PRIMEHPC FX10 の MPI 性能改善状況

富士通株式会社
TC)計算科学) 坂口吉生

1. はじめに

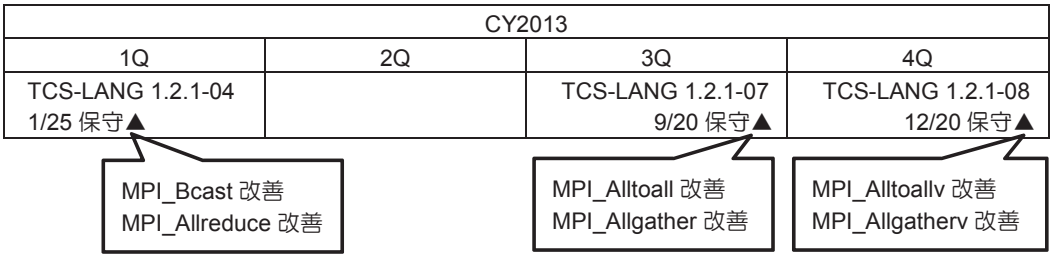
本稿では、東京大学情報基盤センターで 2012 年 4 月から稼働している富士通 PRIMEHPC FX10 (以降、FX10) 向け MPI(Message Passing Interface)ライブラリの性能改善状況について記載します。

FX10 システムの並列計算におけるノード間通信は、富士通独自開発の 6 次元メッシュ/トーラスインターコネクト(Tofu インターコネクト*1)を採用しており、1 次元から 3 次元のトーラス形状のネットワーク構成でアプリケーションプログラムの性能が最大限に引き出せるようになっています。

しかし、1 ノード内複数プロセスの MPI 性能においては性能改善要望があり、2013 年に、チューニングを施した性能改善版 MPI のシステム適用を実施しましたので、改善状況を報告します。

2. 性能改善状況

東京大学情報基盤センター FX10 (Oakleaf-FX) の定期保守時に性能改善を行った MPI 関数を以下に記載します。



MPI は言語処理系ソフトウェア (Technical Computing Suite/Technical Computing Language、以降 TCS-LANG) に含まれており、版数管理が行われています。1.2.1-04 版では、Bcast、Allreduce の性能を改善し、1.2.1-07 版では、Alltoall、Allgather の性能を改善し、最新版の 1.2.1-08 版では、Alltoallv、Allgatherv の性能改善を実施しています。

Oakleaf-FX の言語処理系ソフトウェアの版数は、常に最新版が使用されるように環境変数が設定されています。詳細は、利用支援ポータル(https://oakleaf-www.cc.u-tokyo.ac.jp)に格納されている「Oakleaf-FX 利用手引書」の「1.11.3 実行パス/ライブラリ設定」を参照して下さい。

3. 性能改善内容

Oakleaf-FX 稼働時から、いくつかの Tofu インターコネクト向けにチューニングされたアルゴリズム(以降、従来の Tofu 専用アルゴリズム)が存在していました。従来の Tofu 専用アルゴリズムは、ノード内に 1 プロセスが生成された場合の性能をターゲットとしていましたが、ノード内に複数のプロセスが生成された場合に改善される新アルゴリズムを作成し、性能を改善しています。

*1 Tofu (Torus fusion) は、富士通の高速インターコネクトの呼称です。

新アルゴリズムは、Bcast, Allreduce, Alltoall, Allgather, Alltoallv, Allgatherv の 6 関数に対して適用しています。通信はノード間を RDMA 通信で実装し、ノード内は Tofu ループバック通信または独自の共有メモリ通信を使用しています。

それぞれの関数に適用された新アルゴリズムの特徴を、各項で説明します。

3.1 MPI_Bcast

ノード内に複数のプロセスが生成された場合かつコミュニケータが 3 次元直方体の場合に、従来の Tofu 専用アルゴリズムを発展させた、新規アルゴリズムを開発しました。

ノード内通信は、MPI の共有メモリ通信を使わずに、集団通信独自の共有メモリ通信を実装し、オーバーヘッドを削減しています。

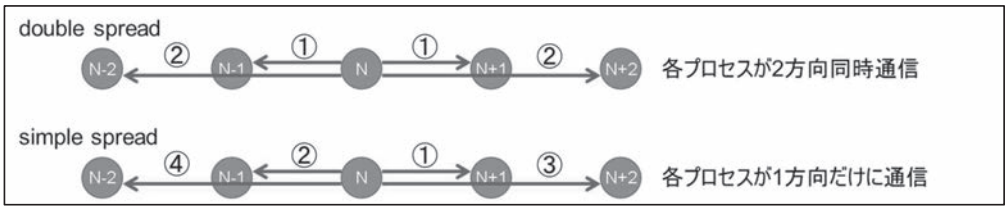
3.2 MPI_Allreduce

ノード内に複数プロセスが生成された場合の MPI_Allreduce は、Reduce + Bcast 部分で性能が出る部分に対し、新アルゴリズムの Bcast を適用し、性能を改善しました。Bcast と同様に、コミュニケータが 3 次元直方体を前提としています。

3.3 MPI_Alltoall

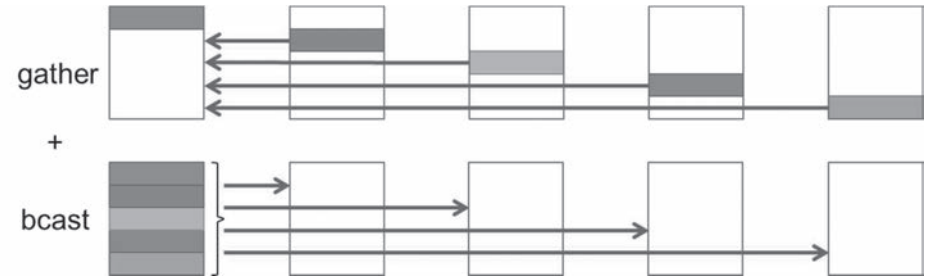
ノード内に複数のプロセスが生成された場合でも性能が改善されるように従来の Tofu 専用アルゴリズムに実装されているトポロジアンアウェアのアルゴリズムを改良しました。各プロセスは、一般的なランク配置では経路が重なりやすい同一ランク方向を避ける送信(ランクベースで近い場所から順番に遠い場所へ転送)し、全ての通信は RDMA 通信で転送し、性能を改善しました。

ノード内 1〜4 プロセスは、double spread アルゴリズムが選択され、ノード内 5〜16 プロセスは、simple spread アルゴリズムが選択されるよう最適化しています。



3.4 MPI_Allgather

ノード内に複数プロセスが生成された場合かつコミュニケータが 3 次元直方体の場合に、MPI_Allgather を Gather + Bcast で実装し、Gather と Bcast のそれぞれに性能改善を行いました。Gather は、自プロセスのデータを root ランク(rank 0)に RDMA 通信で転送するトポロジアンアウェアなアルゴリズムを実装しました。Bcast は、新アルゴリズムを使用しています。



3.5 MPI_Alltoallv

ノード内に複数プロセスが生成された場合の MPI_Alltoallv は、Alltoall の改善と同様の施策を行い、かつ可変長に対応し性能を改善しました。

3.6 MPI_Allgatherv

ノード内に複数プロセスが生成された場合の MPI_Allgatherv は、Gatherv + bcast で実装し、Allgather と同様の施策を行い、かつ可変長に対応し性能を改善しました。

4. 性能改善結果

性能改善効果は、MPI の基本的なベンチマークを行うプログラムである IMB (Intel MPI Benchmarks) を使用して、性能を計測しました。なお IMB による性能測定では、Oakleaf-FX の最大ノード数を利用するような「大規模 HPC チャレンジ」で最大ノード数を指定して実行すると、通信キューが溢れるため修正が必要となりますのでご注意ください。

集団通信 MPI_Alltoall と MPI_Allgather の比較は、性能改善を実施した TCS-LANG 1.2.1-07 とその前の版数である TCS-LANG 1.2.1-06 の比較を行っています。比較のグラフは、「図 4-1 384 ノード(6144 プロセス並列)の Alltoall 性能比較」と「図 4-2 384 ノード(6144 プロセス並列)の Allgather 性能比較」であり、横軸が送信するメッセージサイズ(Byte)、縦軸が実行時間(msec)です。

MPI_Alltoall は、中～大メッセージサイズにおいて、新アルゴリズムが選択されることにより性能が改善されています。

MPI_Allgather は、中メッセージサイズにおいて、新アルゴリズムが選択されることにより性能が改善されています。大メッセージサイズ(32KiB 以上)は、従来のアルゴリズムのため性能は同等です。

アルゴリズムとは、集団通信を行うための通信パターンのことです。それぞれの集団通信関数には、複数のアルゴリズムが実装されています。それぞれのアルゴリズムには特徴があり、小メッセージサイズで高速に動作するアルゴリズムがある一方、大メッセージサイズで高速に動作するアルゴリズムも存在します。これらを組み合わせてなるべくなめらかな曲線にしようとしています。しかし、メッセージ長、プロセス数、形状などの組み合わせがあり、組み合わせが無限であるため、必ずしも最適でないものもあります。

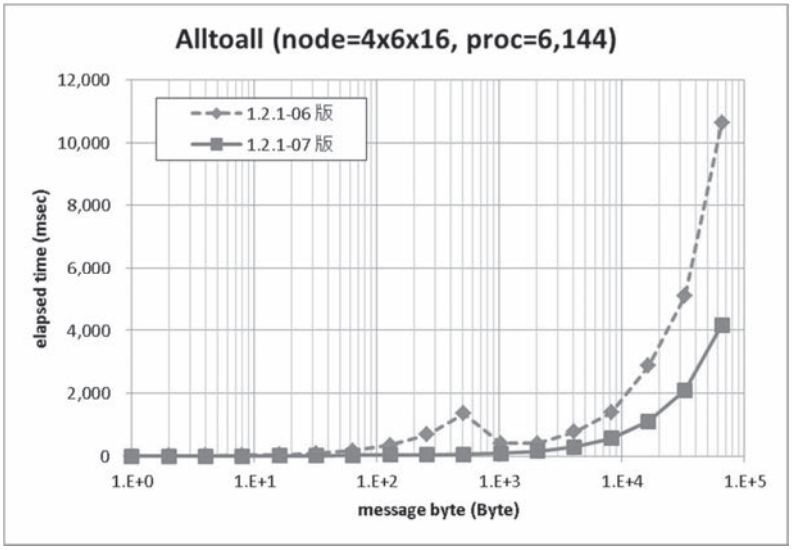


図 4-1 384 ノード(6144 プロセス並列)の Alltoall 性能比較

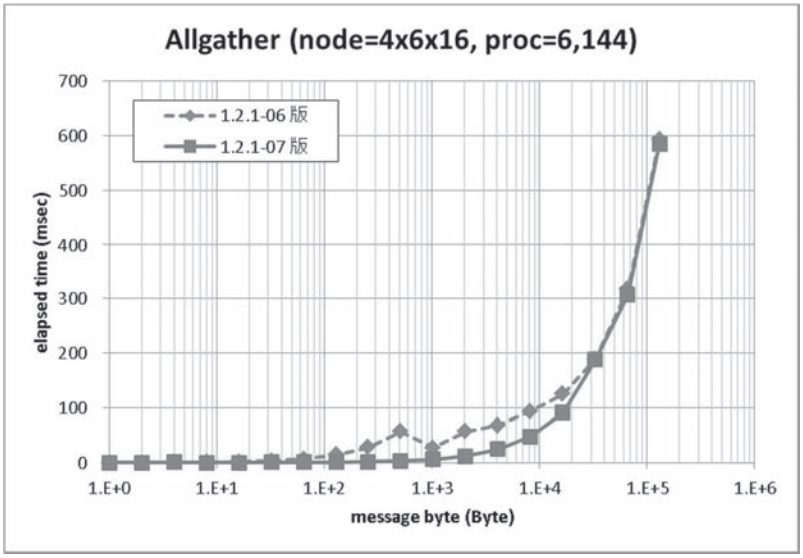


図 4-2 384 ノード(6144 プロセス並列)の Allgather 性能比較

集団通信 MPI_Alltoallv と MPI_Allgatherv の比較は、性能改善を実施した TCS-LANG 1.2.1-08 とその前の版数である TCS-LANG 1.2.1-07 の比較を行っています。比較のグラフは、「図 4-3 384 ノード(6144 プロセス並列)の Alltoallv 性能比較」と「図 4-4 384 ノード(6144 プロセス並列)の Allgatherv 性能比較」であり、横軸が送信するメッセージサイズ(Byte)、縦軸が実行時間(msec)です。

MPI_ Alltoallv は、中～大メッセージサイズにおいて、新アルゴリズムが選択されることにより性能が改善されています。

MPI_Allgatherv は、バラツキのある性能が改善版により、安定して性能が向上しています。

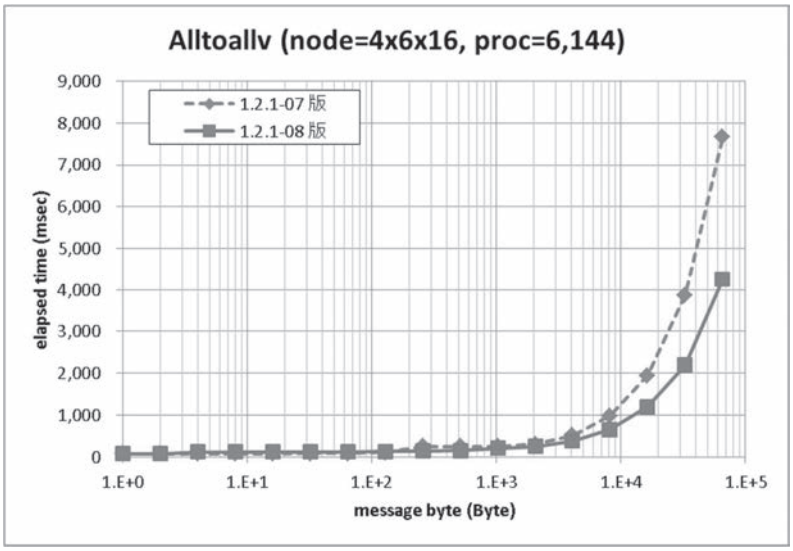


図 4-3 384 ノード(6144 プロセス並列)の Alltoallv 性能比較

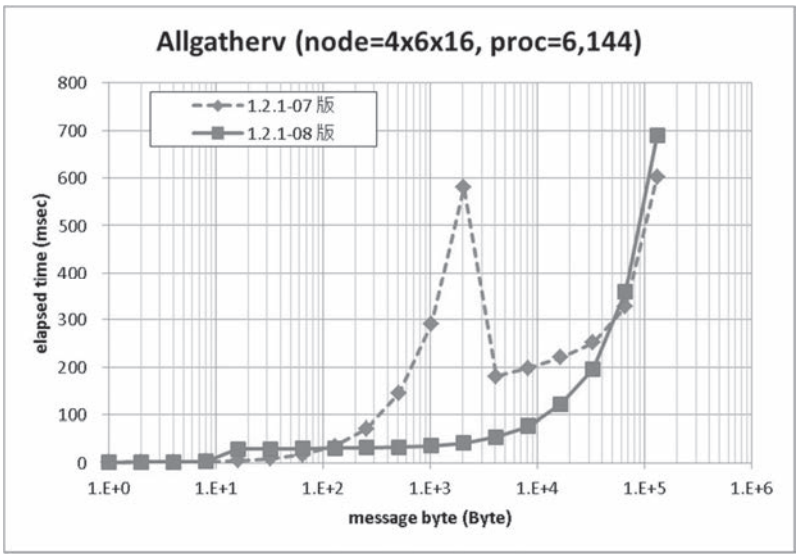


図 4-4 384 ノード(6144 プロセス並列)の Allgatherv 性能比較

5. おわりに

本稿では、富士通 PRIMEHPC FX10 向け MPI ライブラリの性能改善状況について記載しました。最新の性能改善版 MPI ライブラリの利用し、利用者のアプリケーション性能向上に少しでも貢献できれば幸甚です。