大規模データ分散処理プラットフォーム Apache Spark を用いた分散並列機械学習の検討

加藤 香澄 † 竹房 あつ子 ‡ 中田 秀基 § 小口 正人 †

† お茶の水女子大学 ‡ 国立情報学研究所 § 産業技術総合研究所

1. はじめに

近年カメラやセンサ等の発達により，一般家庭でのライフログの取得が可能になり，お年寄りや子供のための安全サービスや，防犯対策・セキュリティといった用途に応用されている．しかし，サーバやストレージを一般家庭に設置して解析するのは困難であり，取得した動画像データはクラウドで解析する必要がある．ただし，動画像はデータサイズが大きいため通信量が膨大になってしまい，その解析に要する計算量も膨大になるので，クラウドでの負荷も大きくなる．本研究では，大規模データ分散処理プラットフォーム Apache Spark(以降，Spark と呼ぶ)[1] を用いてディープラーニングフレームワーク Chainer[2] による機械学習処理を並列化させることで，動画像データ解析処理の効率化を図る．

2. 関連技術

2.1 Apache Spark

Spark は，高速かつ汎用的であることを目的に設計されたクラスタコンピューティングプラットフォームである．マイクロバッチ処理という極小単位でのバッチ処理を行うことが特徴で，他のビックデータのツールと密接に組み合わせることができる．Spark 上では RDD(Resilient Distributed Dataset) にデータを保持し，用意されているメソッドを用いて操作することで自動的に分散が可能である．ポスト Hadoop として注目されている．

2.2 Chainer

Chainer はニューラルネットワークを実装するためのライブラリで，シンプルな記法によりネットワークを直観的に記述でき，畳み込みやリカレントなどの様々なニューラルネットにも対応可能なことが特徴である．このニューラルネットを多層にしたものはディープラーニングと呼ばれ，画像認識・自然言語処理・音声認識など様々な分野に応用されている．Chainer の最大の利点として，CUDA をサポートしているため GPU による高速演算が可能である点が挙げられる．

Study on Distributed Parallel Machine Learning using Apache Spark, a Large-scale Data Distributed Processing Platform
Kasumi Kato†
Atsuko Takefusa‡
Nakada Hidemoto§
Masato Oguchi†
†Ochanomizu University
‡National Institute of Informatics
§National Institute of Advanced Industrial Science and Technology (AIST)

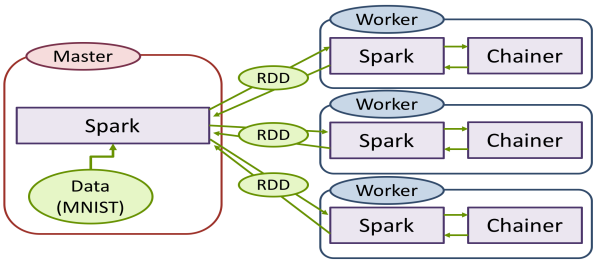

図 1: Spark と Chainer を用いたマスタ・ワーカ処理

3. 実験

本稿では，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[3] を用いて実験を行う．マスタで Python のプログラムを実行することにより，MNIST を Spark に読み込ませて RDD に変換し，同プログラム上で Chainer を呼び出して RDD を渡し，ワーカにて評価を行う実験 (図 1) を Spark の分散機能を利用して実施した．

3.1 実験概要

実験では，マスタ 1 台とワーカとして最大 5 台の端末を Spark Standalone Mode で接続し，マスタでプログラムが実行され，各ワーカでのタスクが完了してワーカからマスタに結果が返って出力されるまでに要する時間を測定した．本実験では，以下 2 つのパラメータを変えて測定した．

1. Spark に読み込ませるデータの partition 数
2. ワーカのノード数

また，この実験を元にタスクがどのように各ノードに分配されているのかを観測した．

実験で用いた計算機の性能を表 1 に示す．マスタ及び全ワーカには同質のノードを用いており，図 2 に示すクラスタ構成とした．

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

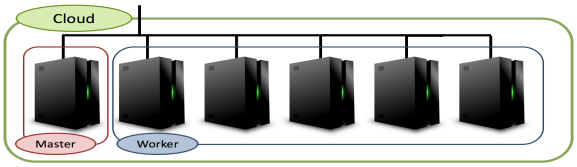

図 2: 実験環境

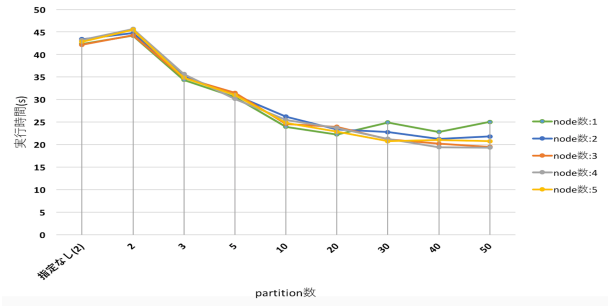

図 3: partition 数及びノード数の変化による実行時間

3.2 実験結果

測定結果を図 3 に示す．このグラフは，ノード数を 1 〜 5 まで変化させ，partition 数を 2,3,5,10,20,30,40,50 と変化させた際の測定結果 10 回の平均値を用いて作成している．実験の結果，partition 数が増加すると実行時間が約 1/2 ほどまで減少することがわかった．また，partition 数の増加による実行時間の減少は partition 数 40 ほどで横ばいになった．一方，ノード数の増加による実行時間の減少はわずかであり，効率よく分散処理が行えていないことがわかった．

図 4 に，ノード数が 2 と 3 の場合についてタスクがどのように各ノードに分配されているのかを観測した結果を，公平性を示す指標である Fairness Index[4] で示す．Fairness Index は以下の式で計算でき，値が 1 に近いほど公平性が高いことを示す．

$$FairnessIndex : f_i = \frac{(\sum_{i=i}^{k} x_i)^2}{\sum_{i=i}^{k} x_i^2}$$

結果から，実行時間に減少が見られた場合でも実際にはタスクが偏って分配されてしまっていたことが判明した．グラフより，partition 数 10 から 20 の間で公平性に大きな変化が見られたため，partition 数 20 までの Fairness Index 及びノードに割り当てられた最多タスク数を図 5 に示す．図 5 から，公平性と実行時間は必ずしも反比例にならず，ノードに割り当てられた最多タスク数はノード数が 2 と 3 の場合でほぼ同じになっていた．

4. まとめと今後の予定

Chainer による解析処理を Spark で並列化し，負荷分散を行った．実行時間とタスク割り当てに関して実験し，ノード数増加による実行時間の減少はわずかであること，

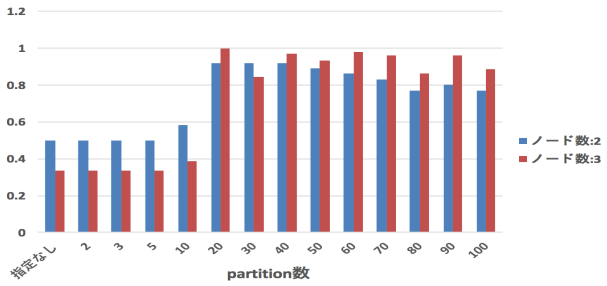

図 4: Fairness Index

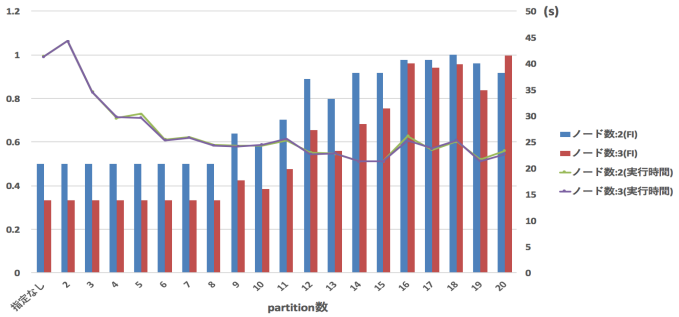

図 5: Fairness Index と実行時間変化

タスク割り当ての公平性と実行時間が反比例しないことが判明した．

今後の課題として，ふるまいの実態を詳しく調査していくことにより現時点での並列処理の課題を明らかにし，処理の効率化を図る．

謝辞

この成果の一部は，JSPS 科研費 JP16K00177 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] Apache Spark，https://spark.apache.org/.

[2] 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.

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

[4] Chiu, D.-M. and Jain, R.: Analysis of the increase and decrease algorithms for congestion avoidance in computer networks, Computer Networks and ISDN Systems, vol. 17, pp. 1-14 (1989).