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　川島 龍太　松尾 啓志

名古屋工業大学大学院

2022/7/27

研究背景

- 分散計算環境で実行する処理の中には，与えられたデータを繰り返し用いながら，小規模なデータを計算機間でやりとりする処理が存在
  - 単回帰分析やロジスティック回帰などの機械学習

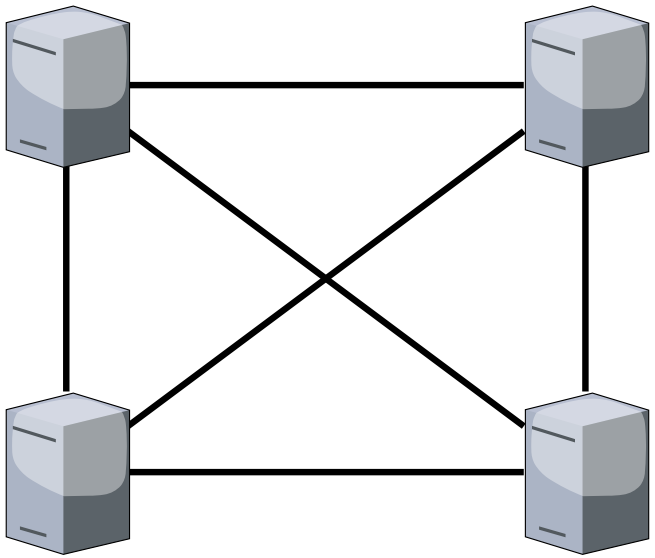
問題点

- TCP/IP による制御は相対的に大きなオーバーヘッド
- カーネルによるパケット I/O 処理は DPDK のパケット I/O 処理に比べて低速

DPDK の Run-to-Completion を用いた L2 分散計算環境

提案する分散処理環境の前提

1. 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
2. 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（ジャンボフレームを含む）より小さい
3. 各計算機で実行される処理はイテレーションが多用され，その実行に必要なデータは小規模である



機械学習において満たされることが多いと考える

TCP/IP による制御

誤り制御

- 各計算機で実行される処理はイテレーションが多用される前提のため，多少の誤りであれば処理結果への影響は軽微

順序制御

- 計算機間でやりとりされるデータのサイズは L2 フレームより小さい前提のため，必ずしも必要ではない

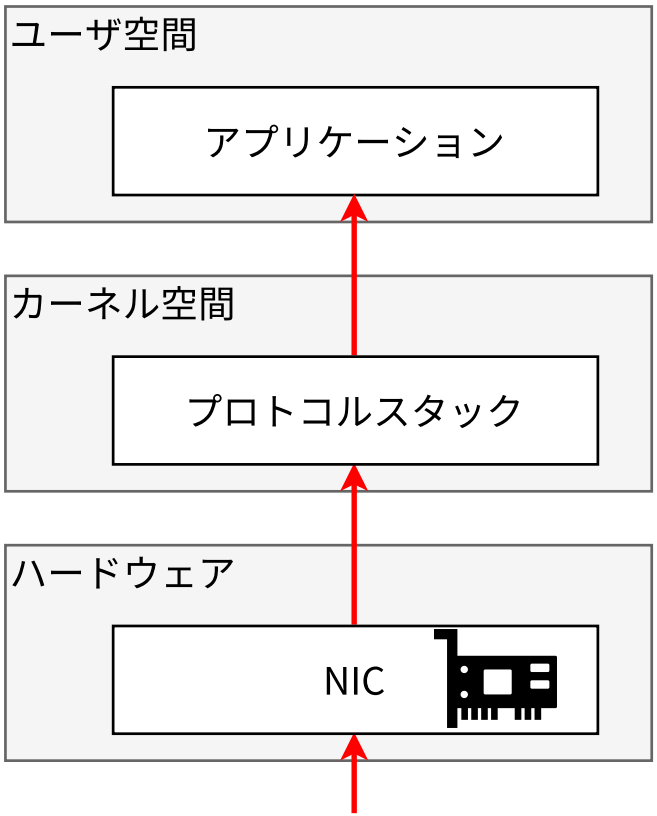
フロー制御

- 各計算機で実行される処理はイテレーションが多用される前提のため，多少のパケットロスであれば処理結果への影響は軽微

TCP/IP による通信ではなく L2 通信を用いる

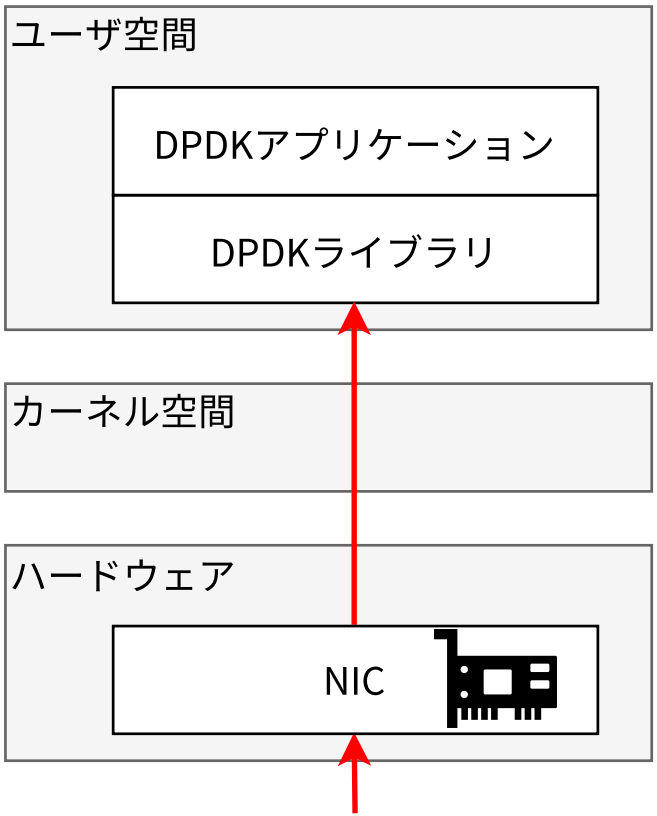
カーネルによるパケットI/O処理

- パケットを受信するたびにNICからのハードウェア割り込みが発生する
- カーネル空間からユーザ空間へのパケットコピーが必要



DPDKによるパケットI/O処理

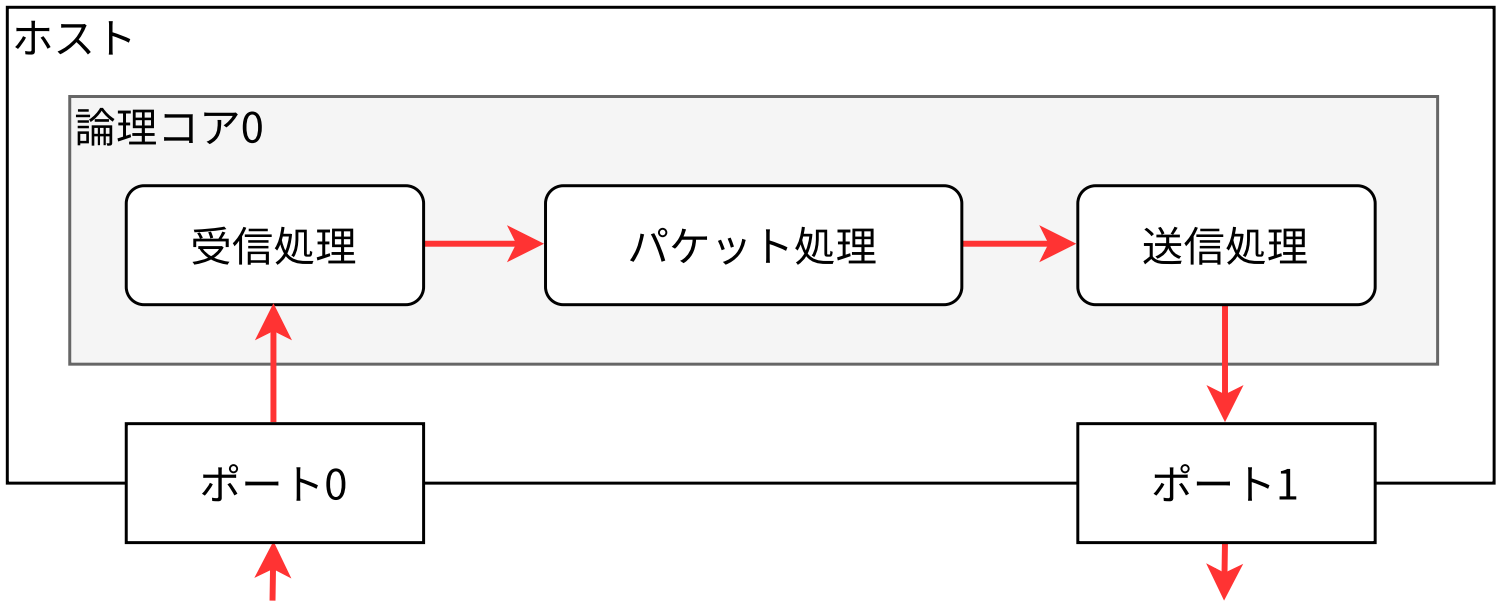
- 特定のCPUコアを専有することによって，NICをユーザ空間から常時ポーリングで監視
  - NICからのハードウェア割り込みが発生しない
  - カーネル空間からユーザ空間へのパケットコピーが不要に



DPDK の実行モデル

Run-to-Completion モデル

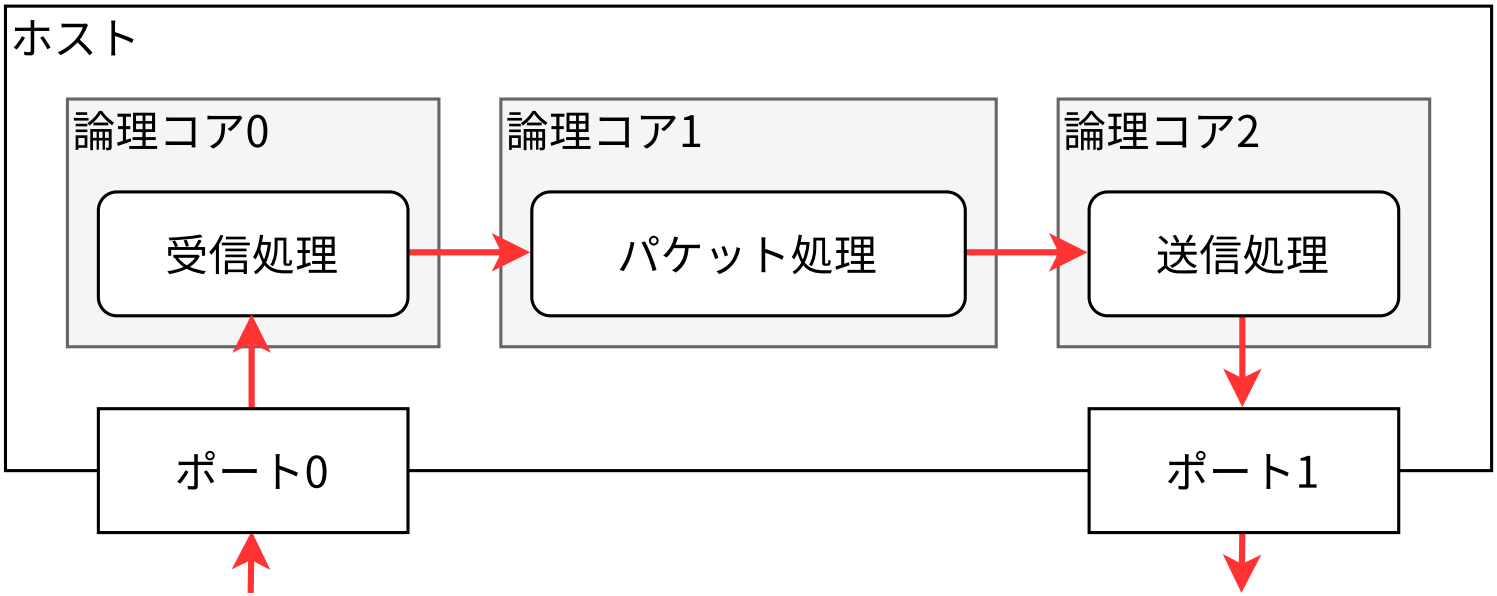
- 受信処理，パケット処理，送信処理を一つの論理コアで行うモデル
- パケット処理が重い場合は受信処理に CPU リソースが割り当たらず，パケットロスが生じる



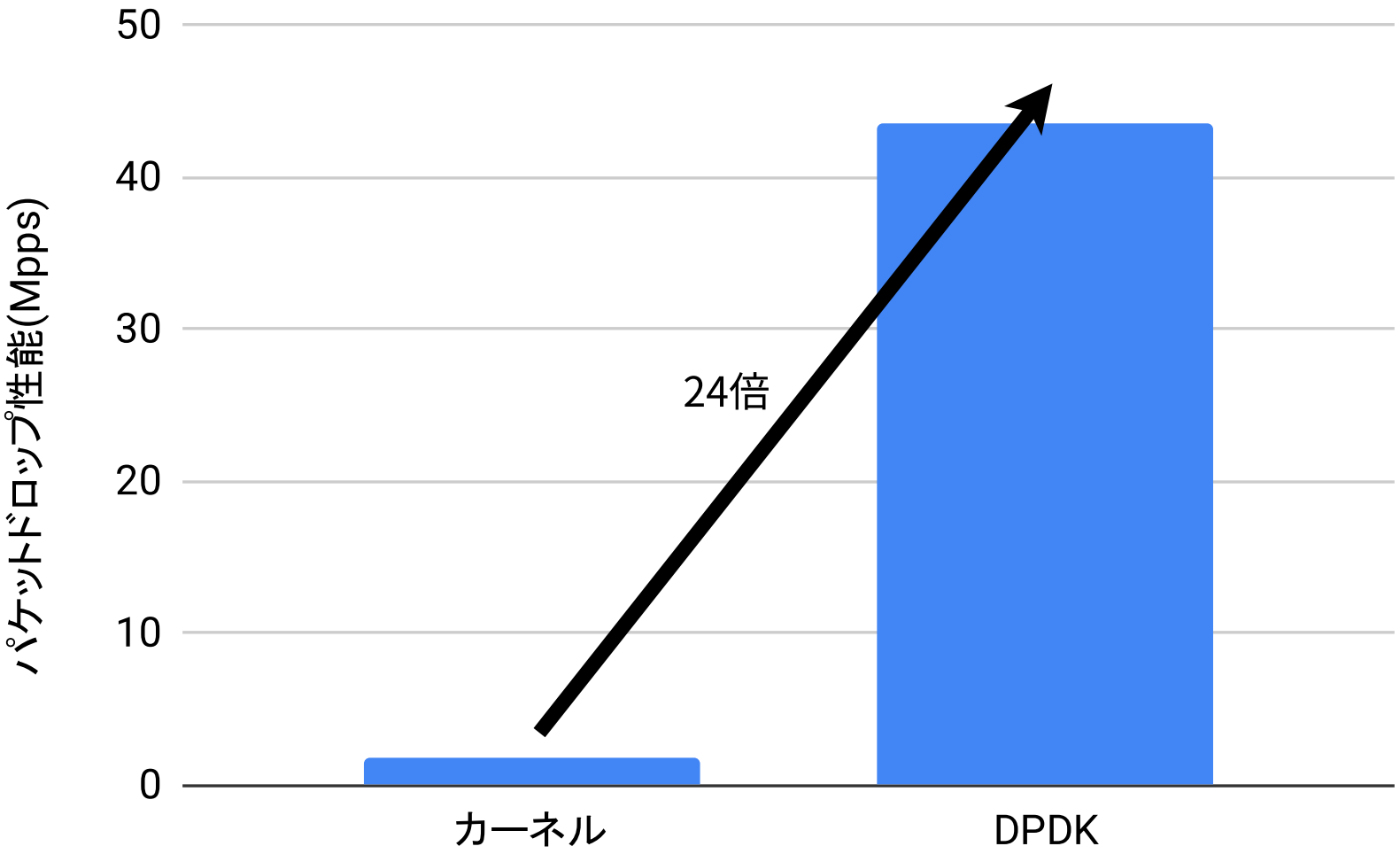
DPDK の実行モデル

Pipeline モデル

- 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
- 必要なデータの大部分が L1 キャッシュに存在して処理速度が向上するという効果や CPU リソースを有効活用できない



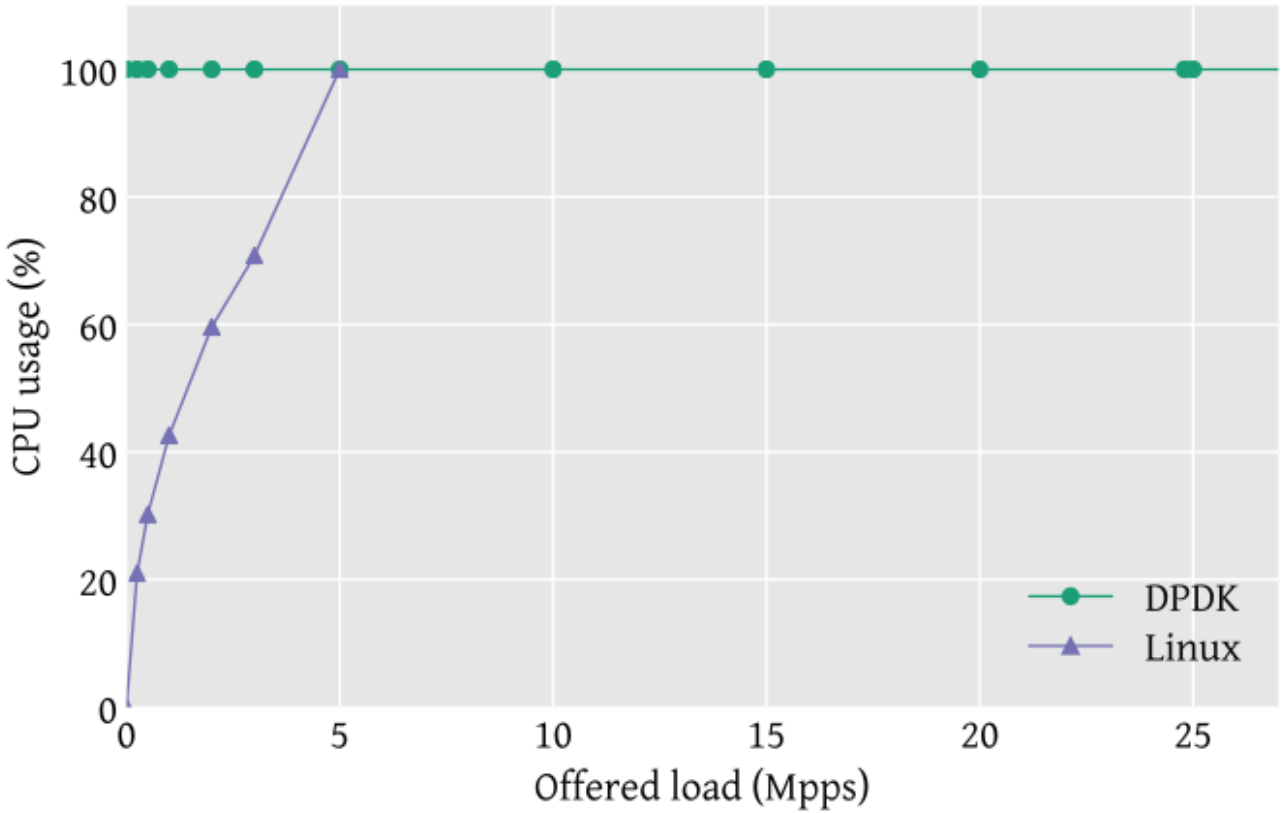
パケットI/O処理の比較



Toke Høiland-Jørgensen et al: The eXpress data path: fast programmable packet processing in the operating system kernel, CoNEXT, 2018

DPDKによるパケットI/O処理の問題点

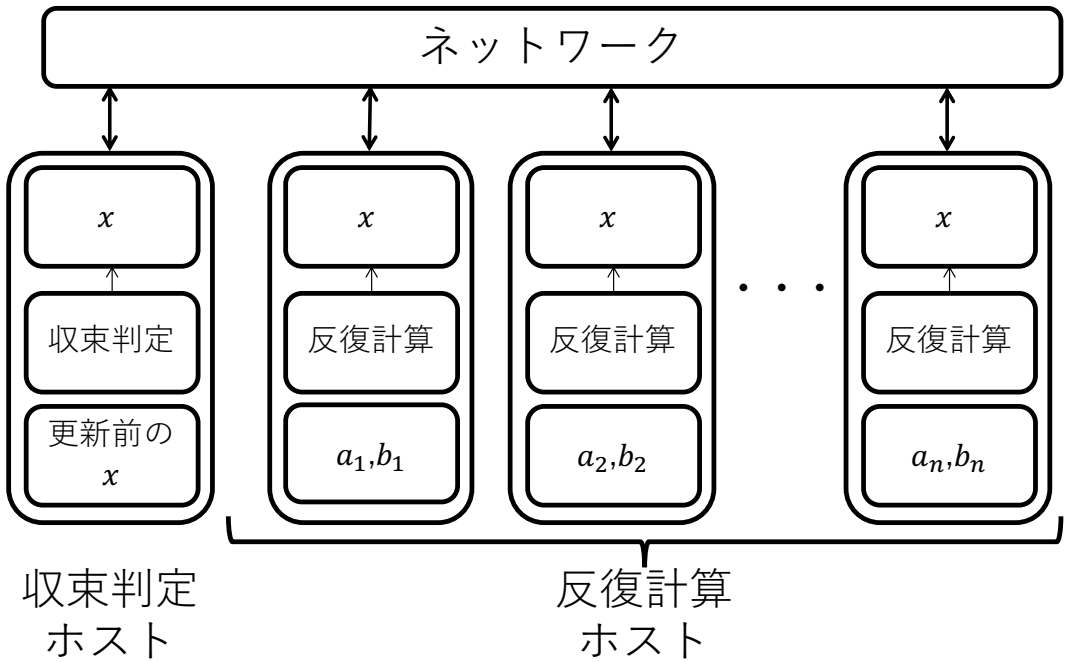
- 特定のCPUコアを専有することによって，NICを常時ポーリングで監視するため，通信負荷が低いときでもCPUリソースを無駄に使用する



Toke Høiland-Jørgensen et al: The eXpress data path: fast programmable packet processing in the operating system kernel, CoNEXT, 2018

関連研究 1

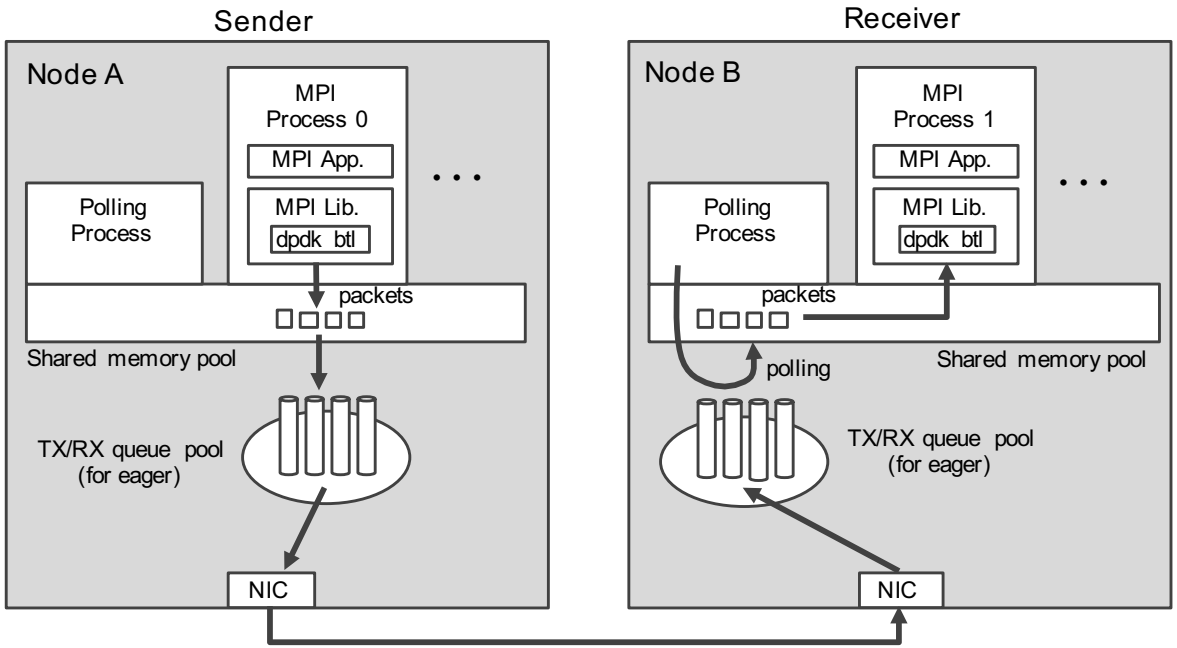
- 多元連立一次方程式の緩和法解析の分散処理に DPDK による通信を用いる研究
  - UDP を用いた分散処理よりも最大 40.1%高速化
  - Pipeline モデルを用いているため，CPU リソースや L1 キャッシュを有効活用できていない



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），2021

関連研究 2

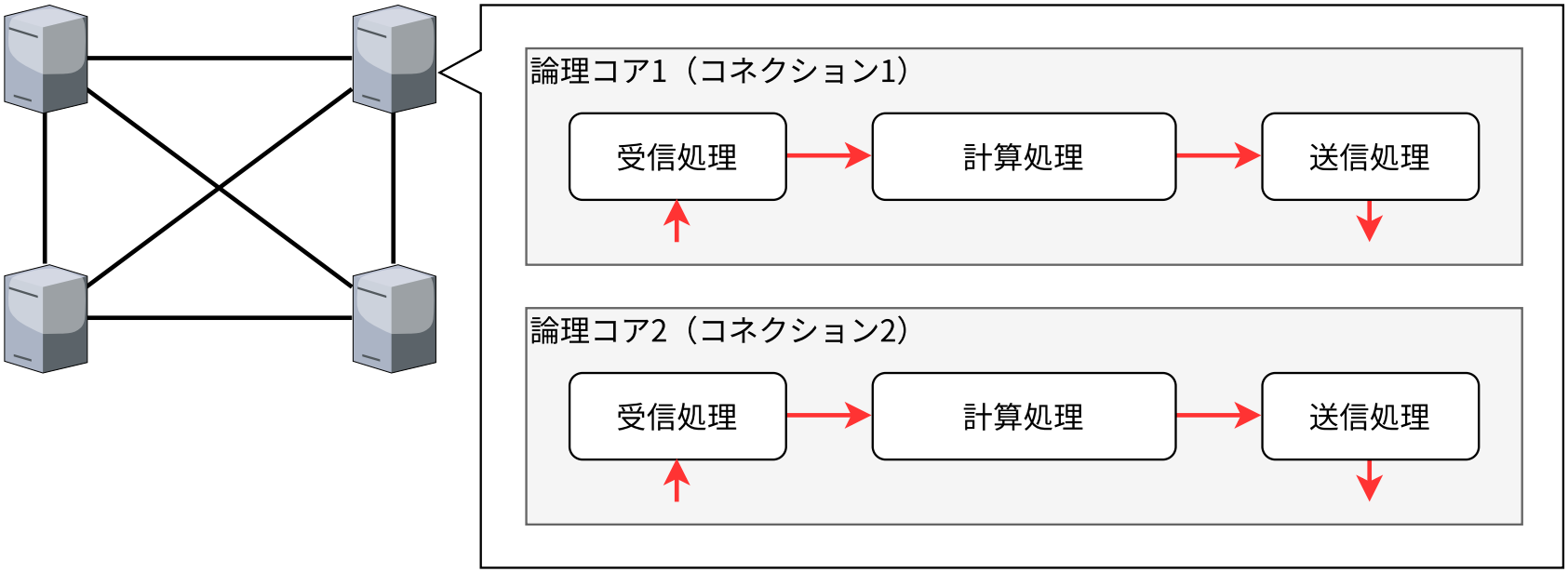
- MPI 通信のデータ転送に DPDK を用いる研究
  - TCP/IP ソケットによるデータ転送を用いた場合に比べて，通信遅延が最大 77%改善
  - パケットロスに対する制御を行っているため，通信のオーバーヘッドが高い



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），2017

提案手法

- DPDK の Run-to-Completion モデルを用いた L2 分散計算環境
  - DPDK による L2 通信の使用
    - 通信処理のオーバーヘッドを低減
    - カーネルによるパケット I/O 処理より高速なパケット I/O 処理
  - Run-to-Completion モデルの採用
    - CPU リソースを有効活用できる
    - 計算処理時に L1 キャッシュを有効活用できる

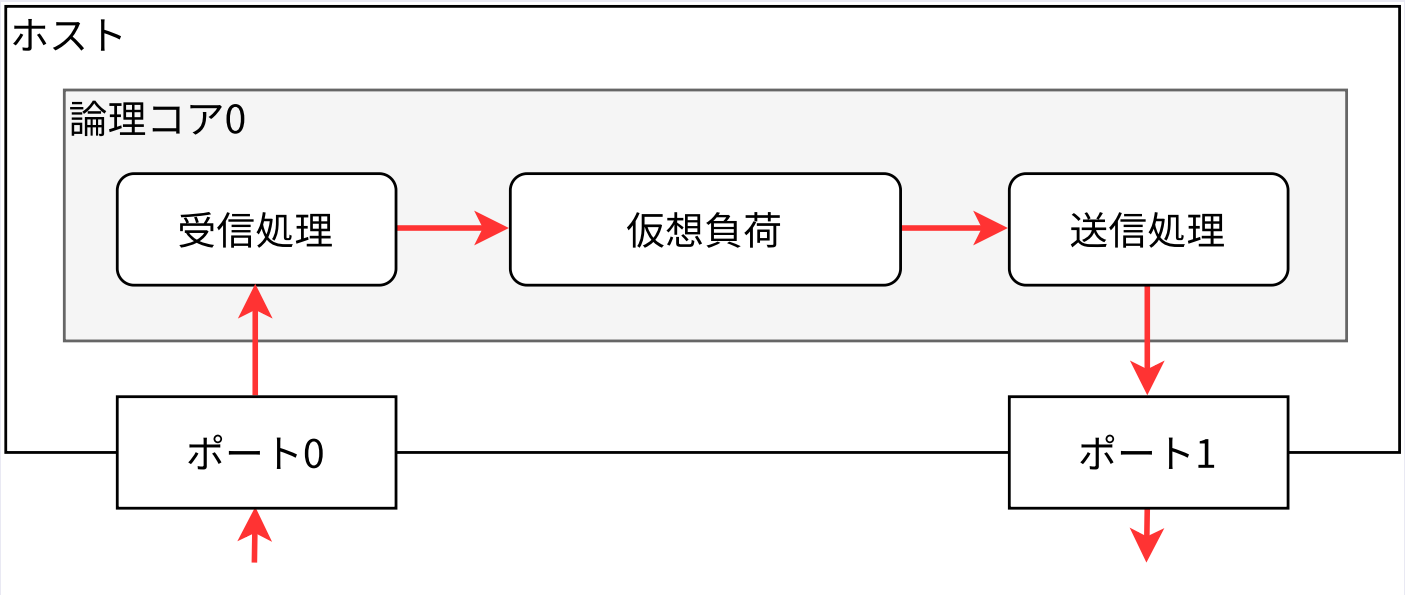


事前実験１

実験内容

- DPDK の Run-to-Completion モデルで実行する処理の負荷とスループットの関係を調査

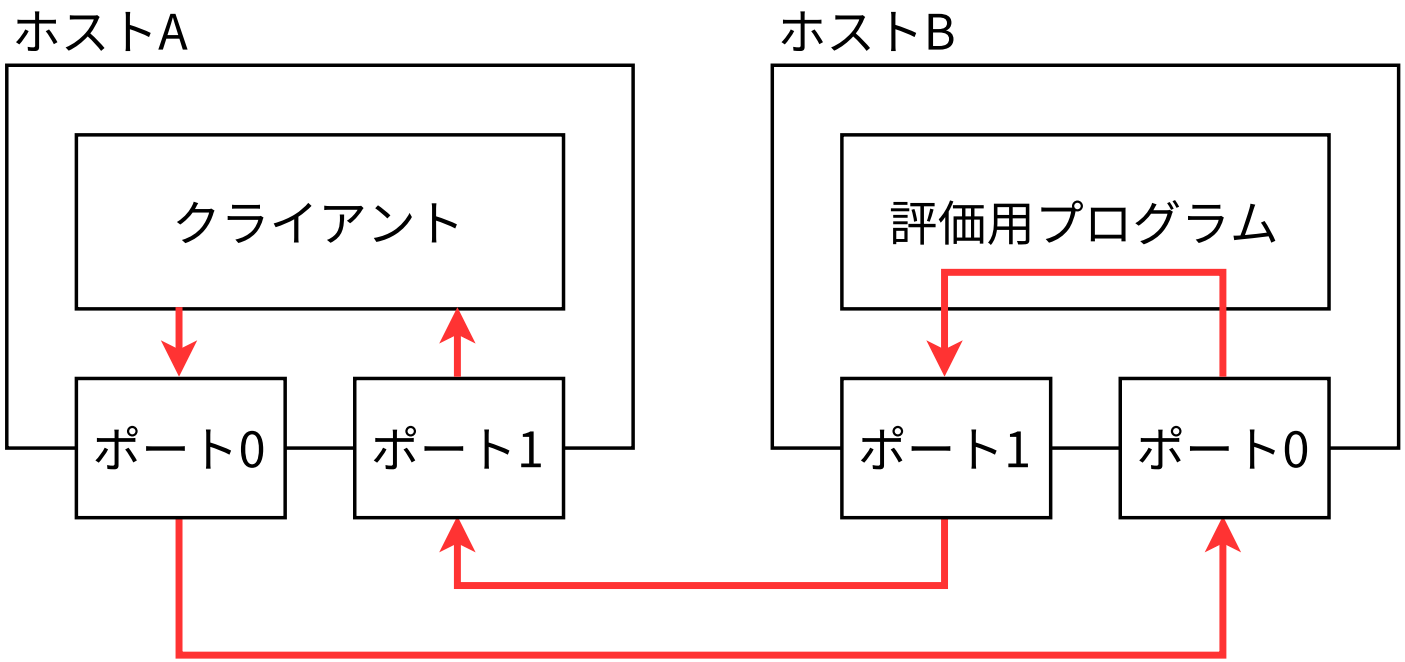
事前実験１用のプログラム



事前実験 1

実験環境

- 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 を用いた



事前実験1

計算機の性能

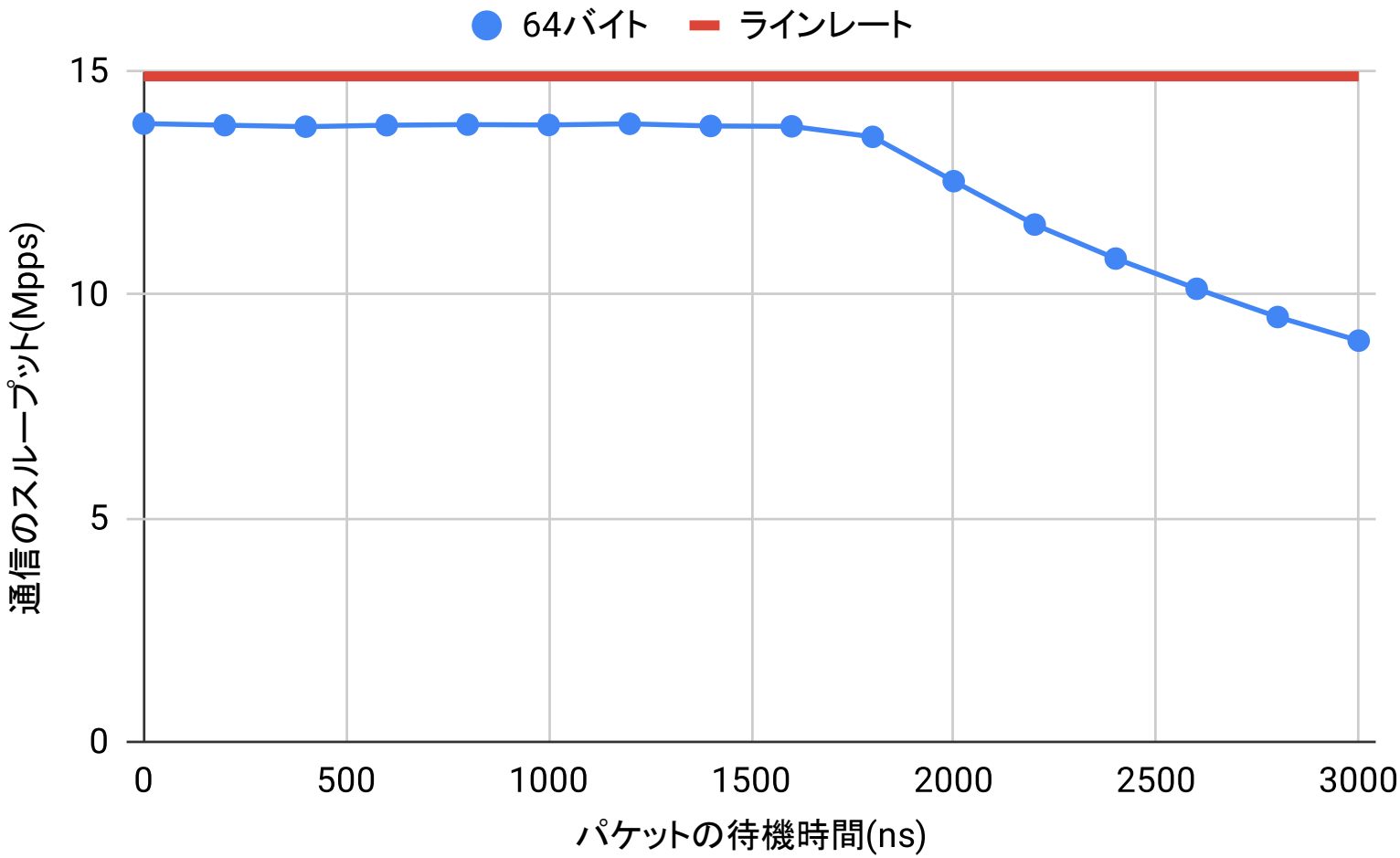
OS	Ubuntu 20.04
CPU	AMD Ryzen 5 3400G (4-cores, 8-threads)
Memory	16GB
NIC	Intel X540-AT2 (10GbE, 2-ports)

Pktgen の設定

パケットのサイズ	64 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%

事前実験１の結果

- 処理時間が 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 Run-to-Completion モデルで実行できる

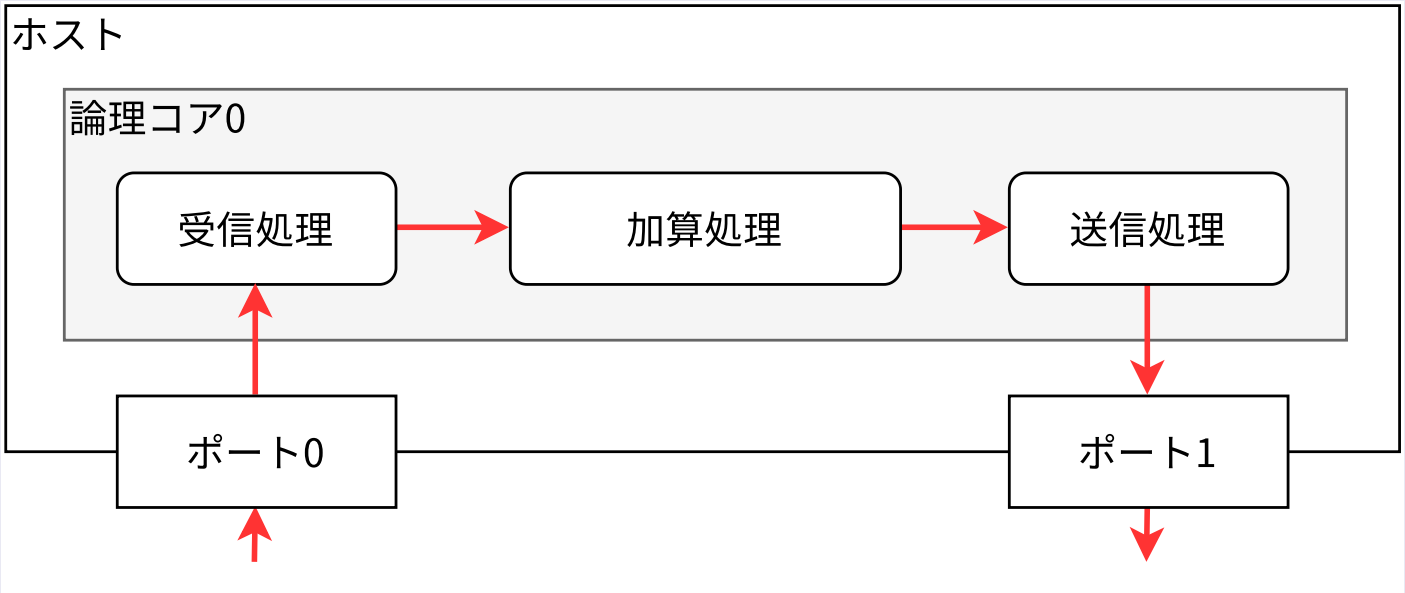


事前実験2

実験内容

- 送られてきた32個の値を加算し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単位時間あたりの演算性能を比較

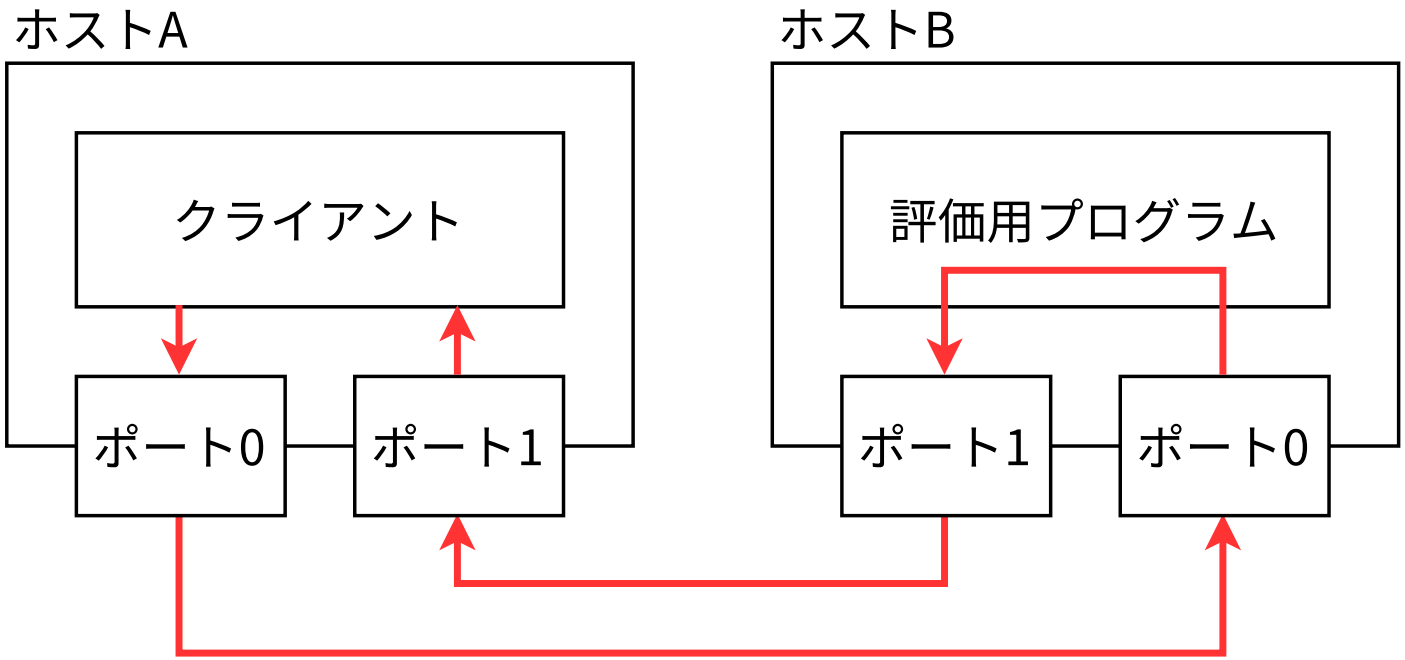
事前実験2用のプログラム



事前実験2

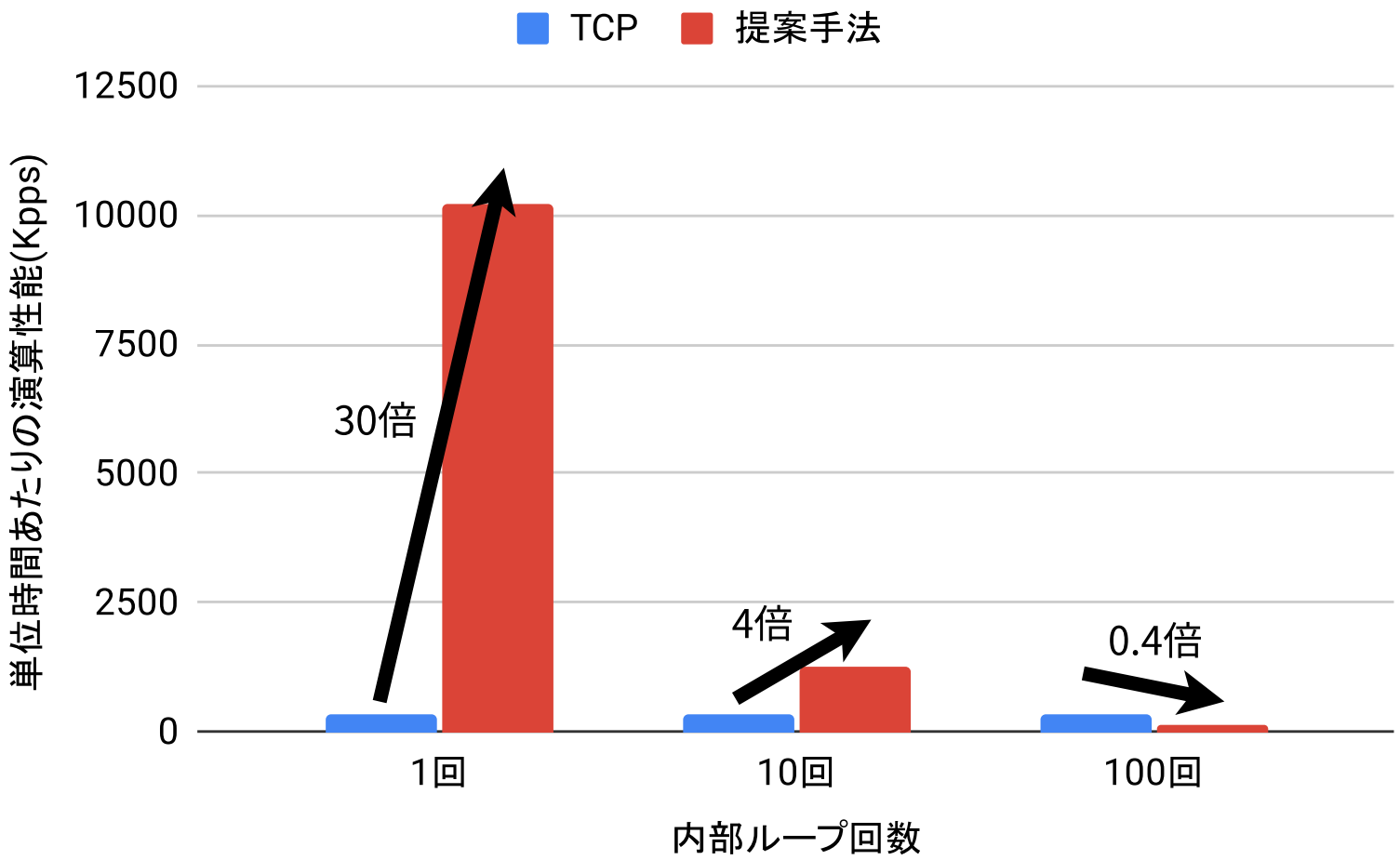
実験環境

- 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



事前実験2の結果

- 事前実験2用のプログラムでは，仮想的に内部演算量を増やすために，同一の加算処理を繰り返しており，その回数を内部ループ回数と定義



評価

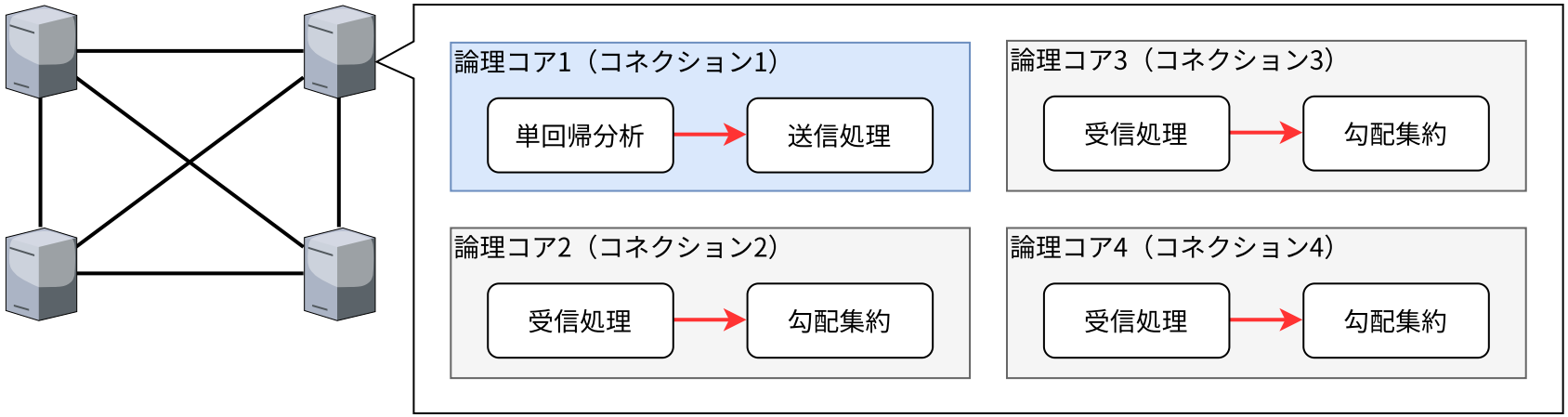
評価内容

- 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

単回帰分析

- 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
  - パラメータの最適化には確率的勾配降下法を用いた
  - パラメータの集約には Gossip Learning を用いた

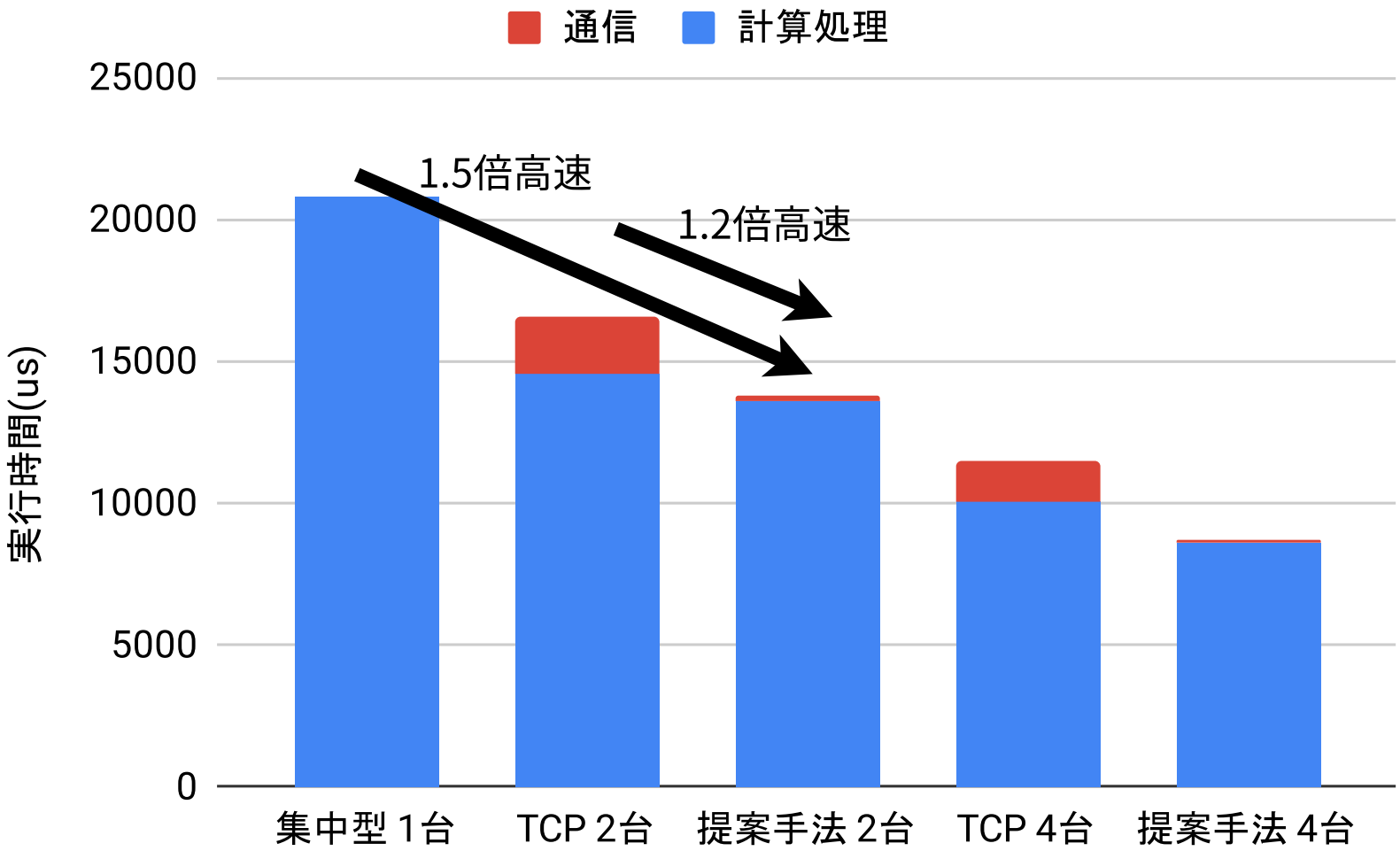
評価



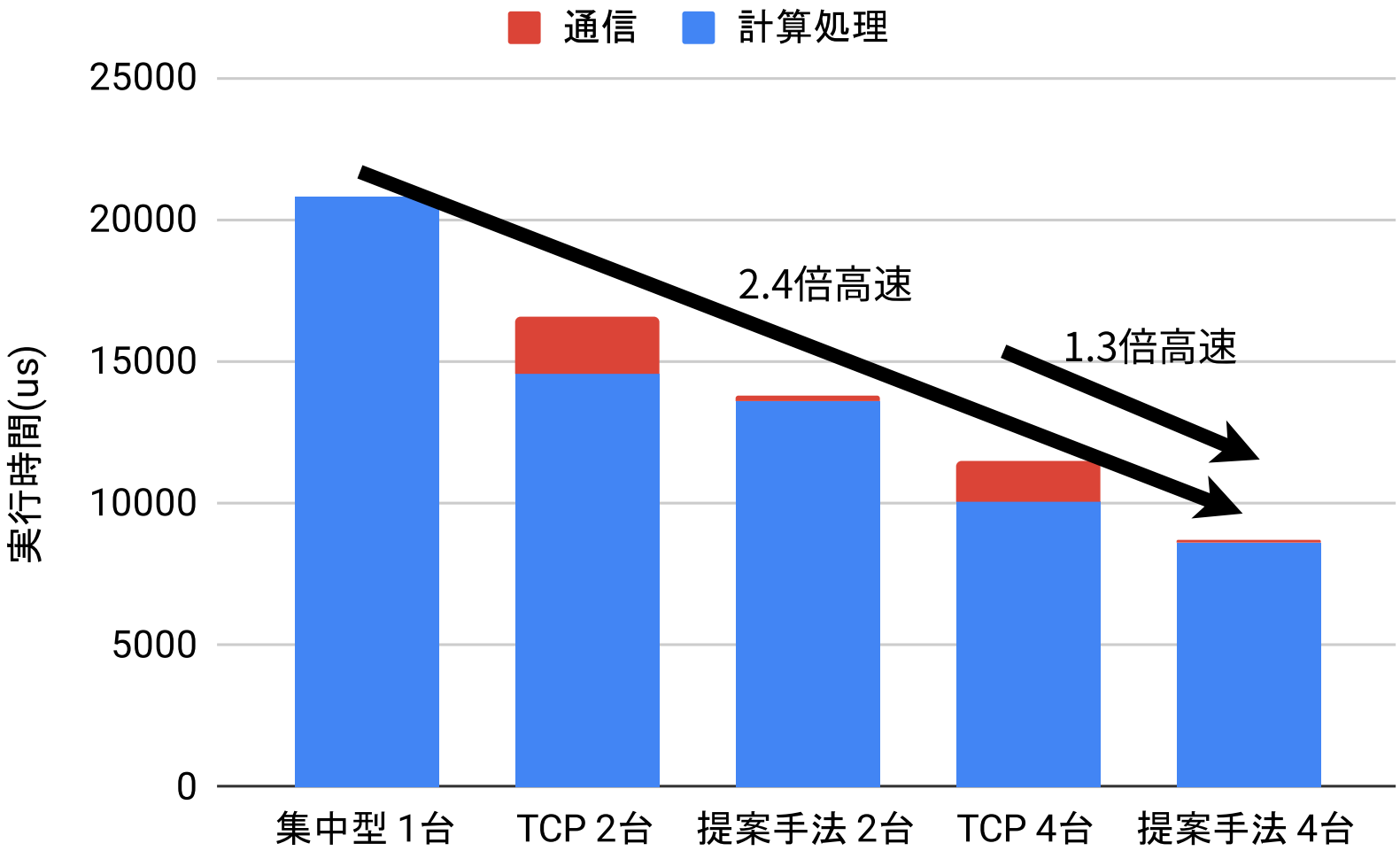
計算機の性能

OS	Ubuntu 20.04
CPU	Intel Core i5-4460 (4-cores, 4-threads)
Memory	16GB
NIC	Intel X540-AT2 (10GbE, 2-ports)

評価結果

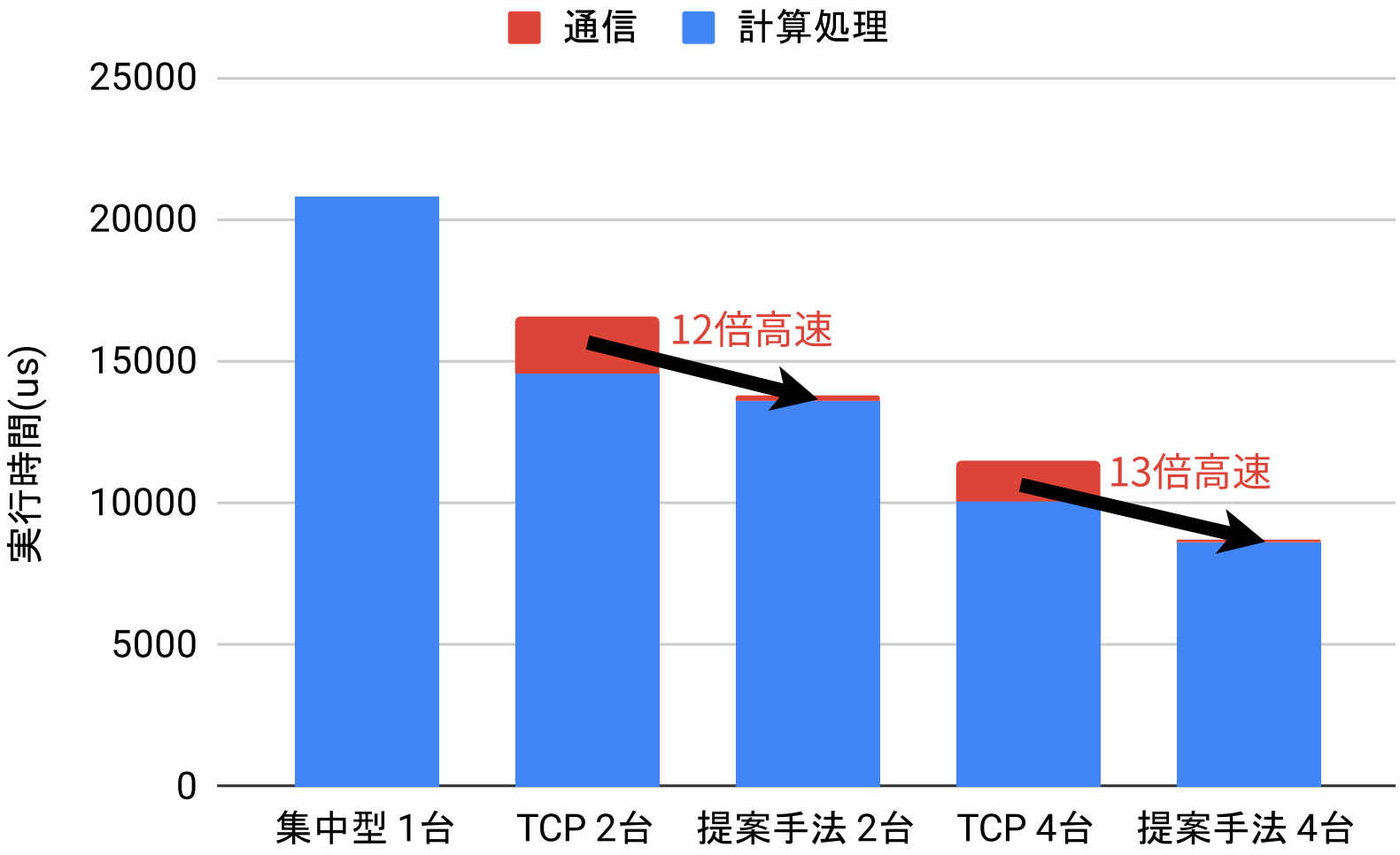

通信
計算処理
25000
20000
15000
10000
5000
0
実行時間(us)
1.5倍高速
1.2倍高速
集中型 1台
TCP 2台
提案手法 2台
TCP 4台
提案手法 4台

評価結果


通信
計算処理
25000
20000
15000
10000
5000
0
実行時間(us)
2.4倍高速
1.3倍高速
集中型 1台
TCP 2台
提案手法 2台
TCP 4台
提案手法 4台

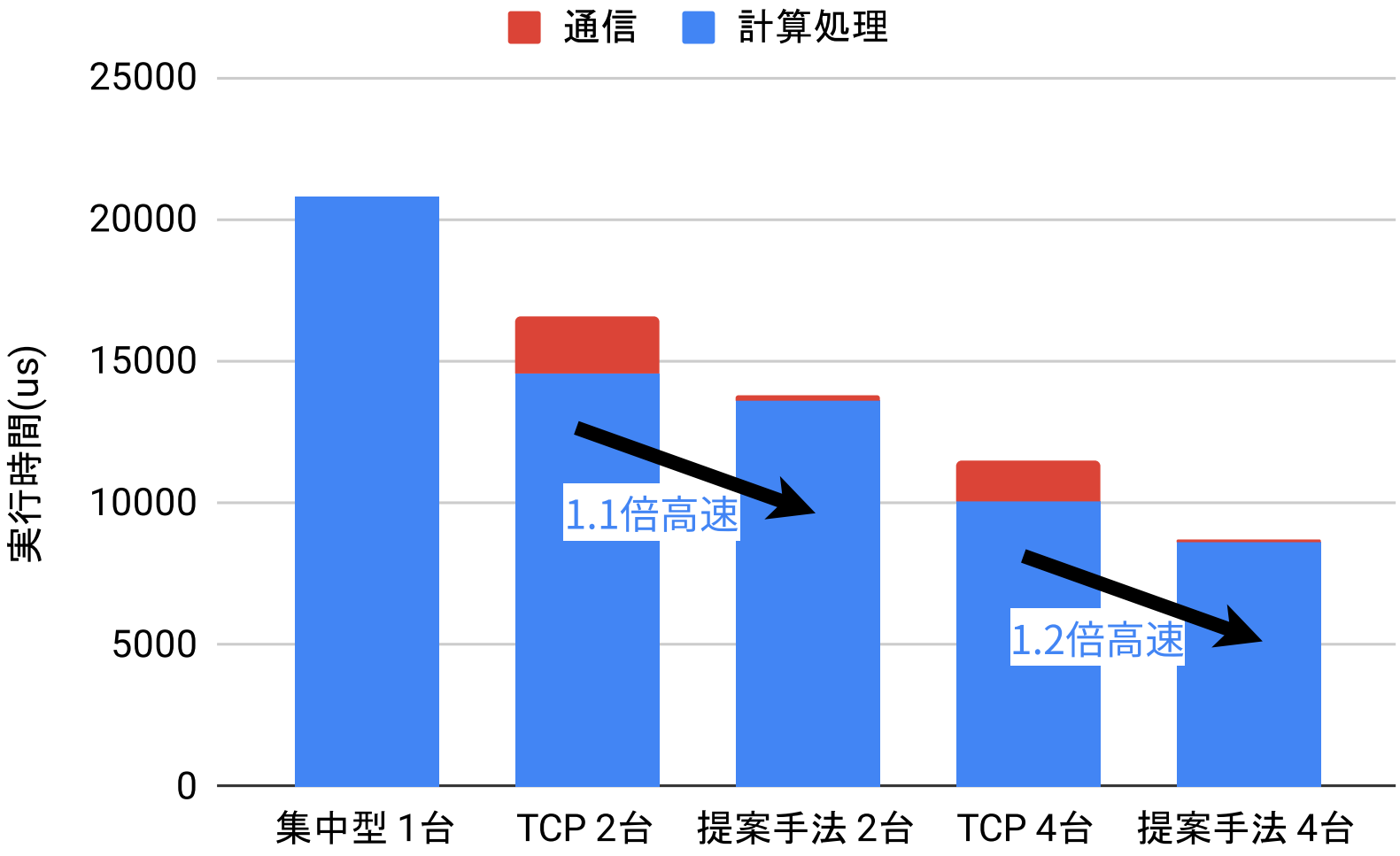
評価結果

- 計算機間の通信に DPDK による L2 通信を使用することは有効



評価結果

- DPDK の Run-to-Completion モデルで処理を行うことは有効



まとめと今後の課題

まとめ

- DPDK の Run-to-Completion モデルを用いた L2 分散計算環境
  - DPDK による L2 通信の使用
    - TCP/IP による制御のオーバーヘッドを低減
    - 高速なパケット I/O 処理を用いることができる
  - Run-to-Completion モデルの採用
    - CPU リソースを有効活用できる
    - 計算処理時に L1 キャッシュを有効活用できる

今後の課題

- カーネルによる L2 通信や DPDK の Pipeline モデルを用いた場合を実装して評価
- ロジスティック回帰やサポートベクターマシンといった複雑な処理も実行できる汎用的な分散計算環境へ拡張