

†Ochanomizu University ‡National Institute of Advanced Industrial Science and Technology (AIST)

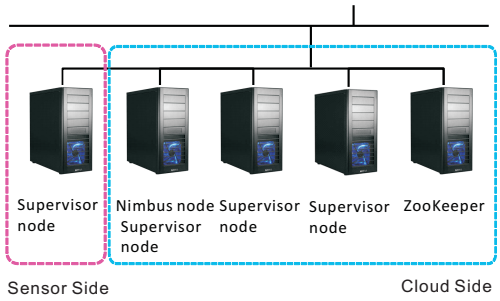

図 1: 実験環境

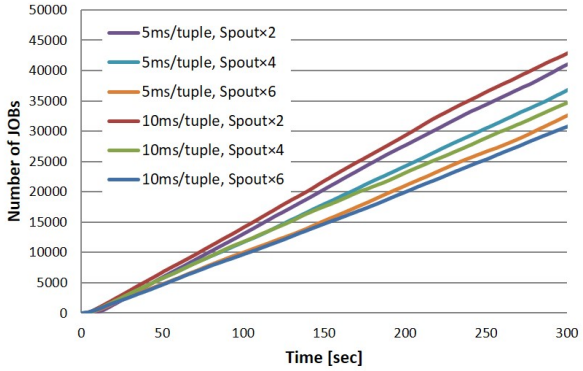

図 2: 経過時間あたりの処理ジョブ数の比較

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．

3.2 Spout 数の変化に伴う性能解析

Spout スレッド数の変化に伴う性能解析を行った．Spout スレッド数を 2，4，6 と変化させ，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 24 とする．

実験結果は図 2 に示す．縦軸は処理した総ジョブ数を表し，横軸は経過時間を秒で表している．実験結果より，スレッド数を増やすと性能が低下し，同じストリーム量 (Spout × 4, 10ms/tuple と Spout × 2, 5ms/tuple) を流した場合ではスレッド数が少なく，かつ速度が速い方が性能がよいことがわかった．これにより，スレッドが必要以上に増えることによるセンサ側の端末への負荷の増加の全体性能への影響が大きいことが示された．

3.3 スレッドのポーリング間隔の変化に伴う性能解析

スレッドのポーリング間隔の変化に伴う性能解析を行った．Storm では Spout および Bolt スレッドが Topology が active かどうか定期的に判断し，active になればそれぞれのスレッド処理を開始する実装になっている．このポーリング間隔を default 時の 100ms から 1ms に短縮させた場合，どのような性能差が出るかを解析した．Spout スレッド数を 4，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 16，24 と変化させた．Sleep 時間は default の 100ms と 1ms の場合で計測を行った．

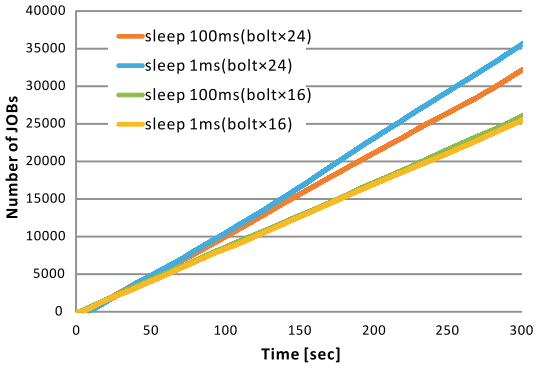

図 3: 経過時間あたりの処理ジョブ数の比較

実験結果は図 3 に示す．縦軸は処理した総ジョブ数を表し，横軸は経過時間を秒で表している．Bolt 数 16 の場合は，ジョブ数の変化は見られなかったが，Bolt 数が 24 の場合，性能向上がみられた．これは，十分な処理スレッド数があれば，ポーリング間隔を短くすることによる性能向上が期待できることを示された．

4. まとめと今後の予定

我々では，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，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，実装している．本研究では，提案フレームワークを用いて性能解析を行い，スレッド数が多い環境において，スレッドのポーリング間隔を調整することによって，性能が向上することがわかった．

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．

参考文献

[1] Apache Storm，https://storm.apache.org/.

[2] Yuko Kurosaki, Atsuko Takefusa, Hidemoto Nakada, asato Oguchi, “A Study of Load Balancing between Sensors and the Cloud for a Real-Time Video Streaming Analysis Application Framework”, Proc. ACM IMCOM, Danang, Vietnam, P3-9, 2016.

[3] Jubatus，http://jubat.us/ja/.

[4] 画像ライブラリ OpenCV，http://opencv.org/.

[5] 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.