Vol.2010-HPC-126 No.16
2010/8/3

多段スイッチ InfiniBand ネットワークにおける全対全通信性能の評価

成瀬 彰[†1] 中島 耕太[†1]
住元 真司[†1] 久門 耕一[†1]

本稿では、Fat-tree InfiniBand(IB) ネットワーク上で全対全通信時の Hot-spot 発生頻度を削減するルーティング手法を提案・評価する。Fat-tree IB ネットワークでは適切に使用ノードを選択しないと Hot-spot 発生により実効通信バンド幅が低下する。提案手法は、各計算ノードに複数の LID(Local Identifier) を割当てる手法の一種であり、具体的には全対全通信でよく用いられるシフト通信パターンに着目し、計算ノード毎に適切な LID を一つ選択・使用することで Hot-spot 発生頻度を削減する手法である。提案手法を OpenMPI に組込み、全対全通信性能を評価する。30 台の計算ノードを 6-ary-2-tree 構成の Fat-tree IB ネットワークに接続した PC クラスタシステム上にて、任意 16 ノードによる全対全通信性能は従来手法と比較して平均で 34%向上することを確認している。

Performance Evaluation of All-to-all Communication on PC Cluster Systems with Multi-Stage InfiniBand Networks

Akira NARUSE,[†1] Kohta NAKASHIMA,[†1]
Shinji SUMIMOTO[†1] and Kouichi KUMON[†1]

In this paper, we propose a method to reduce a hot-spot occurrence during all-to-all communication on fat-tree based InfiniBand networks and evaluate it. Even on fat-tree based networks, an effective network bandwidth degrades due to a hot-spot occurrence unless proper nodes are assigned to a MPI job. Our proposed method utilizes LMC (LID Mask Control) feature of InfiniBand and assigns multiple LIDs (Local Identifier) to each HCA (Host Channel Adapter) in a compute node. It reduces a hot-spot occurrence during all-to-all communication significantly by selecting an appropriate single LID for each HCA according to nodes assignment. We implement a proposed method into Open-MPI and evaluate it on real InfiniBand cluster system consists of 30 compute nodes and 6-ary-2-tree topology networks. All-to-all benchmark shows 1.34 times performance improvement compared to existing method when arbitrary 16-nodes are assigned to a MPI job.

1. はじめに

近年、高いネットワーク性能を必要とする HPC システムでは、高バンド幅・低遅延な InfiniBand(IB)[2] と FBB(Fully Bisectional Bandwidth) 特性を持つ完全 Fat-tree トポロジーを組み合わせた Fat-tree IB ネットワークが良く採用されている。IB の問題はルーティングである。IB は static ルーティングであり adaptive ルーティングをサポートしていない。そのため、FBB 特性を持つ完全 Fat-tree トポロジーであっても、使用する計算ノードを適切に選択しないと、全対全通信中に特定リンクに通信負荷が集中して (Hot-spot 発生)、実効ネットワークバンド幅が低下する。

本稿では、Fat-tree IB ネットワーク上で全対全通信時の Hot-spot 発生頻度を削減するルーティング手法を提案する。提案手法は、各計算ノードに複数の LID(Local Identifier) を割当てる手法の一種であるが、複数 LID を同時使用してマルチパスを形成し、ネットワークトラフィックを複数パスに分散させることで Hot-spot による通信性能低下を軽減する手法ではない。全対全通信でよく用いられるシフト通信パターンに着目し、計算ノード毎に適切な LID を一つ選択・使用することで Hot-spot 発生頻度を削減する手法である。

提案手法を 6-ary-2-tree 構成の Fat-tree IB ネットワークに接続した 30 ノードの PC クラスタシステム上で評価した。全 30 ノードから任意 16 ノードを使用して全対全通信を行うときの実効ネットワークバンド幅は、従来手法と比較して平均で 34%向上することを確認している。

以下、2 章で提案手法を理解する上で最低限必要な情報を説明し、3 章で Fat-tree IB ネットワーク上で Hot-spot が発生する仕組みを明らかにし、4 章で Hot-spot 発生頻度を削減するルーティング手法を提案し、5 章で提案手法を実機上で評価し、6 章で関連研究に言及し、7 章でまとめる。

†1 富士通研究所
Fujitsu Laboratories

2. 背　　景

2.1 InfiniBand

InfiniBand(IB)[2] は低遅延かつ高バンド幅なネットワークであり、HPC クラスタ用のネットワークとして広く普及している。現在では 40Gbps のピークバンド幅を持つ 4xQDR が利用可能で、今後も継続的なバンド幅向上が見込まれている。

IB のルーティング方式は static ルーティングである。各 IB スイッチはそれぞれ転送テーブルを持っており、各パケットのヘッダに記載されている転送先 LID(Local Identifier) と転送先テーブルから、パケットを転送するポートが一意に決まる仕組みとなっている。各 IB スイッチの転送テーブルの設定は SM(Subnet Manager) の仕事である。

通常、各 HCA(Host Channel Adapter) に割り当てられる LID は 1 つである。しかし、IB の LMC(LID Mask Control) 機能を利用することで、各 HCA に最多 128 個の LID を割り当てることが可能である。LMC 機能は、任意サーバ間の通信でマルチパス通信を行うために使われることが多い[10)15)]。我々の提案手法も LMC 機能を利用するが、任意サーバ間でのマルチパス通信は行わない。

2.2 Fat-tree (K-ary-N-tree)

FBB 特性を持つ完全 Fat-tree(K-ary-N-tree) は IB 接続の PC クラスタで良く使われるトポロジーである[1)]。理論上、任意サーバ間の通信をシステム全体で同時実行しても、リンク競合ゼロで通信できる通信パスが存在する。しかし、通信パスを動的に選択できない場合には、通信パターン次第でリンク競合が発生して通信性能が低下する。動的に通信パスを選択する adaptive ルーティング機能を持たない IB ネットワークでは、MPI ジョブに割当てるノードの組み合わせや通信パターン次第で、大幅に通信性能が低下することが報告されている[10)]。

Fat-tree ネットワークのステージ数 (N) は、ネットワークに接続するノード数と (システム規模)、ネットワークの基礎構成要素であるスイッチチップのポート数で決まる。1 つの IB スイッチチップがサポートするポート数は増加傾向にあり、最新の IB スイッチチップは 36 ポート対応である。この 36 ポート IB スイッチチップでステージ数 2 の完全 Fat-tree を構築すると、最大構成は 18-ary-2-tree となり最多 648 台の計算ノードを接続することができる。これは、ステージ数 2 の多段ネットワークで相当規模のシステムを実現可能であることを示している。そこで、本稿ではネットワークトポロジーを 2 ステージ構成の Fat-tree、K-ary-2-tree に限定して議論する。

2.3 シフト通信パターンとルーティング

シフト通信パターンは、全プロセスが全プロセスとメッセージ交換する全対全通信時に良く用いられる通信パターンである[7)]。例えばプロセス数を Np、各プロセス名を P_j $(0 \leq j < Np)$ とすると、全対全通信は Np 回の通信ステージから構成され、通信ステージ i のとき $(0 \leq i < Np)$、プロセス P_j は i 個先のプロセス P_k $(k = (j + i)\ mod\ Np)$ にメッセージを送信する。特徴は、同時刻には全プロセスが基本的には同じ通信ステージを処理しており、同じシフト距離通信を行うことである。

シフト通信パターンは全対全通信に固有の通信パターンではなく、典型的な通信パターンの一種である。隣接ノード間通信など、規則的な通信パターンの多くは、特定のシフト距離通信にマップできることが知られている[12)]。

実際、Fat-tree IB ネットワークではシフト通信に最適化したルーティング設定で良い性能が得られることが報告されている[6)7)]。また、主要な IB SM である OpenSM[8)] にも、この Fat-tree 向けルーティングは組み込まれており、シフト通信パターンに最適化したルーティングは特別なものではない。

2.4 MPI ジョブへのノード割当

Fat-tree IB ネットワークでは MPI ジョブに適切なノード割当てを行えば高い通信性能が得られるが、実運用システムで常に適切なノード割当てを実現するのは簡単ではない。数百台規模のシステムは、単一ユーザに占有されるのではなく、複数ユーザに共有されるのが常であり、複数の MPI ジョブが同時に実行されている状態が一般的である。また、各 MPI ジョブの並列度 (使用ノード数) や実行時間はジョブ毎に様々であるため、MPI ジョブに対して規則的に、例えば割当てノード番号が連番となるようにノードを割当てる方針でシステム運用すると、システムの稼働率 (ノード利用率) が低下してしまう。ノード利用率を最大化するには、各 MPI ジョブに対してノード番号が不連続なノード、つまり任意のノードの割当てを許す必要がある。

しかし、前述のシフト通信パターンに最適化したルーティングでは、適切なノードを割当てたときには良い性能が得られるものの、適切なノード割当てができなかったときには Hot-spot が発生して通信性能が低下することが知られている[10)]。つまり、各ジョブの通信性能とシステムのノード利用率はトレードオフの関係にあり、両立は困難である。これは、Fat-tree IB ネットワークの解決すべき課題である。

Vol.2010-HPC-126 No.16
2010/8/3

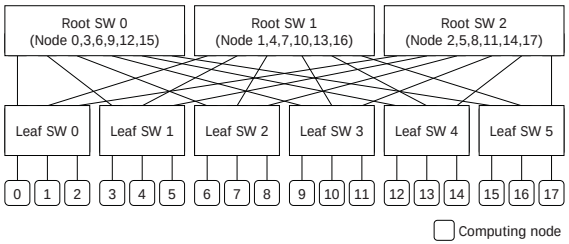

図 1 3-ary-2-tree 構成の Fat-tree ネットワーク例

3. Fat-tree IB ネットワーク上での Hot-spot 発生の仕組み

本章では、Fat-tree IB ネットワーク上で Hot-spot が発生する仕組みを説明する。以下では、ノード数 18、3-ary-2-tree の Fat-tree ネットワーク構成を事例に説明する (図 1)。ルーティング設定は文献 6) と同様の手法を採用し、ノード N_i $(0 \leq i < 18)$ へのパケットは、Root スイッチ $RS_j (j = i \ mod \ 3)$ を経由するルーティング設定になっているものとする。

以下、任意ノード割当て時の Hot-spot 発生の仕組みと、連番ノード割当て時の Hot-spot 発生の仕組みを説明する。

3.1 任意ノード割当て時の Hot-spot 発生

MPI ジョブに任意ノードが割当てられる場合の Hot-spot 発生に関して説明する。図 2 に具体的な Hot-spot 事例を示す。図 2 では、MPI ジョブは 4 ノードジョブであり、ノード N_i $(i \in \{3, 5, 6, 9\})$ の 4 ノードが割り当てられ、ノードあたりのプロセス数は 1、プロセス番号 (rank 番号) はノード番号順に割り当てられている。この条件で、シフト距離が 2 のシフト通信を行うと、ノード N_3 はノード N_6 へ、ノード N_5 はノード N_9 へメッセージを送信する。この 2 通信の送信元であるノード N_3 と N_5 はどちらも Leaf スイッチ LS_1 に接続されたノードである。そして、送信先であるノード N_6 と N_9 へのパケットはどちらも Root スイッチ RS_0 を経由する。従って、この事例ではシフト距離が 2 のシフト通信時に、Leaf スイッチ LS_1 から Root スイッチ RS_0 へのリンクで競合が発生する。

これが任意ノード割当て時の HotSpot 発生の基本的な仕組みである。図 2 では、2 つの通信が 1 つのリンクに集中しているだけだが、状況次第でより多数の通信が 1 つのリンクに集中することがある。その場合には、ネットワークバンド幅が大幅に低下することが報告されている[10)]

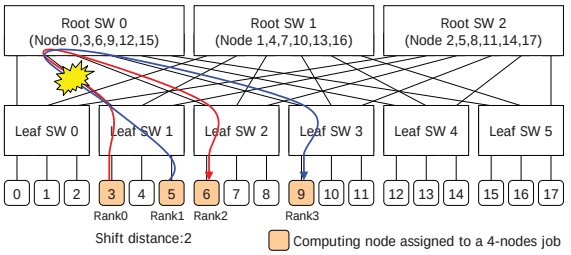

図 2 任意ノード割当て時の Hot-spot 発生例

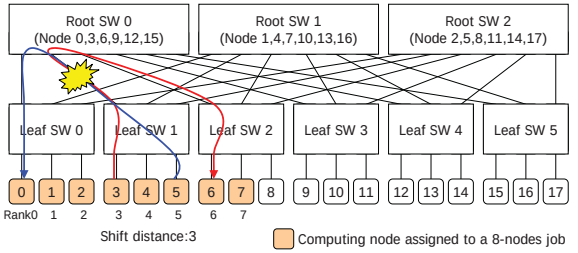

図 3 連番ノード割当て時の Hot-spot 発生例

3.2 連番ノード割当て時の Hot-spot 発生

MPI ジョブに連番ノードを割当てる場合の Hot-spot 発生に関して説明する。図 3 に具体的な Hot-spot 発生事例を示す。図 3 では、ジョブは 8 ノードジョブであり、ノード N_i $(i \in \{0, 1, ..., 7\})$ のノード番号が連続した 8 ノードが割り当てられ、ノードあたりプロセス数は 1 で、プロセス番号 (rank 番号) はノード番号順に割り当てられている。この条件で、シフト距離が 3 のシフト通信を行うと、ノード N_3 はノード N_6 へ、ノード N_5 はノード N_0 へメッセージを送信する。この 2 通信の送信元であるノード N_3 と N_5 はどちらも Leaf スイッチ LS_1 に接続されたノードである。一方、送信先であるノード N_6 と N_0 へのパケットはどちらも Root スイッチ RS_0 を経由する。従って、この事例では少なくともシフト距離が 3 のシフト通信時に、Leaf スイッチ LS_1 から Root スイッチ RS_0 へのリンクで競合が発生する。

一般的には、連番ノード割当て時には Hot-spot は発生しないと思われているが、実際には図 3 に示す通り、Hot-spot が発生する。これは、ネットワーク構成に対して使用ノード

の配置が非対称なときに発生する Hot-spot である。具体的には、Leaf スイッチ毎の使用ノード配置が、Leaf スイッチ間で一致していないときに Hot-spot が発生する可能性がある。図 3 では、Leaf スイッチ LS_0 と LS_1 に接続のノードはそれぞれ 3 台使用されているが、Leaf スイッチ LS_2 に接続のノードは 2 台しか使用されておらず、Leaf スイッチ間で使用ノード数が一致していない。

図 3 の事例で Hot-spot 発生を回避するには、例えばノード N_i $(i \in \{0, 1, 3, 4, 6, 7, 9, 10\})$ のように、ネットワーク構成に対して使用ノードの配置が対称となるよう注意深くノードを選択する必要がある[10)]。そうしないと、Hot-spot が発生し通信性能が低下する。

4. Hot-spot 発生頻度を削減する手法の提案

本章では、Fat-tree IB ネットワークにおける任意ノード割当て時の Hot-spot 発生頻度を削減する手法を提案する。提案手法は、IB の LMC 機能を利用して各ノードに複数の LID を割当てる手法の一種である。既存研究[10) 15)] では複数 LID を用いて 2 ノード間にマルチパスを形成して特定リンクへの負荷集中を回避しているが、我々の提案手法はそれとは異なる。全対全通信時に特徴的なシフト通信パターンに着目し、Hot-spot が発生頻度が低くなるようノード毎に適切な LID を 1 つ選択する手法である。なお、本提案手法は任意ノード割当て時の Hot-spot 発生頻度削減を目的としたものであり、連番ノード割当て時の Hot-spot 発生頻度は削減できない。

以下、各ノードに複数 LID を割り当てる方法と、各ノードが使用する LID を選択する方法を説明する。

4.1 複数 LID 割当てと各 LID のルーティング設定

本節では、複数 LID を割り当てる方法と、各 LID のルーティング設定方法を説明する。

各計算ノードに複数 LID を割り当てる方式は、文献 3) と同様、IB の LMC 機能を使用する。LMC 機能には 0〜7 の値を指定可能であり、指定値を lmc $(0 \leq lmc \leq 7)$ とすると、各ノード[⋆1]には 2^{lmc} 個の LID が連番で割当てられる。ネットワーク構成が K-ary-2-tree の場合は各ノードに K 個の LID を割当てれば十分であり、LMC の適正値は $lmc = \lceil log_2 K \rceil$ となる。例えば、ネットワーク構成が 3-ary-2-tree であれば $lmc = 2$ となり、各ノードには 4 個の LID が割り当てられる。

各 LID のルーティング設定は文献 3) と同様の手法を使用する。計算ノード N_i $(0 \leq i <$ 総計算ノード数) には、$\{BaseLID(i), BaseLID(i) + 1, ..., BaseLID(i) + 2^{lmc} - 1\}$ $(BaseLID(i) = 2^{lmc} \times (i + 1))$ の LID を割り当てる。そして、LID_j 宛のパケットはルートスイッチ RS_k $(k = j \ mod \ 2^{lmc})$ を経由するように設定する。なお、$k \geq K$ となる LID は使用しない。

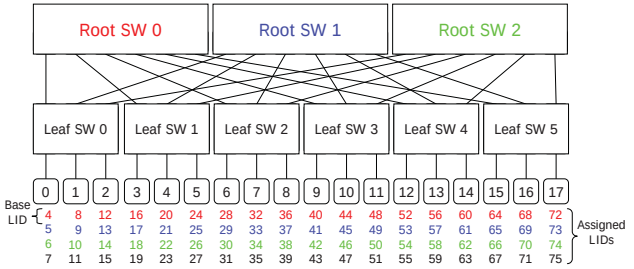

図 4 複数 LID 割当てと各 LID に対するルーティング設定の例

図 4 に具体的な LID 割り当て例を示す。計算ノード N_0 には 4 個の $LID_j (j \in \{4, 5, 6, 7\})$ が割り当てられ、LID_4、LID_5、LID_6 宛のパケットは、それぞれルートスイッチ RS_0、RS_1、RS_2 を経由するように設定される。LID_7 は使用されない。

4.2 使用 LID の選択

MPI プログラム起動時に各ノードの使用 LID を選択する方法を説明する。使用 LID の選択手順は次の 3 ステップである。

(1) ローカルノード ID の割当て
(2) 各ローカルノード宛パケットが経由するルートスイッチの決定
(3) 使用 LID の決定

以下、それぞれのステップを説明する。

手順 1: ローカルノード ID の割当て

MPI ジョブに割当てられた各計算ノードに対して、ローカルノード ID を割り当てる。通常は、hostfile 等に記載の順番で、各ノードに対してローカルノード ID を割り当てるが、本手法ではネットワーク構成に準じたノード番号の順番でローカルノード ID を割り当てる。例えば、図 4 の環境でノード N_i $(i \in 0, 3, 6)$ の 3 ノードが MPI ジョブに割り当てられた場合、N_0、N_3、N_6 のローカルノード ID はそれぞれ 0、1、2 となる。

手順 2: 各ローカルノード宛パケットが経由するルート スイッチの決定

各ローカルノード宛のパケットをどのルートスイッチに転送するかを決定する。ネット

⋆1 厳密には HCA だが、ここでは 1 ノードあたり 1HCA と仮定

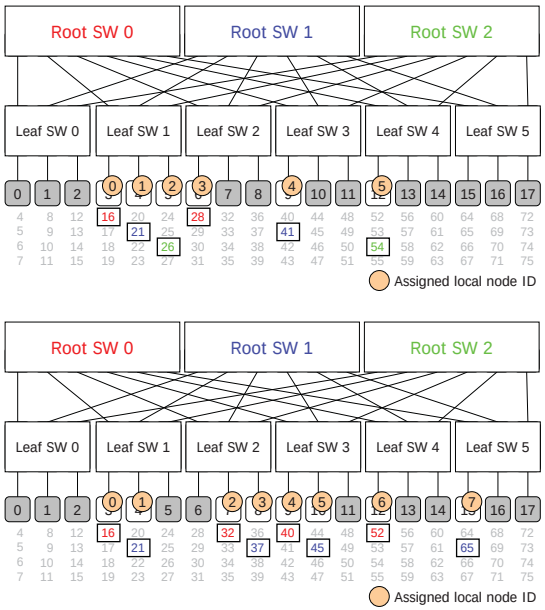

図 5 提案手法による使用 LID 選択の事例

ワーク構成を K-ary-2-tree、MPI ジョブに割当てた計算ノード数を Nn とすると、ローカルノード ID が l であるノードへのパケットは、ルートスイッチ RS_r $(r = l \ mod \ M)$ を経由させる。ここで、基本的には $M = K$ であるが、下記条件を満たす場合は $M = Nln$ とする。

- Leaf スイッチ毎に、直接接続の計算ノードの内、該当 MPI ジョブに割当てられた計算ノード数をカウントする。その最大値を Nln とするとき、$Nn \ mod \ Nln = 0$ を満たす場合には、$M = Nln$ とする。

手順 3: 使用 LID の決定

各ローカルノードの使用 LID を決定する。ローカルノード ID が l であるノードの LID は、下記計算式で決定する。

$$BaseLID(NodeID(l)) + rs(l)$$

ここで、$NodeID(l)$ はローカルノード ID が l であるノードのノード番号、$rs(l)$ は手順 2 で決定したローカルノード ID が l のノード宛のパケットが経由する Root スイッチ番号である。

提案手法による使用 LID 選択の事例を図 5 に示す。図 5(上) は任意 6 ノードを割当て時の事例、図 5(下) は任意 8 ノードを割当て時の事例である。なお、図 5 の事例は、従来手法では全対全通信中に Hot-spot 発生で通信性能が低下するが、提案手法を使うことで Hot-spot 発生を回避できる事例である。

表 1 テスト環境

Server	30x Self-made IA server
CPU	2x Intel Xeon X5570 (2.93GHz)
Mem	24GB (6x 4GB DDR3-1333 DIMM)
IB HCA	Mellanox MHGH29-XTC (4xDDR)
OS	RHEL5.4 (64bit)
MPI	OpenMPI 1.4.1
Compiler	gcc 4.1.2 (option: -O3)
Network	
Topology	6-ary-2-tree
Switch	7x Flextronics F-X430044 (4xDDR, 24-port)
SM	OpenSM 3.2.6

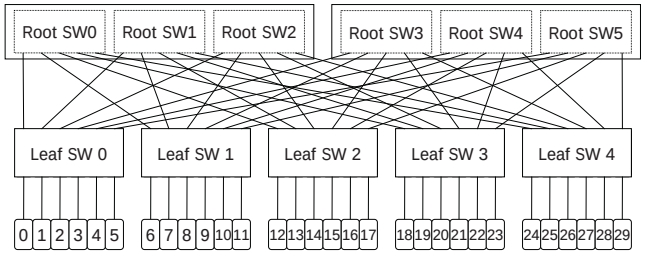

図 6 テスト環境のネットワーク構成 (6-ary-2-tree, 30nodes)

5. 評　　価

従来手法 (単一 LID 割当て方式) と提案手法を、All-to-all ベンチマークで評価した結果を説明する。

5.1 テスト環境

測定環境は表 1 に示す通りである。計算ノードは Intel Xeon X5570 を搭載した 2 socket サーバで、それぞれ 1 枚の IB HCA(4xDDR 対応) を搭載している。ノード数は 30 である。

```
for (j=0; j<Nc; j++) {
    Ic_to   = (my_Ic + j + Nc) % Nc;
    Ic_from = (my_Ic - j + Nc) % Nc;
    for (k=0; k<Nlp; k++) {
        Ilp_to   = (my_Ilp + k + Nlp) % Nlp;
        Ilp_from = (my_Ilp - k + Nlp) % Nlp;
        Ip_to   = (Ic_to    * Nlp) + Ilp_to;
        Ip_from = (Ic_from * Nlp) + Ilp_from;
        Comm("send to Ip_to, recv from Ip_from");
    }
}
// Nc:  Number of compute nodes
// Nlp: Number of process in a node (local process)
// Np:  Total number of process  (Nc * Nlp)
// Ic:  ID of compute node  (0 <= Ic < Nc)
// Ilp: ID of local process  (0 <= Ilp < Nlp)
// Ip:  ID of process  (Ip = Ic * Nlp + Ilp)
```

図 7 2-level ring アルゴリズムの疑似コード

$$Alltoall\_Bandwidth = \frac{Msg \times (Nn-1) \times Nlp \times Nlp}{T}$$

```
Msg: Size of message exchanged
     among each process-pair.
Nn:  Number of nodes.
Nlp: Number of processes in a node.
T:   Elapsed time of all-to-all.
```

図 8 Alltoall バンド幅の定義

ネットワークトポロジーは 6-ary-2-tree であり、7 台の 4xDDR 対応の IB スイッチで構成した (図 6)

Subnet Manager(SM) は OpenSM を使用した。従来手法の性能測定時には、OpenSM を FTREE エンジンで起動した。提案手法の性能測定時には、提案手法に即したルーティングファイルを作成、それを入力ファイルとして OpenSM を FILE エンジンで起動した。

MPI プログラム起動時に使用 LID を選択する仕組みは、OpenMPI[9] に組み込んだ。具体的には、MPI_Init() 直後に各ノードの使用 LID を選択し、OpenMPI の blt_openib コンポーネント内で選択 LID を用いて QP を生成した。

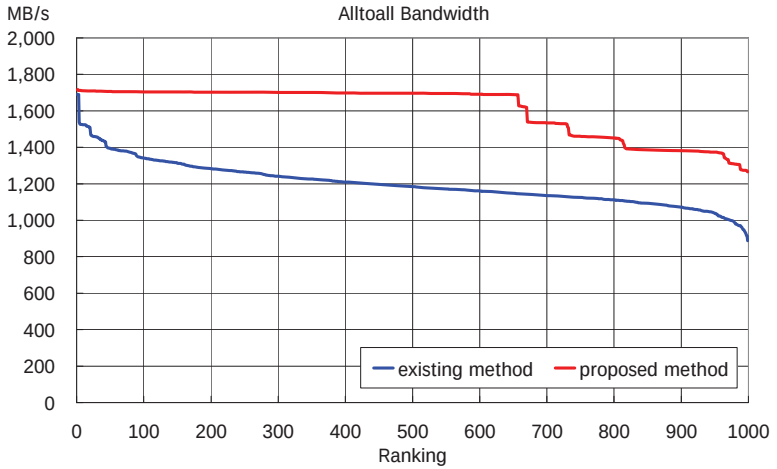

図 9 任意 16 ノードでの全対全通信性能測定結果

All-to-all 通信アルゴリズムにはマルチコアシステムに適切な 2-level ring アルゴリズムを用いた[13]。通常の ring アルゴリズム (pair-wise アルゴリズム) をマルチコアシステムで使用するとスイッチチップ内で HoL(Head-of-line) ブロッキング[5] が発生して通信性能が低下する。2-level ring アルゴリズムではこの HoL ブロッキング発生を回避することができる。図 7 に 2-level ring アルゴリズムの疑似コードを示す。

Alltoall ベンチマークの性能指標には、図 8 に示す Alltoall バンド幅を用いた。Alltoall バンド幅は、全対全通信中のノードあたり実効ネットワーク性能を示す指標である。ネットワーク性能 (ノード間通信性能) を知ることが目的であり、ノード内通信量は計算式から除外している。なお、実行時間 T にはノード内通信時間も含まれており、厳密には Alltoall バンド幅は実効ネットワーク性能を示しているとは言えない。しかし、一般的にはノード内通信時間はノード間通信時間より短く、実行時間 T に占めるノード内通信時間の比率は小さいと考えられるので、本稿では Alltoall バンド幅を実効ネットワークバンド幅とみなす。

5.2 任意ノード割当て時の全対全通信性能

図 9 に、全 30 ノードから任意 16 ノードを割当て Alltoall ベンチマークを実行した結果を示す。測定は、それぞれ割当てノードセットの異なる 1,000 種類のホストファイルを用

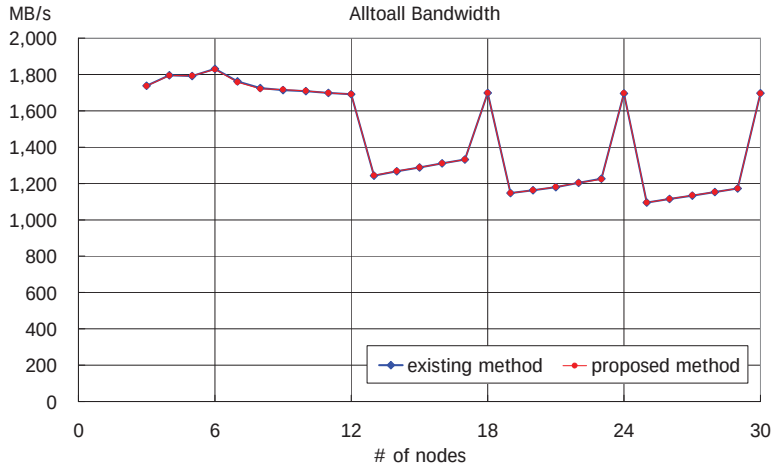

図 10 連番ノードでの全対全通信性能測定結果

い、1,000 回実施した。図は、縦軸が Alltoall バンド幅であり、横軸は 1,000 回の測定結果を Alltoall バンド幅順に並べたものである。青線が従来方式、赤線が提案方式の測定結果を示している。なお、ノードあたりのプロセス数は 8 で、各プロセス間で交換するメッセージサイズは 1MiB とした。

従来方式は、最高性能は 1,690MB/s で提案方針と同程度であるが、平均性能は 1,198MB/s、最低性能は 888MB/s である。割当てノード次第で Alltoall バンド幅は大きく変動しており、1,500MB/s 以上を記録したのは 20 回 (/1,000 回) であった。従来方式では、任意ノード割当て時に安定して高い通信性能が得ることが難しいことが分かる。

提案方式は、最高性能は 1,716MB/s、平均性能は 1,607MB/s、最低性能は 1,265MB/s である。提案方式でも、割当てノード次第で Alltoall バンド幅は変動するが、従来方式と比較すると性能変動は緩やかであり安定している。また、1,500MB/s 以上を記録したのは 732 回 (/1,000 回) であった。提案手法でネットワーク上の Hot-spot 発生頻度を削減した効果である。

5.3 連番ノード割当て時の全対全通信性能

図 10 に、連番ノード割当て時の Alltoall バンド幅測定結果を示す。性能測定はノード数 3〜30 で実施、ノード数 Nn のときにはノード N_i $(0 \leq i < Nn)$ を使って性能測定した。図は、縦軸が Alltoall バンド幅であり、横軸がノード数である。青線が従来方式、赤線が提案方式の測定結果を示している。なお、ノードあたりのプロセス数は 8 で、各プロセス間で交換するメッセージサイズは 1MiB とした。

従来方式と提案方式で同性能となっているが、これは、提案手法では連番ノード割当て時に発生する Hot-spot を削減できないためであり、予想通りの結果である。

図 10 より、割当てノード数がノード数が $12(2 \times K)$ 以下の場合、もしくは、$6(K)$ の倍数である場合には、良い性能が得られていることが分かる。しかし、それ以外の場合には Alltoall バンド幅が低下している。このバンド幅低下は無視できるレベルではない。例えば、ノード数 16 の場合の Alltoall バンド幅は 1,310MB/s であり、任意 16 ノード割当て時に提案手法を用いた場合の平均バンド幅 (1,607MB/s) より 18%低い。これは、連番ノードを割当てより、任意ノード割当てと提案手法の組み合わせで良い性能が得られる可能性を示唆している。

6. 関連研究

Fat-tree ネットワークでの Hot-spot 発生は以前から知られている問題であり、そのため、Fat-tree 向けの adaptive ルーティング方式がいろいろ検討評価されている[14)15)]。しかし、IB は static ルーティング方式であり、提案されている adaptive ルーティング方式は Fat-tree IB ネットワークには適用できない。

各ノード間に複数パスを形成、ネットワークトラフィックを複数パスに分散させて HotSpot 発生の影響を低下させる方式も提案されている[3)4)]。しかし、この方式ではスイッチチップ内で HoL ブロッキングが発生して性能が低下するという問題がある[10)]。

シフト通信パターンに特化した Fat-tree 向け static ルーティング方式が提案されている[6)7)]。この方式は主要 SM である OpenSM に既に組み込まれ一般的に使われているが、本稿で示した通り、ネットワーク構成に対してノード割当てが非対称な場合には Hot-spot が発生する。

7. まとめ

本稿では、2 ステージ Fat-tree IB ネットワークにおける全対全通信時の Hot-spot 発生頻度を削減するルーティング手法を提案した。6-ary-2-tree 構成のネットワークに接続した 30 ノードの PC クラスタシステム上で提案手法を評価した結果、任意 16 ノードによる全対全通信性能は従来手法と比べて平均で 34%向上することが分かり、提案手法で Hot-spot

発生頻度を削減できることを示した。

提案方式は任意ノード割当て時の Hot-spot 発生頻度削減には有効だが、Hot-spot 発生を完全に回避することはできず、また、連番ノード割当て時の Hot-spot 発生頻度を削減することはできない。今後は、より包括的な Hot-spot 発生頻度削減手法を検討する予定である。

参 考 文 献

1) F. Petrini, and M. Vanneschi: k-ary n-trees: High Performance Networks for Massively Parallel Architectures, In *Proceedings of the 11th International Parallel Processing Symposium (IPPS97)*, 1997.
2) InfiniBand Trade Association. InfiniBand Architecture Specification, Release 1.2, October 2004.
3) X. Lin, Y. Chung, and T. Huang: A Multiple LID Routing Scheme for Fat-tree-Based InfiniBand Networks, In *Proceedings of the 18th IEEE International Parallel and Distributed Processing Symposium*, 2004.
4) A. Vishnu, M. Koop, A. Moody, A.R. Mamidala, S. Narravula, and D.K. Panda: Hot-Spot Avoidance With Multi-Pathing Over InfiniBand: An MPI Perspective, In *Proceedings of the 7th IEEE Inernational Symposium on Cluster Computing and the Grid*, 2007.
5) M. Karol, M. Hluchyj, and S. Morgan: Input Versus Output Queueing on a Space-Division Packet Switch, *IEEE Transactions on Communications*, Volume 35, Issue 12, pp. 1347-1356, 1987.
6) C. Gomez, F. Gilabert, M.E. Gomez, P. Lopez, and J. Duato: Deterministic versus Adaptive Routing in Fat-trees, In *Proceedings of the 2007 IEEE International Parallel and Distributed Processing Symposium (IPDPS07)*, 2007
7) E. Zahavi, G. Johnson, D.J. Kerbyson, and M. Lang: Optimized infiniband fat-tree routing for shift all-to-all communication patterns, In *Proceedings of the International Supercomputing Conference 2007 (ISC07)*, 2007.
8) The OpenFabrics Alliance: http://www.openfabrics.org/
9) OpenMPI: http://www.open-mpi.org/
10) T. Hoefler, T. Schneider, and A. Lumsdaine: Multistage switches are not crossbars: Effects of static routing in high-performance networks, In *Proceedings of the 2008 IEEE International Conference on Cluster Computing (Cluster08)*, 2008.
11) A. Faraj, and X. Yuan: Automatic generation and tuning of MPI collective communication routines, In *Proceedings of the 19th Annual international Conference on Supercomputing (SC05)*, 2005.
12) K.J. Barker, A. Benner, R. Hoare, A. Hoisie, A.K. Jones, D.K. Kerbyson, D. Li, R. Melhem, R. Rajamony, E. Schenfeld, S. Shao, C. Stunkel, and P. Walker: On the Feasibility of Optical Circuit Switching for High Performance Computing Systems, In *Proceedings of the 2005 ACM/IEEE conference on Supercomputing (SC05)*, 2005.
13) 成瀬 彰, 中島 耕太, 住元 真司, 久門 耕一: マルチコア PC クラスタ向け All-to-all 通信アルゴリズムの提案と評価, In *Proceedings of SACSIS 2010*, 2010.
14) Z. Ding, R.R. Hoare, A.K. Jones, and R. Melhem: Level-wise scheduling algorithm for fat tree interconnection networks, In *Procddings of the 2006 ACM/IEEE conference on Supercomputing (SC06)*, 2006.
15) P. Geoffray, and T. Hoefler: Adaptive Routing Strategies for Modern High Performance Networks, In *Proceedings of the 16th IEEE Symposium on High Performance Interconnects*, 2008.
16) G. Pfister, M. Gusat, W. Denzel, D. Craddock, N. Ni, W. Rooney, T. Engbersen, R. Luijten, R. Krishnamurthy, and J. Duato: Solving Hot Spot Contention Using InfiniBand Architecture Congestion Control, In *Proceedings of HP-IPC 2005*.