DPDKのRun-to-Completionモデルを用いたL2分散計算環境の提案

山本 竜也[1]　川島 龍太[1]　松尾 啓志[1]

概要：本研究が提案する分散計算環境には特に分散機械学習では満たされることが多い 3 つの前提を設けた．TCP/IP による通信ではルーティング，誤り制御，順序制御といった制御が行われるが，それらは本研究が提案する分散計算環境の前提を踏まえるとオーバーヘッドである．また，カーネルによるパケット I/O は DPDK のパケット I/O に比べて低速である．DPDK には Run-to-Completion モデルと Pipeline モデルの 2 つのモデルがあるが，Pipeline モデルにはコアの使用効率が悪い問題と CPU の L1 キャッシュを有効活用できない問題がある．本研究では，DPDK の Run-to-Completion モデルを用いた L2 分散計算環境を提案する．計算機間の通信に DPDK による L2 通信を用いることによって，TCP/IP による通信のオーバーヘッドをなくすことやカーネルによるパケット I/O に比べて高速なパケット I/O を用いることができる．また，計算機をメッシュネットワークで接続し，Run-to-Completion モデルで処理を実行することによって，Pipeline モデルを用いる場合に比べてコアや CPU の L1 キャッシュを有効活用できる．送られてきた 32 個の値を一定回数足し合わせて送り返すプログラムを用いた実験では，提案手法を用いた場合のスループットは TCP/IP による通信を用いた場合に比べて最大 30 倍程度高いことがわかった．また，単回帰分析を行うプログラムを用いた評価では，提案手法を用いた 2 台での分散学習の実行時間は，1 台での集中学習の 50%程度，TCP/IP による通信を用いた 2 台での分散学習の 83%程度であることがわかった．

キーワード：分散計算環境，DPDK，Run-to-Completion，L2 通信，L1 キャッシュ

1. はじめに

本研究が提案する分散計算環境には 3 つの前提を設ける．1 つ目の前提は L2 通信が可能であるローカルなクラスタ環境で動作することである．2 つ目の前提は計算機間でやりとりされるデータのサイズは L2 フレームの 1.5KB より小さいことである．3 つ目の前提は各計算機で実行される処理はイテレーションが多いもので，その実行に必要なデータは小規模であることである．これらの前提は特に分散機械学習では満たされることが多いと考える．

TCP/IP による通信ではルーティング，誤り制御，順序制御といった制御が行われるが，それらは本研究が提案する分散計算環境においてはオーバーヘッドである．また，カーネルによるパケット I/O には，一定時間に受信するパケットの量が増えると，コンテキストスイッチが増加して，割り込み以外の処理が実行できなくなってしまうといった問題があり，DPDK のパケット I/O に比べて低速である．

DPDK には Run-to-Completion モデルと Pipeline モデルの 2 つのモデルがあるが，Pipeline モデルは受信処理，パケット処理，送信処理をそれぞれ別の論理コアで行うため，コアの使用効率が悪い問題と CPU の L1 キャッシュを有効活用できない問題がある．

そこで本研究では，DPDK の Run-to-Completion モデルを用いた L2 分散計算環境を提案する．計算機間の通信に DPDK による L2 通信を用いることによって，TCP/IP による通信のオーバーヘッドをなくすことやカーネルによるパケット I/O に比べて高速なパケット I/O を用いることができる．また，計算機をメッシュネットワークで接続し，Run-to-Completion モデルで処理を実行することによって，Pipeline モデルを用いる場合に比べてコアや CPU の L1 キャッシュを有効活用できる．

本稿の構成は以下のとおりである．第 2 章で分散計算環境の問題点を述べ，第 3 章で DPDK について説明する．第 4 章で関連研究を述べ，第 5 章で提案手法について述べる．第 6 章，第 7 章，第 8 章で提案手法の予備実験と評価を行い，第 9 章でまとめと今後の課題を述べる．

[1] 名古屋工業大学大学院
Nagoya Institute of Technology

2. 分散計算環境の問題点

本章では，分散計算環境の問題点として TCP/IP による制御とカーネルによるパケット I/O について述べる．

2.1 TCP/IP による制御

TCP/IP による通信ではルーティング，誤り制御，順序制御といった制御が行われる．ルーティングとは端末の相互接続関係や各端末間の通信回線の混み具合の情報を取得し，送信元端末から宛先端末までのルートを決定する制御であるが，本研究が提案する分散計算環境は L2 通信が可能であるローカルなクラスタ環境で動作するため，ルーティングは必ずしも必要ではない．誤り制御とはデータが伝送中に誤ったり失われたりしたとき，それを回復して受信側に正しいデータを送り届ける制御であるが，本研究が提案する分散計算環境の各計算機で実行される処理はイテレーションが多いものであるため，誤り制御も必ずしも必要ではない．順序制御とは受信データの重複をなくし，正しい順序に並び替える制御であるが，本研究が提案する分散計算環境の計算機間でやりとりされるデータのサイズは L2 フレームの 1.5KB より小さいため，順序制御も必ずしも必要ではない．

よって，TCP/IP による制御は本研究が提案する分散計算環境においてはオーバーヘッドであるため，本研究は TCP/IP による通信ではなく L2 通信を用いることにした．

2.2 カーネルによるパケット I/O

バージョン 2.6 以前のカーネルによるパケット I/O（図 1）では，パケットを受信するたびに NIC（Network Interface Card）からのハードウェア割り込みが発生する．そのため，一定時間に受信するパケットの量が増えると，コンテキストスイッチが増加して，割り込み以外の処理が実行できなくなってしまう．また，ユーザ空間からカーネル空間にはアクセスできないため，ユーザ空間のアプリケーションがパケットにアクセスするには，受信したパケットをカーネル空間からユーザ空間にコピーしなければならない．

バージョン 2.6 以降のカーネルには NAPI（New API）と呼ばれる仕組みが導入された．NAPI ではパケット受信によるハードウェア割り込みが発生すると，NIC からのハードウェア割り込みを一時的に無効化し，NIC のデバイスドライバの挙動を割り込み駆動からポーリング駆動に切り替える．そして，処理すべきパケットがなくなるまでポーリング駆動で処理を行い，通常の割り込み駆動に戻る．NAPI によって，一定時間に受信するパケットの量が増えると，コンテキストスイッチが増加して，割り込み以外の処理ができなくなってしまう問題は一定程度解決された．

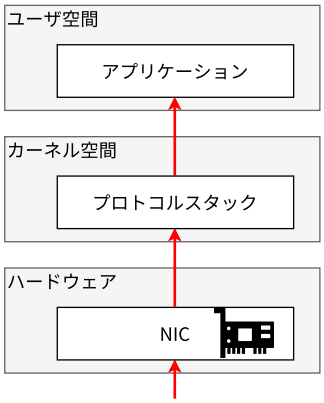

図 1 カーネルによるパケット I/O

3. DPDK(Data Plane Development Kit)

本章では，DPDK によるパケット I/O とその実行モデルについて述べる．また，カーネルによるパケット I/O と DPDK によるパケット I/O の比較や DPDK によるパケット I/O の問題点についても述べる．

3.1 DPDK によるパケット I/O

DPDK [1] とは 2010 年に Intel によって作られたパケット処理を高速化するためのライブラリである．

DPDK は特定の CPU コアを占有することによって，NIC を常時ポーリングで監視する．そのため，DPDK によるパケット I/O（図 2）では，一定時間に受信するパケットの量が増えると，コンテキストスイッチが増加して，割り込み以外の処理ができなくなってしまう問題は発生しない．

また，DPDK によるパケット I/O では，NIC は受信したパケットをユーザ空間からアクセス可能な主記憶領域に書き込む．そのため，受信したパケットをカーネル空間からユーザ空間にコピーしなくても，ユーザ空間のアプリケーションがパケットにアクセスすることができる．

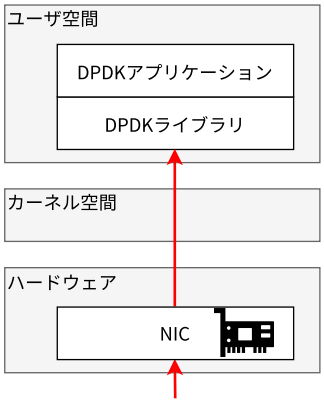

図 2 DPDK によるパケット I/O

3.2 DPDK の実行モデル

DPDK の実行モデルには Run-to-Completion モデルと Pipeline モデルの二つがある．Run-to-Completion モデルは受信処理，パケット処理，送信処理を一つの論理コアで行うモデルである（図 3）．パケット処理が重たいと受信処理に CPU リソースが割当たらなくなり，パケットロスが生じてしまうため，パケット処理ではパケットヘッダの書き換えといった軽い処理が一般的には行われる．Pipeline モデルは受信処理，パケット処理，送信処理をそれぞれ別の論理コアで行うモデルである（図 4）．受信処理を行う論理コアとパケット処理を行う論理コアが別であるため，パケット処理が重たくてもパケットロスが生じることはない．しかし，CPU の L1 キャッシュを有効活用できない．また，それぞれの処理が論理コアを占有するため，分散計算環境においてはコアの使用効率が悪い．

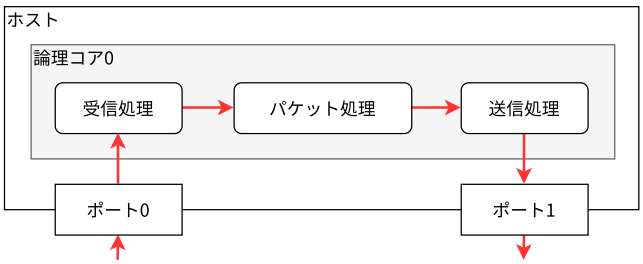

図 3　Run-to-Completion モデル

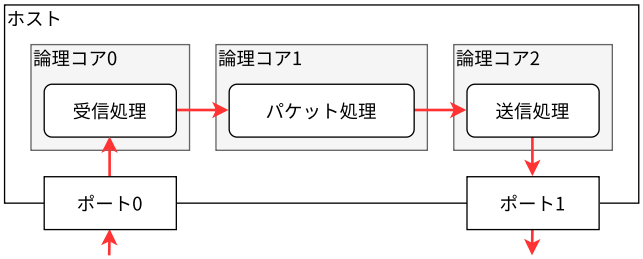

図 4　Pipeline モデル

3.3 パケット I/O の比較

文献 [2] で調査された，カーネルによるパケット I/O と DPDK によるパケット I/O の通信スループットを図 5 に示す．このグラフの横軸は使用した CPU コアの数，縦軸は受信したパケットをすべてドロップしたときのスループットを表している．また，緑の折れ線は DPDK によるパケット I/O，ピンクの折れ線はカーネルによるパケット I/O を表している．グラフより，DPDK のスループットはカーネルのスループットに比べて最大 8 倍程度高いことがわかる．

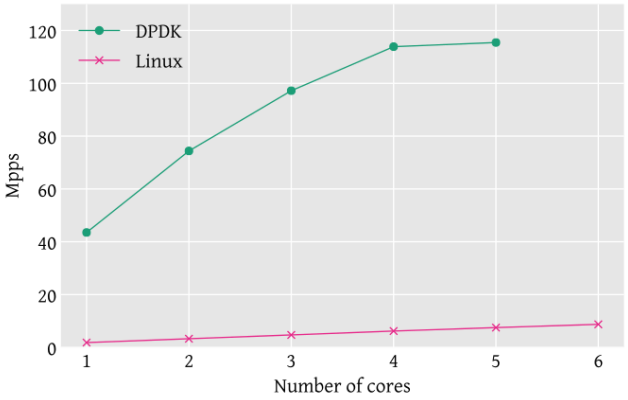

図 5　パケット I/O の比較

3.4 DPDK によるパケット I/O の問題

DPDK は特定の CPU コアを専有することによって，NIC を常時ポーリングで監視するため，通信スループットが低いときでも CPU リソースを無駄に使ってしまう．文献 [2] で調査された，カーネルによるパケット I/O と DPDK によるパケット I/O の CPU 使用率を図 6 に示す．このグラフの横軸は通信スループット，縦軸は CPU 使用率を表している．また，緑の折れ線は DPDK によるパケット I/O，青の折れ線はカーネルによるパケット I/O を表している．グラフから，カーネルによるパケット I/O の CPU 使用率は通信スループットが高くなるにしたがって徐々に増えていくことがわかる．それに対して，DPDK によるパケット I/O の CPU 使用率は通信スループットにかかわらず常に 100%であることがわかる．

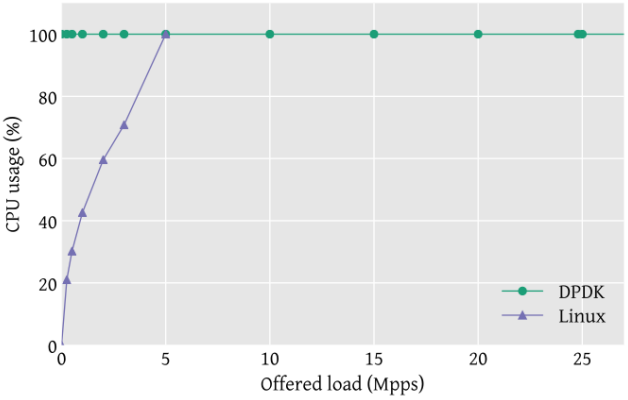

図 6　DPDK によるパケット I/O の問題

4. 関連研究

多元連立一次方程式の緩和法解析の分散処理に DPDK による通信を用いる研究 [3] がある．1992 年に上智大学の小石らによって行われた多元連立一次方程式の緩和法解析の分散処理に関する研究 [4] では，UDP によるブロードキャストが行われていた．そこで，この研究では多元連立一次方程式の緩和法解析の分散処理に DPDK を用いる

ことで，通信オーバーヘッドの削減による計算の高速化を行った．その結果，DPDK を用いた緩和法解析の分散処理アプリケーションは，UDP を用いた分散処理よりも最大40.1%高速化された．しかし，この研究では受信スレッド，収束判定または計算のスレッド，送信スレッドのそれぞれが論理コアを使用する Pipeline モデルを用いているため，コアの使用効率が悪い問題と CPU の L1 キャッシュを有効活用できない問題がある．

MPI 通信のデータ転送に DPDK を用いる研究 [5] もある．多くの MPI ライブラリはデータ転送を行うレイヤが独立しており，ユーザの環境に合わせてデータ転送方式やデバイスを MPI プログラムの実行に柔軟に切り替えることができる．そこで，この研究は MPI ライブラリの新たなデータ転送モジュールとして DPDK によるデータ転送モジュールを提案した．通信のスループットが低いときは Run-to-Completion モデル，高いときは Pipeline モデルを使用するようになっている．その結果，TCP/IP ソケットによるデータ転送を用いた場合と比べ，通信遅延を最大77%改善することができた．しかし，この研究では ACK パケットの授受を実装することによって，パケットロスに対する制御を行っているため，通信のオーバーヘッドがある．

5. 提案手法

本研究は DPDK の Run-to-Completion モデルを用いた L2 分散計算環境を提案する．本章では，その概要として DPDK による L2 通信を用いることと Run-to-Completion モデルで処理を実行することについて述べる．

5.1 DPDK による L2 通信を用いる

第 2 章で述べたとおり，TCP/IP による通信ではルーティング，誤り制御，順序制御といった制御が行われる．しかし，第 1 章で述べた本研究が提案する分散計算環境の前提を踏まえると，これらの制御は本研究においてはオーバーヘッドである．また，カーネルによるパケット I/O には，一定時間に受信するパケットの量が増えると，コンテキストスイッチが増加して，割り込み以外の処理が実行できなくなってしまう問題がある．また，ユーザ空間のアプリケーションがパケットにアクセスするには，受信したパケットをカーネル空間からユーザ空間にコピーしなければならない問題もある．これらの問題により，カーネルのパケット I/O は DPDK のパケット I/O に比べて低速である．

そこで，本研究では分散計算環境の計算機間の通信に DPDK による L2 通信を用いる．これによって，TCP/IP による通信のオーバーヘッドをなくすことができる．また，カーネルによるパケット I/O に比べて高速なパケット I/O を用いることもできる．

5.2 Run-to-Completion モデルで処理を実行する

第 3 章で述べたとおり，DPDK の Pipeline モデルは受信処理，パケット処理，送信処理をそれぞれ別の論理コアで行うため，コアの使用効率が悪い問題と CPU の L1 キャッシュを有効活用できない問題がある．また，DPDK は特定の CPU コアを専有することによって，NIC を常時ポーリングで監視するため，通信スループットが低いときでも CPU リソースを無駄に使ってしまう問題もある．

そこで，本研究では計算機をメッシュネットワークで接続し，Run-to-Completion モデルで処理を実行する（図 7）．受信処理，パケット処理，送信処理を一つの論理コアで行う Run-to-Completion モデルを用いることで，Pipeline モデルを用いる場合に比べて多くの計算機と接続することができ，コアを有効活用できる．また，本研究が提案する分散計算環境の計算機間でやり取りされるデータのサイズは L2 フレームの 1.5KB より小さいため，CPU の L1 キャッシュを有効活用できる．さらに，メッシュネットワークで接続された計算機が協調して処理を実行することによって，通信スループットが低い状態にならず，CPU リソースを有効活用できる．

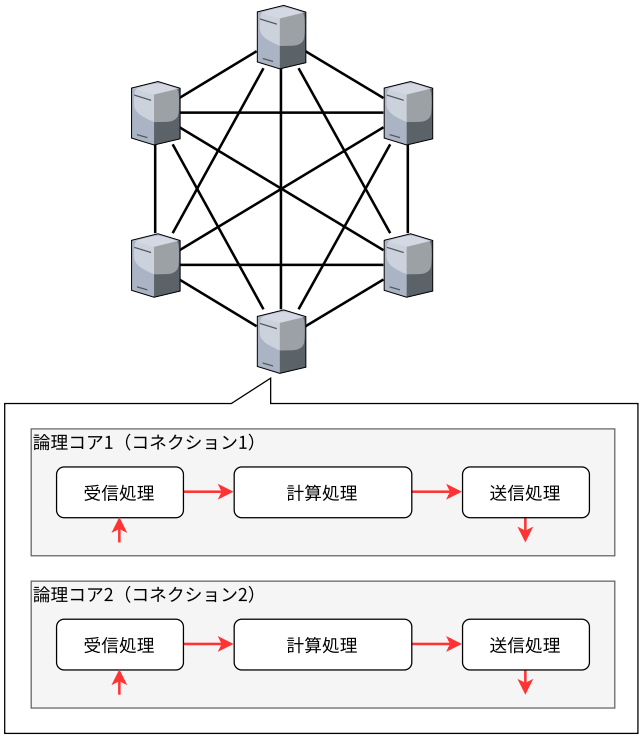

図 7 提案手法

6. 事前実験 1

本章では，DPDK の Run-to-Completion モデルで実行する処理の負荷によって，スループットはどのように変化するかを調査するために行った事前実験 1 について述べる．

6.1 事前実験 1 用のプログラム

事前実験 1 用のプログラムとして，クライアントから送られてきたパケットを一定時間待機させてから送り返すプログラムを作成した（図 8）．DPDK の Run-to-Completion モデルで実行する処理は，受信したパケットを待機させる処理である．パケットを待機させる時間を変更することによって，DPDK の Run-to-Completion モデルで実行する処理の負荷を擬似的に変更することができる．

図 8 事前実験 1 用のプログラム

6.2 実験環境

事前実験 1 で用いたネットワーク構成を図 9 に示す．ホスト A においてクライアントを動作させ，ホスト B において事前実験 1 用のプログラムを動作させた．クライアントには，DPDK ベースのパケットジェネレータである Pktgen [6] を用いた．事前実験 1 で用いた計算機の性能を表 1，Pktgen の設定を表 2 に示す．

パケットはクライアントが動作するホスト A のポート 0 から送信され，事前実験 1 用のプログラムが動作するホスト B のポート 0 に到着する．到着したパケットは事前実験 1 用のプログラムによって待機させられてから，ホスト B のポート 1 からホスト A のポート 1 へと送り返される．

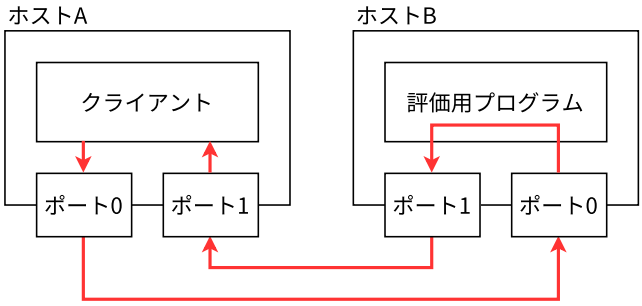

図 9 事前実験 1 で用いたネットワーク構成

表 1 事前実験 1 で用いた計算機の性能

OS	Ubuntu 20.04
CPU	AMD Ryzen 5 3400G (4-cores, 8-threads)
Memory	16GB
NIC	Intel X540-AT2 (10GbE, 2-ports)

表 2 Pktgen の設定

パケットのサイズ	64 バイト, 128 バイト
送信するパケットの数	100,000,000 パケット
送信レート	100%

6.3 実験結果・考察

事前実験 1 の結果を図 10 に示す．このグラフの横軸はパケットの待機時間，縦軸は通信のスループットを表している．また，青い折れ線はパケットサイズが 64 バイトのとき，赤い折れ線はパケットサイズが 128 バイトのときを表している．グラフより，パケットサイズが 64 バイトの場合，待機時間が 1600ns 以下のときスループットは一定であり，それを超えると単調減少することがわかる．また，パケットサイズが 128 バイトの場合，待機時間が 3200ns 以下のときスループットは一定であり，それを超えると単調減少することがわかる．よって，送信するパケットのサイズが小さく，送信レートが 100%という厳しい条件でも，実行時間が 1600ns 以下の計算処理であればスループットに影響を与えずに DPDK の送受信スレッドで実行できる．

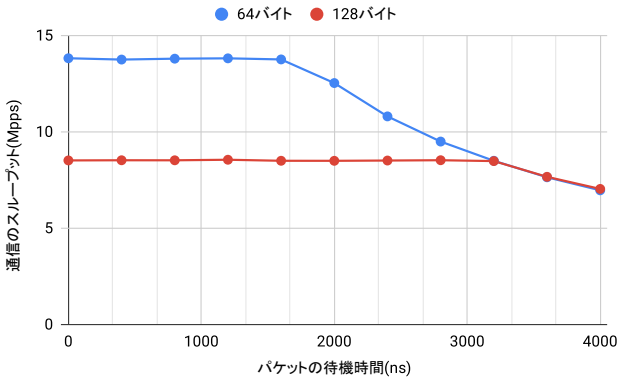

図 10 事前実験 1 の結果

7. 事前実験 2

本章では，提案手法の有効性を確認するために行った事前実験 2 について述べる．事前実験 2 では，送られてきた 32 個の値を一定回数足し合わせて送り返すプログラムにおいて，TCP/IP による通信を用いる場合と提案手法を用いる場合の通信のスループットを比較した．

7.1 事前実験 2 用のプログラム

事前実験 2 用のプログラムとして，クライアントから送られてきた 32 個の値を一定回数足し合わせて送り返すプログラムを作成した（図 11）．DPDK の Run-to-Completion モデルで実行する処理は受信したパケットに含まれる 32 個の値を足し合わせる処理である．このプログラムは TCP/IP による通信を用いるものと提案手法を用いるものの 2 種類を作成した．

図 11　事前実験 2 用のプログラム

7.2　実験環境

事前実験 2 で用いたネットワーク構成は事前実験 1 で用いたもの（図 9）と同じである．ただし，クライアントにはパケットサイズが 64 バイトのパケットを送信レート 100% で送信し続ける自作プログラムを用いた．送信されるパケットのペイロードは要素数が 32 でデータ型が uint16_t の配列である．なお，事前評価 2 で用いた計算機の性能は事前実験 1 で用いたもの（表 1）と同じである．

7.3　実験結果・考察

事前実験 2 の結果を図 12 に示す．このグラフの横軸は受信した 32 個の値を足し合わせる回数，縦軸は通信のスループットを表している．また，青いバーは TCP/IP による通信を用いた場合，赤いバーは提案手法を用いた場合を表している．グラフより，提案手法を用いた場合のスループットは TCP/IP による通信を用いた場合に比べて最大 30 倍程度高いことがわかる．よって，分散計算環境において，DPDK による L2 通信を用いることと DPDK の Run-to-Completion モデルで処理を行うことは有効であると考える．なお，提案手法を用いた場合のスループットが足し合わせる回数が多くなるにつれて下がっていくのは，計算処理が重たくなり受信処理に CPU リソースが割当たらなくなったためである．

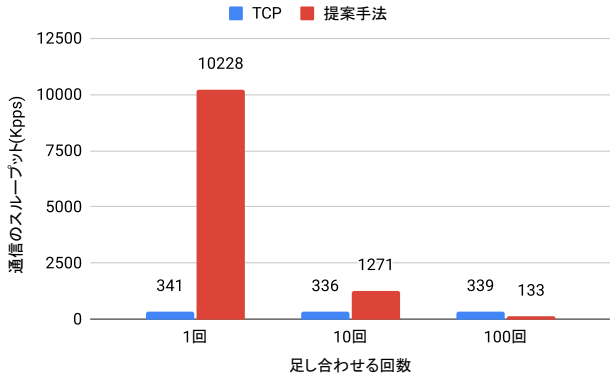

図 12　事前実験 2 の結果

8.　評価

本章では，提案手法の有効性を確認するために行った評価について述べる．評価では，単回帰分析において，1 台での集中学習を行う場合，TCP/IP による通信を用いた分散学習を行う場合，提案手法を用いた分散学習を行う場合の実行時間を比較した．

8.1　評価用プログラム

評価用プログラムとして単回帰分析を行うプログラムを作成した．単回帰分析とは，与えられたデータを用いて $y = ax + b$ の傾き a と切片 b を求める問題である．パラメータの最適化には確率的勾配降下法（Stochastic Gradient Descent, SGD）を用いた．確率的勾配降下法とは，データをランダムに選んで以下の更新を繰り返し，f を最小化するアルゴリズムである．なお，パラメータを w，学習率を α，目的関数を f とする．

$$w \leftarrow w - \alpha \nabla f(w) \tag{1}$$

複数台の計算機で分散して学習を行う場合はパラメータの集約が必要となる．パラメータの集約には Gossip Learning [7], [8] を用いた．Gossip Learning において，各計算機は確率的勾配降下法によってパラメータを更新する．そして，定期的にランダムに選んだ他の計算機と通信し，パラメータの平均を計算する．このようなパラメータの更新を繰り返していくことによって学習が進む．

このプログラムは 1 台での集中学習を行うもの，TCP/IP による通信を用いた分散学習を行うもの，提案手法を用いた分散学習を行うものの 3 種類を作成した．

8.2　評価環境

評価で用いたネットワーク構成を図 13 に示す．計算機をメッシュネットワークで接続し，それぞれの計算機で評価用プログラムを動作させた．なお，評価で用いた計算機の性能は事前実験 1 で用いたもの（表 1）と同じである．

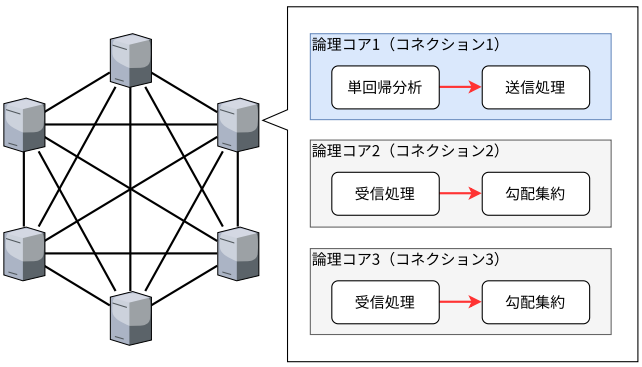

図 13　評価で用いたネットワーク構成

8.3　評価結果・考察

評価結果を図 14 に示す．このグラフの横軸は左から 1 台での集中学習の場合，TCP/IP による通信を用いた 2 台

での分散学習の場合，提案手法を用いた 2 台での分散学習の場合を表しており，縦軸は実行時間を表している．グラフより，提案手法を用いた 2 台での分散学習の実行時間は，1 台での集中学習の 50%程度，TCP/IP による通信を用いた 2 台での分散学習の 83%程度であることがわかる．よって，分散計算環境において，DPDK による L2 通信を用いることと DPDK の Run-to-Completion モデルで処理を行うことは有効であると考える．

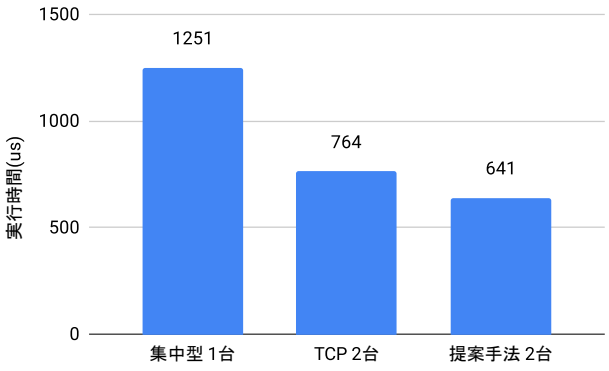

図 14 評価結果

9. まとめと今後の課題

本稿では，DPDK の Run-to-Completion モデルを用いた L2 分散計算環境を提案した．計算機間の通信に DPDK による L2 通信を用いることによって，TCP/IP による通信のオーバーヘッドをなくすことやカーネルによるパケット I/O に比べて高速なパケット I/O を用いることができる．また，計算機をメッシュネットワークで接続し，Run-to-Completion モデルで処理を実行することによって，Pipeline モデルを用いる場合に比べてコアや CPU の L1 キャッシュを有効活用できる．送られてきた 32 個の値を一定回数足し合わせて送り返すプログラムを用いた事前実験 2 では，提案手法を用いた場合のスループットは TCP/IP による通信を用いた場合に比べて最大 30 倍程度高いことがわかった．また，単回帰分析を行うプログラムを用いた評価では，提案手法を用いた 2 台での分散学習の実行時間は，1 台での集中学習の 50%程度，TCP/IP による通信を用いた 2 台での分散学習の 83%程度であることがわかった．

今後の課題としては，使用する計算機の台数を増やして評価を取ること，提案手法のどの部分が性能向上に寄与しているのかを調べるために Raw Socket を用いてカーネルによる L2 通信を実装して評価を取ること，ロジスティック回帰やサポートベクターマシン（Support Vector Machine, SVM）といった複雑な機械学習を実行することなどがある．

参考文献

[1] The Linux Foundation: Home - DPDK, The Linux Foundation (online), available from ⟨https://www.dpdk.org/⟩ (accessed 2022-06-10).
[2] Toke Høiland-Jørgensen and Jesper Dangaard Brouer and Daniel Borkmann and John Fastabend and Tom Herbert and David Ahern and David Miller: The eXpress data path: fast programmable packet processing in the operating system kernel, *Proceedings of the 14th International Conference on emerging Networking EXperiments and Technologies*, Association for Computing Machinery, pp. 54–66 (2018).
[3] 伴野隼一，矢吹道郎：DPDK の超高速通信を活用した，緩和法解析の分散処理に関する研究，研究報告マルチメディア通信と分散処理（DPS），Vol. 2021-DPS-186, No. 15, pp. 1–7 (2021).
[4] 小石　昇：分散処理におけるブロードキャストを利用した共有データの参照および更新に関する研究，修士論文，上智大学理工学研究科・電気電子工学専攻 (1992).
[5] 島田明男，早坂光雄：DPDK によるデータ転送を用いた MPI 通信の実装，研究報告ハイパフォーマンスコンピューティング（HPC），Vol. 2017-HPC-158, No. 13, pp. 1–7 (2017).
[6] Keith Wiles: pktgen/Pktgen-DPDK: DPDK based packet generator, Intel Corporation (online), available from ⟨https://github.com/pktgen/Pktgen-DPDK⟩ (accessed 2022-06-10).
[7] Peter H Jin and Qiaochu Yuan and Forrest Iandola and Kurt Keutzer: How to scale distributed deep learning?, *arXiv preprint arXiv:1611.04581* (2016).
[8] 小国英明，高橋良希，首藤一幸：広域分散を想定した深層学習手法の比較，データ工学と情報マネジメントに関するフォーラム (2019).