# 大規模機械学習向けクラスタにおける ネットワーク構造とパラメータ交換手法

黎 明曦 †, 谷村 勇輔 ‡,†, 中田 秀基 ‡,†
† 筑波大学, ‡ 産業技術総合研究所
{rei-mingi, yusuke.tanimura, hide-nakada}@aist.go.jp

**概 要**

大規模なデータを対象とする機械学習システムの高速化には並列化が必須である。データ並列機械学習システムにおいては、何らかの方法で機械学習機内のパラメータの値を交換する必要があるが、パラメータ交換手法とネットワーク構造の関係は知られていない。本研究では、ネットワークシミュレータ SimGrid を用いて、さまざまなネットワークと、パラメータ交換手法を組み合わせて評価を行った。その結果、パラメータ交換手法によっては、ネットワークバイセクションバンド幅の影響をほとんど受けず、比較的貧弱なネットワークでも遜色のない性能で実行できることがわかった。

## 1 はじめに

ディープラーニングに代表される近代的な機械学習システムでは、大量の学習データを処理することが必要であるため、並列化による高速化が必須である。機械学習システムを並列化する方法には大別して、複数の機械学習機がデータセットを分割して学習するデータ並列と呼ばれる手法と、一つの機械学習機の内部を並列化するモデル並列と呼ばれる手法の二種類がある。本研究では、データ並列機械学習システムを対象とする。

データ並列機械学習システムでは、複数の機械学習機内のモデルを同期させる必要がある。この同期は必ずしも厳密である必要はないが、ある程度の幅の範囲で、すべての機械学習機内のモデルが近似している状態を保つ必要がある。モデルの同期、すなわちモデルを構成するパラメータの同期には、大別して 2 つの方法がある。一つは、パラメータサーバと呼ばれる専用の中央サーバを設けこれにデータを集約して分配する方法、もう一つは、機械学習機同士が直接相互に通信して情報を交換し、全体として同期をとる方法である。

同期には当然ながらネットワーク通信が必要となり、計算機システムのネットワーク構成の影響を受ける。われわれは、パラメータ同期手法と、計算機システムのネットワーク構成の関係を知るために、分散並列環境シミュレータ SimGrid[1] を用いて、大規模環境を仮想的に構築し、定量的な評価を行った。ファットツリー構成のネットワークにおいてバイセクションバンド幅を調整し、バイセクションバンド幅とパラメータ同期手法の相関を調べた。

本稿の貢献は、パラメータ同期手法とバイセクションバンド幅の関係を定量的に評価し、同期手法を選定する事によりバイセクションバンド幅が比較的小さい計算システムにおいても、十分な性能が出ることを示したことにある。

本論文の構成は以下の通りである。2 節に本研究の背景を示す。3 節では本稿が仮定するネットワークモデルを示す。4 節に SimGrid を用いたシミュレーションとその結果を示す。5 節で実験結果に対する議論を行う。6 節に関連研究を示す。7 節はまとめである。

## 2 背景

### 2.1 並列機械学習システムにおけるパラメータの交換手法

大規模な機械学習システムには、大規模な分散システムが必須である。例えば文献 [2] においては、16CPU コアを持つ計算機 1000 台を 3 日間もちいた学習を行っている。

機械学習システムを並列化によって高速化するには、大別して 2 つの手法がある。一つは、単一の機械学習機の内部を並列化する手法で、**モデル並列**と呼ばれる手法である。もう一つは、複数の機械学習機を、ゆるく同期した状態で並行して動作させることで高速化する手法で、**データ並列**と呼ばれる。個々の機械学習機は、データセットのそれぞれ異なるサブセットを学習し、パラメータを更新する。更新されたパラメータを定期的に集約して再分配することで、機械学習機群をゆるく同期させ、学習を加速する。この 2 つの手法は排他ではなく、多くの場合双方を併用することになるが、本稿ではデータ並列のみを対象として議論する。

パラメータを交換する手法としては、集中サーバに集約して再分配する方法と、個々のサーバが相互に通信し

*NOTICE: xSIG 2017 does not publish any proceedings, and this manuscript is provided only to the xSIG 2017 attendees during the workshop. Thus, the TPC expects that acceptance in xSIG 2017 should not preclude subsequent publication in conferences or journals.

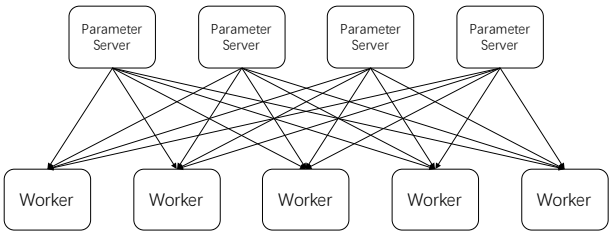


図 1: パラメータサーバを用いた並列機械学習機の構造

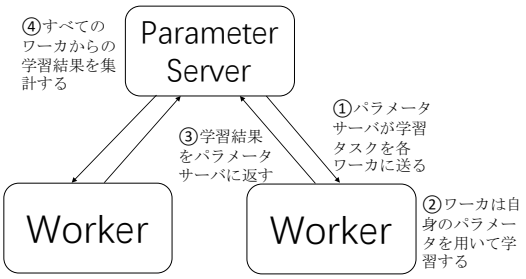


図 2: パラメータサーバの動作

てパラメータを交換する方法の 2 つがある。以下詳しく述べる

### 2.1.1 パラメータサーバによるパラメータ交換手法

パラメータを定期的に集約、再分配するために用いるサーバ機構を、一般にパラメータサーバ [3] [4] と呼ぶ。

このような手法は文献 [2] でも用いられている [5]。われわれは、これまでにマスターワーカを用いた簡易的なパラメータサーバを構築し、大脳皮質モデル BESOM[6] の高速化を行ってきた [7]。

図 1 にパラメータサーバを用いた並列機械学習機の構造を示す。システムは、パラメータサーバとワーカ（機械学習機）から構成される。ワーカは定期的にパラメータサーバにパラメータ更新情報を送信し、最新のパラメータを受け取る。パラメータサーバはワーカからの更新情報を受信し、内部に保持したパラメータを更新し、ワーカに返信する (図 2)。

図 3 に、パラメータサーバ群とワーカ群間の通信を時間軸に沿って示す。パラメータサーバとワーカの間では全対全で通信が行われる。計算全体としては一種の BSP（Bulk Synchronous Parallel[8]）を構成しており、ワーカ群がパラメータサーバ群を介して周期的に同期を繰り返す構造となっている。

パラメータサーバが複数あるのは、パラメータサーバがボトルネックになることを避けるためである。大規模な機械学習においてはパラメータの数が膨大になるため、単一のパラメータサーバではネットワーク通信ボトルネックになることが避けられない。パラメータの更新

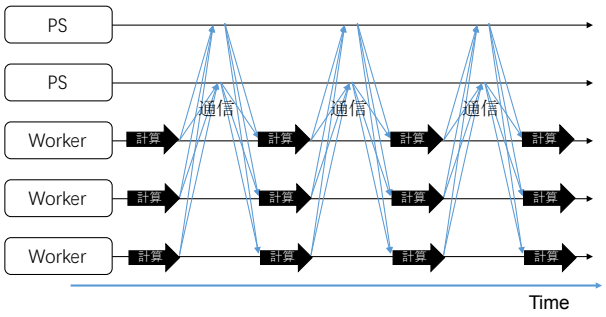


図 3: パラメータサーバを用いた並列機械学習システムの通信パターン

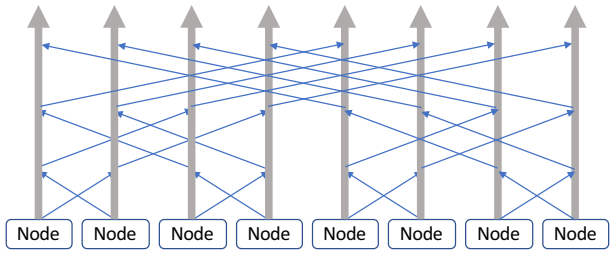


図 4: バタフライ通信パターン

は個別に独立して行う事ができるため、パラメータを複数のパラメータサーバに分割することが、容易に可能である。

機械学習モデルのパラメータは均等に分割され、すべてのパラメータサーバに配置される。つまり、パラメータサーバは、パラメータの一部をそれぞれ担当することになる。ワーカは分割したパラメータを、それぞれ担当するパラメータサーバに送信する。

### 2.1.2 直接交換によるパラメータ同期

パラメータサーバを用いずにパラメータを同期する手法としては、機械学習機同士が直接パラメータの勾配を交換する手法がある。これを繰り返すことですべてのノードがすべての勾配の和を入手する事ができる [9]。

8 ノードの場合の様子を図 4 に示す。このような通信パターンはバタフライとして知られ、MPI の allreduce 等で広く用いられている [10]。$N'$ ノードに対して $Log_2 N'$ 回の通信ですべてのノードに情報を行き渡らせることができる

## 2.2 SimGrid

SimGrid[1][11] は、並列計算アプリケーション向けに開発されたシミュレーションフレームワークである。並列計算シミュレータは大別して、エクスペリメンタルプラットフォーム、エミュレータ、ネットワークシミュ

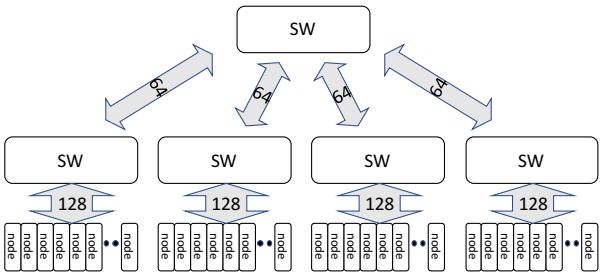

図 5: ハーフバイセクション 4 サブクラスタ構成

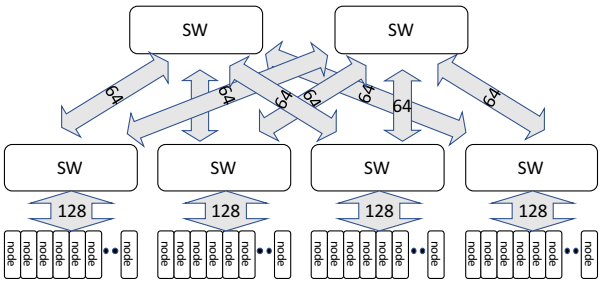

図 6: フルバイセクション 4 サブクラスタ構成

レータ、アプリケーションシミュレータの 4 種類があり、SimGrid はアプリケーションシミュレータに属する。

アプリケーションシミュレータの特徴は、シミュレーション時に実際イベントを発生させず、イベントのコストだけを抽出して、シミュレーションを行うことである。このため計算負荷が小さく、単一計算機でも大規模なシステムのシミュレーションを短時間で実行することが可能になる。

SimGrid ではプラットフォーム設定とデプロイメント設定を XML ファイルで記述し、シミュレーションコードを C++で記述する。SimGrid のプロセスは、典型的には、メッセージを受信し、タスクを実行し、メッセージを送信する、というループを実行する。このようなプロセスが多数存在し、それぞれ相互に通信することでシステムとして動作する。

SimGrid の設定ファイルではホストやリングなどの低位の構成要素だけでなく、「クラスタ」のような抽象度の高い構成要素を直接記述できるため、大規模なシステムを表現することが容易である。

# 3 ネットワークモデル

本稿で仮定するネットワークのモデルを述べる。本稿では、比較的大規模なスイッチで構成した複数のサブクラスタを、上位のスイッチで接続してスケールアップする大規模クラスタを想定する。この際、上位のスイッチを複数設けてファットツリーを構成することでバイセクションバンド幅を確保する。各クラスタは 128 ノードとし。各スイッチは最大 256 接続を収容できるものとする。

4 サブクラスタの例を図 5 と図 6 に示す。左右の 2 つづつのクラスタに分離して考えた場合、上位スイッチが 1 つの場合には、左右の間には 128 本の接続が存在しハーフバイセクションバンド幅となる (図 5)。上位スイッチが 2 つになると、左右間の接続は 256 本となり、フルバイセクションバンド幅となる (図 6)。このように、上位スイッチの数を変更することでバイセクションバンド幅を調整することができる。

# 4 実験

下に述べるネットワーク構成、およびパラメータ交換方法を用いて、SimGrid を用いてシミュレーションを行った。パラメータ交換は、ミニバッチ 1 回の実行ごとに交換を行うことを想定し、1 秒間隔とした。10 回パラメータ交換を行うのに要した時間を計測した。

## 4.1 クラスタとネットワークの構成

上述のように、本稿で対象とするクラスタは、128 ノードのサブクラスタを上位スイッチで複数接続することで構成する。サブクラスタ数を 16、32、64、128 とした。

バイセクションバンド幅を制御するために、上位スイッチの数をサブクラスタ数の 1/2, 1/4, 1/8.. とした。それぞれフルバイセクションバンド幅、ハーフバイセクションバンド幅、1/4 バイセクションバンド幅に相当する。ネットワークのバンド幅はすべて 1GByte/s とした。

## 4.2 パラメータ交換方法

パラメータ交換方法としてはパラメータサーバを利用する方法を 2 通り、バタフライをベースとした方法を 2 通りテストした。

パラメータサーバを利用する方法としては、1 つのサブクラスタにパラメータサーバを集中させる方法 (図 7) と、各サブクラスタにパラメータサーバを分散する手法 (図 8) を比較した。図中の赤いノードがパラメータサーバを意味する。いずれの場合もサブクラスタの数によらず、常にパラメータサーバは 128 ノードとした。分散する場合には各サブクラスタに均等にパラメータサーバを配置した。例えば、サブクラスタ数が 32 の場合には、各サブクラスタの 128 ノードの内 4 ノードがパラメータサーバとなる。

バタフライをベースとする方法では、単純にすべてのノードでバタフライを構成する方法と、階層的に 2 段階に分けて行う方法を比較した。前者では、$N$ ノードに対して単純に $log_2 N$ 回の相互通信を行う。一方後者では、まずサブクラスタ内で分木カスケード通信を行って、サブクラスタの情報を集約し、それからサブクラス

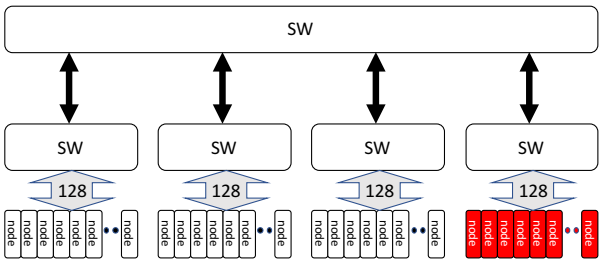


図 7: 集中型パラメータサーバ構成

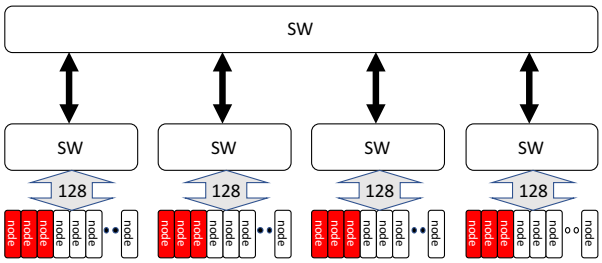


図 8: 分散型パラメータサーバ構成

タ内のノード 0 番だけが、クラスタ間でバタフライ通信を行って全体の情報を集約する。さらにサブクラスタ内で 2 分木カスケード通信で集約した情報を分散する。この様子を (図 9) に示す (簡単のため各サブクラスタのノード数を 4 としている)。サブクラスタのノード数を $N$、サブクラスタ数を $M$ とすると、サブクラスタ内でのカスケード通信が $log_2 N$ 回が 2 回、サブクラスタ間のバタフライに $log_2 M$ 回で、総計 $2log_2 N + log_2 M$ 回の通信が必要となる。

バタフライベースの手法とパラメータサーバベースの手法では、機械学習機に用いるノードの数が厳密には異なる（パラメータサーバベースの場合が常に 128 少ない）が、バタフライベース手法で 128 ノード減らしても実行時間は同じになることが予期されるため、ここでは考えない。

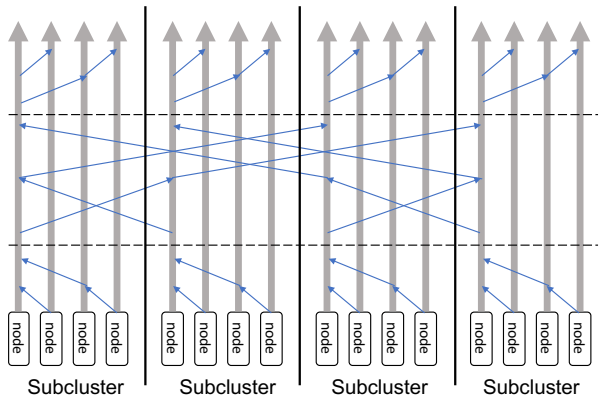


図 9: 2 層バタフライ

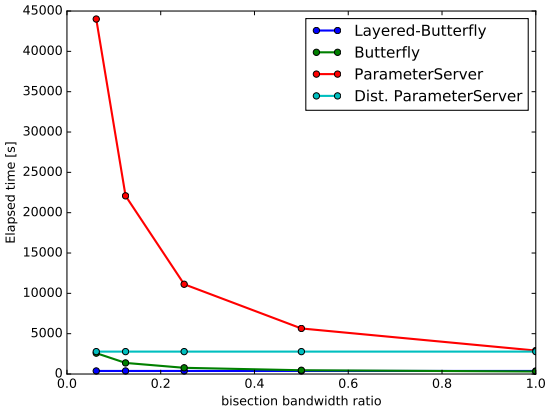


図 10: 128 サブクラスタの結果

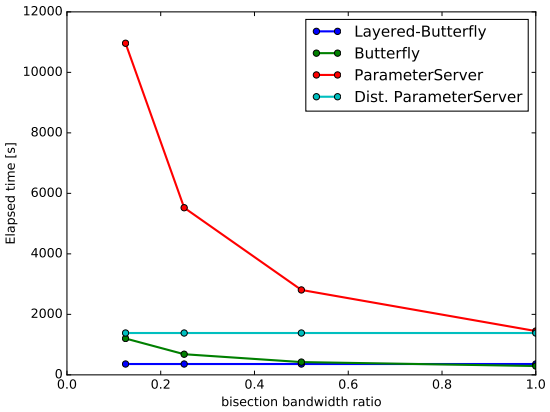


図 11: 64 サブクラスタの結果

### 4.3 シミュレーション結果

図 10、図 11、図 12、図 13 にシミュレーションの結果を示す。グラフの縦軸は実行時間である。横軸はフルバイセクションに対する比率で、一番右の 1 がフルバイセクションバンド幅に相当する。左に行けばいくほどネットワーク帯域が制限されていることに相当する。

## 5 議論

### 5.1 実験結果に関する考察

この結果から、パラメータサーバを使う方法はバタフライをベースとする手法と比較して基本的な性能が劣る事がわかる。これはパラメータサーバノードへの通信が集中するため、パラメータサーバノードの通信性能がボトルネックとなるためである。

ナイーブな集中型パラメータサーバにおいては、ネットワークのバイセクションバンド幅が小さくなると大幅に性能が低下する事がわかる。一方、各サブクラスタにパラメータサーバを分散する分散パラメータサーバは、バイセクションバンド幅低下の影響をほとんど受けな

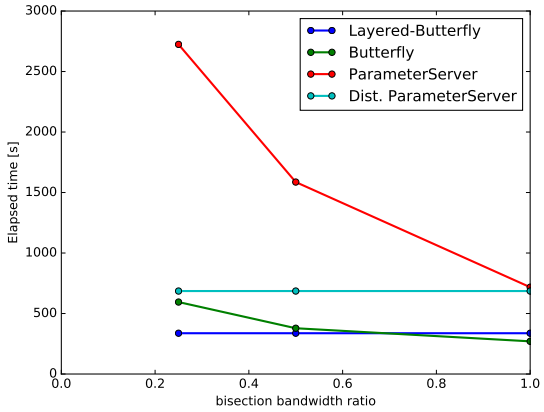


図 12: 32 サブクラスタの結果

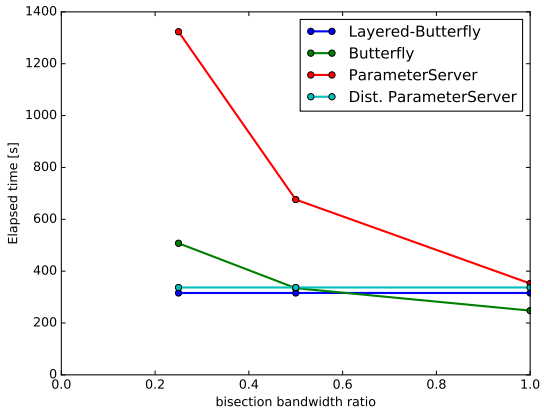


図 13: 16 サブクラスタの結果

い。これは、パラメータサーバが分散することにより、ネットワークトラフィックがクラスタ全体で平滑化されるためである。

バタフライベースの手法はパラメータサーバベースの手法よりも高速である。全ノードでフラットにバタフライを行うナイーブなバタフライではネットワークに負荷がかかるため、バイセクションバンド幅が低下するとともに性能が低下する。これに対して、階層的に 2 段階でバタフライを行う手法では、サブクラスタネットワークへの負荷が小さくなるので、バイセクションバンド幅の低下が抑えられる。

図 14 に図 10 の一部を拡大したものを示す。この図からわかるように、フルバイセクションバンド幅のネットワークではフラットなバタフライの方が高速である。サブクラスタのノード数を $N$、サブクラスタ数を $M$ とするとフラットなバタフライでは $log_2N + log_2M$ 回のノード間通信が行われるのに対し、2 段階バタフライでは $2log_2N + log_2M$ 回の通信が必須となり、回数にして $log_2N = 7$ 回の余分な通信が発生する。このため、ネットワークへの負荷が問題となりにくいフルバイセクションバンド幅ネットワークでは、回数の少ないフラットなバタフライのほうが高速なのだと思われる。

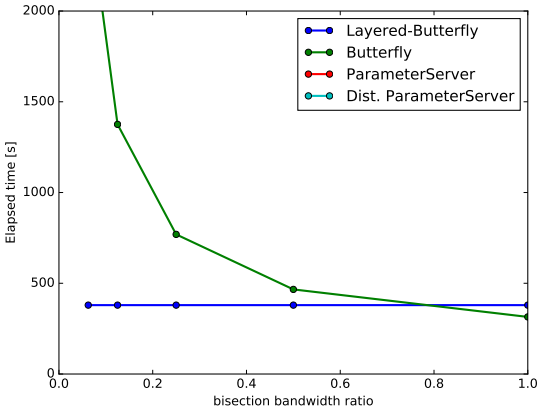


図 14: 128 サブクラスタの結果 (拡大)

### 5.2 パラメータサーバの利点と欠点

前項でみたとおり、パラメータサーバを用いる手法は性能的な面では、バタフライをベースとした手法と比較して低速であると言える。

一方、性能以外の面に目を向けると、パラメータサーバにはいくつかの利点がある。ひとつは、耐故障性である。筆者らが [12] で議論したとおり、パラメータサーバを用いることで、ノード故障時の復旧プロトコルを単純にすることができる。

また、昨今ではパラメータの同期条件をリラックスし、厳密に同期をとることを諦めることで通信負荷を低減する手法も提案されている。この場合にも、パラメータサーバのような集中サーバのほうが、実装が容易である。

## 6 関連研究

本稿では、パラメータを全対全で交換する手法を前提とした。これはすべての機械学習機においてパラメータを完全に同期することを前提としたためである。しかしパラメータを完全に同期しなくても並列学習が進むことが知られている ([13], [14])。

[15] は、ゴシッププロトコルを用いてミニバッチごとに、ランダムに 1 ノードを選択し勾配を交換する方法を提案している。このような手法はネットワークバンド幅に対する影響が非常に小さいことが予想される。一方で、計算の収束性への影響も認められるため、慎重な評価が必要である。

## 7 おわりに

われわれは、パラメータ同期手法と、計算機システムのネットワーク構成の関係を知るために、分散並列環境シミュレータ SimGrid を用いて、バイセクションバン

ド幅とパラメータ同期手法の相関を調べた。その結果、同期手法を選定する事によりバイセクションバンド幅が比較的小さい計算システムにおいても、十分な性能が出ることを示した。

今後の課題は以下のとおりである。

- 非同期プロトコルを用いた場合のネットワークバンド幅要求についても同様のシミュレーションを行う。
- 非同期プロトコルや通信インターバルを拡大した場合のコンバージェンスへの影響を理論的に解明し、大規模並列機械学習に適したシステム構築の指針を得る。

## 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。本研究は JSPS 科研費 JP16K00116 の助成を受けたものです。

## 参考文献

[1] Henri Casanova, Arnaud Giersch, Arnaud Legrand, Martin Quinson, and Frédéric Suter. Versatile, scalable, and accurate simulation of distributed applications and platforms. *Journal of Parallel and Distributed Computing*, 74(10):2899–2917, June 2014.

[2] Quoc Le, Marc'Aurelio Ranzato, Rajat Monga, Matthieu Devin, Kai Chen, Greg Corrado, Jeff Dean, and Andrew Ng. Building high-level features using large scale unsupervised learning. In *International Conference in Machine Learning*, 2012.

[3] Parameter server: `http://parameterserver.org/`. Accessed: 2015-06-20.

[4] Mu Li, David G. Andersen, Jun Woo Park, Alexander J. Smola, Amr Ahmed, Vanja Josifovski, James Long, Eugene J. Shekita, and Bor-Yiing Su. Scaling distributed machine learning with the parameter server. In *11th USENIX Symposium on Operating Systems Design and Implementation (OSDI 14)*, pages 583–598, Broomfield, CO, October 2014. USENIX Association.

[5] Jeffrey Dean, Greg S. Corrado, Rajat Monga, Kai Chen, Matthieu Devin, Quoc V. Le, Mark Z. Mao, Marc'Aurelio Ranzato, Andrew Senior, Paul Tucker, Ke Yang, and Andrew Y. Ng. Large scale distributed deep networks. In *NIPS 2012: Neural Information Processing Systems*, 2012.

[6] Yuuji Ichisugi and Naoto Takahashi. An efficient recognition algorithm for restricted bayesian networks. In *Proc. of 2015 International Joint Conference on Neural Networks (IJCNN 2015).*, 2015.

[7] 黎 明曦, 谷村 勇輔, 一杉 裕志, and 中田 秀基. マスタ・ワーカ型パラメータサーバの実装と besom への適用. In **信学技報**, *vol. 115, no. 174, CPSY2015-33*, pages 179–184, 2015.

[8] Leslie G. Valiant. A bridging model for parallel computation. *Commun. ACM*, 33(8):103–111, August 1990.

[9] Huasha Zhao and John Canny. Butterfly mixing: Accelerating incremental-update algorithms on clusters. In *Proceedings of the 2013 SIAM International Conference on Data Mining*, 2013.

[10] Rajeev Thakur and W. D. Gropp. Improving the performance of mpi collective communication on switched networks. Technical Report ANL/MCS-P1007-1102, Argonne National Laboratory, 11/2002 2002.

[11] Simgrid: Versatile simulation of distributed systems: `http://simgrid.gforge.inria.fr/index.php`. Accessed: 2016-07-11.

[12] 黎 明曦, 谷村 勇輔, and 中田 秀基. パラメータサーバを用いた並列機械学習システムにおける耐故障性のシミュレーション. In **信学技報**, *vol. 116, no. 177, CPSY2016-20*, pages 125–130, 2016.

[13] Alekh Agarwal, Olivier Chapelle, Miroslav Dudík, and John Langford. A reliable effective terascale linear learning system. *J. Mach. Learn. Res.*, 15(1):1111–1133, January 2014.

[14] Qirong Ho, James Cipar, Henggang Cui, Jin Kyu Kim, Seunghak Lee, Phillip B. Gibbons, Garth A. Gibson, Gregory R. Ganger, and Eric P. Xing. More effective distributed ml via a stale synchronous parallel parameter server. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NIPS'13, pages 1223–1231, USA, 2013. Curran Associates Inc.

[15] Peter Jin, Qiaochu Yuan, Peter Jin, and Kurt Keutzer. How to scale distributed deep learning? In *ML Systems Workshop, NIPS 2016*, 2016.