Vol.2018-HPC-165 No.11
2018/7/30

NICオフロード機能を用いたノード内永続型ブロードキャストの予備評価

森江 善之[1,a)] 畑中 正行[1] 高木 将通[1] 堀 敦史[1] 石川 裕[1]

概要：MPI 永続型ブロードキャスト関数が規格化されようとしている。本関数を富士通製 PRIMEHPC FX100 に搭載されている Tofu2 インターコネクトのオフロード機能を用いて設計・実装する。実装した関数を富士通が提供する MPI ブロードキャスト関数と比較し、初期設計の問題点を議論し改善方法を提案する。

Evaluation of Intra Node of Persistent Collective Communication using NIC Offloading

Yoshiyuki Morie[1,a)] Masayuki Hatanaka[1] Masamichi Tagaki[1] Atsushi Hori[1]
Yutaka Ishikawa[1]

Abstract: The persistent MPI broadcast function is being standardized. This paper designs and implements this persistent collective for intra-processes within a single node using an offloading capability of Tofu2 interconnect on Fujitsu PRIMEHPC FX100. It is evaluated and compared with Fujitsu's MPI broadcast function, and its issue and improvement are discussed and proposed, respectively.

1. はじめに

近年，スーパーコンピュータは大規模化が進み，数万〜数十万規模の並列計算機が導入されている．これらの大量な計算ノードが参加する集団通信の高速化は重要な課題となっている．これに伴い，新しい非同期集団通信関数である永続型集団通信の導入について議論がされている [1].

永続型集団通信とは集団通信が繰り返し実行されるプログラムにおいてその初期化処理を準備関数として独立させ，集団通信の各インスタンスにおいて初期化処理を省略し，高速化するものである．

現在の大規模並列計算機は，高速かつ高機能なインターコネクトで結合されており，NIC オフロード機能を有するインターコネクトが登場している [2][3]．この NIC オフロード機能は，オフロードのための事前準備が必要であるため，初期化処理を独立させる永続型集団通信と親和性が高く，これを用いた実装により通信性能が向上することが期待される．

これに対し著者ら [4] は，NIC オフロード機能を備える Fujitsu PRIMEHPC FX100 (FX100) を対象として永続型集団通信のプロトタイプ実装を行い，その通信性能や各種オーバヘッドに関して評価を行った．しかし，ノード内マルチプロセスの場合の永続型集団通信に関する実装，評価は行っていない．

そこで，本稿では，FX100 の NIC オフロード機能を用いて永続型ブロードキャストのノード内通信についてノード間およびノード内をバケツリレー方式で実装する方法を提案，これを用いた実装と Fujitsu MPI_Bcast との性能比較を行う．評価の結果，計算ノード毎に 2 プロセス実行する場合は実装方式のほうが Fujitsu MPI_Bcast の性能を上回っているが，それ以上のプロセス数では実装方式の性能は大きく劣化する．これを改善する方法として計算ノード間の通信とノード内通信を 2 つのフェーズに分離し，計算

[1] 理化学研究所 計算科学研究センター
Riken Center for Computational Science (R-CCS), kobe, hyogo, 650-0047, Japan
a) yoshiyuki.morie@riken.jp

ノード内通信において NIC が有する DMA 機能を最大限使用する方式を提案し，その性能を通信モデル上で予測する．予測の結果，Fujitsu MPI_Bcast に比べて計算ノードごとに 4 プロセス実行する場合でも 2 倍以上の性能向上が見込めるという結果が得られた．

本稿は以下のような構成となる．まず，第 2 節にて永続型集団通信の説明を行う．また，第 3 節では今回利用する FX100 の NIC オフロード機能である SMCQ について説明する．次に，第 4 節で永続型ブロードキャストのノード内オフロード実装としてバケツリレー方式の実装手法について述べ，第 5 節で実装した永続型ブロードキャスト通信の関して性能評価実験を行う．第 6 節において．性能改善案である 2 フェーズ通信転送方式を提案し通信モデルによる性能推定を行う．また，第 7 節では，関連研究について述べ，最後にまとめと今後の課題を述べる．

2. 永続型集団通信

永続型集団通信は，初期化関数である MPI_(Bcast/Allgather/...)_init， 開始関数の MPI_Start(), 完了確認関数の MPI_Wait(), 資源解放関数 MPI_Request_free() の 4 個の関数で構成される．

また，MPI_Start() による永続型集団通信の発行，MPI_Wait() で完了確認を行うインターフェースは非同期な動作が可能となる．

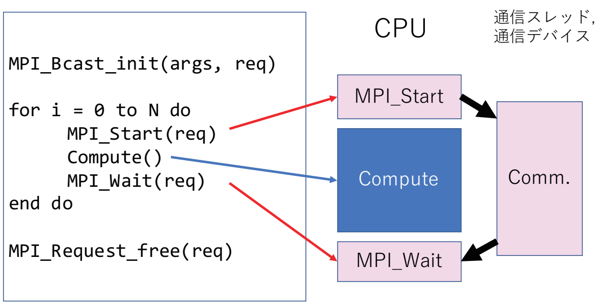

図 1 永続型集団通信の動作概要

永続型集団通信の実行例として図 1 に疑似コードを示す．この疑似コードに示したように，準備関数では，通信に必要な初期化処理をループの外に記述できるため，ループ内の集団通信処理から準備処理に要する時間を取り除くことが可能となる．また，MPI_Start() 後に計算を開始することで，通信と計算の並列動作が可能となる．このような永続型集団通信の具体的な実装としては，集団通信の通信処理を通信スレッドによる実装，もしくは，通信デバイスへのオフロード実装が考えられる．

3. Tofu2 オフロード機能

本章では，本論文の対象となる FX100 の Tofu2 における NIC オフロードの詳細に付いて記す．

Tofu2 では，通信命令を書き込む制御キュー (CQ) を持つ．この制御キューは Session Mode CQ (SMCQ) と呼ばれる NIC オフロード機能を実現するモードを有する．このモードを利用することで通信処理を通信デバイスへオフロードすることができる．

CQ は，送信指示キューを持ち，これにより送信発行等の処理を制御する．この送信指示キューは，Transmit Order Queue (TOQ) と呼ばれ，SMCQ を利用する際には，図 2 のように FIFO 動作を行うためのリードポインタ，ライトポインタに加え，スケジューリングポインタを利用することになる．

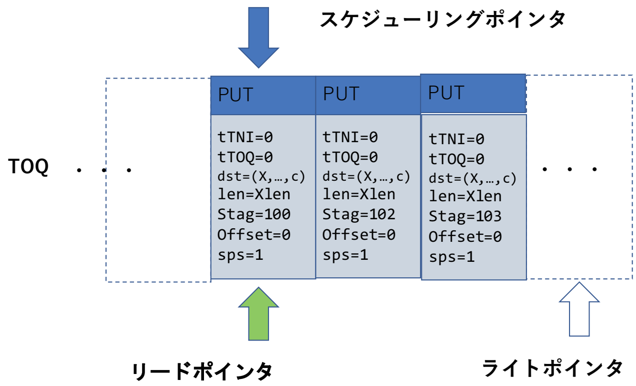

図 2 Session mode CQ

TOQ には，各通信の情報を記述する TOQ ディスクリプタが投入される．この TOQ ディスクリプタを元に TOQ の先頭から Tofu Network Interface (TNI) が通信を実行する．TOQ ディスクリプタは通信に関する各種情報，通信タイプ，ターゲット TNI, ターゲット CQ，宛先物理座標，転送データサイズ，メモリ領域のアドレスに紐付けされたタグ情報 (stag)，転送位置のオフセット情報，スケジューリングポインタの移動量を指定する Session Progress Step (SPS) などを保持する．

ユーザは TOQ ディスクリプタを投入後にキューの末尾を指すようライトポインタを更新する．また，TNI によりリードポインタは TOQ の先頭ディスクリプタを指すように動作する．SMCQ でないモードでは，TNI はリードポインタに従い，先頭から TOQ ディクリプタを元に通信コマンドを発行し，リードポインタが末尾に至るまでコマンドを発行し続ける．スケジューリングポインタは，利用されない．

一方，SMCQ においては，スケジューリングポインタの動作が重要となる．図 3 のようにこのモードの制御においては，スケジューリングポインタまでリードポインタは進行できるが，スケジューリングポインタを追い越すことはできない．このスケジューリングポインタは，Put パケットを受け取った時にパケットの SPS フィールドで指定された値と同数前進する．

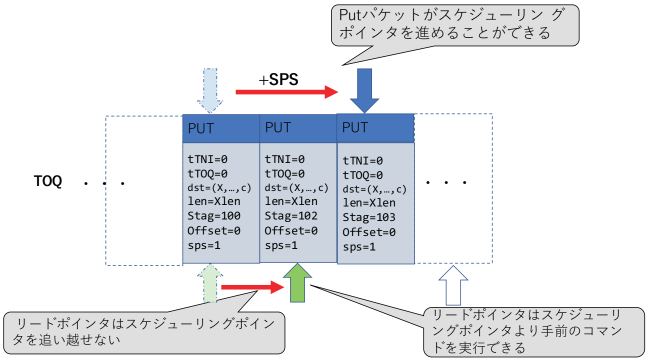

図 3　SMCQ の動作

このスケジューリングポインタの機能により Put パケットが到着しない間は通信の発行を待つことが可能となり，Put パケットが到着した時に通信コマンドをどこまで発行するかを事前に決めることができる．これによりパケットの到着を起点として集団通信などの発行順に依存関係のある通信を CPU の介在なしに進行させる NIC オフロード機能を実現できる．

また，この NIC オフロード機能はマルチプロセスにおける同一ノード内でのデータ転送にも利用可能である．通信の対象となるメモリ領域は，TNI と CQ の組み合わせで登録され，その際にタグづけされ一意に決まる．ノード内マルチプロセスにおいては各プロセスは一意の組み合わせで TNI と CQ を割り付けるため，各プロセスを区別できる．これらから，ノード内通信は自ノードの物理座標，タグ (stag), オフセット，対象メモリ領域を登録した TNI,CQ の組み合わせを TOQ ディスクリプタで指定すること実現できる．

4. ノード内永続型ブロードキャストの NIC オフロード化

本稿では，永続型ブロードキャストに関してノード内通信も含めた NIC オフロード化を行う．

ノード間通信部のブロードキャストアルゴリズムは京コンピュータおよび FX100 でブロードキャストの通信アルゴリズムとして利用された Trinaryx3 アルゴリズム [5] を用いる．このアルゴリズムは，ネットワークトポロジを考慮して通信競合を起こさないようにパイプラインパスを複数生成し，複数搭載されている DMA の帯域幅を効率的に利用する通信アルゴリズムとなっている．

Tofu2 では，各計算ノードにおいて TNI が 4 基搭載されている．Trinaryx3 は，ノード間通信を行う際に各ノードの TNI を 3 基利用する．本節では，余った TNI をノード内通信に専有し，セグメントが到着する度に並列にノード内通信をすることでノード間通信とオーバラップする実装を提案する．

通信データはセングメント分割してバケツリレー転送をおこなうものとする．図 4 では，TNI1〜TNI3 をノード間通信，TNI4 をノード内通信に用いている．ノード間通信用の各 TNI に上流ノードからのセグメントが到着したら，ノード内通信専用の TNI4 が，受信したセグメントをノード内の他プロセスのメモリ領域へ転送する．この時，上流ノードからセグメントが 1 個到着するたびにノード内通信が（ノード内プロセス数-1）回発生する．

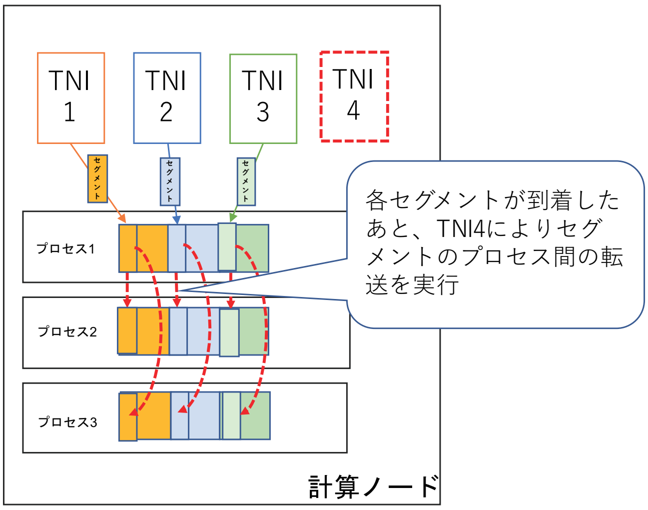

図 4　ノード内通信用 TNI を用いた実装の概要

SMCQ を用いたノード内通信の実装を動作概要を図 5 に示す．ここではノード間通信用の TNI1 とノード内通信用の TNI4 の挙動例を示している．TNI1 の TOQ には 2 つの TOQ ディスクリプタが登録されている．最初の TOQ ディスクリプタが下流ノードへのセグメント通信のためのコマンドであり，後のディスクリプタが TNI4 の TOQ のスケジューリングポインタを進めるためのコマンドである．TNI4 に登録されている 2 つの TOQ ディスクリプタは，それぞれ，プロセス 1 のセグメントをプロセス 2 に，プロセス 1 のセグメントをプロセス 3 に，送信するためのコマンドである．

上流ノードから TNI1 に送信されるセグメント送信 (PUT) パケットの SPS 値は 2 となっている．図 5 において TNI1 にこのパケットが到着すると，スケジューリングポインタ（青矢印）は，+2 される．これにより 2 つのディスクプリタが処理されて，下流ノードへのセグメント通信とノード内通信が行われる．

5. 性能評価実験

今回，性能評価実験では永続型集団通信に関して前節で紹介したノード内通信専用 TNI を用いた実装と Fujitsu MPI_Bcast に関して，それぞれの通信性能の評価を行った．比較対象である Fujitsu MPI_Bcast は FX100 で提供されるよくチューニングされたブロッキング型の bcast 実装であり，通信処理に CPU の介在が必要である一方でノード

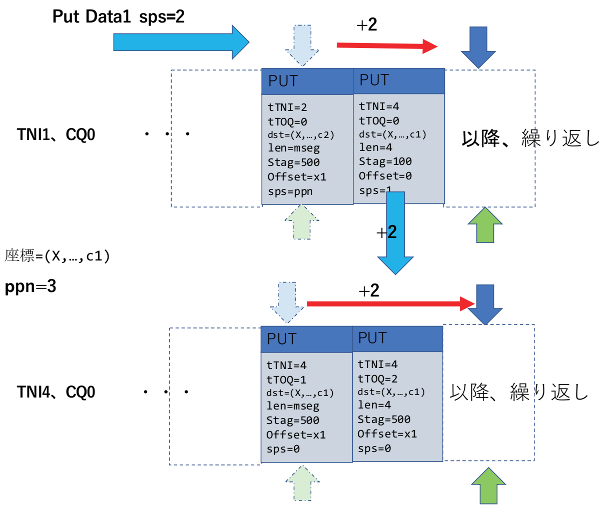

図 5 ノード内通信専用 TNI を用いたバケツリレー実装における SMCQ の動作概要

内のデータ移動に CPU を活用できる利点がある．この実験では，ノード内プロセス数を変えたときの各実装の性能を評価する．

5.1 実験環境

実験環境とした Fujitsu PRIMEHPC FX100 の仕様を以下に述べる．CPU は Fujitsu SPARC64^{TM} XIfx (32 演算コア+2 アシスタントコア)，メモリは 32GB，メモリバンド幅は，480GB/s，計算ノード数は 36 基である．また，OS やコンパイラ，MPI ライブラリはそれぞれ富士通株式会社独自開発のものである．また，ネットワークは，Tofu2[2] を用いる．この通信帯域幅は，12.5GB/s で，ネットワークトポロジは 6 次元メッシュ/トーラスでユーザは仮想 3 次元トーラスとしてアクセスする．また，各計算ノードでは RDMA エンジンを 4 基搭載する．

5.2 実験結果

図 6 に永続型ブロードキャストの実効通信帯域幅を示す．本稿で提案したノード内通信用 TNI を用いた実装による永続型集団通信では，24 プロセス，12 ノードにおいておよそ最大 9GB/s の通信帯域を示した．これは，Tofu2 の論理通信帯域幅と近い値である．また，36 プロセス，12 ノードでは，最大約 5GB/s，48 プロセス，12 ノードでは，最大 3.5GB/s となった．ノード内通信を 1TNI で実行しているため，ノード内通信時間がノード内のプロセス数に比例するため，全体の通信帯域幅が減少する．これに対し，MPI_Bcast は，24 プロセス時点では，同等の性能で 36, 48 プロセスでは，最大約 7GB/s の通信帯域幅であった．これは，ノード内のメモリコピーは，並列に処理されているからだと考えられる．

一方，Fujitsu MPI_Bcast では，図 7 に示すように 12 プロセス，12 ノードの性能は，1 ノード，1 プロセスでの実行に比べ，大きく低下している．これについては今後調査していく予定である．

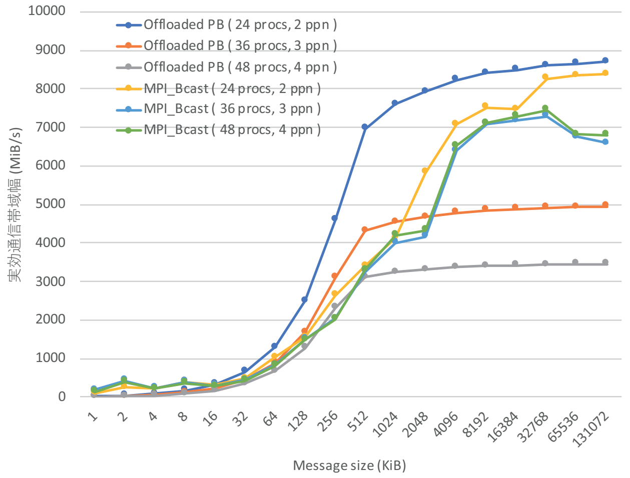

図 6 性能評価

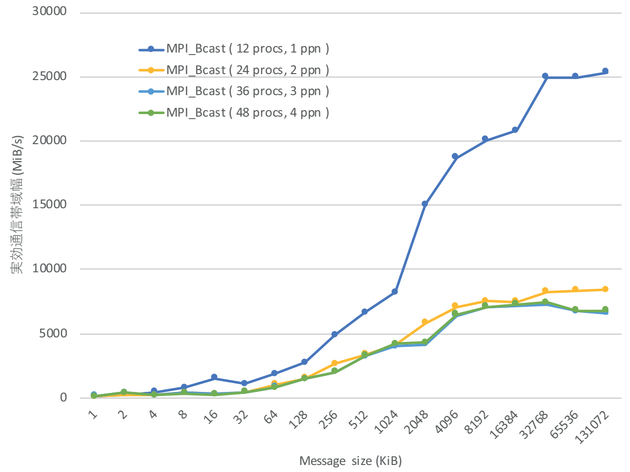

図 7 Fujitsu MPI_Bcast 1 プロセス/ノード vs ノード内マルチプロセス

6. 議論

今回の提案した実装は 3TNI からバケツリレーで転送されたセグメントデータをノード内通信用の TNI1 基でノード内プロセスへデータ転送を行う実装であった．この実装では，ノード内通信に関連しては TNI を 1 基しか用いないため，通信帯域幅が律速したと考えられる．

そこで，この点を改良する実装法としてノード間通信の完了後に 4TNI を利用し，並列にノード内通信を実行する改善方法について検討する．

本改善案では，ノード間の通信が完了したら，4TNI を利用してノード内通信を実行する 2 フェーズ転送方式による実装を行う．ノード内通信では，セグメント分割によるバケツリレー転送を行わずに転送するデータ全体を 4 個の大きなフラグメントに分割し，全 TNI に割り付け並列に転送することでノード内転送の高速化を行う．

図 4 にこの改善案の概要を示す．

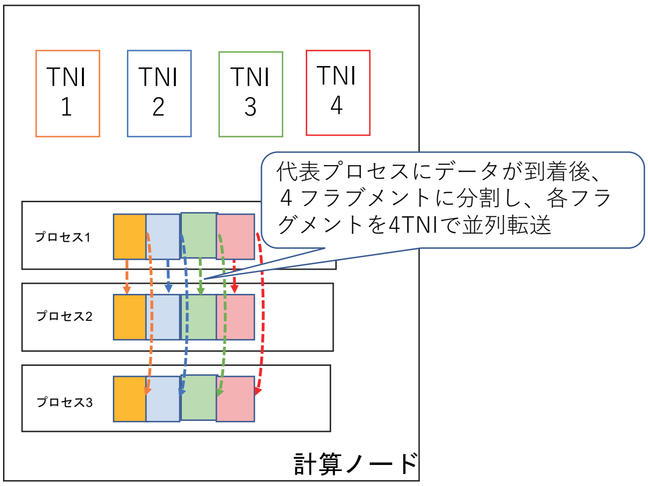

図 8　ノード内通信を複数 TNI により並列実行する実装

以下，改善案による実装の SMCQ での動作について述べる．ノード間通信部のすべてのパイプラインが転送が終了したあとノード内通信を開始する．この通信を起動するために，各 TNI へ Put パケットを送付する．

このため，図 9 に示すように隣接ノードからのノード間通信の最後のセグメントを転送する Put パケットの sps を tni+1 と指定し，これが到着した際にスケジューリングポインタを tni+1 前進させる．これにより，本図においては，TNI2，TNI3 にスケジューリングポインタを進めさせる．進めれらた TNI では，ノード間通信が完了したと判断して完了通知通信を各 TNI へ発行する．

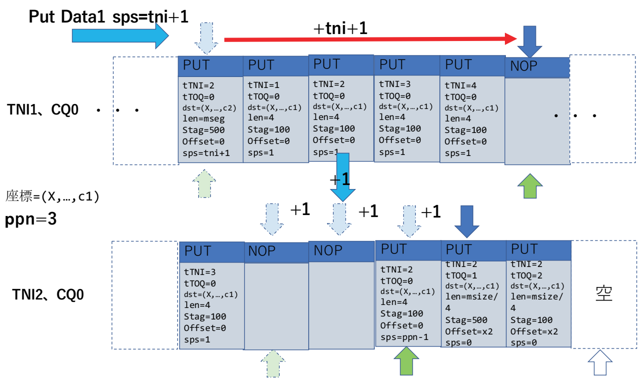

図 9　ノード内通信を複数 TNI により並列実行する実装における SMCQ の動作概要

各 TNI では，2NOP，ノード内通信のトリガー通信，代表プロセス以外の各プロセスへの Bcast データのノード内データコピー通信の TOQ ディスクリプタが登録されている．最初の 2 つの NOP が登録されていることによって，各 TNI が処理しているノード間通信処理終了後に処理されるコマンドによってスケジューリングポインタが移動するが，何の処理もされない．3 個の TNI 全てがノード間通信処理終了後に処理されるコマンドが実行されることにより 2 個の NOP の次に登録されているノード内データコピーのための TOQ ディスクリプタが実行される．言い換えれば，この 2 個の NOP によって各 TNI が処理しているノード間通信処理終了同期を行っている．

これにより代表プロセスの Bcast 受信バッファを 4 基の TNI で分担して代表プロセス以外のプロセスに受信バッファにコピーすることが可能となる．

6.1　性能モデルによる評価

LogGP モデル [6] により今回実装したバケツリレー方式の予測実行時間は，式 1 のようになる．各変数は，L_n は 1 ホップあたりレイテンシ，G_n は 1 バイト送信する際の時間，g は次の通信を発行するまでの時間となる．また，XYZ_{hops} は FX100 の仮想 3 次元の X 軸，Y 軸，Z 軸の最大ホップ数の総和，m はメッセージサイズ，m_{seg} はセグメントサイズ，n_{seg} はセグメント数，ppn はノード内プロセス数である．

$$
\begin{aligned}
T_1 = &((XYZ_{hops}+1)(L_n + m_{seg}G_n) \\
&+(n_{seg}(ppn-1)-1)(m_{seg}G+g) \qquad (1)
\end{aligned}
$$

これから，バケツリレー方式の予測通信帯域幅は式 2 となる．

$$B_1 = m/T_1 \qquad (2)$$

また，２フェーズ転送方式の実装の予測実行時間は，式 3 のようになる．n_{tni} は TNI 数である．

$$
\begin{aligned}
T_2 = &((XYZ_{hops}+1)(L_n + m_{seg}G_n) \\
&+(n_{seg}-1)(m_{seg}G+g) \qquad (3) \\
&+(L_n + m/n_{tni}G+g)
\end{aligned}
$$

これから，予測通信帯域幅は式 4 となる．

$$B_2 = m/T_2 \qquad (4)$$

二つの実装の予測通信帯域幅を図 10 に示す．

各パラメタは，それぞれ $L_n = 1.0e-7sec$，$G_n = 1.04762e-10sec$，$g = 1.0e-9sec$，$XYZ_{hops} = 4$，$n_{tni} = 4$ である．

また，ノード内プロセス数は，2 から 4 個とし，セグメント数は 1KiB-32KiB は 1 個，64KiB-128KiB は 4 個，256KiB-512KiB は 8 個，1MiB-8MiB は 16 個，16MiB-64MiB は 32 個，128MiB は 64 個，256MiB は 128 個，512MiB は 256 個，1024MiB は 512 個とした．

バケツリレー転送方式による実装は，計算ノード内のプロセス数が 2 の場合でも最大約 9GB/s となる．さらに，ノード内プロセスが数が増加すると約 4.5B/s，3GB/s とな

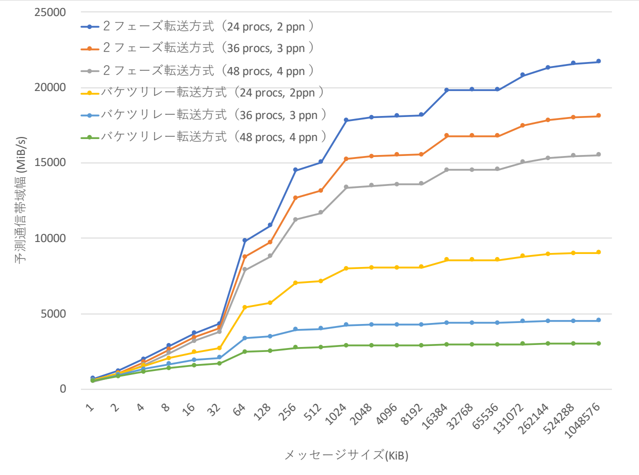

図 10　予測通信帯域幅

る．これはノード内通信に TNI を 1 基しか利用しないことから TNI の通信帯域幅で律速しているためである．一方，２フェーズ転送方式による実装は，計算ノード内プロセス数が 2 の場合に最大で約 22GB/s となる．これはノード間通信を 3TNI で完了させ，その後ノード内通信を 4TNI で並列実行することにより高速化したと考えられる．また，ノード内プロセス数が 4 の場合でも，最大で約 16GB/s の性能となる．これらから，ノード内プロセス数が増加した際にも提案実装では Fujitsu MPI_Bcast より高速となると考えられる．

7.　関連研究

筆者ら [4][7] は，Tofu2 のオフロード機能を用いてブロードキャストアルゴリズムの実装を行った．また，プログレススレッドを用いた際の永続型集団通信の実装を行い，性能を比較する形での評価を行った．しかし，永続型集団通信のノード内転送関して通信デバイスの NIC オフロード機能を利用した実装に関する評価はしていない．

また，MPI でまだ正式に採用されてないことなどから永続型集団通信に関する研究自体が少ない．

Ajima ら [2] は，Tofu2 のオフロード機能を利用する例として Bcast や Gather の非同期集団通信のアルゴリズムを報告している．しかしながら，これらは Tofu2 のネットワークトポロジを考慮したものではない．このため，実際の運用では性能の観点から実用的であるとは言えない．さらに，対象は非同期集団通信を想定しており，永続型集団通信として実装の考慮してものでもない．

また，Hatanaka ら [8] は，Tofu2 のオフロード機能を利用した永続型隣接集団通信の実装を行った．しかし，ここでは多段の依存関係を持つブロードキャストのような永続型集団通信の実装は行われていない．

オフロード機能を持つ通信デバイスとして Mellanox ConnectX-2 や Bull eXascale Interconnect[3] などが挙げられる．これらを利用して非同期集団通信の実装 [9] などが行われているが，永続型集団通信の実装については報告はない．

南里ら [10] は，非同期集団通信の通信隠蔽効果の調査を行った．FX100 のアシスタントコアによる非同期集団通信や Mellanox 社のオフロード機能である SHArP 機能を用いた非同期集団通信の隠蔽効果を調査を行っている．しかし，永続型集団通信の調査は行っていない．

8.　おわりに

本稿では，現在 MPI フォーラムで規格化されようとしている MPI 永続型ブロードキャスト関数のノード間およびノード内通信を富士通製 PRIMEHPC FX100 に搭載されている Tofu2 インターコネクトのオフロード機能を用いた実装を提案し，富士通が提供する MPI ブロードキャスト関数と比較した．計算ノード内 2 プロセスであれば提案手法は Fujitsu MPI_Bcast よりも高い性能が得られたが，それ以上のプロセス数では性能が劣ることが判明した．

この原因は，計算ノード間および計算ノード内のデータ転送を同時に行うために計算ノード間転送用に TNI を 3 つ，計算ノード内データ転送用に TNI を 1 つ割当てたために，TNI1 つによる計算ノード内複数プロセスデータ転送時間が全体の性能律速になっている．その改善方法として，計算ノード間でバケツリレーするフェーズと計算ノード内でデータを転送する 2 フェーズに分離し，計算ノード内において NIC の DMA 機能を最大限使用する方式を提案し，その性能を通信モデルにより予測した．この予測から，各計算ノード内 4 プロセスで実行する場合でも Fujitsu MPI_Bcast に比べ，2 倍以上の性能向上が見込める結果となった．

今後，本改良版を実装し基本性能評価を行うとともに，計算と通信のオーバラップを行った際の性能の評価を行う．また，今回はブロードキャストのみの実装となったが，他の永続型集団通信についても同様に評価実験を行う．

謝辞　本論文の一部は，「特定先端運営費等補助金/ 次世代超高速電子計算機システムの開発・整備等」で実施された内容に基づくものである．

参考文献

[1] Message Passing Interface Forum. MPI STANDARD EFFORTS / MPI 4.0. `http://www.mpi-forum.org/mpi-40/`.

[2] Y. Ajima, T. Inoue, S. Hiramoto, S. Ando, M. Maeda, T. Yoshikawa, K. Hosoe, and T. Shimizu. Tofu interconnect 2. In *2014 IEEE 22nd Annual Symposium on High-Performance Interconnects*, pp. 57–62, aug 2014.

[3] S. Derradji, T. Palfer-Sollier, J. Panziera, A. Poudes, and F. W. Atos. The bxi interconnect architecture. In *IEEE 23rd Annual Symposium on High-Performance Interconnects (HOTI'15)*, pp. 18–25, aug 2015.

[4] Y. Morie, M. Hatanaka, M. Takagi, A Hori, and Y. Ishikawa. Prototyping of offloaded persistent broadcast on tofu2 interconnect. In *SC17*, nov 2017.

[5] T. Adachi, N. Shida, K. Miura, S. Sumimoto, A. Uno, M. Kurokawa, F. Shoji, and M. Yokokawa. The design of ultra scalable mpi collective communication on the k computer. *Computer Science - Research and Development*, Vol. 28, No. 2-3, pp. 147–155, may 2013.

[6] Albert Alexandrov, Mihai F. Ionescu, Klaus E. Schauser, and Chris Scheiman. LogGP: incorporating long messages into the LogP model - one step closer towards a realistic model for parallel computation. In *SPAA '95 Proceedings of the seventh annual ACM symposium on Parallel algorithms and architectures*, jun 1995.

[7] 森江善之, 畑中正行, 高木将通, 堀敦史, 石川裕. FX100 における永続型集団通信関数のプロトタイプ実装と評価. 情報処理学会研究会報告ハイパフォーマンスコンピューティング（HPC), 第 2018-HPC-163 巻, feb 2018.

[8] M. Hatanaka, M. Takagi, A Hori, and Y. Ishikawa. Offloaded mpi persistent collectives using persistent generalized request interface. In *EuroMPI/USA 2017*, sep 2017.

[9] S. Di Girolamo, P. Jolivet, K. D. Underwood, and T. Hoefler. Exploiting offload-enabled network interfaces. *IEEE Micro*, Vol. 36, No. 4, pp. 6–17, aug 2016.

[10] 南里豪志, 大島聡史, 小野 謙二. 非ブロッキング集団通信の通信隠蔽効果に関する調査. 情報処理学会研究会報告ハイパフォーマンスコンピューティング（HPC), 第 2017-HPC-162 巻, pp. 1–11, dec 2017.