フォグコンピューティングにおける並列分散機械学習基盤

2014SE095 鈴木慎一郎　2014SE102 塚原郁仁

指導教員：宮澤元

1　はじめに

スマートフォンなどの Internet-of-Things (IoT) デバイスの増加により，現在我々が取り扱うデータは増加の一途を辿っている．このような巨大なデータ集合はビッグデータと呼ばれる．これを分析して有益な情報を取り出すために様々な分析手法が提案されている．こうしたビッグデータ分析の手法の一つとして機械学習が注目を集めている．ビッグデータを機械学習で処理することによって，データに隠されていたパターンを見つけ出したり，新たなデータについての予測を行ったりすることができる．あらかじめ用意されたデータ集合を処理するだけでなく，逐次的に生成されるデータをリアルタイムに分析するオンライン機械学習も利用されている．

センサを備えた IoT デバイスはデータを次々に生成することから，IoT アプリケーションではこれらのデータを機械学習などを用いて適切に処理することが必要である．しかし，IoT デバイスの計算リソースは限られており，こうした分析を行うことは困難である．そこで，生成されたデータの処理をクラウド上で動作する IoT アプリケーションによって行うことが普通である．しかし，IoT デバイスが生成するデータ量が多すぎると，クラウドに送信するだけでネットワーク帯域が消費されてしまったり，クラウドのデータセンターへトラフィックが集中することにより通信レイテンシが増加する，などといった問題がある．

このような問題を解決するためにフォグコンピューティングが提案されている．フォグコンピューティングでは，IoT デバイスの近くにエッジと呼ばれる計算機リソースを配置して，これらを利用して IoT デバイスが生成するデータを処理することができる．IoT デバイスからエッジまでのネットワーク的な距離は短いので，通信レイテンシの問題が解決できる．また，IoT デバイスが生成するデータをエッジを用いて処理することで，これらのデータを直接クラウドに送った場合に比べて，クラウドの中枢のデータセンターへのトラフィックが集中することによってネットワーク帯域が消費されてしまう問題を解決することができる．IoT デバイスが生成したデータに対する機械学習処理もフォグコンピューティングによって効率化できる可能性がある．

本研究の目的は，フォグコンピューティング環境におけるオンライン機械学習の処理状況を調査することである．広域に分散した IoT デバイスからのデータを機械学習で処理する場合，クラウドで集中処理するのと比較して，機械学習の処理結果にどのような特徴が現れるかを調べる．具体的には，フォグコンピューティング環境を擬似的に再現した環境で，オンライン機械学習向けの分散処理フレームワーク Jubatus を用いてオンライン機械学習処理を行い，動作を確認する．同様の処理をクラウドを想定した環境でも行い，結果を比較する．

2　研究の背景

IoT の普及などにより，その量が爆発的に増加したデータはビックデータと呼ばれている．本節では，ビッグデータを処理するにあたり，その処理形態と処理環境に関して述べる．

2.1　オンライン機械学習

オンライン機械学習とは，逐次的に発生するデータを次々に処理できるような機械学習の事である．逐次生成されるデータを受け取ると，それを蓄積することなく次々に処理していくことができる．そのため，データを蓄積する巨大なストレージが必要なく，処理が高速であるといった利点がある．一方，通常の機械学習はバッチ処理と呼ばれ，あらかじめ用意されたデータに対して一括して処理を行う．しかし，次々に生成される大規模データを処理する場合，全てのデータが揃うのを待つのが困難であるし，一度処理を始めた後で新たなデータが生成されると，これを処理に追加できないといった問題がある．そこで，IoT のように次々にデータが生成されるビッグデータの機械学習処理には，オンライン機械学習が適していると考えられる．

2.1.1　Jubatus

オンライン機械学習向けの並列分散処理フレームワークとして Jubatus がある [3]．Jubatus とは株式会社 Preferred Networks と NTT ソフトウェアイノベーションセンタが共同開発した日本初のオープンソースプロダクトである．Jubatus には以下の特徴がある．

並列分散機械学習　巨大なデータセットを高速に扱うために，データを複数のサーバで分担して処理する．この際，同じデータを複数のサーバで処理することによるネットワークトラフィックの増加を避けるために，サーバ間ではデータそのものを共有せず，分析に必要なモデル情報呑みを共有している．

リアルタイムオンライン処理　逐次的に発生するデータをオンラインで処理するには，データ処理をできるだけ高速にする必要がある．ハードディスクなどの低速な装置にデータを保存するとデータ処理の遅れにつながるので，データ処理を全てメモリ上で行う．

2.1.2 Jubatus における並列分散機械学習

Jubatus では，複数のサーバを用いて大規模データに対する機械学習を並列に行うことが出来る．図 1 に Jubatus を用いて並列分散機械学習を行う場合の構成を示す．学習データを送信するクライアントは，サーバには直接接続せず，jubatus proxy に接続して学習データを含む機械学習リクエストを送信する．jubatus proxy はクライアントから送られたリクエストをサーバに中継する．この時，単純に特定のサーバに中継するのではなく，複数のサーバにリクエストを分散させつつ中継することで，サーバの負荷分散をはかるとともに，複数サーバの並列処理によって機械学習の性能を向上できる．なお，jubatus proxy とサーバ間の協調は，分散アプリケーションの為のオープンソースの分散コーディネーションサービスである Zookeeper[4] を用いて行われている．例えば，サーバは起動するとそのサーバ識別子を Zookeeper に登録し，jubatus proxy は Zookeeper が管理するサーバ識別子リストを使ってリクエストを中継する先を決定する．

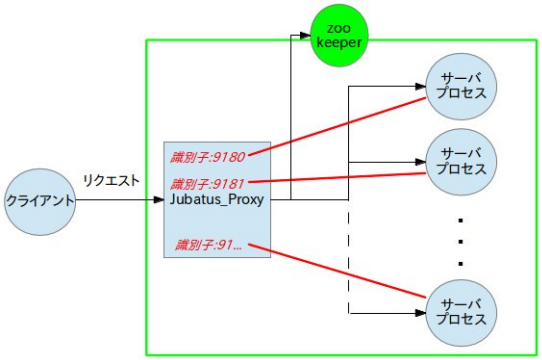

図 1　Jubatus による並列分散処理の構成

複数サーバを用いて並列分散機械学習を行う上での一番の課題は，各サーバにおける学習結果から構築される学習モデルの扱いにある．Jubatus では，各サーバで独立して学習モデルを構築しながら，適切なタイミングで学習モデルの MIX 処理を行うことによって，システム全体で学習モデルを厳密に共有しないので, 同じ学習データに対して常に同じ学習結果を与えるとは限らない．しかし，このことによってサーバ間での通信量を低減する効果がある他，サーバ台数を増やしてスケールアウトすることが可能なシステムとなっている．

MIX 処理が開始されると，まず複数サーバから MIX マスタを決める．その後，MIX マスタは他のサーバから前回の MIX 処理以降に学習された学習モデルの差分を受け取る．MIX マスタは，受け取った差分と自身の差分を結合させて新しい差分を作り出す．最後に MIX マスタが新しい差分を他のサーバに配ることで MIX 処理は完了する．

MIX 処理の形式には Linear MIX と Push/Pull MIX の 2 つがある．Linear MIX はクラスタ内の全サーバで同期的に MIX 処理を実行するもので，最初の MIX 処理で MIX マスタに選ばれたサーバが定期的に MIX 処理を行い続ける．Push/Pull MIX はクラスタ内の各サーバがそれぞれ非同期的に MIX 処理を実行するもので，以下の MIX 開始条件を満たしたサーバが MIX 対象サーバを 1 つ以上選択し，自身と選択したサーバグループに MIX 処理を行う．MIX 開始条件は，前回の MIX 処理からの経過時間と各サーバでの学習モデルのローカルな更新回数のいずれかがあらかじめ指定された閾値を超えた場合となっており，これらの条件は独立して判断される．

2.2 フォグコンピューティング

大規模データの処理に関する問題の一つに処理性能がある．ビッグデータのサイズは非常に大きく，計算リソースに乏しい IoT デバイスなどで処理するのは困難である．そこで，こうしたビッグデータを処理するため，クラウド上で動作するアプリケーションが広く利用されている．しかし，今後 IoT が進展し処理すべきデータ量がさらに増大すると，すべての処理をクラウドで行う場合，データセンターにトラフィックが集中し，通信レイテンシが増加する．

クラウドを用いて大量のデータを処理する際の問題点を解決するために，デバイスからネットワーク的に近い場所に配置されたエッジと呼ばれる計算リソースを用いて処理を行うフォグコンピューティングが提案されている，複数のエッジを使用して，IoT デバイスからデータセンターへ送られるデータの前処理を行うことで，データセンターで処理すべきデータ量を削減できる．処理速度の点においても，データセンターに加えて複数のエッジに負荷を分散させることで処理の高速化につながる [2].

2.3 フォグコンピューティングにおけるオンライン機械学習

本研究の目的は，広域に配置された IoT デバイスが逐次生成するデータをオンライン機械学習によって処理する際の状況を調査することである．例えば，Jubatus をフォグコンピューティングで動作させる際，複数の Jubatus サーバをそれぞれフォグコンピューティングのエッジで動作させることが考えられる．しかし，Jubatus はもともと比較的密に接続された計算機クラスタなどで実行されることを想定しており，通信遅延の大きいフォグにおいてどのように振る舞うかは明らかではない．また，複数のエッジの計算リソースの性能が全て同等とは限らないので，処理するデータ量や利用できるネットワーク帯域，通信レイテンシなどに応じて，Jubatus サーバの最適な配置は変化する．また，複数の Jubatus サーバにおける学習結果から構築される共有の学習モデルに影響を与える可能性もある．

3 実験

本研究では，擬似的に構築したフォグ環境を用いて実際に Jubatus を動作させる実験を行う．Jubatus サーバをデータセンターに配置したような擬似クラウド環境と，エッジに配置したような擬似フォグ環境で，機械学習の結

果や処理時間を比較する．

3.1 実験環境

サーバ PC2 台とクライアント PC2 台を 1000Base-T Eithernet で接続して実験環境を構築した．tc コマンドを用いてサーバ間，サーバ-クライアント間の通信遅延を様々に設定して擬似的なフォグ環境とした (図 2) 使用した PC の仕様を表 1 に示す．

表 1 PC の仕様

クライアント 1	LIFEBOOK P772/G
クライアント 2	Raspberry Pi 3
サーバ 1	プロセッサ数:8，全て 800MHz
サーバ 2	プロセッサ数:8，全て 800MHz 以上

3.2 実験内容

クライアントとサーバ 1 台ずつのペアを用意し，それぞれで同時に Jubatus を用いて，mnist[5] を動作させる．mnist とは，複数の解答ラベル付き画像データを学習用と解析用に分け，学習用データを学習した結果を元に解析用データを解析し，正答率を判定するプログラムである．画像処理ライブラリ OpenCV を利用して，テストデータの数字画像から複数の特徴点を抽出し，その特徴量を抜きだし，この特徴量の重みから解答ラベルを判断する分類を行っている．

今回は，クライアントは解答ラベル付きの数字画像データ 1000 枚をサーバに送る．サーバは送られたデータのうち 900 枚を学習用，100 枚を解析用とする．サーバは解析用データの分類結果をその解答ラベルと付き合わせて答え合わせを行い，クライアントに正答率を送信する．

サーバ-クライアント間の通信に遅延を入れた擬似クラウドサーバを用いる場合 (図 3) とサーバ間の通信に遅延を入れた擬似フォグサーバを用いる場合 (図 4) とで比較を行った．なお，使用する遅延時間は 10ms，50ms，100ms のいずれかとした．比較対象は次の 4 つである．

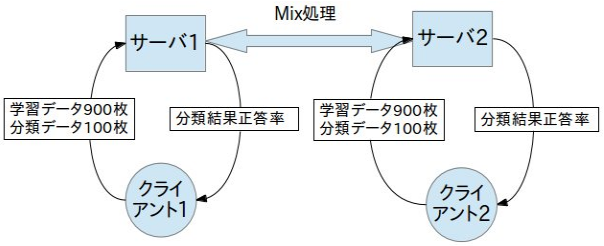

図 2 実験イメージ図

MIX 時間：一回の mnist 処理につき，複数回行われた MIX の内の 1 回の MIX 処理にかかった時間．

プログラム実行時間：mnist プログラム開始から終了までにかかった時間．クライアントが画像を送り初めてからサーバがテスト結果を返すまで．

MIX 回数：一回の mnist 処理につき，実行された MIX 処理の回数．

正答率：サーバがクライアントに返す 100 枚のテストデータの正答率．

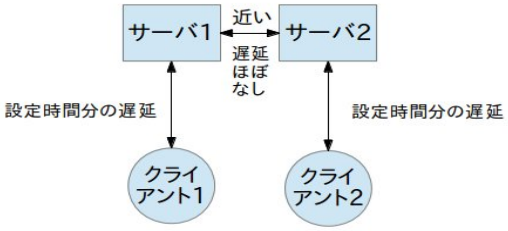

図 3 擬似クラウドサーバの構成

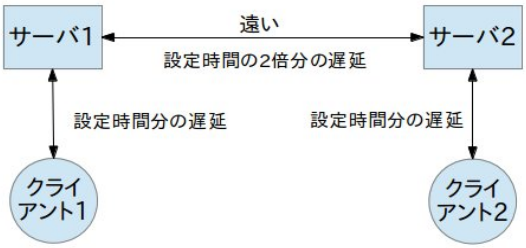

図 4 擬似フォグサーバの構成

3.3 実験結果

3.3.1 mnist の実行による MIX 処理

送られた 900 枚の画像データを学習する過程で，2 つのサーバ間で MIX 処理が複数回行われていた．

3.3.2 実験結果平均

ここでは mnist 処理を 3 回行ったときの平均データを記述する．

表 2 実験結果平均

	擬似クラウド	擬似フォグ
MIX 時間	0.425962secs	8.564459secs
プログラム実行時間	3m10.6422s	3m10.4995s
MIX 回数	11.67 回	10 回
正答率	0.73	0.678333

サーバ同士を遠くすることで 1 回の MIX 処理にかかる時間が長くなり，MIX 回数が減って正答率は悪くなった．しかし，MIX 処理にかかる時間が長くなったことによるプログラム全体の実行時間にかける影響はほぼなかった．

3.3.3 MIX マスタとなるサーバマシンの差

表 3 はサーバ 1，サーバ 2 ごとに，マスタサーバとなった時の 6700000byte 近くの差分を送った MIX についての

1 回につきかかる時間の平均を示している．

マスタサーバがサーバ 1 である時の MIX 時間が，同じ環境においてマスタサーバがサーバ 2 である時の MIX 時間より長くなる傾向があった．

表 3　マスタサーバの MIX 処理平均時間

	サーバ 1	サーバ 2
10ms 擬似クラウド	0.489412 secs	0.436947 secs
10ms 擬似フォグ	0.898070 secs	0.829673 secs
50ms 擬似クラウド	0.490926 secs	0.417835 secs
50ms 擬似フォグ	2.874092 secs	2.8532 secs
100ms 擬似クラウド	0.482184 secs	0.423102 secs
100ms 擬似フォグ	9.30319 secs	9.06905 secs

4　考察

4.1　サーバマシン性能による影響

サーバ 1 より比較的性能が上であるサーバ 2 が MIX マスタとなったときの MIX 処理時間は，サーバ 1 が MIX マスタの時より早く処理された．

これは MIX 処理中におけるマスタサーバが別のサーバから受け取る差分を自身の差分と合成する処理の時間が，マスタサーバとなるマシンの性能によって変わってくる事が考えられる．

よって早い MIX 処理を行うためにはマシンの性能が良いサーバを常に MIX マスタにし続ける必要がある．

4.2　MIX 回数と正答率の関係

プログラムの最後の MIX 処理にかかった時間が全体の MIX 処理時間平均より大幅に減っている状態が複数回見受けられ，その時に配っている差分データも通常より小さいサイズであった．

擬似クラウド環境と擬似フォグ環境の間には 1 回の MIX 処理時間について大きな差が見られたが，全体の MIX 処理回数については 1，2 回程度の差しか見られなかった．

まず，MIX 処理に使われた差分データの量のほとんどが 6700000byte に近い傾向があった．このことから MIX 処理の開始条件はどちらかのサーバに一定量の差分データ容量が溜まったら行われるものであり，サーバ距離の遠さに関わらず MIX 処理の時間間隔は一定であると考えられる．

次に，オンライン機械学習向けのフレームワークである Jubatus はリアルタイム処理手法を行うことが出来る．このことから Jubatus はプログラムの終了時間を優先し，残りの差分データで MIX 処理を行ってプログラムを終わらせている可能性がある．

よって，サーバー同士を遠くに置いた時に 1 回の MIX 時間が伸びることによって，プログラムの最後に行う MIX 処理の回数に図 5 のような変化が生じ，赤い部分の残りの差分データを用いた MIX 処理が行われて MIX 回数が変わったことにより正答率に変化が見られたと考えられる．

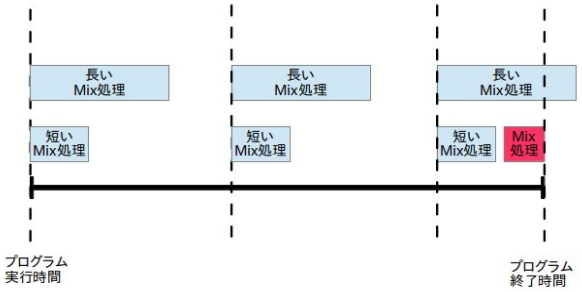

図 5　最後の MIX の変化

5　おわりに

今回の実験ではフォグを使用するオンライン並列分散機械学習の動作を確かめるため、実験として擬似的にサーバの距離を変えて比較を行った．

結果として性能が高いサーバは比較的 MIX 処理時間が早いことが分かった．また，サーバ同士の距離は 1 回の MIX 処理時間の違いを生みだし，そのことによって MIX 回数の変化が影響して正答率に変化が見られると推測できる．

今後は，クライアントとサーバ間の距離を変えることでよりフォグ環境に近づけた実験を行う．また，分類機能以外の機械学習のプログラムについての調査も行っていきたい．

参考文献

[1] Bonomi, Flavio, et al.“Fog computing and its role in the internet of things.” Proceedings of the first edition of the MCC workshop on Mobile cloud computing. ACM, 2012.

[2] 黒崎 裕子, 竹房あつ子, 中田 秀基, 小口 正人: “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,” 第 8 回データ工学と情報マネジメントに関するフォーラム, 2016.

[3] Jubatus： 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/#

[4] Hunt, Patrick, et al. “ZooKeeper: Wait-free Coordination for Internet-scale Systems.” USENIX annual technical conference. Vol. 8. 2010.

[5] jubatus-example/mnist at master jubatus/jubatus-example GitHub
https://github.com/jubatus/jubatus-example/tree/master/mnist