Vol.2019-HPC-170 No.33
2019/7/26

マルチスレッドプログラムにおける通信ライブラリの性能比較

小田嶋 哲哉[1,a)]　李 珍泌[1]　佐藤 三久[1]

概要：メニーコアプロセッサが主流になりつつある現在，多数のコアを有効に活用するためにタスク並列が注目されている．タスク並列のプログラミングモデルは，複数のスレッドが計算だけではなく，スレッド同士で通信を行うことで同期のオーバヘッドを削減することが期待される．我々は，XcalableMP2.0 において，従来の SPMD モデルだけではなく，タスク並列によるマルチスレッドプログラミングの導入を提案している．この通信レイヤにどの通信ライブラリを採用するかは，現在検討段階であり，本稿では，各種通信ライブラリの性能評価を行うことで，XcalableMP2.0 の通信レイヤに利用するライブラリの検討を行う．これまでの研究 [1] より，MPI をスレッド間通信に用いると，スレッド数が増えることで通信性能が著しく低下することがわかっている．そこで，PGAS 向けの通信ライブラリはスレッド間で高い通信性能を達成することができるか検証を行う．本稿では，複数の MPI 実装と GASNet，GASPI および InfiniBand Verbs を用いて，マルチスレッド PingPong レイテンシとバンド幅の評価を行った．これらより，メッセージサイズが小さい場合において，PGAS 向け通信ライブラリは低レイテンシ通信を実現したが，本来の thread multiple の性能には達しないことがわかった．一方，InfiniBand Verbs による直接通信は，どの条件でも高い通信性能を達成し，他の PGAS 通信ライブラリにはオーバヘッドがあることがわかった．

1. はじめに

大規模分散メモリ環境における次世代の並列プログラミング言語として，理化学研究所計算科学研究センターが中心となり，PGAS (Partitioned Global Address Space) 並列言語 XcalableMP (以降「XMP」と略す) を開発している．XMP は，「#pragma xmp」からはじまる指示文を提供し，「グローバルビュー」と「ローカルビュー」という 2 種類のプログラミングモデルによりユーザを支援する．グローバルビューは，ノード間におけるデータの分散や並列処理を全ノード（プロセス）が実行する SPMD (Single Program Multiple Data) モデルである．分散されたデータはローカルへのメモリ参照に置き換えられるが，リモートメモリのデータを参照するには，XMP の指示文を用いて通信を明示的に行う必要がある．ローカルビューは，Co-array Fortran 記法によるノード間の片方向通信を容易に記述することができる．これにより，部分的なノードグループ間通信を行うことで，最小限の通信や同期による性能向上を期待することができる．

XMP はこれまでに，ノードを主体としたプログラミングモデルを提供していたが，現在主流となっているメニーコアプロセッサでは，それでは粒度が大きすぎることが問題となっている．そこで，メニーコアプロセッサにおける性能と生産性を両立させるために，タスク並列プログラミングをベースとした XcalableMP2.0 が PC クラスタコンソーシアムの並列言語 XMP 企画部会で議論されている．データの依存関係に応じて，タスク（スレッド）を動的に生成し，演算や通信をタスクの粒度で行うことで，プロセス間よりも同期のオーバヘッドを削減することができ，性能向上が期待される．しかしながら，これまでの研究 [1], [2] によって，通信ライブラリとしてデファクトスタンダードとなっている MPI (Message Passing Interface) において，スレッド間通信がプロセス間通信よりも性能が低下してしまう問題があることがわかっている．

前回の研究会では [1]，MPI の thread multiple の性能悪化や InfiniBand Verbs を用いることで高い通信性能を実現することを示した．本稿では，ノード間通信において InfiniBand Verbs だけでなく，PGAS 向けの通信ライブラリにおけるスレッド間通信性能を網羅的に比較することで，XMP2.0 の通信レイヤへの適用を検討する．

2. マルチスレッド通信

本節では，マルチスレッド通信の種類と，MPI 通信と PGAS 通信ライブラリを併用した場合における制限事項を

[1] 理化学研究所 計算科学研究センター
[a)] tetsuya.odajima@riken.jp

示す．

2.1 マルチスレッド通信の種類

広く利用されている通信ライブラリとして MPI がある．MPI には様々な実装が存在しているが，各実装やハードウエアの組み合わせでサポートするマルチスレッド通信のレベルが変わる．MPI では，以下の 4 つのレベルが定義されている．

MPI_THREAD_SINGLE
: 1 スレッドのみが通信を行うことができる．つまり，マルチスレッド通信には対応していないということである．

MPI_THREAD_FUNNELD
: プロセスはマルチスレッドに対応しているが，MPI の呼び出しはメインスレッドのみに許されている．つまり，実際の MPI の呼び出しは逐次化されてしまう．

MPI_THREAD_SERIALIZED
: プロセスはマルチスレッドに対応しているが，MPI の呼び出しは 1 つのスレッドにしか許されない．つまり，メインスレッド以外も MPI を呼び出すことができるが，あるスレッドの通信が終わったあとでないと別のスレッドが MPI を呼び出すことができない．これも FUNNELD と同様に MPI の呼び出しは逐次化されてしまう．

MPI_THREAD_MULTIPLE
: MPI の呼び出しに関する制限はない．つまり，多数のスレッドが同時に MPI を呼び出しても動作するということを示す．

我々が XMP2.0 で実現しようとしている，タスク並列によるマルチスレッドプログラミングでは，MPI_THREAD_MULTIPLE のようにスレッドから MPI の呼び出しが制限されないことが必須である．

2.2 マルチスレッド通信による問題点

MPI 以外にも，PGAS 向けの通信ライブラリや低レベル通信 API を直接利用することで，より高速な通信を実現する手段がある．一方，MPI はこれまで広く利用されてきたことによる使いやすさや，集団通信やタグマッチングなどの高い機能を有している．そのため，XMP2.0 では，MPI と PGAS 通信ライブラリや低レベル通信 API を適材適所で利用することを考えている．しかし，これらを混在して利用する場合，MPI と別の通信ライブラリを同時使用してはいけないという制限がある．つまり，MPI を使用する区間，別の通信ライブラリを使用する区間を明確に分ける必要がある．この理由として，エンドポイントとなる NIC のハードウェア資源の取り合いが挙げられる．そのため，ユーザは注意してプログラミングを行うか，直接低レベル通信 API を用いて MPI とは異なるエンドポイントを使用する必要がある．

さらに，MPI のマルチスレッド通信は，プロセス通信に対して性能が低いことが示されている [1], [2]．そのため，あえて通信用のスレッドを作成し通信を移譲することでこの問題を回避したり，ユーザが直接低レベル通信 API を使用してプログラミングを行ったりする必要がある．

3. マルチスレッド通信ライブラリ

本節では，マルチスレッド通信の性能評価の対象とするライブラリの詳細を示す．

3.1 MPI

MPI には，商用・オープンソースを含め様々な実装が存在するが，本稿ではその中からいくつかを紹介する．

3.1.1 MPICH

MPICH はオープンソースソフトウェアとして開発が進められている MPI の実装である．MPI-3 までの機能に準拠しており，拡張性が高いという特徴がある．そのため，多くの MPI 実装のリファレンスモデルとして利用されている．

3.1.2 MVAPICH2

MVAPICH2 [3] はオハイオ州立大学で開発が進められている MPICH ベース実装の一つである．これまでに，InfiniBand [4] 向けの通信最適化を行っており，低レイテンシ通信を実現している．MVAPICH2 は，InfiniBand だけではなく，Intel Omni-Path にも対応しており，NVIDA GPU 間直接通信 MVAPICH2-GDR や PGAS 向けの拡張実装 MVAPICH2-X などがある．本稿では，MVAPICH2 実装の評価を行う．

3.1.3 Intel MPI

Intel MPI は, Intel で開発が進められている MPICH ベースの商用ライブラリである．Intel Compiler 同様，Intel プロセッサ環境向けの最適化が施されており，高い性能を期待することができる．近年，Intel も独自のインターコネクションである Omni-Path を提供しており，Intel MPI により高い通信性能を達成することができる．本環境では Intel MPI の準備ができていないため，本稿での評価は省略する．

3.1.4 OpenMPI

OpenMPI [5] は，オープンソースソフトウェアとして開発が進められている MPI 実装の一つである．MPI-3.1 までの機能に準拠しており，機能別に実装が分かれている．また，MPICH と同様に，様々な MPI 実装のベースとなっている．京コンピュータなどで使用されている富士通 MPI も，OpenMPI をベースに開発を行っている．

3.2 GASNet

GASNet [6] は Lawrence Berkeley National Laboratory

で開発が進められている低レベル通信ライブラリである．PGAS を対象としたライブラリで，Remote Memory Access (RMA) ベースの高速な通信性能と Active Messages による低レイテンシ通信を実現している．2019 年 6 月に，これまでベースとしていた GASNet-1 から，GASNet-EX [7] へベースを移すことになった．GASNet-EX では，GASNet-1 をエクサスケールシステム向けに拡張したものである．さらに，メニーコアプロセッサのようなマルチスレッド環境において，性能向上を図っている．GASNet も様々な PGAS 言語の通信レイヤとして使用されている．GASNet は MPI との同時使用ができないため，通信の区間を切り分ける必要がある．GASNet では，1 つの大きな配列を全プロセスで共有し，通信に必要なデータサイズと開始アドレスを用いて Put/Get といった片方向通信でやり取りを行う．

3.3 GASPI

GASPI (Global Address Space Programming Interface) [8] は，GASPI Forum によって仕様が策定されている PGAS 向けの通信ライブラリである．GASPI も，GASNet のように Remote DMA (RDMA) をベースに通信を行う．プロセス間だけではなく，スレッド間でも非同期通信が可能である．しかし，MPI との同時使用ができないため，通信区間を分割する必要がある．

3.4 OpenSHMEM

OpenSHMEM [9] は，PGAS プログラミングのための標準化された API を提供する通信ライブラリの一つである．その他の PGAS 向け言語のように，RDMA 通信をベースに直接ハードウェアを利用することができるため低レイテンシ通信が実現可能である．

3.5 UCX

UCX (Unified Communication X) [10] は，様々なハードウェアに対応する抽象化レイヤを提供する通信ライブラリである．異なる通信 API，プロトコル，実装を 1 つのフレームワークで効率的に実行できることを目標に開発が進められている．また，コミュニティには様々なハードウェアベンダーが参加している．UCX は 3 つのコンポーネントから構成される．UCP は高レベル API で使用される機能を提供している．UCT は通信レイヤに近い低レベル API を提供しており，効率的かつ低オーバヘッドにアクセスできる．ここでは，CPU のメインメモリだけではなく，GPU 上のメモリに対する操作も可能となっている．UCS はサービス層に関する API を提供しており，メモリ管理やデータ構造などを司る．近年，UCX はいくつかの通信ライブラリの通信レイヤとして利用されている．上記に示した OpenMPI は，バージョン 4.0 からその低レベル通信 API として UCX を採用した．同様に，OpenSHMEM もバージョン 1.3 までは GASNet を採用していたが，バージョン 1.4 からは UCX を採用している．

表 1 評価環境

CPU	Intel Xeon E5-2670 2.6GHz 8 cores × 2 sockets
Main memory	DDR3 1600MHz 32GB
Interconnection	InfiniBand: Mellanox ConnectX-3 Single-port FDR
OFED	MLNX_OFED_LINUX-4.4-2.0.7.0
OS	CentOS release 7.5

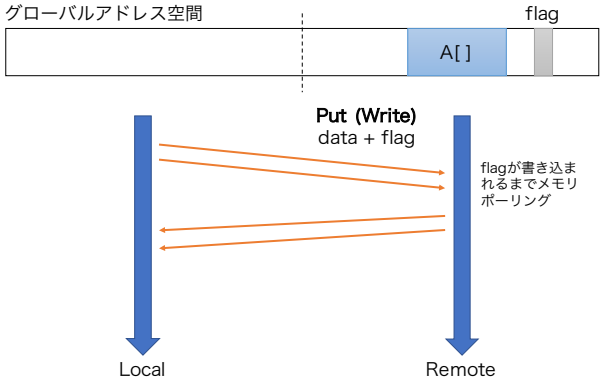

図 1 片方向通信による PingPong 通信の例

3.6 低レベル通信 API

3.6.1 InfiniBand Verbs

InfiniBand Verbs (IB Verbs) は，InfiniBand 向けの低レベル通信 API である．そのため，通信ライブラリというカテゴリではないが，ハードウェアに非常に近いレイヤでプログラミングを行える．しかしながら，IB Verbs は仕様が曖昧かつ，プログラミングコストが非常に高いことが問題となっている．例えば RDMA 通信を行う際に，ローカルの通信キューの管理だけではなく，リモートのキューを絶えずエンキューしておかなければ，リモート側から通信完了通知がなくなり最悪デッドロックが発生してしまう．プログラムが複雑という点を除けば，ハードウェアをほぼ直接利用できるため軽量な通信が可能になり，高い通信性能を発揮することが期待される．InfiniBand 向けの通信として，GASNet は引き続き openib コンポーネントを使用しているが，OpenSHMEM や OpenMPI などは UCX を使用する方向に転換している．

3.7 uTofu

uTofu は，Tofu インターコネクトシリーズ向けの低レベル通信 API である．Tofu はスーパーコンピュータ「京」を始め，FX100 やスーパーコンピュータ「富岳」(a.k.a. ポスト「京」) でも使用されるインターコネクトである．IB Verbs と同様のレイヤでプログラミングを行うことができ，

複数の DMA エンジンである Tofu Network Interface を同時に叩くことができ，高い通信バンド幅を実現することができる．京コンピュータや FX100 では，富士通製の MPI や RDMA ライブラリを使用することができるが，これらの問題として「MPI_THREAD_MULTIPLE」に対応していないということが挙げられる．そのため，通信は逐次化されてしまい，タスク並列を有効に利用することができない．一方，uTofu を使用して，ユーザが明示的にスレッドセーフにプログラミングを行うことで，マルチスレッド対応の通信が可能になる．本稿では，時間の関係上 Tofu インターコネクトに関する評価は省略するが，今後の課題としたい．

4. 性能測定

本節では，各種通信ライブラリの通信性能をベンチマークにより比較する．そして，XMP2.0 で使用する通信ライブラリの検討材料とする．以下に，評価に使用した通信ライブラリとバージョンを示す．

- MVAPICH2-2.3.1
- MPICH-3.3
- OpenMPI-4.0.1
- GASNet-EX 2019.3.2
- GASPI-2 1.3.0
- InfiniBand Verbs

評価に用いた環境は **表 1** に示すサーバを 2 台使用する．InfiniBand をスイッチ経由で接続し，そのサーバ間の通信性能を評価する．スレッド数は $\{1, 2, 4, 8\}$ という組み合わせを用いる．

評価には PingPong ベンチマークを用いる．MPI は P2P 通信である Send/Receive 通信と片方向通信である Put の性能を評価する．MPI における片方向通信では，MPI-3 の RMA を使用する．その他の通信ライブラリについては，片方向通信である Put (Write) の性能を評価する．PGAS をターゲットとした評価であるため，片方向通信の場合 **図 1** のように，ローカルスレッドはまずデータを相手の PGAS 領域に書き込み，最後にフラグを書き込む．リモートスレッドはフラグの書き込みをメモリポーリングして待ち，反転を検出したら同じようにデータとフラグを相手の PGAS 領域に書き込む．

4.1 性能評価できなかった通信ライブラリ

本稿では，紹介した通信ライブラリや低レベル通信 API をすべて評価対象としたかったが，動作が不安定であったものや環境構築が未完了であるため評価を行うことができなかったものがいくつかある．

uTofu は，Tofu インターコネクト向けの低レベル通信 API であるが，ドキュメントを手に入れたのが直前であったため実装が間に合わなかった．しかし，API としては IB Verbs と同様のレイヤであるため，比較的容易に実装が可能であると考えている．OpenMPI は，ビルド自体は完了しているが，PingPong を実行するとデータサイズによっては異常に低いレイテンシになってしまうこが確認されている．おそらく，フラグによるスレッド間の同期が想定通り動作していないことが考えられるが，現在調査中である．OpenSHMEM および UCX については，OpenSHMEM の通信レイヤとして UCX を使用する予定であった．そのテストとして，OpenMPI で UCX を試したところ，通信が発生するタイミングでプログラムが abort してしまう問題が発生した．これがビルドの方法に失敗したのか，使用方法が正しくないのかといった切り分けができなかったため，本稿では評価をすることができなかった．

4.2 レイテンシ性能

PingPong ベンチマークを用いて，各通信ライブラリの片道のレイテンシを評価する．**図 2** から**図 5** にそれぞれスレッド数 1 から 8 までのレイテンシを示す．図の縦軸はレイテンシ $[\mu sec]$，横軸はスレッドごとの通信データサイズ $[Byte]$ である．これらのグラフに OpenMPI のデータを掲載していない．本環境で OpenMPI を使用すると性能が大きく振れたり，異常な値が得られたりと安定して動作していないため，本稿では除外することとした．

図 2 より，1 スレッドの通信性能について評価を行う．1 スレッドの通信は，言い換えればプロセス間通信と同等である．これより，MVAPICH2 (Put) および MPICH (Put) は他の通信ライブラリよりレイテンシが大きい事がわかる．続いて，GASNet と GASPI が約 $2\mu sec$，MVAPICH2 (P2P) ，MPICH (P2P) および IB Verbs が約 $1.6\mu sec$ のレイテンシである．MPI の P2P 通信の場合，データと共に通信の同期を行うことができるため， 図 1 のように 2 回データを転送する必要がなくなる．これによって，メッセージサイズが小さい場合において非常に低いレイテンシで通信を行うことができる．一方，IB Verbs は 図 1 のように 2 回通信を行っているが MPI P2P 通信と同等のレイテンシであり，低レベル通信 API を用いることで軽量な通信が可能であることが示された．GASNet や GASPI などの PGAS 向けの通信ライブラリは，それらの通信レイヤに低レベル通信 API を用いているため，MPI 片方向通信と比較して低いレイテンシを達成している．MPI の片方向通信による実装では，他の実装と比較して約 3 倍のレイテンシとなっている．MPI は他の通信ライブラリと比較して機能が充実している一方，実装のレイヤが深くなってしまい，通信のオーバヘッドが大きくなる傾向にある．そのため，メッセージサイズが小さい場合にはそのオーバヘッドが通信時間の大半を占めてしまい，性能低下に至ったと考えられる．

図 3 より，2 スレッドの通信性能について評価を行う．

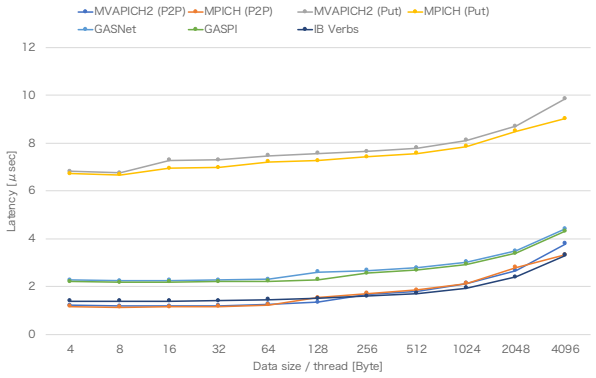

図 2 PingPong レイテンシ（〜4KB）：1 スレッド

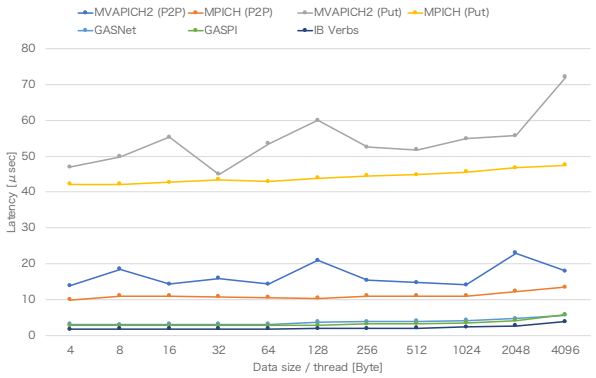

図 3 PingPong レイテンシ（〜4KB）：2 スレッド

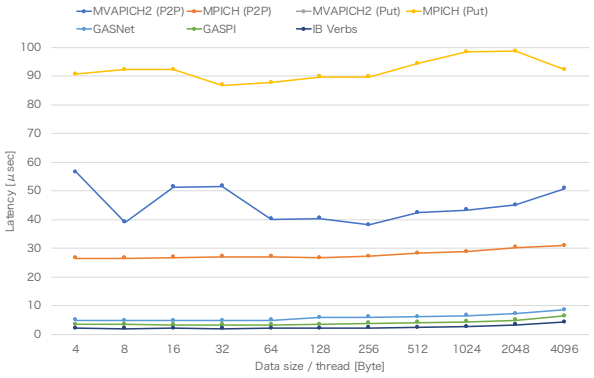

図 4 PingPong レイテンシ（〜4KB）：4 スレッド

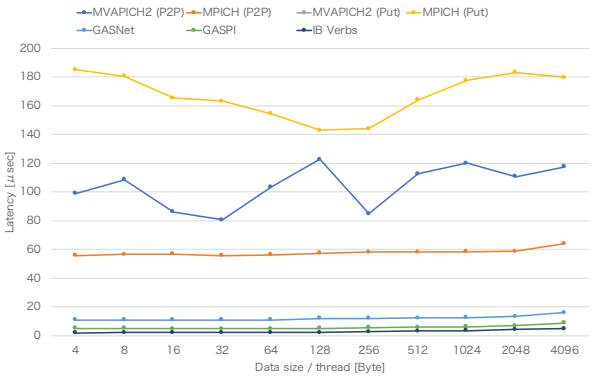

図 5 PingPong レイテンシ（〜4KB）：8 スレッド

性能の傾向としては図 2 と類似しているが，MPI P2P のレイテンシが 1 スレッドと比較して 5 倍以上に増加している．これは従来の研究 [1], [2] でも示しているように，マルチスレッド時の MPI はエンドポイント資源の排他制御を行うため，性能が著しく低下する．そのため，タスク並列においてスレッド間で自由に通信させる用途には，現在の MPI は向かないということがわかる．一方，IB Verbs をはじめ GASNet や GASPI は，1 スレッドと比較して性能低下が小さいことがわかる．

図 4 および図 5 より，4 スレッドおよび 8 スレッドの通信性能について評価を行う．MVAPICH2 (Put) はレイテンシが大きすぎたため，これらのグラフには収まりきっていない．性能の傾向は図 3 と同じだが，スレッド数が大きくなることで徐々に GASNet と GASPI の性能差が開いていることがわかる．しかしながら，やはり IB Verbs はスレッド数によるレイテンシの悪化がほとんど見られない．そのため，特に通信レイテンシがアプリケーションの性能に大きく影響する場合，IB Verbs などの低レベル通信 API を使用することが重要であることがわかった．

4.3 バンド幅性能

図 6 から**図 9** にそれぞれスレッド数 1 から 8 までのバンド幅を示す．図の縦軸はバンド幅 $[MB/s]$，横軸はスレッドごとの通信データサイズ $[Byte]$ である．マルチスレッドで PingPong を行う場合，イテレーションごとにスレッド間の同期を取ることはないため，バンド幅は双方向通信の値となる．

図 6 より，1 スレッドの通信性能について評価を行う．これもレイテンシの評価と同様に，1 スレッドはプロセス間通信と同等である．これより，MPICH を除きデータサイズが大きくなるとバンド幅はおおよそ同じ値に収束する．MPICH のバンド幅が 6GB/s に達していない原因は現在追求中であるが，環境変数による最適化不足や OFED の組み合わせによるものと想定している．その他の通信ライブラリでは，データサイズが 4KB から 512KB まで，バンド幅の立ち上がりに違いが出ている．MVAPICH2 では，片方向通信よりも P2P のほうが同期のオーバヘッド小さいため，バンド幅は高くなっているが，データサイズが 1MB を超えるとオーバヘッドが相対的に小さくなり性能差はなくなる．IB Verbs はデータサイズが 512KB まで高いバンド幅を実現しているが，それ以上では MPI や PGAS 通信ライブラリに劣っている．本評価に用いた IB Verbs 実装は，単純に片方向通信を発生させているだけであるが，各通信ライブラリではペーシングなどの最適化を行っている可能性がある．IB Verbs で常に高い性能を実現させるためには，単純な実装だけではなく最適化を取り入れる必要がある．

図 7 より，2 スレッドの通信性能について評価を行う．

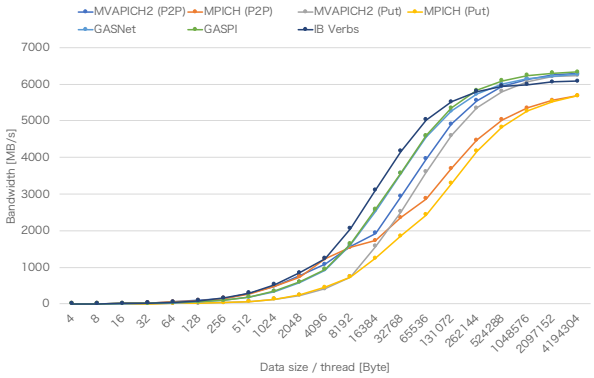

図 6 PingPong バンド幅：1 スレッド

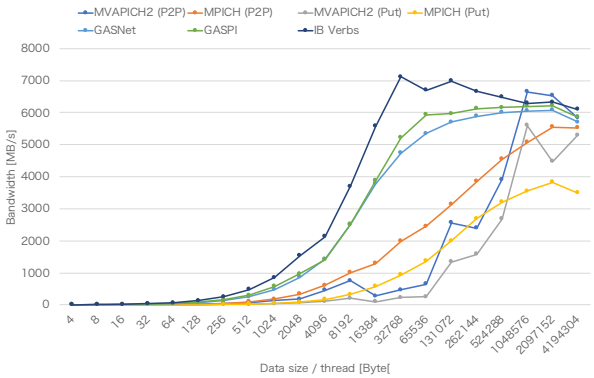

図 7 PingPong バンド幅：2 スレッド

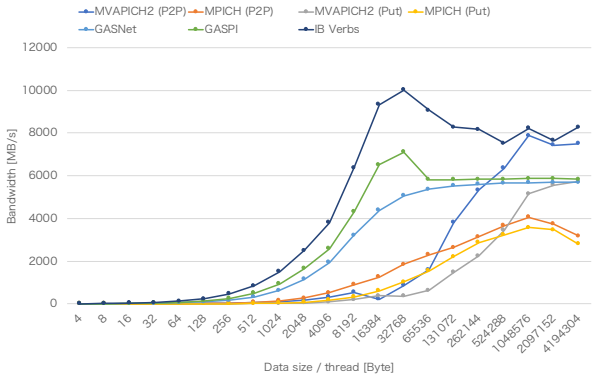

図 8 PingPong バンド幅：4 スレッド

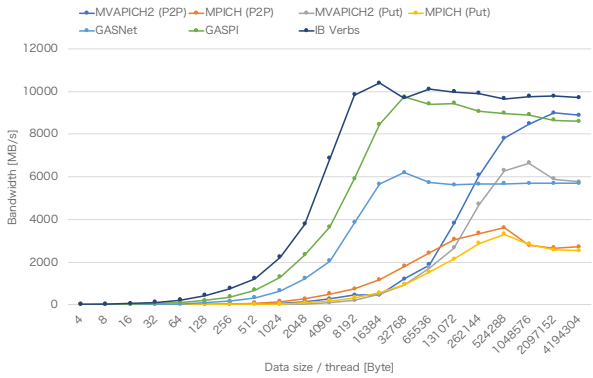

図 9 PingPong バンド幅：8 スレッド

MPI の通信性能は P2P，片方向通信によらずレイテンシ評価の場合と同様に，マルチスレッド時は性能が 1 スレッド時と比較して低下している．MVAPICH2 の P2P だけはデータサイズが 1MB 以上のときに 6GB/s 以上のバンド幅を達成しているが，これは MVAPICH2 内部の通信最適化の影響であると考えている．IB Verbs および PGAS 通信ライブラリは，スレッド数が増えたことによってバンド幅の立ち上がりが向上している．特に IB Verbs は 32KB で 7.1GB/s と最も高い性能を達成している．

図 8 より，4 スレッドの場合，性能の傾向は図 7 と同様であるが，GASNet のバンド幅が約 6GB/s で飽和していることがわかる．各スレッドが自由に通信を行うことができる場合，スレッド数が増加することでバンド幅は双方向バンド幅に近づく．しかし，図 8 の場合，InfiniBand FDR x4 の片方向の実行バンド幅に律速している．そのため，GASNet は真に thread multiple ではない可能性がある．GASPI もデータサイズ 64KB 以上で GASNet ど同様にバンド幅が律速している．一方，IB Verbs は依然高いバンド幅性能を実現している．

図 9 より，8 スレッドの場合，IB Verbs はデータサイズ 8KB 以上でほぼ 10GB/s に達している．一方，4 スレッドのときには 6GB/s でバンド幅性能が律速していた GASPI が 8.5GB/s 以上を達成している．1〜4 スレッドまでは性能が律速していたにもかかわらず，8 スレッドでは急激に性能が向上した原因については現在調査中である．

4.4 議論

本評価では，PingPong ベンチマークを用いて各通信ライブラリのレイテンシおよびバンド幅を評価してきた．これらより，これまでの研究でも指摘されていたように現状の MPI は thread multiple で動作はするものの，性能最適化が十分ではないことが改めてわかった．また，GASNet はメッセージサイズが小さい場合では低いレイテンシ性能を実現していたが，thread multiple で通信を行っていない可能性があることがわかった．そのため，タスク並列などのマルチスレッド通信では十分に高い性能を発揮することができない．GASPI はメッセージサイズが小さいときに，低いレイテンシで通信が可能であったが，4 スレッドまではバンド幅がスレッド数を増やしても 6GB/s に律速してしまっている．8 スレッドではバンド幅性能も向上しているが，性能としては安定していない．一方，IB Verbs はどのような条件でも非常に高い性能を発揮していた．そのため，XMP2.0 におけるタスク並列では IB Verbs を通信レイヤにすることで，高い性能を達成できる可能性が高い．プログラミングのコストが高いことが懸念されるが，PGAS 通信における Put，Get，同期を行うための wrapper 関数などを作成することで，プログラミング言語の通信レイヤへの適用が比較的容易になると考えている．この実装につ

いては，新たに検討をする必要がある．

5. まとめ

本稿では，タスク並列を想定したマルチスレッド通信性能について，複数の通信ライブラリを用いて評価を行った．これより，MPI は高機能ではあるが，マルチスレッド通信ではプロセス間通信と比較して性能が極端に低下してしまうことがわかった．一方，PGAS 向けの通信ライブラリは，小メッセージでは低いレイテンシを達成したが，スレッド数を増やしても 1 スレッドと同等のバンド幅に律速する場合や，スレッド数を多くしなければ十分にバンド幅性能が得られない場合があることがわかった．これまでの研究 [1] でも示していたように，IB Verbs はどの条件においても高い通信性能を達成している．

今後の課題として，本稿では掲載できなかった Open-MPI，OpenSHMEM，UCX といった通信ライブラリの通信性能を評価する．その結果より，XMP2.0 の通信レイヤに対してどの通信ライブラリを使用するかを検討し，実際に実装を行う予定である．

参考文献

[1] 小田嶋哲哉，森江善之，李　珍泌，佐藤三久：マルチタスク PGAS モデルをサポートするノード間軽量通信レイヤの検討，情報処理学会研究報告 (ハイパフォーマンスコンピューティング)， Vol. 2019-HPC-168, No. 1 (2019).
[2] Tsugane, K., Lee, J., Murai, H. and Sato, M.: Multi-tasking Execution in PGAS Language XcalableMP and Communication Optimization on Many-core Clusters, *Proceedings of the International Conference on High Performance Computing in Asia-Pacific Region*, HPC Asia 2018, pp. 75–85 (2018).
[3] : MVAPICH: MPI over InfiniBand, Omni-Path, Ethernet/iWARP, and RoCE. `http://mvapich.cse.ohio-state.edu/`.
[4] Mellanox: RDMA Aware Networks Programming User Manual Rev 1.7. `http://www.mellanox.com/related-docs/prod_software/RDMA_Aware_Programming_user_manual.pdf`.
[5] : Open MPI: Open Source High Performance Computing. `https://www.open-mpi.org/`.
[6] : GASNet Communication System. `http://gasnet.lbl.gov/`.
[7] Bonachea, D. and Hargrove, P. H.: GASNet-EX: A High-Performance, Portable Communication Library for Exascale, *Languages and Compilers for Parallel Computing (LCPC'18)* (2018).
[8] : GASPI-Forum – Forum of the PGAS-API GASPI. `http://www.gaspi.de/`.
[9] : OpenSHMEM.org. `http://www.openshmem.org/site/`.
[10] : UCX-Unified Communication X. `http://www.openucx.org/`.