Vol.2010-ARC-192 No.4
Vol.2010-HPC-128 No.4
2010/12/16

InfiniBandにおける最適なアドレス割り当てによる経路更新処理の高速化

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

本稿では，InfiniBand における最適なアドレス割り当てによる経路更新処理の高速化手法について述べる．InfiniBand では，障害発生時に障害箇所を迂回する経路へ更新することで，障害回復処理を行う．この処理を高速化するため，スイッチ上の経路情報である FDB の構造に着目し，更新対象となる FDB 上のエントリが連続配置となるように，アドレス割り当てを行う．これにより，経路更新時に必要な MAD (Management Datagram) 転送量を削減し，障害回復処理を高速化する．

提案する最適なアドレス割り当て手法を OpenSM に対して適用し，5,832 ノード及び 11,664 ノードの Fat Tree ネットワークにて評価した結果，経路更新時に必要な MAD 転送量を，従来方式と比較して，1/11〜1/12 に削減できることを確認した．また，詳細な解析の結果，設計どおりの効果が得られたことを確認した．

A Fast Forwarding Database Update Method by Proper Address Assignment for InfiniBand

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

This paper describes a fast forwarding database (FDB) update method by proper address assignment for InfiniBand. In InfiniBand network, in order to recover failed network, a routing in network is updated to newer routing to avoid failed switch and links. To accelerate the updating process, we focus structure of FDB, addresses are assigned for updated entries in FDB to be set in continuous region. The proper address assignment reduces the number of MAD (Management Datagram) used for routing update and accelerate updating process.

We apply the proper address assignment to OpenSM and evaluate the update method. In 5,832 nodes and 11,664 nodes Fat Tree network environment, the update method by the proper address assignment achieve to 11-12 times faster performance than by conventional address assignment. And it is verified that the evaluated performance is best as designed by detail analysis.

1. はじめに

近年，HPC 分野においてクラスタシステムの大規模化が進んでおり，特に最近では，1,000 ノード以上の大規模なクラスタシステムが構築されるようになっている．例えば，理化学研究所の RICC[1] の超並列 PC クラスタは，1,024 ノード構成であり，日本原子力研究開発機構[2] の PC サーバによるクラスタシステムは 2,157 ノード構成[3] である．今後はさらにノード数の増加が進むと推測される．このような大規模クラスタシステムでは，高いサーバ間通信性能が求められるため，Fat Tree 構成の InfiniBand[4] によりサーバ間を接続する事例が多い．

大規模クラスタシステムでは，バッチジョブ管理システムを用いて多数のユーザが共用利用する形態を採っている場合がほとんどである．このようなシステムにおいて，ネットワーク障害が発生すると，障害箇所を経由する経路を用いているジョブは，障害回復まで一時停止されるかタイムアウトによりキャンセルされる．いずれの場合も，多数のユーザに影響を与えるため，できるだけ高速に障害回復する必要がある．

InfiniBand では，障害が発生すると，障害範囲を検出し，障害範囲を回避する経路に切り替えることで障害回復を実現している．この障害回復処理は Subnet Manager (SM) というネットワーク管理機構が実施している．経路情報は，各スイッチが保持しているので，障害回復処理では，これを更新する必要がある．SM は，MAD (Management Datagram) という管理用パケットを各スイッチに送付することで経路情報を更新する．ネットワークが大規模化するとスイッチ数の増加や経路情報の増加により，経路情報の更新に必要な MAD 転送量が増加する．したがって，MAD 転送量の増加を抑え，経路更新を高速化する必要がある．

そこで，本稿では，経路更新処理時の MAD 送付処理時間を短縮し，経路更新処理の高速化をはかる．スイッチ上の経路情報である FDB の構造に着目し，更新対象となる FDB 上のエントリが連続配置となるように，ノードのアドレス割り当てを行う．これにより，経

†1 (株) 富士通研究所
Fujitsu Laboratories Ltd.

Vol.2010-ARC-192 No.4
Vol.2010-HPC-128 No.4
2010/12/16

路更新時に必要な MAD 転送量を削減し，障害回復処理を高速化する．これを実現するために，SM がノードに割り当てるアドレスの割り当て方式を改良する．

本稿で提案する最適なノードアドレス割り当て手法を OpenSM に対して適用し，シミュレータを用いて，5,832 ノード及び 11,664 ノードの Fat Tree ネットワークにて評価した結果，経路更新時に必要な MAD 転送量を，従来方式と比較して 1/11〜1/12 に削減できることを確認した．また，詳細な解析の結果，設計どおりの効果が得られたことを確認した．

2. InfiniBand

2.1 概　　要

InfiniBand[4] とは InfiniBand Trade Association によって規格化されている高速ネットワークである．現在は，主に HCP システムにおけるサーバ間接続に用いられている．

InfiniBand では，適切に経路制御を行うことで，Fat Tree のようなマルチパスを持つネットワークトポロジーを実現することができる．ノード間に複数の通信経路を設けることができるため，帯域と信頼性を高めることができる．

また，InfiniBand では，Subnet Manager(SM) と呼ばれるネットワーク管理機構によりスイッチの経路情報の管理を行っている．SM が適切な経路情報をスイッチに設定することで経路制御を実現している．

2.2 SM による経路制御

InfiniBand では，スイッチやノードに対して Local Identifier (LID) というアドレスが割り当てられる．SM は，スイッチやノードに対してこのアドレスの割り当てを行う．InfiniBand 内で転送される通信用パケットのヘッダには，転送先アドレスが格納されている．

スイッチには転送先アドレスに対応する出力ポート番号 (FDB: Forwarding Database) が設定されている．スイッチは，到着したパケットの転送先アドレスを参照し，FDB の情報から出力ポートを特定し，転送する．これにより，パケット転送を実現している．

図 1 のように，FDB は 64 個の連続する転送先アドレスを単位 (FDB ブロック) として管理されており，FDB の設定は，この FDB ブロック毎に行われる．したがって，各スイッチの FDB ブロック数は使用されるアドレスの個数にほぼ比例する．一般にネットワーク規模が大きくなれば，スイッチ数も増加するため，ネットワーク規模の増加に対して，FDB ブロック数は，線形以上に増加する．なお，本稿では，InfiniBand Specification の慣例に習い，アドレスは 16 進数表記，ポート番号は 10 進数表記する．

SM は，ネットワーク全体の経路を探索し，それを実現する各スイッチの FDB を算出し，

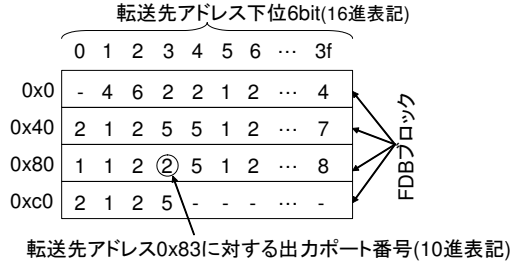

図 1　FDB の構成

これを設定する．各スイッチへの FDB 設定では，MAD (Management Datagram) という管理用パケットが使用される．SM は設定対象スイッチに対して FDB ブロックを格納した MAD(経路情報 MAD) を送付することで設定処理を実現している．

2.3 MAD 転送時間

MAD の転送時間は，MAD 種別，スイッチ機種，ネットワーク環境により大きく異なり，ばらつきもあるが，MAD の転送時間は 1 つあたり概ね数 100μs〜数 10ms 程度である．36 ポートスイッチである Voltaire 製 Grid Director 4036 において，経路情報 MAD の転送時間を測定すると，1 つの MAD の転送時間は，平均 265μs である．さらに，MAD 転送は転送失敗により再送となる場合もある．この場合は，さらに数倍〜数 100 倍の時間が必要となる．

2.4 障害発生時の経路切り替え

ネットワーク上で障害が発生し，スイッチがリンクダウンを検出すると，これを MAD の一種である Trap により SM に通知する．SM は，Trap を受信すると障害箇所を検出し，障害箇所を回避する経路を再計算する．その後，障害箇所を回避する経路に切り替えるため，FDB の更新を行う．この FDB の更新は，2.2 節で述べたように，MAD の送付により実現する．ネットワーク規模の増加に対して，FDB ブロック数は，線形以上に増加するため，ネットワークが大規模化すると，FDB 更新のための MAD 転送時間の長大化を招く可能性がある．

3. Fat Tree

3.1 標準的な経路制御

大規模 PC クラスタにおける InfiniBand ネットワークでは，図 2 のような Fat Tree が広

Vol.2010-ARC-192 No.4
Vol.2010-HPC-128 No.4
2010/12/16

表 1 経由する Spine とノード番号

経由する Spine	Spine 1	Spine 2	Spine 3	Spine 4
ノード番号 (16 進表記)	1,5,9,d	2,6,a,e	3,8,b,f	4,8,c,10

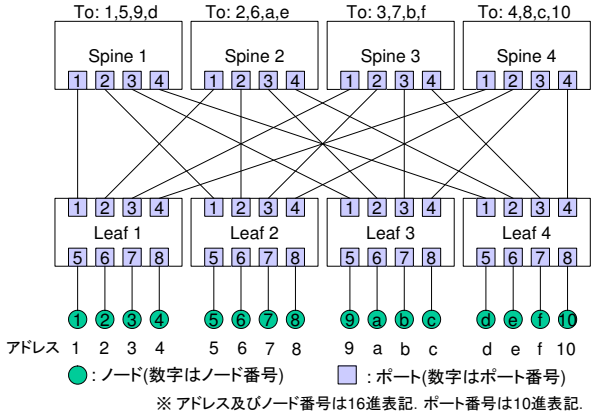

図 2 Fat Tree の構成例

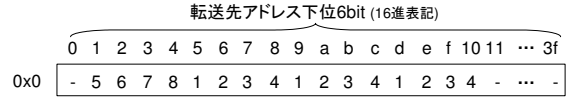

図 3 16 ノード構成における FDB

く用いられている．InfiniBand における Fat Tree における標準的な経路制御では，各ノードへの経路を決める際，経由する Spine スイッチを順番に選択することで経路の負荷分散を実現している．

例えば，図 2 の例では，ノード 1 から順番に経路を決定していき，ノード 1 へ向かう経路は Spine 1 経由，ノード 2 へ向かう経路は Spine 2 経由というように経路を決定していく．このように決定するため，表 1 のように転送先のノード番号によって経由する Spine 番号が決定する．この時，ノード番号と同じアドレスを割り当てていくと，例えば Leaf 1 の FDB は図 3 のようになる．このように，転送先アドレスの順に出力ポート番号が順番に出現するような FDB となり，同一の出力ポート番号が離散的に配置される．

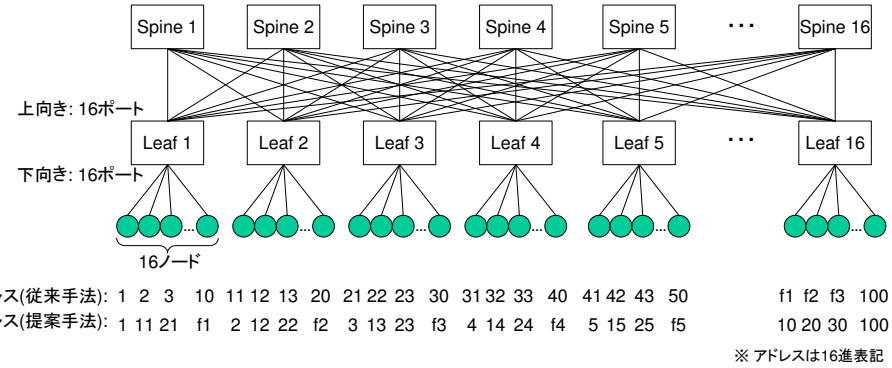

図 4 256 ノード構成

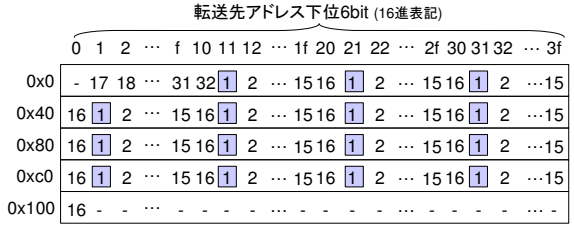

図 5 256 ノード構成における FDB

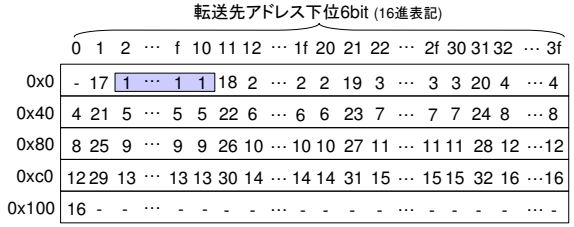

図 6 アドレス割り当て変更による FDB

3.2 障害発生時の経路更新処理

Leaf-Spine 間の経路に障害が生じたり，Spine スイッチに障害が生じた場合は，Leaf から Spine 側へ向かう経路において，障害が生じている Spine へ向かう経路を，正常な別の Spine スイッチへ向かう経路へ変更することで障害回避を行う．したがって，障害発生時の

Vol.2010-ARC-192 No.4
Vol.2010-HPC-128 No.4
2010/12/16

ノードへの経路変更は，Leaf スイッチ上の FDB を更新することで実現される．Spine スイッチ上の FDB は変更されない．

更新される FDB エントリは，障害が生じた方向の出力ポート番号を持つ FDB エントリであり，これを別の出力ポートを変更する．例えば Spine 1 が故障する場合を考えると，各 Leaf スイッチにおいて，出力ポート番号が 1 となる FDB エントリを更新する必要がある．

図 4 に示すような 32 ポートの Leaf スイッチ 16 台を用いた 256 ノード構成の場合は，FDB は図 5 のようになる．この FDB において出力ポート番号が 1 となる FDB エントリは，離散的に存在し，ほぼ全ての FDB ブロックに跨って存在する．FDB ブロックを単位として更新するので，経路を切り替えるためには，ほぼ全ての FDB ブロックを更新する必要がある．図 5 の事例では，4 つの FDB ブロックを更新する必要がある．

4. 課　　題

大規模 InfiniBand ネットワークを利用したシステムは，バッチジョブ管理システムを用いて多数のユーザが共用利用する形態を採っている場合がほとんどである．このようなシステムにおいて，ネットワーク障害が発生すると，障害箇所を経由する経路を用いているジョブは障害回復まで一時停止されるかタイムアウトによりキャンセルされる．いずれの場合も，多数のユーザに影響を与えるため，できるだけ高速に障害回復する必要がある．

障害が発生すると，SM は経路を再計算し，更新後の経路に切り替えるために FDB 更新を行う．ネットワーク規模が大きくなると更新が必要となるスイッチ数と各スイッチ上の FDB ブロック数が増加する．このため FDB 更新のために必要となる MAD 数が増加する．36 ポートの Leaf スイッチ用いた場合における，ノード数，Leaf スイッチ数，各 Leaf スイッチ上の FDB ブロック数の関係を表 2 に示す．Leaf スイッチ数と FDB ブロック数は共にノード数に比例して増加するため，Leaf スイッチの FDB ブロック数の合計はノード数の 2 乗に比例する．

ノード番号順にアドレスを割り当てる場合，3.2 節で述べたように，1 台の Spine スイッチに障害が生じた場合，各 Leaf スイッチのノードへの経路に対する FDB ブロックをほぼ全て更新する必要がある．2.3 節で述べた MAD 転送時間を用いて FDB ブロック更新処理時間を見積もると，表 2 に示すように，648 ノード構成では 0.10 秒程度で完了するのに対して，11,664 ノード構成では 31.31 秒必要になる可能性がある．この見積もりは，MAD 転送が再送しないことを仮定している．再送を含めて考えると，さらに更新処理が長大化する可能性がある．したがって，10,000 ノードクラスの大規模構成では，経路更新処理の MAD 送付処理時間が長大化する．これを解決することが課題である．

表 2　ネットワーク規模と Leaf スイッチの FDB 数

ノード数	648	5,832	11,664
Leaf スイッチ数	36	324	648
各 Leaf の FDB ブロック数	11	92	183
FDB ブロック数合計	396	29,808	118,584
更新時間見積 (s)	0.10	7.87	31.31

```
for(port=0;port<portnum;port++){
      for(leaf=0;leaf<leafnum;leaf++){
            assignaddr( node(leaf, port), addr);
            addr++;
      }
}
```

※ `port`: ポート番号，`leaf`: Leaf スイッチ番号
`node(leaf, port)`: (Leaf 番号, ポート番号) = (`leaf`, `port`) に接続されるノード

図 7　アドレス割り当て処理の疑似コード

5. アドレス割り当てによる経路更新処理の高速化手法

本章では，アドレス割り当てによる経路更新処理の高速化手法を提案する．Fat Tree における標準的な経路制御の場合，各 Leaf スイッチに接続される同一ポート番号に接続されるノードへ向かう経路は，同一の Spine を経由する．したがって，これらのノードのアドレスが連続するように割り当てを行う。すなわち，Leaf 1 の 1 番ポートに接続されるノードにアドレス 1 を，Leaf 2 の 1 番ポートに接続されるノードに アドレス 2 を割り当てる．

具体的なアドレスの割り当て方法を図 7 の疑似コードに示す．疑似コードに示すように，同一ポート番号に接続されるノードに連続してアドレスを割り当てる．このように，障害発生時に更新対象となる FDB 上のエントリが連続する配置となるようにし，更新が必要となる FDB ブロック数を削減し，MAD 送付処理時間を削減する．

256 ノード構成の Fat Tree における従来手法と提案手法によるアドレスの割り当てを図 4 に示す．図 4 からわかるように，従来手法では，同一 Leaf 内に接続されるノードのアドレスが連続するのに対し，提案手法では，各 Leaf における同一ポートに接続されるノードのアドレスが連続する．この結果，従来手法の FDB は図 5 のようになるのに対し，提案手法では図 6 のようになる．この結果，出力ポート番号が 1 となる箇所は連続しているため，

Vol.2010-ARC-192 No.4
Vol.2010-HPC-128 No.4
2010/12/16

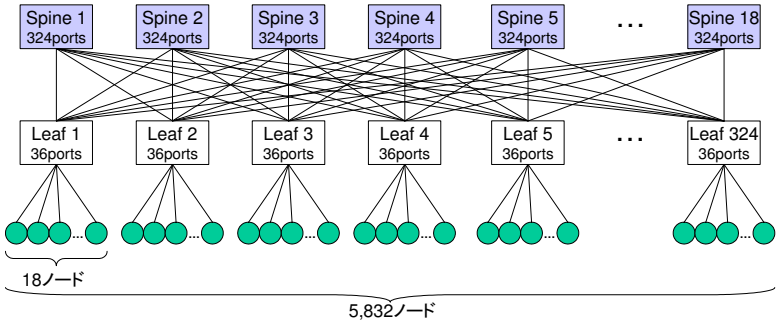

図 8　5,832 ノード構成

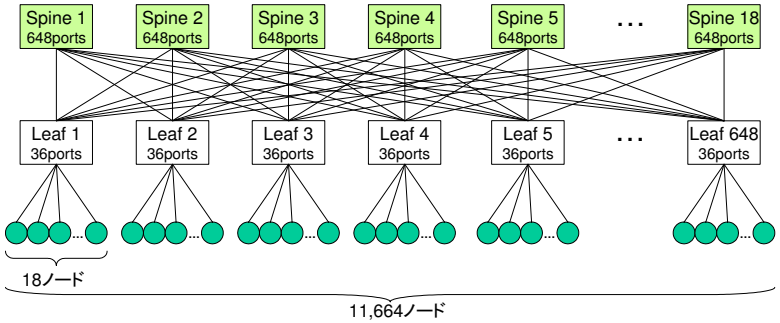

図 9　11,664 ノード構成

Spine 1 に障害が生じた場合には，FDB ブロックを 1 箇所更新するだけで経路更新できる．このようにアドレスを割り当てることで，更新が必要となる FDB ブロック数を削減する．

6. 評　　価

6.1 評 価 方 法

アドレス割り当てによる経路更新処理の高速化手法の有効性を確認するため，提案手法によるアドレス割り当てを適用した場合のシミュレータによる評価を行った．評価では，提案手法と従来手法における Fat Tree 構成におけるスイッチ故障時の経路更新処理時に必要となる FDB ブロックを格納した MAD 数を比較した．

評価には，以下の 2 つのネットワークを用いた．各構成を図 8 及び図 9 に，諸元を表 3 に示す．

表 3　ネットワークの諸元

	5,832 ノード構成	11,664 ノード構成
ノード数	5,832	11,664
ノードのアドレス範囲	0x1 – 0x16c8	0x1 – 0x2d90
ノードへの経路に対する FDB ブロック数	92	183
Leaf スイッチ数	324	648
Leaf ポート数	36	36
Spine 接続ポート	18	18
ノード接続ポート	18	18
Leaf スイッチのアドレス範囲	0x4001 – 0x4144	0x4001 – 0x4288
Spine スイッチ数	18	18
Spine ポート数	324	648

(1)　5,832 ノード構成
(2)　11,664 ノード構成

また，以下の 2 パターンのアドレス割り当てを行った．

(1)　同一 Leaf 番号に接続されるノードのアドレスを連続配置 (従来手法)
(2)　同一ポート番号に接続されるノードのアドレスを連続配置 (提案手法)

各アドレス割り当てを行った場合において，Spine スイッチを 1 台切断し，SM に経路更新処理を実行させる．そして，この際に SM が経路更新のために送信する経路情報 MAD 数を計測した．

6.2 評 価 環 境

評価では，OpenSM 3.3.7[5] を用いた．OpenSM に対し，提案手法と従来手法におけるアドレス割り当てを設定し，各場合の経路更新処理を評価した．また，Open Fabrics Alliance が開発する ibsim 0.5 というシミュレーション環境を用いた．ibsim に対して，ネットワークの結線情報を入力としてあたえると，SM とネットワーク間の MAD 送受信処理を模擬することができる．ibsim 上でスイッチ切断を指示すると，当該スイッチ部分をシミュレータ上で切断し，SM に Trap を通知する．SM はシミュレータ上の切断された領域を走査し，シミュレータに対して，経路更新のための MAD を送付する．この際送付された MAD をシミュレータ側で記録することで計測を行う．

6.3 評 価 結 果

6.3.1 経路情報 MAD 数の比較

経路更新処理時に OpenSM から送信される経路情報 MAD 数を測定した．経路情報 MAD

Vol.2010-ARC-192 No.4
Vol.2010-HPC-128 No.4
2010/12/16

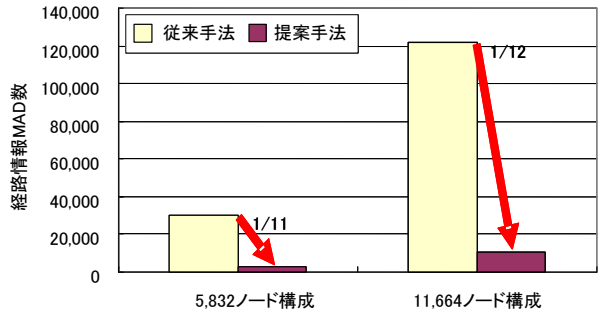

図 10　経路情報 MAD 数の比較

数を図 10 に示す．従来手法と提案手法を比較すると，提案手法では，更新処理で必要となる経路情報 MAD 数を 5,832 ノード構成の場合は約 1/11，11,664 ノード構成の場合は約 1/12 に削減できていることが分かる．これにより，提案手法の適用により，経路更新における MAD 転送時間を削減できることが確認できた．

6.3.2 詳 細 検 証

Leaf スイッチから Spine スイッチへ向かう経路は 18 本あるため，Fat Tree の標準的な経路制御では，この 18 本に均等に転送先アドレスを割り当てる．このうちの 1 本が故障した場合には，全体のアドレスのうち 1/18 が更新対象アドレスとなるため，提案方式を適用した場合，経路情報 MAD 数は最大 1/18 に削減できる．この削減量と評価結果を単純に比較すると，一致しない．この理由は，Fat Tree における経路更新時には，ノードへの経路に対する FDB だけでなく，スイッチへの経路に対する FDB も更新されるためである．スイッチへの経路に対する FDB 更新については，提案手法では，特に考慮していないため，この部分の経路情報 MAD 数は変化しない．そこで，検証のため，経路情報 MAD のうち，スイッチへの経路に対する経路情報 MAD を取り除き，ノードへの経路に対する経路情報 MAD 数を用いて検証を行った．検証結果を表 4 に示す．

各 Leaf のノードへの経路に対する更新対象 FDB ブロック数は，ノード数を n とすると $\lceil n/64 \rceil$ である．従来方式では，この全ての FDB ブロックを更新する．これに対し，本方式では，各 Leaf の Spine 側のポート数を p とすると各 Leaf における更新対象 FDB ブロック数は $\lceil n/64/p \rceil$ である．各 Leaf における更新対象 FDB ブロック数の算出値から理論上のノードへの経路に対する経路情報 MAD 数 (表 4(e)) を算出すると (b) の実測値とほぼ一致する．このため，設計通りの効果が得られたといえる．

表 4　評価結果の検証

		5,832 ノード構成		11,664 ノード構成	
		従来手法	提案手法	従来手法	提案手法
(a)	総経路情報 MAD 数	30,267	2,727	121,715	10,260
(b)	ノードへの経路に対する経路情報 MAD 数	29,484	1,944	118,583	7,128
(c)	各 Leaf の更新対象 FDB ブロック数	92	6	183	11
(d)	Leaf 数	324	324	648	648
(e)	(c)×(d)	29,808	1,944	118,584	7,128

7. 関 連 研 究

FDB 更新処理の高速化に関連する研究事例として文献[6] がある．文献[6] では，FDB の構造に変換テーブル機構を導入することで同一の出力ポートへの経路情報を集約し，FDB 上の更新箇所を削減することで経路更新処理の高速化を図っている．しかし，提案手法では，InfiniBand 準拠であればスイッチハードウェアの変更は不要であるのに対し，文献[6] による手法は，スイッチハードウェアの変更が必要である．また，提案手法では，障害箇所へ向かう出力を多数の出力ポートへ分散させることができるが，文献[6] による手法は，更新後の出力ポートが限定される．このため，負荷分散の点で提案手法の方が優位である．

また，InfiniBand の経路更新に関連する研究事例として文献[7] がある．文献[7] では，スイッチの経路更新順序を考慮することで，経路更新処理途中において経路のループが生じない手法について述べており，トーラストポロジにおいて，評価している．本稿では，Fat Tree における経路更新処理について議論している．文献[7] でも言及されているように Fat Tree においては，経路更新処理順序を考慮しなくても，経路のループが生じることはなく，安全に経路を更新できる．

8. お わ り に

本稿では，最適なノードアドレス割り当てによる InfiniBand における経路更新処理の高速化について述べた．スイッチ上の経路情報である FDB の構造に着目し，更新対象となる FDB 上のエントリが連続配置となるように，ノードアドレス割り当てを工夫することで，更新対象となる FDB 上の領域を集約し，経路更新処理時に必要となる MAD 転送量を削減した．

本稿で提案する最適なノードアドレス割り当て手法を OpenSM に対して適用し，5,832 ノード及び 11,664 ノードの Fat Tree ネットワークにて評価した結果，経路更新時に必要な

Vol.2010-ARC-192 No.4
Vol.2010-HPC-128 No.4
2010/12/16

MAD 転送量を従来方式と比較して，1/11〜1/12 に削減できることを確認した．また，詳細な解析の結果，設計どおりの効果が得られたことを確認した．

本稿で提案した最適なノードアドレス割り当て手法は，特に 10,000 ノードを越えるような大規模な Fat Tree 構成において効果が大きい．したがって，今後さらに大規模なクラスタシステムを構成する場合には，提案手法の適用が非常に有効であると言える．

今後の課題として，スイッチへの経路まで考慮したアドレス割り当て方式の検討や，実環境における評価がある．

参 考 文 献

1) RIKEN Integrated Cluster of Clusters, http://accc.riken.jp/ricc.html
2) 日本原子力研究開発機構，http://www.jaea.go.jp/
3) 「日本原子力研究開発機構様の新スーパーコンピュータシステムが稼動」，富士通株式会社，プレスリリース，http://pr.fujitsu.com/jp/news/2010/03/1.html (2010).
4) InfiniBand Architecture Specification Release 1.2.1, InfiniBand Trade Association, http://www.infinibandta.org.
5) OpenSM, OpenFabrics Alliance, http://www.openfabrics.org.
6) 奥 智行，赤羽 真一 : “ネットワークノード装置，” 公開特許公報 2005-333220 (2005).
7) 中島 耕太，久門 耕一，成瀬 彰，住元 真司 : “大規模 InfiniBand システムにおける経路更新手法の提案，” 電子情報通信学会 技術情報報告．CPSY．コンピュータシステム，Vol. 109, No. 168 (2009).