ビッグデータ処理基盤 Apache Spark のストリーミング機能を利用したセンサデータ解析フレームワークの検討

一瀬 絢衣[1]　竹房 あつ子[2]　中田 秀基[3]　小口 正人[1]

概要：各種センサの普及やクラウドコンピューティング技術の習熟に伴い，お年寄りや子供のための安全サービスなどを目的としたライフログの利用が普及してきている．しかし，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．また，近年ディープラーニング技術の発達で，その高い精度から画像や音声の認識などに広く用いられているが，計算負荷が高いことが課題の一つとなっている．本研究では，複数カメラからの動画収集とその解析処理を，効率よく高速に行う解析フレームワークの構築を目指し，Apache Kafka によるデータ収集と Apache Spark のストリーミング機能と Chainer を用いた機械学習を行う．本稿では，提案フレームワークの性能向上に向け，まずはデータ転送を行う Apache Kafka の性能調査を行った．実験から，データソース側と Kafka 間のスループットはデータの分割数である Partition 数によって変化すること，Kafka Cluster とデータ処理プロセス間はデータプロセス数によって全体のスループットが変化することが確認できた．

A Study of Sensor Data Analysis Framework Using Streaming Function of Big Data Processing Infrastructure Apache Spark

Ayae ICHINOSE[1]　Atsuko TAKEFUSA[2]　Hidemoto NAKADA[3]　Masato OGUCHI[1]

1. はじめに

各種センサの普及やクラウドコンピューティング技術の習熟に伴い，お年寄りや子供のための安全サービスなどを目的としたライフログの利用が普及してきている．このようなサービスでは，一般家庭にサーバやストレージを設置して全ての処理を行うことは困難であるため，センサデータをそのまま送信してクラウドで処理するのが一般的である．センサデータの解析をクラウドで行う研究は多くなされており，Twitter のセンチメント分析など小規模かつ大量のデータをクラウド内で効率よく解析する手法が提案されている．しかし，動画像解析は連続的に大容量データを転送する必要があり，計算量，データ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．

画像や映像の解析にはディープラーニング技術が広く使われている．ディープラーニングはニューラルネットワークの中で識別を行う中間層を多層化したものを用いた機械学習であり，精度やスピードの向上という点で注目されている．Chainer[1] や Caffe[2]，TensorFlow[3] といったディープラーニングのフレームワークも多く利用されているが，計算負荷が高いことが課題の一つとなっている．

本研究では，複数カメラからの動画収集とその解析処理を，効率よく高速に行う解析フレームワークの構築を目指し，Apache Kafka によるデータ収集と Apache Spark のストリーミング機能と Chainer を用いた機械学習を行う．Apache Spark(以降, Spark と呼ぶ)[4] とは，大規模データ処理のための高速かつ汎用性の高いエンジンであり，Chainer はディープラーニングフレームワークの一つである．本稿では提案フレームワークの性能向上に向け，データ転送を行

[1] お茶の水女子大学
Ochanomizu University, Bunkyo, Tokyo 112-8610, Japan
[2] 国立情報学研究所
National Institute of Informatcs, Chiyoda, Tokyo 101-8430, Japan
[3] 産業技術総合研究所
AIST, Tsukuba, Ibaraki 305-8560, Japan

う Apache Kafka(以降, Kafka と呼ぶ) の性能調査を行った. Kafka は，データを保存する Kafka Cluster(Broker)，Kafka Cluster にデータの配信を行う Producer，Kafka Cluster からデータの参照を行う Consumer という 3 つのコンポーネントで構成される．実験から，Producer と Kafka Cluster 間のスループットは Consumer の動作に関わらずデータの分割数である Partition 数によって変化すること，Kafka Cluster と Consumer 間は Consumer の数によって全体のスループットが変化することが確認できた．

2. 関連技術

2.1 Apache Spark

Spark は，カリフォルニア大学バークレー校で開発が開始された，大規模データの格納，処理を目的とした分散処理フレームワークである．同じく分散処理フレームワークである Apache Hadoop で用いられている，MapReduce と呼ばれるバッチ処理に特化した処理方法に対し，Spark はデータをメモリに保存することで入出力を高速化することにより，処理全体の実行速度の向上を図る．このアプローチは，レスポンスの速さが必要とされる対話的処理や，データを繰り返し処理する機械学習処理などに適している．Spark は複数のコンポーネントで構成されており，その一つにストリームデータを処理する Spark Streaming がある．Spark Streaming は，数秒から数分ほどの短い間隔で繰り返しバッチ処理を行うマイクロバッチ方式によりストリームデータ処理機能を提供する．Spark Streaming は Apache Kafka や Amazon Kinesis，Twitter からのデータ取得や MQTT のプロトコルにも対応しており，それぞれのデータストリームに合ったアプリケーションを容易に作成できる．

2.2 Apache Kafka

Kafka は，大容量データを高スループット/低レイテンシに収集，配信することを目的に開発されている分散メッセージングシステムである．複数のサーバ上でクラスタとして実行可能であり，Kafka クラスタはトピックと呼ばれるカテゴリにキー，値，タイムスタンプから構成されたレコードのストリームを格納する．メッセージングモデルには送信側がデータの転送を開始する Push 型と，受信側がデータのリクエストを送ることによりデータの転送が開始される Pull 型という 2 つの大きな概念が存在する．Kafka は図 1 に示す構造のように，Producer，Kafka Cluster，Consumer で構成され，Producer と Kafka Cluster 間は Push 型のメッセージングモデル，Kafka Cluster と Consumer 間は Pull 型のメッセージングモデルとなっている．これにより，Push 型の複数プロセスへのデータのブロードキャストと，Pull 型のデータの処分割による処理のスケーラビリティの両方を可能にしている．また，Pull 型

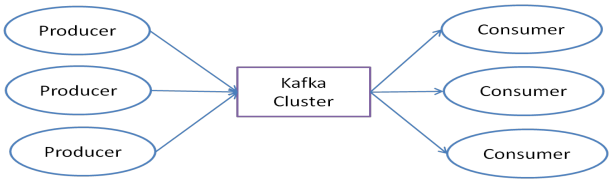

図 1 Apache Kafka

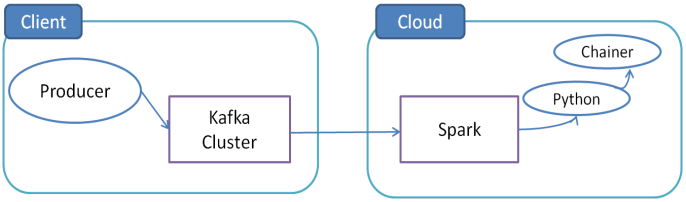

図 2 ストリーミング機械学習

では Consumer がデータの到着量を管理するため，多様な Consumer の性能に応じた効率のよいデータ転送が可能である．こういった技術から速度，耐久性に優れており，リアルタイムのデータパイプラインやストリーミングアプリケーションの構築に広く使われている．

2.3 Chainer

Chainer は，Preferred Networks 社が開発したディープラーニングフレームワークである．Python のライブラリとして提供されており，制御構造はすべて Python での記述が可能である．Chainer の特徴として，柔軟性，直感的，高速の 3 つが掲げられている．畳込みニューラルネットやリカレントニューラルネット，再帰型ニューラルネットなど様々なネットワークアーキテクチャをシンプルに実装でき，また，GPU にも対応している．他のフレームワークの多くが一度ニューラルネット全体の構造をメモリ上に展開し，その処理通りに順伝搬，逆伝播を実行するというアプローチであるのに対し，「Define-by-Run」方式と呼ばれる，ネットワーク構築と学習を同時に行う方式を採用しているのも大きな特徴である．動的にネットワークを定義するため手法の自由度が高く，複雑化していくディープラーニングの開発への対応が可能であると考えられている．また，インストールも容易にできることから，広く使われている．

3. ストリーミング機械学習

本研究では図 2 に示す構成で，ストリーム処理を行ってきた．クライアント側は，Kafka を用いてデータの送信を行い，クラウド側では Spark で Kafka からデータを受け取り，Python プログラムを実行して Chainer を呼び出す．ここで，Spark ではデータは RDD (Resilient Distributed Dataset) という，分散処理を前提としたオブジェクトのコレクションとして扱われるため，データは RDD として読み込み，RDD から Chainer の要求する型に変換する．

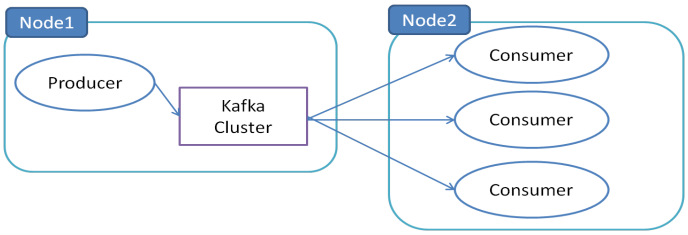

図 3　実験構成

表 1　実験で用いた計算機の性能

OS	Ubuntu 16.04LTS
CPU	Intel(R) Xeon(R) CPU W5590 @3.33GHz (8 コア) × 2 ソケット
Memory	8Gbyte

Spark のマイクロバッチ処理に用いられるマイクロバッチサイズを指定し，指定した時間ごとに区切られたデータに対し Chainer を呼び出して識別処理を行う．
これまでの実験 [5] から，Kafka の構造によりクラウド側の処理が全体の処理のボトルネックとなってしまっていることが確認できた．また，高帯域環境でのデータ転送では，Producer と Kafka Cluster 間の転送がボトルネックとなっていることが確認できた．

4. 実験

Kafka の性能を詳細に調査するため，2 つの端末を用いて Kafka を用いたデータ転送を行い，Producer と Kafka Cluster 間，Kafka Cluster と Consumer 間のスループットを測定した．

4.1 実験環境

図 3 に示す構成で，2 つの端末を用いてデータ転送を行う．Node1 内で Producer と Kafka Cluster を動作させ，Node2 では Consumer を複数動作させる．転送するデータは 0 から 9 の手書き数字の 28 × 28 画素の画像データに正解ラベルが与えられているデータセットである MNIST[6] を用いた．Consumer の動作はデータを参照するのみである．データのカテゴリを生成する Topic においてデータの Partition 数を指定し，複数の Consumer にデータを分散させた．Partition 数は 1，2，4，8，16 に設定し，Consumer 数は 1，2，4 とした．実験に用いた計算機の性能を表 1 に示す．Producer，Kafka Cluster 側及び Consumer 側で同質のノードを用い，その間のネットワーク帯域は 1Gbps となっている．(1)Consumer を作動させた状態で Producer から Kafka Cluster へデータを転送する場合，(2)Producer と Kafka Cluster 間のみでデータ転送を行う場合，(3) あらかじめ Kafka Cluster に転送されたデータを Consumer が参照した場合の 3 つについて，スループットを計測した．MNIST データを 50 万枚転送した際のスループットを測定し，Consumer の数が複数である場合にはスループットの

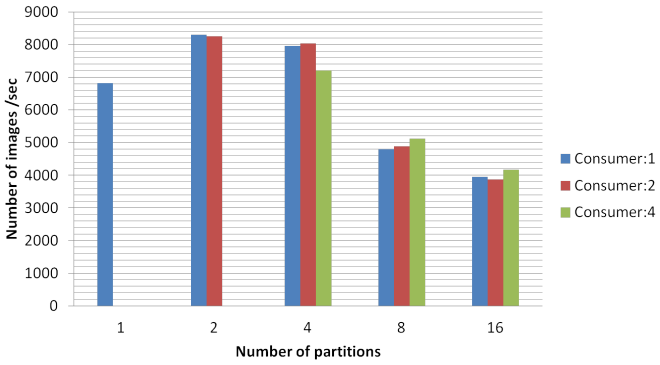

図 4　(1)Consumer を作動させた状態で Producer から Kafka Cluster へデータを転送した場合の Producer と Kafka Cluster 間のスループット

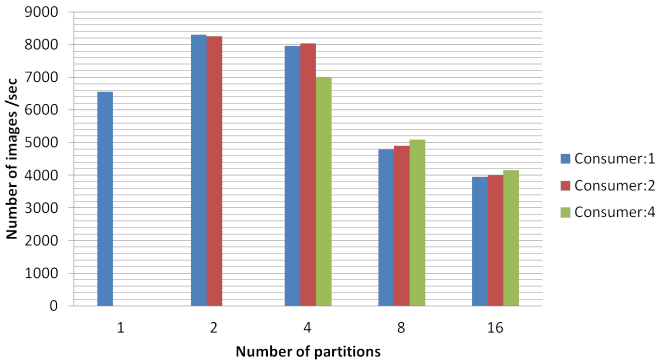

図 5　(1)Consumer を作動させた状態で Producer から Kafka Cluster へデータを転送した場合の Kafka Cluster と Consumer 間のスループット

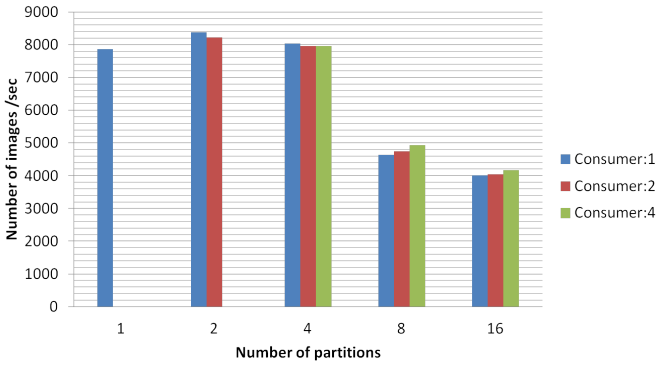

図 6　(2)Producer と Kafka Cluster 間のみでデータ転送を行った場合の Producer と Kafka Cluster 間のスループット

合計値を結果に用いた．

4.2 実験結果

(1)Consumer を作動させた状態で Producer から Kafka Cluster へデータを転送した場合の Producer と Kafka Cluster 間のスループットを図 4，Kafka Cluster と Consumer 間のスループットを図 5，(2)Producer と Kafka Cluster 間のみでデータ転送を行った場合の Producer と Kafka Cluster 間のスループットを図 6，(3) あらかじめ Kafka Cluster に転送されたデータを Consumer が参照した場合の Kafka

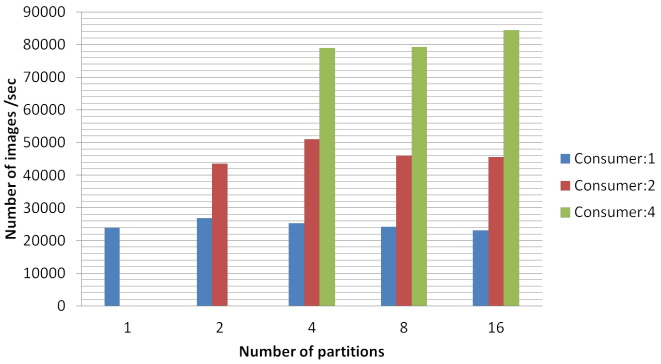

図 7　(3)Kafka Cluster に転送されたデータを Consumer が参照した場合の Kafka Cluster と Consumer 間のスループット

Cluster と Consumer 間のスループットを図 7 に示す．横軸がの Partition 数を示し，縦軸が 1 秒間に転送した画像の枚数を示す．図 4，図 6 から，Producer と Kafka Cluster 間のスループットは Consumer の動作に関わらず Partition 数により大きく変化していることが確認できた．また図 4，図 5 から，Kafka Cluster と Consumer 間のスループットは Producer と Kafka 間のスループットとほぼ一致していることがわかる．Consumer での処理はデータ参照のみであり，ボトルネックが Producer と Kafka 間にあると考えられる．図 7 のあらかじめ Kafka Cluster に転送されたデータを参照する場合には，Consumer の数に比例して全体のスループットが向上している．つまり，Consumer の数を増やした場合にも Kafka の転送性能の劣化が少ないことが確認できた．

5. 関連研究

ディープラーニングを用いたストリームデータ解析は近年数多く研究されており，高速に処理するためのアルゴリズムや精度を向上させるアーキテクチャが検討されている [7][8][9]．しかし，これらはストリームデータを一つの計算機で実行することが前提とされている．本研究はセンサ・クラウド間のデータ送信を考慮した全体のスループットを考慮している点で異なる．また，Spark Streaming を利用してストリーム処理を行っているため，Spark の機能を利用した拡張を容易に行うことが可能である．

Spark Streaming は様々な技術に応用されており，Miucin らは Spark Streaming で実装された解析ツールを提供するツールキットである DINAMITE を提案している [10]．DINAMITE の解析ツールは高度なデバッグ情報ですべてのメモリアクセスを計測し，プログラマがメモリのボトルネックを特定するのを支援する．Chen らは，現在広く使われているルールベースのシステムのスピード，スケーラビリティ，フォルトトレランスといった課題への対処として Spark Streaming を利用している [11]．我々は Spark Streaming を用いた機械学習処理により，一般家庭のライフログ解析の効率化を図る．

6. まとめと今後の課題

本研究では大規模データ処理フレームワーク Spark のストリーミング機能を利用して Chainer を用いた機械学習処理を行う事により，リアルタイムセンサデータ解析処理における性能要件を検討した．本稿ではデータ転送におけるボトルネックを解析するため，Apache Kafka の性能を調査した．実験から，Producer と Kafka Cluster 間のスループットは Consumer の動作に関わらず Partition 数によって変化すること，Kafka Cluster と Consumer 間は Consumer の数を増やした場合にも Consumer 間での影響はほとんどなく，Consumer の数によって全体のスループットが向上することが確認できた．今後の課題としては，Kafka Cluster として複数のノードを用いた場合の性能を調べ，提案手法の分散処理について検討する．

謝辞

この成果の一部は，JSPS 科研費 JP16K00177，平成２９年度国立情報学研究所公募型共同研究および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．

参考文献

[1] Tokui, S., Oono, K., Hido, S. and Clayton, J.: Chainer: a Next-Generation Open Source Framework for Deep Learning, *In Proceedings of Workshop on Machine Learning Systems (LearningSys) in The Twenty-ninth Annual Conference on Neural Information Processing Systems (NIPS)* (2015). 6 pages.

[2] Jia, Y. et al.: Caffe: Convolutional Architecture for Fast Feature Embedding, *arXiv preprint arXiv:1408.5093* (2014).

[3] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y. and Zheng, X. TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems (2015). `http://download.tensorflow.org/paper/whitepaper2015.pdf`. pp. 1-19.

[4] Apache Spark, https://spark.apache.org/.

[5] Ichinose, A., Takefusa, A., Nakada, H. and Oguchi, M.: 大規模データ処理フレームワークのストリーミング機能を利用した機械学習処理の評価, 第 9 回データ工学と情報マネジメントに関するフォーラム (DEIM2017).

[6] Lecun, Y., Cortes, C. and Burges, C. J. The MNIST Database of handwritten digits, http://yann.lecun.com/exdb/mnist/.

[7] Qing, L., Zhaofan, Q., Ting, Y., Tao, M., Yong, R. and Jiebo, L.: Action Recognition by Learning Deep Multi-Granular Spatio-Temporal Video Representation, *Pro-*

ceedings of the 2016 ACM on International Conference on Multimedia Retrieval, ICMR '16, New York, NY, USA, ACM, pp. 159–166 (2016).

[8] Ye, H., Wu, Z., Zhao, R.-W., Wang, X., Jiang, Y.-G. and Xue, X.: Evaluating Two-Stream CNN for Video Classification, *Proceedings of the 5th ACM on International Conference on Multimedia Retrieval*, ICMR '15, New York, NY, USA, ACM, pp. 435–442 (2015).

[9] Read, J., Perez-Cruz, F. and Bifet, A.: Deep Learning in Partially-labeled Data Streams, *Proceedings of the 30th Annual ACM Symposium on Applied Computing*, SAC '15, New York, NY, USA, ACM, pp. 954–959 (2015).

[10] Miucin, S., Brady, C. and Fedorova, A.: End-to-end Memory Behavior Profiling with DINAMITE, *Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering*, FSE 2016, New York, NY, USA, ACM, pp. 1042–1046 (2016).

[11] Chen, Y. and Bordbar, B.: DRESS: A Rule Engine on Spark for Event Stream Processing, *Proceedings of the 3rd IEEE/ACM International Conference on Big Data Computing, Applications and Technologies*, BDCAT '16, New York, NY, USA, ACM, pp. 46–51 (2016).