Apache Stormを用いたリアルタイム動画像データ解析フレームワークの性能解析

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　〒 305–8560 茨城県つくば市梅園 1–1–1
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，一般家庭やオフィスビルでもデータを取得して，防犯対策，お年寄りや子供のための安全サービスを目的とした動画像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は一般的にクラウドで全ての処理が行われている．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．我々は，動画像解析処理全体の高速化を目的として，負荷分散機能をもつ分散型リアルタイム計算基盤である Apache Storm を導入し，前処理をセンサ側とクラウド側で負荷分散させた．本研究では，提案フレームワークの性能について，より詳しく解析する．解析結果より，処理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．

キーワード　ストリーム処理，Apache Storm，リアルタイム処理，性能評価

Performance Evaluation of a Real-time Video Stream Analysis Application Framework Using Apache Storm

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 305–8560, 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．

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分散型リアルタイム計算基盤 Apache Storm(以降，Storm と呼ぶ) [3] を導入し，センサ・クラウド間で負荷分散を行う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 [4] 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 Apache Storm に焦点を当ててより詳しく解析する．解析結果より，処理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. 関 連 技 術

我々が開発しているアプリケーションでは，前処理のセンサ側とクラウド側の負荷分散の基盤として Storm を，クラウド側の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 [3]．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 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 クラスタは，Master Node，Slave Node，ZooKeeper から構成される．Master Node 上には Nimbus と呼ばれるデーモンを走らせる．Nimbus は主に Slave Node へのタスクの割り振りを行っている．Slave Node 上には Supervisor と呼ばれるデーモンを走らせる．Supervisor 上には，Supervisor を起動する際に指定した数だけ Worker スレッドが立ち上がる．Topology を受け取った Nimbus は，各 Supervisor 上の Worker スレッドに Spout スレッド，Bolt スレッドを割り振り，処理を実行する．Zookeeper は，Storm クラスタのノード間の状態管理，同期に用いられている．

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．

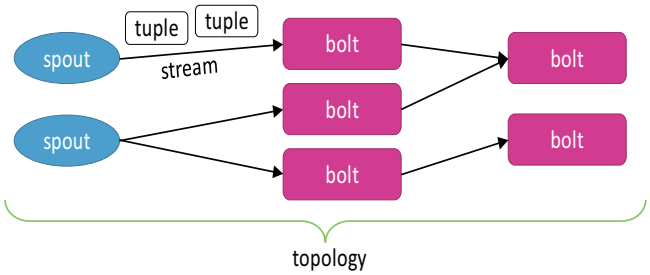

図 1　Topology の流れ

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，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. 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 (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，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，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を削減する場合，(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．

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

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となる Bolt に，構造体を渡す．

3.1.2 Bag-of-Features を用いた特徴抽出

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．2.2 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．

3.1.3 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 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，

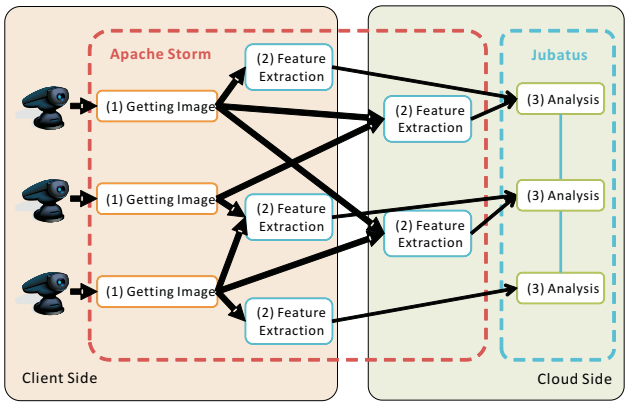

図 2　提案するフレームワーク

解析結果が送信される．

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，分散型リアルタイム計算基盤の Storm を導入した．本研究では，図 2 に示すように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．3.1 節で説明した (1) はセンサ側，(2) はセンサ側とクラウド側で処理される．また，(3) の処理はクラウド側で行われるようにした．(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) の処理をセンサ側とクラウド側の両方で行う場合は，クラウドで処理する場合は前処理の負荷が分散できるものの，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．

既存研究 [4] 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て提案フレームワークが有効かどうか評価実験を行った．実験結果より，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．

4. 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 Storm の性能解析を行った．

（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

各実験では経過時間あたりに完了したジョブ数を比較する．

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．

構築した Storm クラスタは図 3 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．Su-

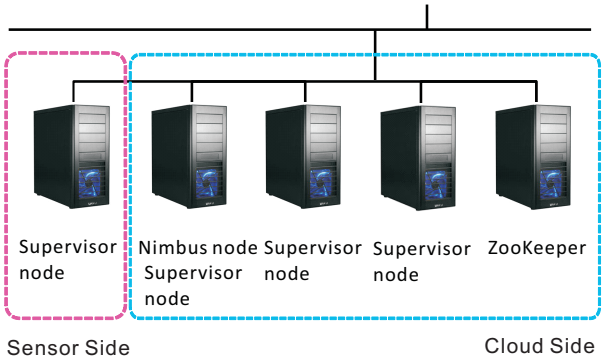

図 3　実 験 環 境

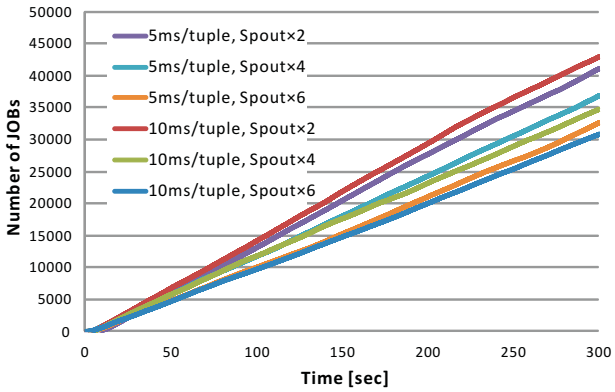

図 4　異なる Spout スレッド数，データ生成速度における処理ジョブ数の比較

pervisor 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，Supervisor ノードの 4 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　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 5ms/tuple と 10ms/tuple とし，Bolt スレッド数を 24 とする．

実験結果は図 4 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．実験結果より，計算上同じストリーム量 (spout × 4, 10ms/tuple と spout × 2, 5ms/tuple) を流した場合，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．これはスレッドが増えることによるスレッド切り替えオーバーヘッドがボトルネックになったと考えられる．

4.3　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，Spout スレッドが active かどうか定期的に判断し，active になければ Sleep する実装になっている．これは，Storm はストリーム処理であるため，常にスレッドが動作している

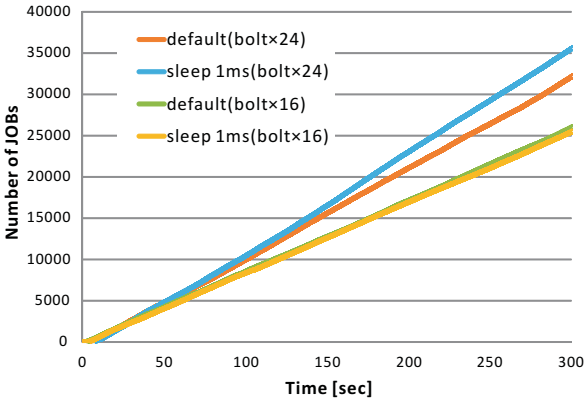

図 5　異なるポーリング間隔，Bolt 数における処理ジョブ数の比較

必要があり，そのための監視であると考えられる．この Sleep 時間を 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．

実験結果は図 5 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る場合，Sleep 時間が短いほうが処理の持ち時間が少なくなるためと考えられる．よって，Sleep 時間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．

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，人々の異常行動の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 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．

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．

また，Storm の高速化を目指して Twitter 社は Apache Heron(以降，Heron と呼ぶ) を開発した [13]．Storm では Master Node が多くの機能を持っており，負荷が集中し，ボトルネックになっていた．Heron では，Topology 毎に Storm における Nimbus にあたる Topology Manager を持たせ，各 Topology

が互いに独立して管理されるようにすることで，ボトルネックを解消した．また，大規模な Topology において，高いスループットと低レンテンシを両立しており，Storm よりも高性能なフレームワークとなっているが，まだ公開されていない．

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するため，Storm を導入して，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動画像データ解析フレームワークを設計，実装している．提案フレームワークの性能について，使用している Apache Storm に焦点を当ててより詳しく解析した結果より，処理スレッド数が多い環境では，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．

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．

謝　　辞

この成果の一部は，国立研究開発法人新エネルギー・産業技術総合開発機構 (NEDO) の委託業務の結果得られたものです．

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] Apache Storm，https://storm.apache.org/
[4] 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”, IMCOM2016, Danang, Vietnam, January 2014.
[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] 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 CloudsTraffic Monitoring Using Video Analytics in Clouds”, 7th International Conference on Utility and Cloud Computing，pp.39-48，2014.
[13] Sanjeev Kulkarni, Nikunj Bhagat, Maosong Fu, Vikas Kedigehalli, Christopher Kellogg, Sailesh Mittal, Jignesh M. Patel, Karthik Ramasamy, Siddarth Taneja, “Twitter Heron: Stream Processing at Scale”, The 2015 ACM SIGMOD International Conference on Management of Data, pp.239-250，2015.