# 低コストで大規模化可能なラテン方陣Fat-Treeにおける All-to-all通信の高速化の実現

清水 俊宏[1]　中島耕太[1]

**概要**：近年，クラスタシステムは大規模化の傾向にあり，より多くのサーバを接続して並列に処理するシステムが増えてきた．サーバを接続する際のトポロジー構造としては Fat-Tree が広く用いられている．サーバ間通信の中継に用いられるスイッチは高価であるため，出来るだけスイッチ数を削減したいという要求がある．Fat-Tree でサーバを接続すると，サーバ間の経路が複数あることからネットワーク性能は高いものの，多くのスイッチを必要とするため，コストも高くなるという問題がある．このコストの問題を解消するため，ラテン方陣 Fat-Tree という従来の Fat-Tree と比較して少ないスイッチで多くのサーバを接続可能なトポロジーが提案されている．このトポロジーは，サーバ接続のためのスイッチ台数が少なくて済む一方で，サーバ間の経路が 1 通りしかなく，All-to-all 通信などの高負荷な通信処理を行うと経路競合が原因で性能低下が生じる．
本稿ではラテン方陣 Fat-Tree が有限射影平面に基づいて構成されていることに注目し，All-to-all 通信の経路競合を回避する手法を提案する．実用的なクラスタシステムの運用においては，一部のサーバ群に限定してジョブを投入することも多いため，一部のサーバ群のみの間での All-to-all 通信の経路競合を回避する方法も提案する．この経路競合回避手法によって高速化を実現する．
さらに，シミュレーションおよび実機の両方で実験し，提案手法の実用性を検証する．その結果，大規模化されたラテン方陣 Fat-Tree でも Fat-Tree と同等の性能が出ることが実証できた．実機での評価の結果，最大で 2.7 倍の高速化を実現し，シミュレーションでも同様の結果を得た．

**キーワード**：トポロジー，ラテン方陣 Fat-Tree，All-to-all 通信，競合回避

# Acceleration of All-to-all Communication in Latin Square Fat-Tree, Low Cost Scalable Network Topology.

Toshihiro Shimizu[1]　Kohta Nakashima[1]

***Abstract:*** Recently, the scale of the cluster systems becomes larger and many of these cluster systems have many servers connected by switches. Fat-Tree is one of the major topology for cluster system. Since the cost of the switch which is used to connect servers is high, to reduce the number of switches becomes more important. Since Fat-Tree has multiple paths between two servers, we can attain high network performance using Fat-Tree. However, many switches are required and hence the cost will rise. Latin Square Fat-Tree was proposed as a topology which enables to connect more servers than normal Fat-Tree. Using the topology, we can reduce the number of switches. However, since there is only one route between each two servers, it tends to cause link congestion during high-load network operation such as All-to-all communication. Thus, the topology is not used practically.
In this paper, we focus on the fact that Latin Square Fat-Tree is based on the finite projective plane to invent a method of All-to-all communication avoiding link congestion. Since, in practical cluster system, we submit a job to a server group which has some selected servers in the cluster system, we propose a method of All-to-all communication in a server group (consists of all servers or partial servers) avoiding link congestion. We also made experiments by both simulation and actual servers and demonstrated practicality of our method. As a result, even in the case of large-scale Latin Square Fat-Tree, we get a prospect of equivalent performances as normal Fat-Tree. Moreover, we achieved 2.7 times acceleration of throughput from the experiment of actual servers. This result is similar to the simulation.

***Keywords:*** topology, Lagin Square Fat-Tree, All-to-all communication, avoid link congestion

## 1. はじめに

近年は，計算科学の進歩が著しく，科学技術計算の高速化・大規模化の要求が高まっている．クラスタシステムの大規模化のために，多くの計算サーバを接続して並列に処理する技術が求められているが，並列化した場合の性能は1台での性能に比べるとサーバ間通信分の処理時間がかかるため，この処理時間を削減する機構が必要となってくる．その一つのアプローチとしてサーバ同士をどのようにつなぐか，そのトポロジー構造に着目して性能向上を図る方法がある．同一のリンク上で同じ方向に複数の通信が通過した場合，経路競合が起こり通信に遅延が生じ，性能低下の原因となる．よって，経路競合の少ないトポロジー構造及び通信手段が必要となる．また，サーバ間の中継にはスイッチを用いるが，スイッチはサーバと比べて高価であるため，より少ないスイッチで多くのサーバを接続することが望ましい．

現在のクラスタシステムのトポロジー構造はFat-Treeが広く採用されている．Fat-Treeを用いることで，各サーバから他のすべてのサーバに通信するAll-to-all通信という非常に負荷の高い通信処理を競合なく実行できる．しかし，Fat-Treeにはサーバ間の経路に冗長性があり，段数を固定したとき，接続可能なサーバ数が少ないという問題がある．したがって，より多くのサーバをつなぐにはFat-Treeの段数を高くしなくてはならず，1スイッチあたりのサーバ台数が増えてしまい効率が悪い．そこで，Fat-Treeよりもさらに少ないスイッチ数でより多くのサーバを接続できるトポロジー構造が求められている．

多層full-mesh[1]やラテン方陣Fat-Tree[2]は従来のFat-Treeより多くのサーバを接続することを目的として提案されたトポロジー構造である．ラテン方陣Fat-Treeは従来のFat-Treeや多層full-meshに比べてスイッチのコスト面での効率はよいが，その分サーバ間の結合が疎になっているために，All-to-all通信といった高負荷な通信を競合なく行う手法は確立されていない．ラテン方陣Fat-Treeで経路競合を避けることができれば，コストを低く抑えられるため有用である．

また，実用的なクラスタシステムの運用を考えるとクラスタシステム全体に一つの計算ジョブを割り当てる運用と共に，一部のサーバ群のみに限定してジョブを割り当てる運用も考えられる．そこで，本稿ではラテン方陣Fat-Treeにおける経路競合のないAll-to-all通信の手法として以下の2つを提案し，評価する．

(1) 全体でのAll-to-all通信
(2) 一部のサーバ群のみを使用したAll-to-all通信およびそのサーバ群の選択方法

[1] (株) 富士通研究所

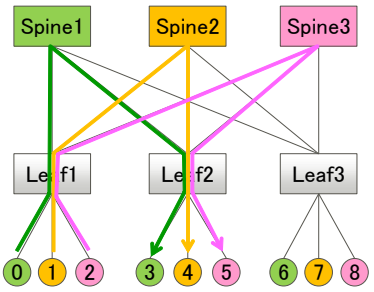


**図 1** Fat-Treeによるトポロジー構造

手法(1)はラテン方陣Fat-Treeのleafスイッチ間の経路の一意性に注目し，手法(2)はラテン方陣Fat-Treeの元となっている有限射影平面上で異なる方向に通信を行うことで経路競合を回避するものである．なお，既に我々は文献[3]にて，手法(2)のみ提案している．本稿はこれらの手法の提案に加えて実機での評価を追加したものである．

実機での評価の結果，従来のFat-Treeで用いられるシフト通信パターンの代わりに提案手法を採用することで2.7倍のスループット向上を達成した．

また，シミュレーションでの評価の結果，ラテン方陣Fat-Tree上で従来のシフト通信パターンを採用した場合，大規模になるほどより競合が発生しやすくスループットが低下していくことが分かった．このため提案手法の効果は大規模になるほど増大していくことが明らかになった．

## 2. 従来手法と課題

### 2.1 従来のFat-Treeにおける All-to-all 通信

#### 2.1.1 従来のFat-Tree

多くのクラスタシステムではInfiniBand[4]による図1のようなFat-Treeによるトポロジー構造が採用されている．

Fat-Treeはどの2サーバ間の経路も次数分存在する密なトポロジー構造であるため，通信負荷の高いAll-to-all通信を経路競合なく行うことができる．All-to-all通信の性能を向上させることで，All-to-all通信を内部で複数回実行する高速フーリエ変換(FFT)などの処理を高速かつ並列に処理することができる[5]．

図1の上段のスイッチをspineスイッチ，中段のスイッチをleafスイッチと呼ぶ．各スイッチが他のスイッチと接続している部分のポート数を次数と呼ぶ．図1のスイッチの次数は3である．

なお，図1では各spineスイッチの片側のみトポロジーを構築しており，ポートが半分余っている．そこで，spineスイッチの反対側にも同様の構造を構築することで，サーバ台数を2倍にすることが可能である．本稿では簡単のため，図1のようにspineスイッチの片側のみを構築したトポロジーを考える．この片側のみのトポロジー構造で競合のないAll-to-all通信が達成された場合，対称性により台数を2倍にして両側に構築したトポロジー構造でも容易に

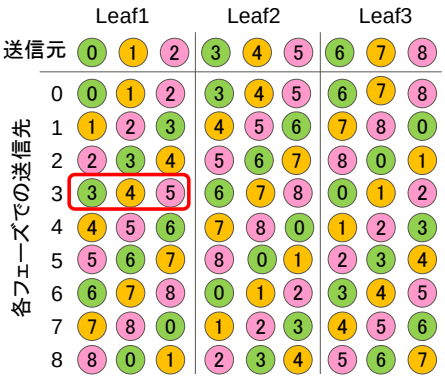


**図 2** シフト通信パターンの例

競合のない All-to-all 通信が達成できる．

### 2.1.2 Fat-Tree における All-to-all 通信の競合回避

Fat-Tree 上で All-to-all 通信を行う場合，シフト通信パターンを採用することで競合を回避することが知られている．シフト通信パターンとは，$i$ 番目のフェーズでサーバ番号 $S$ のサーバは $(S+i)\%N$ に送信する，という通信パターンである [6][7]．ただし，$N$ はサーバ台数である．例えば，図 1 のような Fat-Tree のすべてのサーバを用いたシフト通信パターンでの All-to-all 通信は図 2 のようになる．図 2 において，第 3 フェーズでサーバ 0,1,2 はそれぞれサーバ 3,4,5 に送信することを表しており，図 1 の経路で送信することで，競合を回避できる．

### 2.1.3 実用的な運用を考えた All-to-all 通信

実用的には，常にクラスタシステムのすべてのサーバを利用した大規模な All-to-all 通信を行うことはなく，一部のサーバ群にジョブを割り当て，それらの間で All-to-all 通信を実行することも頻繁に行われる．例えば，前項の Fat-Tree の，一部のサーバ群で All-to-all 通信を行う場合，使用するサーバ群を leaf スイッチ単位で選択することで同様にシフト通信パターンによる競合のない All-to-all 通信が可能である．

## 2.2 ラテン方陣 Fat-Tree

クラスタシステムの大規模化に伴い，少ないスイッチ数でより多くのサーバを並列に処理するため，従来の Fat-Tree よりも多くのサーバを接続することが可能なラテン方陣 Fat-Tree というトポロジー構造が提案されている．ラテン方陣 Fat-Tree の構造を説明するために，その元となっている有限射影平面 [8] の構造を述べる．

### 2.2.1 有限射影平面の定義

位数 n(n は素数) の有限射影平面は，以下で定義される点と直線 (点の集合) の集合からなる．点は以下の名前を持つ $n^2+n+1$ 個の集合である．

- $P, P(c)(0 \le c < n), P(c,r)(0 \le c,r < n)$

直線および直線上の点集合は以下の $n^2+n+1$ 本とする．

- $L = \{P\} \cup \{P(c) \mid 0 \le c < n\}$

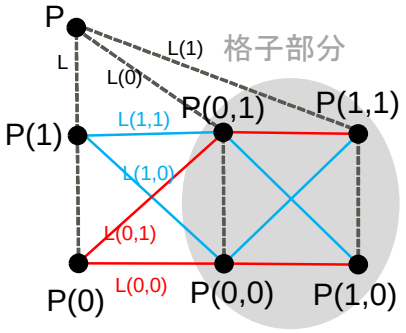


**図 3** 位数 2 の有限射影平面

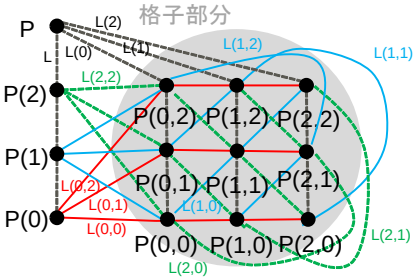


**図 4** 位数 3 の有限射影平面

- $L(c) = \{P\} \cup \{P(c,i) \mid 0 \le i < n\}(0 \le c < n)$
- $L(c,r) = \{P(c)\} \cup \{P(i,(r+ci)\%n) \mid 0 \le i < n\}(0 \le r,c < n)$

$P(c,r)(0 \le c,r < n)$ の部分を格子部分と定義する．位数 2 の有限射影平面の構造を図 3 に，位数 3 の場合の有限射影平面の構造を図 4 に示す．位数 $n$ の有限射影平面はサイズ $n$ の直交ラテン方陣 $n-1$ 枚と等価であることが知られている [2].

### 2.2.2 有限射影平面とトポロジーの対応

ラテン方陣 Fat-Tree のトポロジー構造は位数 $n$ の有限射影平面に対して以下の変換を施すことで得られる [2].

- 各直線 $\ell$ と同名の spine スイッチ $\ell$, 各点 $p$ と同名の leaf スイッチ $p$ を用意し，対応させる．
- 直線 $\ell$ 上に点 $p$ がある場合，対応する spine スイッチ $\ell$ と leaf スイッチ $p$ をリンクで結ぶ
- 各 leaf スイッチと $n+1$ 台のサーバをリンクで結ぶ．

図 5 上のように 1 本の直線とその直線上の点は spine スイッチとそれに接続された leaf スイッチに対応する．図 5 下のように有限射影平面のすべての点と直線に対して同様の操作を行うとトポロジー構造が完成する．図 3，図 4 の有限射影平面はそれぞれ図 6，図 7 のラテン方陣 Fat-Tree に対応する．接続サーバ台数は $(n+1)(n^2+n+1)$ 台で，従来の Fat-Tree の $(n+1)^2$ 台と比べて格段に多い．位数 $n$ の有限射影平面に対応するラテン方陣 Fat-Tree の次数は $n+1$ となる．有限射影平面の性質はラテン方陣 Fat-Tree の性質に置き換えることが可能である．例えば，有限射影平面ではどの異なる 2 点についてもその 2 点を通る直線

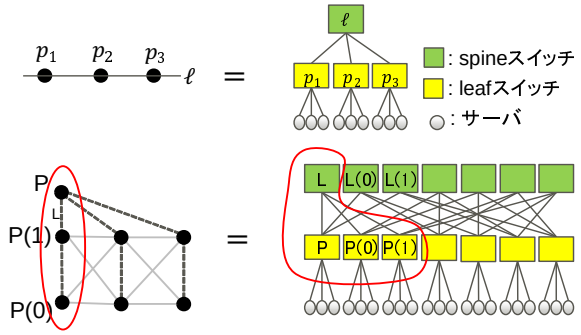


**図 5** 有限射影平面とトポロジーの対応

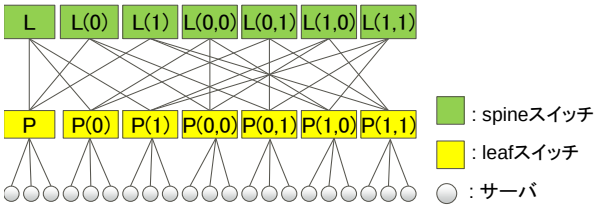


**図 6** 位数 2(次数 3) のラテン方陣 Fat-Tree

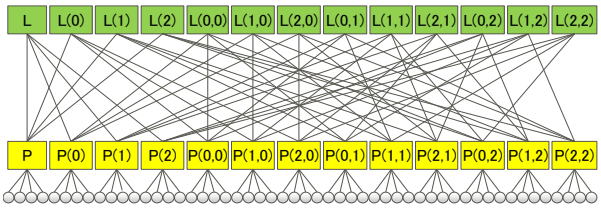


**図 7** 位数 3(次数 4) のラテン方陣 Fat-Tree

がただ 1 本あるという性質がある．この性質をラテン方陣 Fat-Tree に置き換えると，どの異なる 2 つの leaf スイッチ間においても spine スイッチを経由する通信経路がただ 1 本存在する，という性質となる．

### 2.3 課題

ラテン方陣 Fat-Tree は少ないスイッチで多くのサーバを接続できるというメリットがあるが，どの 2 サーバについても最小ホップ数で到達できる経路数が少なく (例外を除きほぼ 1 本のみ)，トポロジー構造が疎である．このため，競合のない All-to-all 通信が可能かどうかも，知られていなかった．このような理由により，ラテン方陣 Fat-Tree は実用的なクラスタシステムの接続方式としても広く用いられていない．本稿では，ラテン方陣 Fat-Tree でも競合のない All-to-all 通信が可能であることを示し，一部のサーバのみを用いた競合のない All-to-all 通信も可能であることを示す．

## 3. 提案手法

本章では，全台を用いた All-to-all 通信と一部のサーバ群のみを使用した All-to-all 通信の手法をそれぞれ別の手法として提案する．

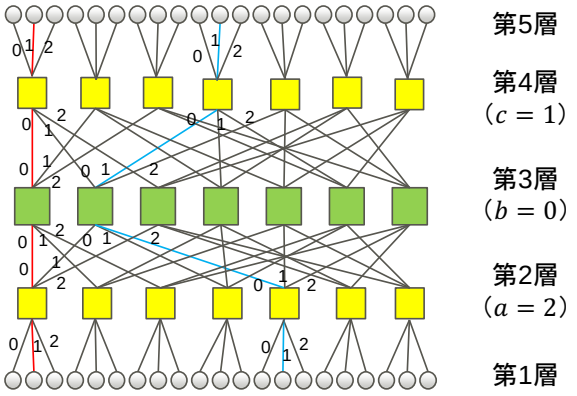


**図 8** spine スイッチで折り返したトポロジー構造 ($n = 2$)

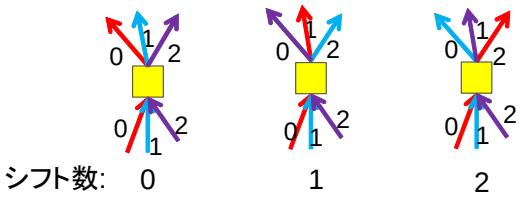


**図 9** スイッチにおけるシフト数ごとの転送パターン

### 3.1 全台を用いた All-to-all 通信

本節では，ラテン方陣 Fat-Tree 上の全サーバを用いた All-to-all 通信の手法を提案する．この手法は文献 [9] の手法を参考に，ラテン方陣 Fat-Tree での方式に拡張した．

#### 3.1.1 シフト数のトリプルと通信パターンの対応

まず，図 8 のように spine スイッチ以降の復路となる経路を上側に折り返し，第 1 層のサーバから第 5 層のサーバへの All-to-all 通信として考える．

シフト通信パターンでは，全サーバに対して，宛先を 1 フェーズごとにシフトさせて All-to-all 通信を実現させていたが，本手法では，シフトを各スイッチ内で行う．各スイッチに対して，下位のリンクと繋がるポートを入力ポート，上位のリンクと繋がるポートを出力ポートとしておき，それぞれ 0 から $n$ までのポート番号を付与する．ただし，spine スイッチは上下対称であるが，その対応するリンク同士は同じポート番号になるようにしておく．図 8 は $n = 2$ の場合のポート番号付与例である．

スイッチ内で通信を転送する際，入力ポートと出力ポートのポート番号の差をシフト数と呼ぶ．すなわち，ある通信が番号 $x$ の入力ポートから入り，それを番号 $y$ の出力ポートへ転送した場合，そのシフト数は $(y - x)\%(n + 1)$ と定義する．例えば，$n = 2$(次数 3) の場合におけるスイッチの転送パターンは図 9 のようになる．

各フェーズでシフト数は層ごとに共通の値としておく．すなわち，同じフェーズの同じ層に属するすべてのスイッチは共通のシフト数で転送すると決めておく．すると，第 2 層のシフト数を $a$, 第 3 層のシフト数を $b$, 第 4 層のシフ

ト数を $c$ として，トリプル $(a,b,c)$ の値と各サーバの送信元から宛先までの送信経路が対応する．例えば図 8 の赤，青の線はいずれも $(a,b,c)=(2,0,1)$ における 2 つのサーバペア間の通信を表している．赤い線については第 2 層でポート 1 から入り，ポート $0(=(1+a)\%3)$ へ，第 2 層でポート 0 から入り，ポート $0(=(0+b)\%3)$ へ，第 3 層でポート 0 から入り，ポート $1(=(0+c)\%3)$ へ転送される．他のサーバについても同様に経路を決定することで宛先が決定する．ここで，第 2 層と第 4 層は実体としては同じスイッチであるが，転送の向きが異なっており，向きによって別のシフト数が割り当てられていることに注意しておく．

#### 3.1.2 全台での All-to-all 通信

次に，全台での All-to-all 通信を実現するためにどのトリプルに対応する通信を実行するかを決める．ここで，$0\le a,b,c\le n$ であることから，考えられるトリプル $(a,b,c)$ は $(n+1)^3$ 通りであり，サーバ台数を上回る．よって，すべてのトリプルに対応する通信を実行してしまうと，重複が発生する．そこで，以下の条件をみたすトリプルのみをリストアップして実行する．

**条件:** $b\neq 0$ or $(b=0$ and $c=0)$

このときのフェーズ数，すなわち実行するトリプル数を数える．$b\neq 0$ の場合は $n(n+1)^2$ 個，$(b=0$ and $c=0)$ の場合は $n+1$ 個であることから，合計 $(n+1)(n^2+n+1)$ フェーズであり，サーバ台数と一致する．

#### 3.1.3 All-to-all 通信が競合なく行われることの証明

実際にこの条件に従って通信を行った場合，競合なく All-to-all 通信が行われることを証明する．

まず，各スイッチは $n+1$ 本の通信をそれぞれ別の下位リンクから受け取りそれを別の上位リンクへ転送しているので，競合が起こらないことは明らかである．

異なる leaf スイッチ配下のサーバ間での通信の場合，ラテン方陣 Fat-Tree の構造上，通信経路がただ一本存在するので，これに基づいて自動的に $(a,b,c)$ が定まる．このとき，spine スイッチへの入力元と出力先の leaf スイッチが異なるため，ポート番号も異なり，$b\neq 0$ となる．よって，このトリプル $(a,b,c)$ は上記のリストに入っている．

同一 leaf スイッチ配下のサーバ間の通信の場合，通信経路は複数あるが，どの経路を通過しても $b=0$ となる．このうち，$c=0$ となるためには第 4 層について入力ポートと同じ番号の出力ポートへ転送されなくてはならず，この条件をみたす経路がただ 1 本存在する．この経路における第 2 層でのシフト数から $a$ の値も決定する．このようにして得られたトリプル $(a,b,c)$ は上記のリストに入っている．

よって，提案手法で競合なく All-to-all 通信されることが証明された．

### 3.2 一部のサーバ群のみを使用した All-to-all 通信

本節では，必要なサーバ台数に応じて柔軟に All-to-all 通信を行うため，ラテン方陣 Fat-Tree の一部のサーバ群のみを使用して競合なく All-to-all 通信する手法を述べる．

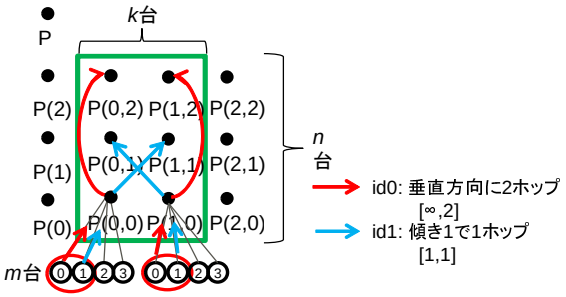


**図 10** leaf スイッチの選択と送信方向

**表 1** フェーズ群内の転送手順 (左: $m=2$, 右: $m=4$)

| 通信番号 \ フェーズ | 0 | 1 |
|---|---|---|
| 0 | 0 | 1 |
| 1 | 1 | 0 |

| 通信番号 \ フェーズ | 0 | 1 | 2 | 3 |
|---|---|---|---|---|
| 0 | 0 | 1 | 2 | 3 |
| 1 | 1 | 2 | 3 | 0 |
| 2 | 2 | 3 | 0 | 1 |
| 3 | 3 | 0 | 1 | 2 |

#### 3.2.1 使用するサーバ群の決定方法

まず，投入するジョブのサーバ群を選択する方式について説明する．ラテン方陣 Fat-Tree に対応する有限射影平面の格子部分に注目し，格子部分 ($n\times n$ の形状) の $n\times k$($k$ は任意の自然数) の長方形を選ぶ．選んだ点に対応する leaf スイッチそれぞれに対して，配下のサーバのうち $m$ 台 ($m$ は $m\le k$ をみたす自然数) を選択する．したがって，合計 $nkm$ 台が選択される (図 10 参照，緑枠の長方形に属する leaf スイッチ配下のサーバを $m$ 台ずつ選ぶ)．本手法では，$m\le k\le n$ なる自然数 $n,k$ を用いて $D=nkm$ と表せるような $D$ 台を用いて All-to-all 通信が可能である．

なお，以降の説明のため，各 leaf スイッチ配下のサーバ $m$ 台に $0,1,2,\ldots,m-1$ という leaf 配下内 ID を割り当てる．

#### 3.2.2 フェーズ群内の通信

フェーズ数はサーバ台数と同じく，$nkm$ フェーズとなる．この $nkm$ フェーズを $m$ フェーズごとに $nk$ 個のフェーズ群に分ける．同一のフェーズ群で各サーバは同じ宛先 leaf スイッチに送信する．ただし，この宛先 leaf スイッチ配下の宛先サーバは次のようにして，フェーズ群内のフェーズごとにそれぞれ異なるようにする．

宛先 leaf スイッチは各フェーズで $m$ 台からの通信を受信する．本節の手法では，この $m$ 本の通信の送信元 leaf 配下内 ID は異なっており，その leaf 配下内 ID に応じて $0,1,\ldots,m-1$ と通信番号をつけておく．フェーズ群内の $i$ 番目のフェーズにおいて，通信番号が $j$ である通信は，leaf スイッチ配下内 ID が $(i+j)\%m$ であるサーバに送るものとする．$m=2,4$ の場合の具体例は表 1 のようになる．

### 3.2.3 各フェーズ群に対応する送信先 leaf スイッチの決定

格子状に並べた各 leaf スイッチについて，leaf スイッチ間の移動を「ベクトル」として表現する．このベクトルが右に $x$ マス，上に $y$ マス移動しているとき，$(x, y)$ と成分表示する．次に，ベクトルの傾きを定義する．成分表示が $(0, 0)$ であるベクトルをゼロベクトルと呼ぶ．成分表示が $(x, y)$ である (ゼロでない) ベクトルの傾きを，$x \not\equiv 0 \pmod n$ であるときは $yx^{-1}\%n$ と，$x \equiv 0 \pmod n$ であるときは $\infty$ と定義する．ただし，$x^{-1}$ は $\pmod n$ における $x$ の乗法の逆元であり，$xx^{-1} \equiv x^{-1}x \equiv 1 \pmod n$ をみたす．例えば，$n = 3$ の場合において $\pmod 3$ で考えると，ベクトル $(1, 2)$ の傾きは $2 \times 1^{-1} = 2$ である．ここで，$2 \times 2 \equiv 2 \times 2 \equiv 1$ であることから，$\pmod 3$ における 2 の逆元は 2 であるので，ベクトル $(-1, 2)$ の傾きは $2 \times (-1)^{-1} \equiv 2 \times 2^{-1} = 1$ であり，ベクトル $(2, 1)$ の傾きは $1 \times 2^{-1} = 2$ である．また，ベクトル $(0, 2)$ の傾きは $\infty$ である．

次に，ベクトルのホップ数を定義する．成分表示が $(x, y)$ であるベクトルのホップ数は，$x \not\equiv 0 \pmod k$ である場合は $x\%k$，$x \equiv 0 \pmod k, y \not\equiv 0 \pmod n$ である場合は $y\%n$, ゼロベクトルのホップ数は 0 と定義する．傾き $s$, ホップ数 $h$ のベクトルを $[s, h]$ と表す．また，ゼロベクトルを $[*]$ と表す．

格子上の点 $(a_x, a_y)$ とベクトル $[s, h]$ によって，移る点 $(b_x, b_y)$ を次のように定義する．$s \neq \infty$ であるときは，$b_x = (a_x + h)\%k, b_y = (s(b_x - a_x) + a_y)\%n$ とする．これは，$(a_x, a_y)$ を通る傾き $s$ の直線上の点のうち，$x$ 座標が $(a_x + h)\%k$ である点を表している．$s = \infty$ であるときは，$(b_x, b_y) = (a_x, (b_y + h)\%n)$ とする．これは，$(a_x, a_y)$ から上に $h$ 個移動した点である．例えば, 点 $(1, 2)$ をベクトル $[1, 1]$ によって移すと, $b_x = (1+1)\%2 = 0, b_y = 1 \times (0-1)+2 = 1$ より，$(0, 1)$ に移る．なお，どの点もゼロベクトル $[*]$ によって同一の点に移るものとする．どのベクトル $v$ についても，選択した長方形部分の各 leaf スイッチを $v$ によって同時に移したとき，それぞれ別々の点に移る．

上記の傾きの定義によって，ゼロベクトルを除く各ベクトルの傾きは 0 以上 $n-1$ 以下の整数または $\infty$ で表される．各 $s = 0, 1, \ldots, n-1$ について，傾きが $s$ であるベクトルは $k-1$ 本存在し，傾きが $\infty$ であるベクトルは $n-1$ 本，ゼロベクトルが 1 本存在し，これらを合計すると $nk$ 本となる．

各フェーズ群で leaf 配下内 ID ごとに共通のベクトルを指定しておき, leaf スイッチ間の移動はこのベクトルに従って行うものとする．これによって，同一の leaf 配下内 ID のサーバ同士が競合することはない．また，同一の leaf スイッチ配下の全サーバはそれぞれ傾きの異なるベクトルを指定することで経路競合を回避することができる．

**表 2** $(n, k, m) = (3, 2, 2)$ の場合のベクトル表

| | | leaf 配下内 ID 0 | leaf 配下内 ID 1 |
|---|---|---|---|
| フェーズ群 | 0 | $[\infty, 1]$ | $[2, 1]$ |
| | 1 | $[\infty, 2]$ | $[*]$ |
| | 2 | $[0, 1]$ | $[\infty, 1]$ |
| | 3 | $[1, 1]$ | $[\infty, 2]$ |
| | 4 | $[2, 1]$ | $[0, 1]$ |
| | 5 | $[*]$ | $[1, 1]$ |

例えば，図 10 のように leaf スイッチ $P(0, 0)$ 配下の leaf 配下内 ID 0 番のサーバと leaf スイッチ $P(1, 0)$ 配下の leaf 配下内 ID 0 番のサーバは同一のベクトル $[\infty, 2]$(赤いベクトル) を指定することで，経路競合は発生しない．また，leaf スイッチ $P(0, 0)$ 配下の leaf 配下内 ID 0 番と 1 番のサーバにおいて，別々の傾きであるベクトル $[\infty, 2]$(赤), $[1, 1]$(青) に送ると，同一の spine スイッチを共有することがないため，経路競合は発生しない．

以上をまとめると $nk$ 個のフェーズ群について，leaf 配下内 ID ごとに $m$ 本のベクトルを指定することになるが，その制約は以下のようになる．

(1) どのフェーズ群においても，指定するベクトルの傾きはすべて異なる．
(2) 各 leaf 配下内 ID について，全 $nk$ フェーズ群で現れるベクトルは上記の $nk$ 本が網羅されている．

この制約をみたすような通信順序を決定する．

### 3.2.4 ベクトル表の作成方法

ここで，通信パターンをベクトル表で表現する．ベクトル表は，行方向にフェーズ群，列方向に送信元の leaf 配下内 ID を並べ，フェーズ群ごとに leaf スイッチ間を移動する際のベクトルを指定した表である．この表によって，送信元 leaf スイッチを指定したとき，宛先 leaf スイッチが定まる．

leaf 配下内 ID 0 の列 (最左列) には順に，傾きが $\infty$ のベクトル ($n-1$ 本), 各 $s = 0, 1, \ldots, n-1$ について，傾きが $s$ であるベクトル ($k-1$ 本ずつ) を並べ，最後にゼロベクトルを並べる．それ以降の列は左隣の列を垂直方向下向きに $n-1$ 個シフトした列として作成する．$(n, k, m) = (3, 2, 2)$ の場合は表 2 のようになる．

このようにして作成したベクトル表が実際に前節の制約をみたしていることを証明する．制約 (2) については自明であるため，制約 (1) を示す．各列において傾きが同一の区間は連続しており，長さは最長で $n-1$ である (傾き $\infty$ の場合)．よって，垂直方向下向きに $v (0 \leq v < nk)$ 個シフトしたとき，この区間が元の区間と重ならないための条件は，$n - 1 \leq v < nk - (n-1)$ となっていることである．任意の整数 $i, j (0 \leq i < j < m)$ について，第 $j$ 列は第 $i$ 列を $v = (j-i)(n-1)$ 個分垂直方向下向きにシフトしたものであるので，この $v$ が上記の不等式をみたすこ

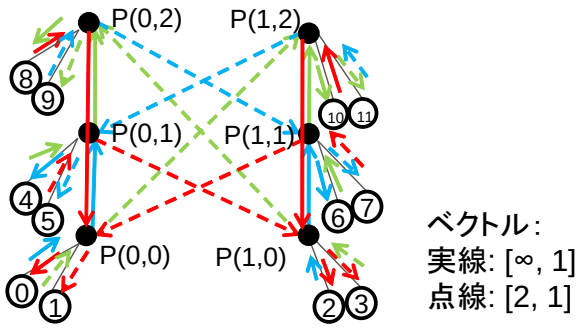


**図 11** 第 0 フェーズ群の 0 番目のフェーズの通信

**表 3** 各サーバの宛先

| フェーズ群 | フェーズと通し番号 | | サーバ ID | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
| 0 | 0 | 0 | 4 | 11 | 6 | 9 | 8 | 3 | 10 | 1 | 0 | 7 | 2 | 5 |
| | 1 | 1 | 5 | 10 | 7 | 8 | 9 | 2 | 11 | 0 | 1 | 6 | 3 | 4 |
| 1 | 0 | 2 | 8 | 1 | 10 | 3 | 0 | 5 | 2 | 7 | 4 | 9 | 6 | 11 |
| | 1 | 3 | 9 | 0 | 11 | 2 | 1 | 4 | 3 | 6 | 5 | 8 | 7 | 10 |
| 2 | 0 | 4 | 2 | 5 | 0 | 7 | 6 | 9 | 4 | 11 | 10 | 1 | 8 | 3 |
| | 1 | 5 | 3 | 4 | 1 | 6 | 7 | 8 | 5 | 10 | 11 | 0 | 9 | 2 |
| 3 | 0 | 6 | 6 | 9 | 4 | 11 | 10 | 1 | 8 | 3 | 2 | 5 | 0 | 7 |
| | 1 | 7 | 7 | 8 | 5 | 10 | 11 | 0 | 9 | 2 | 3 | 4 | 1 | 6 |
| 4 | 0 | 8 | 10 | 3 | 8 | 1 | 2 | 7 | 0 | 5 | 6 | 11 | 4 | 9 |
| | 1 | 9 | 11 | 2 | 9 | 0 | 3 | 6 | 1 | 4 | 7 | 10 | 5 | 8 |
| 5 | 0 | 10 | 0 | 7 | 2 | 5 | 4 | 11 | 6 | 9 | 8 | 3 | 10 | 1 |
| | 1 | 11 | 1 | 6 | 3 | 4 | 5 | 10 | 7 | 8 | 9 | 2 | 11 | 0 |

とを示す．$j - i \geq 1$ であることから左の不等号は成立する．右の不等号は $(j - i + 1)(n - 1) < nk$ と同値であり，$(j - i + 1)(n - 1) \leq m(n - 1) < kn$ より従う．

以上により，上記の制約をみたしていることがわかり，経路競合が起こらないことが証明された．

#### 3.2.5 各フェーズでの各サーバの宛先

3.2.4 項のベクトル表から leaf スイッチ間の転送方法が決まり，3.2.2 項より leaf 配下の転送方法が決まるため，各フェーズでの各サーバの宛先が決まる．例えば，第 0 フェーズ群の 0 番目のフェーズの通信経路は図 11 のようになる．なお，各サーバに通し番号をつけた．

0 番のサーバの通信経路 (青い実線の矢印) について説明する．表 2 のベクトル表によると，第 0 フェーズ群では leaf 配下内 ID 0 のサーバにはベクトル $[\infty, 1]$ が指定されているため，leaf スイッチ $P(0,0)$ を経由して leaf スイッチ $P(0,1)$ に到達する．ここで，表 1 によって，送信元 leaf 配下内 ID が 0 の通信 (通信番号 0) は宛先 leaf 配下内 ID も 0 に転送されるので，最終的にサーバ 4 に転送される．他のフェーズについても同様に経路と宛先を決定する．その結果を表 3 に示す．

**表 4** ラテン方陣 Fat-Tree と従来の Fat-Tree のパラメータ

| 構成 | 1 | 2 | 3 | 1’ | 2’ | 3’ |
|---|---|---|---|---|---|---|
| 次数 | 3 | 3 | 4 | 5 | 5 | 7 |
| 通信方式 | 全体 | 部分 | 部分 | - | - | - |
| $n$ | 2 | 2 | 3 | - | - | - |
| $k$ | - | 2 | 3 | - | - | - |
| $m$ | - | 2 | 3 | - | - | - |
| 選択サーバ数 | 21 | 8 | 27 | 25 | 10 | 28 |
| 全サーバ数 | 21 | 21 | 52 | 25 | 25 | 49 |

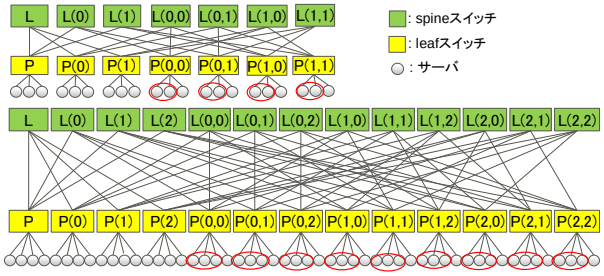


**図 12** ラテン方陣 Fat-Tree で用いるサーバ群

## 4. シミュレーションによる評価

### 4.1 実機との比較のためのシミュレーション

本節では実機での構成が可能な比較的小規模な台数でのシミュレーション評価を行う．なお，次章で本節と同じ構成で実機評価を行う．

#### 4.1.1 評価方法

トポロジー構造として 4.1.2 項で述べるラテン方陣 Fat-Tree の 3 種類と従来の Fat-Tree の 3 種類による合計 6 種類の構成を用い，それぞれ以下の通信パターンでシミュレーションを行い，そのスループット比を計算した．

- シフト通信パターン: 2.1.2 項を参照．ラテン方陣 Fat-Tree および従来の Fat-Tree で用いる．
- 提案手法: 3 章を参照．全台の通信は 3.1 節，一部のサーバを使用した通信は 3.2 節の方式を用いた．ラテン方陣 Fat-Tree でのみ実行．

#### 4.1.2 シミュレーションに用いるトポロジーの構成

シミュレーションでは 3 つの構成のラテン方陣 Fat-Tree(構成 1,2,3) と比較対象として従来の Fat-Tree(構成 1 ’ ,2 ’ ,3 ’ ) を用いる．これらのパラメータを表 4 に示す．サーバの全台数および All-to-all 通信に使用する台数が対応するラテン方陣 Fat-Tree に近くなるように設定した．

各構成のトポロジー構造と用いるサーバを図 12, 図 13 に示す．構成 1 では図 12 上の全サーバを，構成 2 では図 12 上の赤丸部分のサーバを，構成 3 では図 12 下の赤丸部分のサーバを使用する．また，構成 1 ’ では図 13 上の全サーバを，構成 2 ’ では図 13 上の赤丸部分のサーバを，構成 3 ’ では図 13 下の赤丸部分のサーバを使用する．

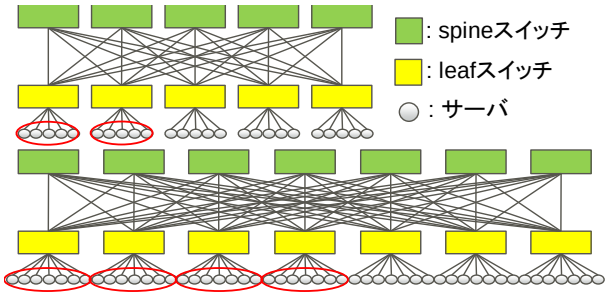


**図 13** 従来 Fat-Tree で用いるサーバ群

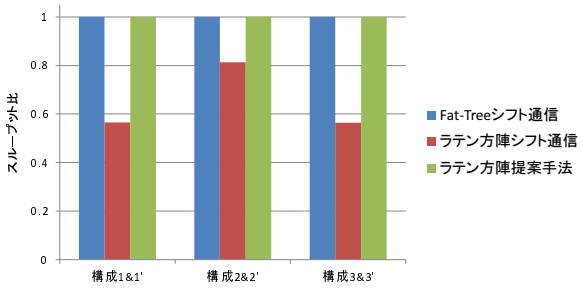


**図 14** 従来方式と提案方式のシミュレーション結果の比較

#### 4.1.3 スループット比の計算

本項では，スループット比の計算方法について述べる．スループット比は文献 [1] の評価で用いられている指標で，一本のリンクが単位時間あたりに通信できる情報量 (理論限界) を基準にしたとき，All-to-all 通信実行時に出ているスループットの割合を見積もっている．

各通信フェーズにおいて，各リンクを同時に通る通信の本数をそのリンクに対する経路競合数と定義する．経路競合数が 1 であれば，そのリンク上で経路競合が発生していないことを表している．各サーバ間通信において，途中で通るそれぞれのリンクの経路競合数の最大値を算出し，これを最大経路競合数と呼ぶ．この最大経路競合数の逆数を平均した数値を通信フェーズにおけるスループット比と定義し，このスループット比を全フェーズで平均した数値を All-to-all 通信時のスループット比と定義する．

#### 4.1.4 評価

スループット比のシミュレーション結果を図 14 に示す．

従来の Fat-Tree のシフト通信パターンの結果がスループット比 1.0 であり，競合は発生していない．これに対して，ラテン方陣 Fat-Tree でシフト通信パターンを採用してしまうと，スループット比が低下し，競合が起こっていることがわかる．提案手法では，スループット比が 1.0 であり，ラテン方陣 Fat-Tree でも競合なく All-to-all 通信ができていることが確認された．

### 4.2 大規模な場合のシミュレーション

本稿の手法は大規模を意図しているため，大規模な場合の効果を検証する必要がある．そこで，本節では実機では評価できない規模のシミュレーションを行う．最大で 5000 台程度のサーバを用いることを想定した，大規模なシミュレーションを実施した．シミュレーションには次節の表 6 に示す 27 台のサーバを利用した．

**表 5** 大規模なシミュレーションにおける構成 (ラテン方陣)

| | 構成 | 1 | 2 | 3 | 4 | 5 | 6 |
|---|---|---|---|---|---|---|---|
| 規模 | 次数 | 4 | 6 | 8 | 12 | 14 | 18 |
| | $n$ | 3 | 5 | 7 | 11 | 13 | 17 |
| | 全台数 | 52 | 186 | 456 | 1596 | 2562 | 5526 |
| 75% | $k$ | - | - | 7 | 11 | 13 | 16 |
| | $m$ | - | - | 7 | 10 | 11 | 16 |
| | 部分台数 | - | - | 343 | 1210 | 1859 | 4352 |
| | 比率 | - | - | 75.2 | 75.8 | 72.6 | 78.8 |
| 50% | $k$ | 3 | 5 | 6 | 9 | 10 | 13 |
| | $m$ | 3 | 4 | 6 | 9 | 10 | 13 |
| | 部分台数 | 27 | 100 | 252 | 891 | 1300 | 2873 |
| | 比率 | 51.9 | 53.8 | 55.3 | 55.8 | 50.7 | 52.0 |
| 25% | $k$ | 2 | 3 | 4 | 6 | 7 | 9 |
| | $m$ | 2 | 3 | 4 | 6 | 7 | 9 |
| | 部分台数 | 12 | 45 | 112 | 396 | 637 | 1377 |
| | 比率 | 23.1 | 24.2 | 24.6 | 24.8 | 24.9 | 24.9 |

#### 4.2.1 評価方法

4.1.1 項と同様に次項で述べるトポロジー構成ごとにシフト通信パターンと提案手法を実行しそのスループットを比較する．

#### 4.2.2 トポロジーの構成

ラテン方陣 Fat-Tree での規模ごとのトポロジー構成を表 5 に示す．次数として，位数 $n$ が素数となるようなものを定める．次に各構成に対して一部のサーバのみを用いた通信をするため，$k, m$ を定めた．このとき，用いるサーバ台数がそれぞれ全体の約 75%,50%,25%となるように 3 種類設定した．

なお，構成 1, 2 における 75%のサーバを用いた通信は，75%程度になる適切な $k, m$ の値が取れないため，省略した．

#### 4.2.3 評価

前項の構成で実験した結果を図 15 に示す．規模が大きくなるに従って，従来法であるシフト通信パターンではスループット比が減少していく一方で，提案手法はすべてスループット比 1.0 であるため，競合が全く発生していないことがわかる．

また，同規模であれば通信に用いるサーバ台数が多いほど競合が多く発生していることがわかる．これは，サーバ台数を増やすことで通信の密度が増加しているためであると考えられる．

Infiniband の現在の実装では 36 ポート (有限射影平面の位数 $n = 17$ に対応) が主流である．例えば，2016 年 4 月現在，Mellanox 社製の Infiniband スイッチは EDR 世代で 36 ポートである [10]．本シミュレーションによって，このスイッチを用いて $(n+1)(n^2+n+1) = 5526$ 台 (spine ス

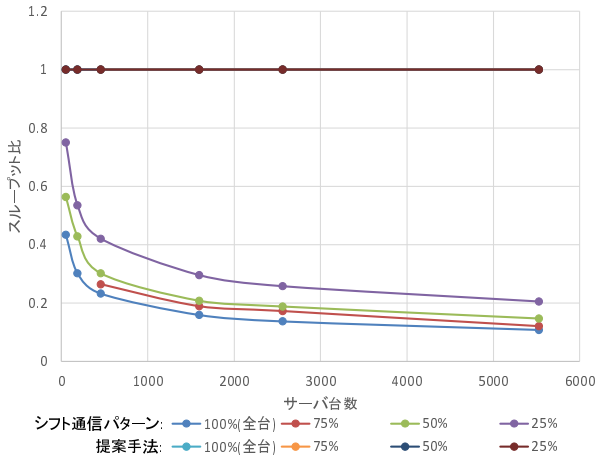


**図 15** 大規模なシミュレーションの実行結果

イッチの両側につなげば倍の 11052 台) のサーバが接続でき，競合のない All-to-all 通信ができるということが確認された．この場合，全台のシフト通信パターンのスループット比は 0.108 であるため，約 9.3 倍のスループット向上が見込める．

## 5. 実機での評価

### 5.1 評価方法

シミュレーションで挙げた構成 1, 2, 3 について，実機を用いてトポロジー構造を構築する．

評価は IMB(Intel MPI Benchmark) を用いた [11]．IMB の All-to-all では内部で MPI_Alltoall 関数を呼び出す．そのスループットを換算し，さらに理論限界とも比較する．評価に用いる MPI は OpenMPI 1.10.0 とした [12]．

OpenMPI の MPI_Alltoall 関数は宛先を決定するアルゴリズムとして shift(pairwise という名前で実装) と linear が既に用意されており，オプションとして指定できる．いずれも，シフト通信パターンでの All-to-all 通信を行うが，shift がフェーズごとに同期を取っているのに対して，linear はフェーズごとの同期は取らずに前のフェーズの通信が完了するとすぐに次のフェーズを開始する．これらに加えて提案手法であるラテン方陣 Fat-Tree での宛先決定アルゴリズムを新たに実装した．測定方法は 1 バイトから $2^{28}$ バイト (=256MiB) までの 2 のべき乗のメッセージサイズを指定して，それぞれ 100 回ずつ All-to-all 通信を行い，その所要時間の平均値から，サーバ 1 台が単位時間に送信したメッセージサイズ (スループット) を算出する．実験に用いたサーバ，ソフトウェア，InfiniBand の諸元をそれぞれ表 6，表 7，表 8 に示す．

なお，構成 3 のトポロジー構造を構築する際，leaf スイッチの格子部分以外と有限射影平面上の直線 L に対応する spine スイッチを省略した．いずれの構成も FDR と QDR が混在するものの，すべてのリンクが QDR としてリンクアップされる．

**表 6** サーバの諸元

| | |
|---|---|
| サーバ | RX100S8 |
| CPU | Intel(R) Xeon(R) CPU E3-1230 v3 @ 3.30GHz |
| コア数 | 8 コア |
| メモリ | 1600MHz, 15.4GiB |

**表 7** ソフトウェアの諸元

| | |
|---|---|
| OS | Cent OS 7.2 |
| Linux Kernel | 3.10.0-327.el7.x86_64 |
| MPI | OpenMPI 1.10.0 |
| ベンチマーク | Intel MPI Benchmark(IMB) 4.1 |

**表 8** Infiniband の諸元

| | |
|---|---|
| Spine スイッチ | SX6005 (FDR X4 12 ポート) |
| Leaf スイッチ | IS5022 (QDR X4 8 ポート) |
| HCA | Connect X3 (Single Port FDR) |

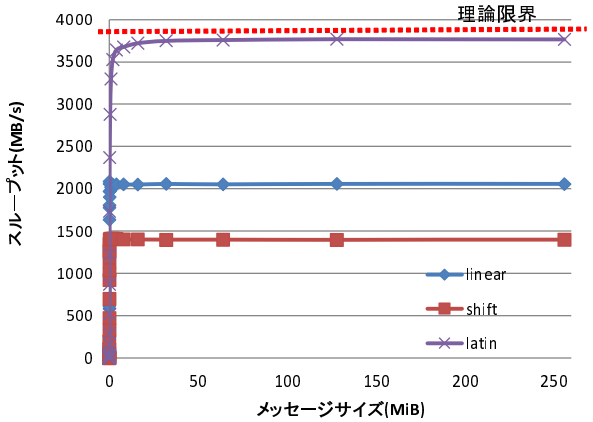


**図 16** 構成 1 の実験でのスループット実測値

### 5.2 評価

QDR 4X の最大スループットは 4000MB/s であるが，事前に測定した IMB における PingPong 通信のスループットは 3717MiB/s であった．本評価ではこの値を理論限界として用いる．転送したメッセージサイズを横軸に，サーバ 1 台あたりのスループットを縦軸にグラフをプロットした結果を構成ごとにそれぞれ図 16，図 17，図 18 に示す．シフト通信パターンでのスループットはそれぞれ 1400MiB/s, 2700MiB/s, 1500Mib/s となっており理論限界には程遠い．これに対して，ラテン方陣 Fat-Tree でのスループットはメッセージサイズが十分に大きければいずれも 3700Mib/s 程度であり，ほぼ理論限界を達成したといえる．この結果はシフト通信パターンと比べて最大で 2.7 倍の向上である．

これらの実機での結果を理論限界との比率の形で表現し，同一構成のシミュレーションと比べた結果を図 19 に示す．シミュレーションと実機の結果は同じ傾向であるも

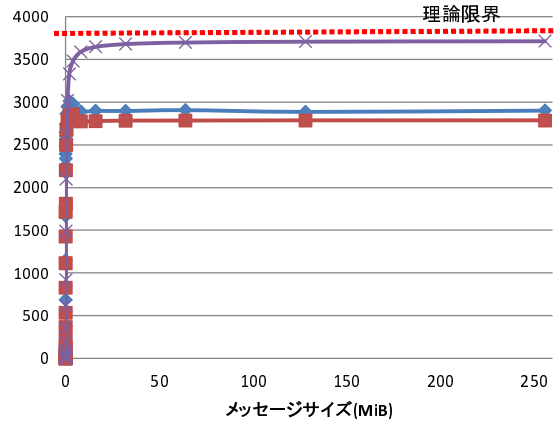


**図 17** 構成 2 の実験でのスループット実測値

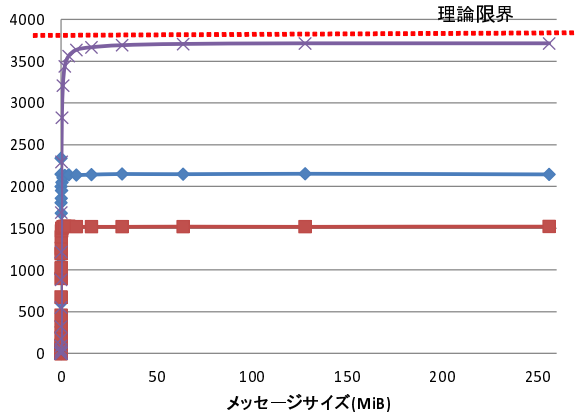


**図 18** 構成 3 の実験でのスループット実測値

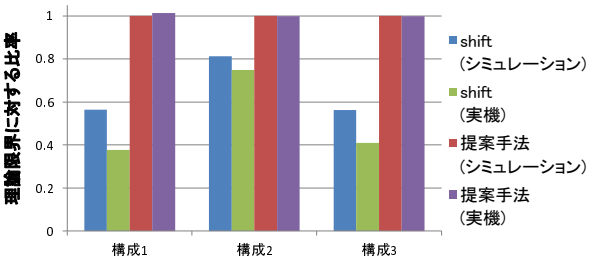


**図 19** シミュレーションと実機のスループット比較

のの，シミュレーション結果は実機に比べてスループットが過大に見積もられることがわかる．以上のことから，ラテン方陣 Fat-Tree のトポロジー構造が実用的にも有用であることが明らかとなった．

## 6. 関連研究

ネットワーク・トポロジーにおけるルーティングの一手法に通信経路を状況に応じて変化させて送信するアダプティブルーティング [13] がある．これは比較的柔軟なルーティングが行える一方で，トポロジー構造全体を俯瞰することが難しいため本稿のように経路競合を完全に回避することは難しい．アダプティブルーティングによるスケーラブルなトポロジー構造として，Dragonfly というトポロジー構造が提案されている [14], [15], [16]．Dragonfly はランダム通信を行う場合に高性能で競合が少ないことが知られている．しかしながら，アダプティブルーティングの特性上，MPI レベルでの経路競合回避方法は考えられていないため，各フェーズでの経路競合を完全に避ける手法にまで言及したものはない．一方，本稿のように事前に経路を静的に決定しておくスタティックルーティングは，柔軟性や耐故障性に弱く，トポロジー構造の変化に対応することが難しいものの [17]，経路の決め方によってはアダプティブルーティングよりも良い性能が得られる．文献 [18] では，単一の Fullmesh における All-to-all 通信の経路競合回避方法が提案されている．単一の Fullmesh は接続可能台数が少なく，実用性が低い．実際，次数 18(36 ポート) のスイッチであれば，接続できるサーバ台数は 342 台である．上記のいずれの文献においてもラテン方陣 Fat-Tree トポロジー構造における All-to-all 通信および一部のサーバ群での All-to-all 通信についての言及はない．

## 7. おわりに

本稿では，低コストで大規模化可能なラテン方陣 Fat-Tree を用いたクラスタ環境上で All-to-all 通信を競合なく行うことができることを明らかにした．また，実際に実機 (InfiniBand) でトポロジー構造を構築し，通常のシフト通信パターンと比較して最大で 2.7 倍のスループットを達成した．また，メッセージサイズが十分大きければ，スループットの理論限界に近い性能を達成していることも確認した．シミュレーションでも実機評価と同様の結果を得られ，大規模になるほどその効果が増大していくことが明らかになった．特に，現在主流の 36 ポートスイッチでは 9.3 倍程度のスループット向上が見込めることが明らかになった．

これによってラテン方陣 Fat-Tree を構築することで，従来の Fat-Tree に比べて格段に多くのサーバ (36 ポートスイッチで 18 倍) を用いつつ Fat-Tree と同様に実用上ほとんど競合のない高スループットでの All-to-all 通信ができることが示された．

今後の課題としては，競合のない All-to-all 通信が実現可能なサーバ台数とサーバ選択方法のバリエーションの制限緩和がある．現状では，トポロジー全体および，$nkm$ という形で書き表されるサーバ台数でのみ All-to-all 通信が可能である．これ以外の規模で All-to-all 通信を行う場合，現状ではサーバを適宜補って All-to-all 通信を行うこととなるが，この場合コスト面でのオーバーヘッドが生じる．この問題を解決するため，より細かく調整できるような台数指定を可能とする手法改良が必要となる．また，ラテン方陣 Fat-Tree にも従来の Fat-Tree と同様に 3 段以上の構成でより多くのサーバをつなげる手法が存在する [2] が，このトポロジー構造に対する経路競合のない All-to-all 通

信を実現する手法についても知られていない．さらに，ラテン方陣 Fat-Tree には経路に冗長性がないため，耐故障性についての議論も知られていない．これらの解決も今後の課題である．

## 参考文献

[1] 中島 耕太, 三輪 真弘: スイッチ台数の削減と高い All-to-all 通信性能を両立する多層 Fullmesh トポロジーの提案, 情報処理学会研研究報告 2014-HPC-145(12), pp.1–7, (2014).

[2] M. Valerio, L. E. Moser and P. M. Melliar-Smith: *Recursively Scalable Fat-Trees as Interconnection Networks*, Proceedings of the 13th IEEE International Phoenix Conference on Computers and Communications, pages 40–46, (1994).

[3] 清水 俊宏，中島 耕太: 接続台数を最大化するラテン方陣 Fat-Tree における All-to-all 通信での経路競合回避, 情報処理学会研研究報告 2015-HPC-151(3), pp.1–8, (2015).

[4] InfiniBand Architecture Specification Release 1.3, InfiniBand Trade Association, http://www.infinibandta.org.

[5] D.Takahashi: *Implementation of Parallel 1-D FFT on GPU Clusters*, IEEE 16th International Conference on Computational Science and Engineering,pp.174–180 (2013).

[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), pp.1–8, (2007).

[7] E. Zahavi, G. Johnson, D. J. Kerbyson, and M. Lang: *Optimized Infiniband Fat-Tree Routing for Shift All-to-all Communication Patterns* , Concurrency Computation Practice and Experience 22 (2), pp.217–231(2009).

[8] A. A. Adrian, S. Reuben: *An Introduction to Finite Projective Planes*, New York, Holt, Rinehart and Winston (1968).

[9] B. Prisacari, G. Rodoriguez and C. Minkenberg: *Generalized Hierarchical All-to-all Exchange Patterns*, In Proceedings of the 2013 IEEE International Parallel and Distributed Processing Symposium (IPDPS13), pp.537–547(2013).

[10] Mellanox, http://www.mellanox.com

[11] Intel MPI Benchmark, https://software.intel.com/en-us/articles/intel-mpi-benchmarks

[12] OpenMPI, http://www.open-mpi.org

[13] P.Geoffray, and T. Hoefler: *Adaptive Routing Strategies for Modern High Performance Networks*, In Proceedings of the 16th IEEE Symposium on High Performance Interconnects pp.165–172, (2008).

[14] J. Kim, W.J. Dally, S. Scott, D. Abts: *Technology-Driven, Highly-Scalable Dragonfly Topology*, 35th International Symposium on Computer Architecture (ISCA '08), pp.77–88, (2008).

[15] N. Jain, A. Bhatele, N. Xiang, N. J. Wright, L. V. Kale: *Maximizing Throughput on a Dragonfly Network*, In Proceedings of the International Conference for High Performance Computing, Networking, Storage and Analysis, pp.336–347, (2014).

[16] B. Prisacari, G. Rodriguez, P. Heidelberger, D. Chen, C. Minkenberg, T. Hoefler: *Efficient Task Placement and Routing of Nearest Neighbor Exchanges in Dragonfly Networks*, In Proceedings of the 23rd international symposium on High-performance parallel and distributed computing (HPDC '14), pp.129–140, (2014).

[17] 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), pp.116–125, (2008).

[18] E. Totoni, A. Bhatele, E.J. Bohm, N. Jain, C. L. Mendes, R. M. Mokos, G. Zheng, and L. V. Kale: *Simulation-based Performance Analysis and Tuning for a Two-level Directory Connected System*, In Proceedings of IEEE 17th International Conference on Parallel and Distributed Systems(ICPADS), pp.340–347, (2011).