Vol.2018-OS-142 No.12
2018/2/28

ラック間をまたぐ リモートGPUおよびSSD間通信への光無線割り当てに関する研究

原 弘明[1]　森島 信[1]　鯉渕 道紘[2]　天野 英晴[1]　松谷 宏紀[1]

概要：ラックスケールアーキテクチャでは、計算機を構成する専用ハードウェアコンポーネントをサーバラックに格納しておき、これらをハードウェアリソースの共有プールとして利用する。異なるラックのハードウェアコンポーネントをラック間ネットワークで接続することで、ラック間をまたいだハードウェアリソースの共有プール化も可能であり、本論文ではラックスケールアーキテクチャにおけるこのようなラック間通信に焦点を当てる。本論文ではこのようなラック間ネットワークに、光無線通信を用いた再構成可能ネットワークを導入することでネットワーク性能を強化することを提案する。ハードウェアコンポーネント間通信を想定したものとして、MongoDB、Chainer といった実アプリケーションを動作させた際のコンポーネント間通信トレースを取得した。その通信トレースを基にネットワークシミュレータによるシミュレーション評価を行った。評価結果より、本論文が対象とするラックスケールアーキテクチャにおいて、光無線通信によるネットワーク再構成の効果を確認できた。

A Study on FSO Assignment of Inter-Rack Communication between Remote GPUs and SSDs

Hiroaki Hara[1]　Shin Morishima[1]　Michihiro Koibuchi[2]　Hideharu Amano[1]　Hiroki Matsutani[1]

1. はじめに

ラックスケールアーキテクチャでは、CPU およびメモリ、ストレージ、GPU など計算機を構成するハードウェアリソースの利用率向上を目的として、これらのコンポーネントをサーバラックに格納しておき、ハードウェアリソースの共有プールとして利用する。異なるラックのハードウェアコンポーネント同士をラック間ネットワークで接続することで、ラック間をまたいだハードウェアリソースの利用率向上を図ることもできる。ラックスケールアーキテクチャにおいてはコンポーネント間の密なネットワークが重要となり、ラック間をまたいだラックスケールアーキテクチャを想定する場合、ラック内のみならずラック間でのネットワークを強化する必要がある。

そこで、本研究ではラック間ネットワークを強化する技術として、光無線（Free Space Optics： FSO）を用いたラック間の再構成可能ネットワークに着目する。FSO リンクを用いた相互結合網 [1][2] では、図 1 に示すようにラック上部に FSO デバイスを設置し、FSO リンクを用いて一部あるいはすべてのラック間通信を行う。有線リンクとは異なり物理的にケーブルを張り替えることなく、FSO デバイスのコリメーターレンズの向きを変えるだけで任意のラック間に動的にリンクを構築できる。ラック上部と天井間の空間を利用するため、遮蔽物による影響を受けずに長距離通信が可能であり、ラック間ネットワークのトポロジを動的に変更可能である。

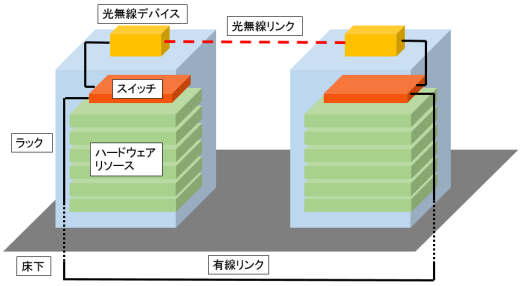


図 1　FSO リンクを用いたラック間通信

本研究ではラック間に FSO リンクを動的に割り当てることでラック間のネットワークトポロジを動的に再構成し、ラック間をまたぐラックスケールアーキテクチャ全体のネットワーク性能を強化することを提案し、その効果を評価する。評価ではまず、ラックスケールアーキテクチャにおけるコンポーネント間通信を想定した実機による通信トレースを取得する。その後、取得した通信トレースを反映したネットワークシミュレータによって FSO によるトポロジ再構成をラック間をまたぐコンポーネント間通信に導入する効果を評価する。

本論文の構成は以下の通りである。2 章で関連技術と関連研究を紹介する。3 章ではラックスケールアーキテクチャにおけるラック間をまたぐコンポーネント間通信に FSO を用いた再構成可能ネットワークを導入することを提案する。4 章にて本提案を評価し、5 章で本論文をまとめる。

[1] 慶應義塾大学大学院 理工学研究科
Graduate School of Science and Technology, Keio University
[2] 国立情報学研究所
National Institute of Informatics

2. 関連研究

2.1 ラックスケールアーキテクチャ

本論文で想定するラックスケールアーキテクチャ（Rack Scale Architecture: RSA）を図 2 に示す。図 2(a) のような従来型ラックマウントサーバでは、必要なハードウェアコンポーネント（CPU およびメモリ、ストレージ、GPU など）が一通り揃った計算機をサーバラックに格納している。対して、図 2(b) の RSA では、計算機を構成するハードウェアコンポーネントをラック内に格納し、それらをネットワーク接続することでラック内もしくはラック間をまたいで計算機を構築する。ラック間をまたいで計算機を構築している例を図 2(c) に示す（この例では同一リソースを同一ラックに格納している）。この場合、アプリケーション毎に必要なハードウェアリソースのみをラック間をまたいで柔軟に割り当てることができ、従来型ラックマウントサーバよりもハードウェアリソースの利用率向上を図れる。このようなアイディアを Software defined infrastructure とも呼ぶ。

RSA のアイディア自体は 2013 年ごろから広く知られるようになってきた [3]。最近では、RSA を想定した研究事例も登場している [4][5]。例えば、文献 [4] ではラック内ネットワークに PCI-Express ベースの RDMA を使用する手法、文献 [5] では計算コンポーネント毎に付与された小規模スイッチを用いてラック内ネットワークを実現する手法が提案されている。これらは主としてラック内通信を高効率化するための提案であり、RSA におけるラック間をまたぐコンポーネント間通信に焦点を当てた研究はほとんど存在していない。

2.2 NEC ExpEther

ExpEther では Ethernet を利用して PCI-Express を延長することにより、空間的に離れた位置にあるハードウェアリソースが直接ホスト PC の PCI-Express スロットに接続されているかのように扱うことができる [6]。また、通信に Ethernet を利用するため、既存の Ethernet スイッチを用いてネットワークを構成することができる。本研究で構築した RSA のプロトタイプシステムにおいても計算コンポーネント間通信には ExpEther を利用している。本研究においても各種ハードウェアコンポーネントは ExpEther 技術を利用して 10GbE スイッチに接続されているものと想定する。

2.3 FSO 通信

FSO リンクによって 40Gbps 以上の高いスループットを光電変換なしに実現できる。これまでにホームネットワーク、室内ネットワーク、ビル間、衛星通信など幅広い分野において FSO リンクが利用されてきた [7]。図 3 は、本研究で使用した FSO デバイスの写真である。装置上部にコリメータレンズが設置されており、コリメータレンズは FC コネクタを介して光ケーブルで 40GbE 光トランシーバに接続されている。コリメータレンズは位置を調節するための台座に固定されている。図 3 では 2 台の PC 上に 2 個のコリメータレンズを設置し、双方向に通信できるようにしている。本論文では図 1 のようにサーバラック上部にコリメータレンズを設置し、ラック間通信に FSO リンクを用いることを想定している。

文献 [8] では、このような FSO リンクをデータセンタ内のラック間通信に利用することを提案している。具体的には、FSO を用いたネットワークの動的再構成によって、既存のデータセンタで使用されている有線リンクによる Fat-tree トポロジよりも、低コストかつ高性能なネットワークを実現できることを示した。本研究では、RSA を対象にラック間をまたぐコンポーネント間通信に FSO リンクを導入し、動的にネットワークを再構成することを提案する。

FSO とは別の無線通信技術として、60GHz 帯無線を用いた研究が報告されている [9][10]。しかし、FSO 通信は、1) 従来の 60GHz 無線通信に比べてはるかに低いビットエラーレートで通信でき、2) 多数の無線通信を利用しても干渉が起こりにくいなどの利点を有している。そこで本論文では無線リンクを実現するための手法として FSO リンクを選択した。

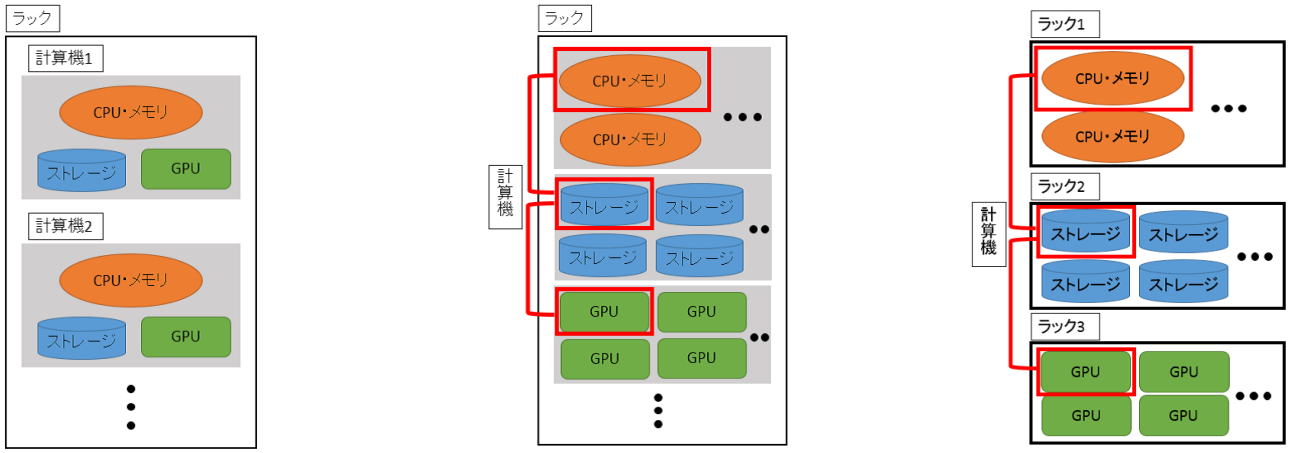
3. FSO を用いた RSA のための動的トポロジ再構成

RSA のおいては、異なるラックに格納された計算コンポーネント同士をラック間ネットワークで接続することで、ラック間をまたいだ計算リソースの利用率向上を実現できる。このようなラック間通信は、通常、スイッチを介した有線リンクによって実現されるため、一度構築されたネットワークトポロジは変更できない。そこで、一定時間ごとに FSO リンクによる動的なネットワーク再構成を行うことで、ワークロードに適したネットワークトポロジを構成することが可能となる。とりわけ、RSA においては接続主体が計算機よりも細粒度な計算コンポーネントであるため、コンポーネント間のネットワークを強化することは重要な課題であると考えられる。本研究では、RSA を対象にラック間をまたぐコンポーネント間通信に FSO リンクを導入し、動的にリンクを再構成することでラック間ネットワーク性能を強化を目的とする。

3.1 全体アーキテクチャ

本研究の主題はラック間をまたぐ計算コンポーネント間通信であるため、ラック間通信が重要となるような RSA を想定する。具体的には、図 2(c) に示すように、計算機を構成するハードウェアリソースとして CPU・メモリ、ストレージ（SSD）、GPU の 3 種類が存在し、それぞれが単一ラックに集中して格納されているような RSA を想定する。このような集中配置は、例えば、計算負荷の高い GPU が格納されているラックに強力な冷却装置を導入するなど、ラック毎に格納されているリソースに適した管理手法を導入しやすいという利点がある。

今回、それらの各リソースの 1 つのデバイスを 1 ノードとしたネットワークを想定する。ラック内に格納される 1 つのデバイスを 1 ノードとしたとき、ラック内とラック間で 2 階層のネットワークを構築する。それぞれのネットワークは再構成可能であるが、元の静的なトポロジは 3D トーラスとする。文献 [5] では、ラック内ネットワークに着目したとき、1 つのノードとなるマイクロサーバに小規模のスイッチ機能を実装し、それらのスイッチ間をクロスポイントスイッチを用いることでネットワークの再構成を可能としている。本研究のラック内のネットワークとしてはこの文献 [5] で実現されている再構成可能ネットワークを前提とする。ラック間ネットワークにおいては通常の有線リンクと FSO リンクの混在するネットワークを想定し、全リンクにおける FSO リンクの割合は制約条件となるパラメータとして変化させる。なお、ここでは FSO リンクを構成するための FSO デバイスは図 1 のようにラックの上部に設置され、FSO デバイスは 1 つのラックに複数個設置できる [2] ものとする。これにより、2 階層の 3D トーラストポロジを基として、ラック内、ラック間の 2 階層共に再構成可能な RSA が実現される。



(a) 従来型ラックマウントサーバ　(b) ラックスケールアーキテクチャ　(c) ラック間にまたがるラックスケールアーキテクチャ

図 2　ラックスケールアーキテクチャの概念図

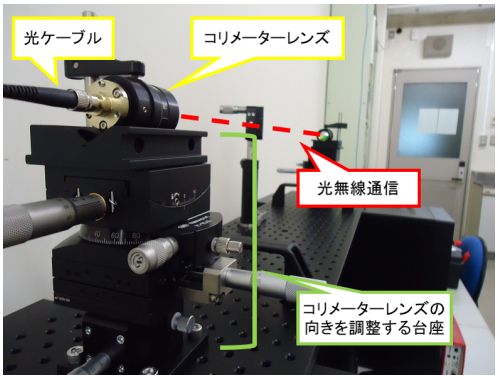


図 3　2 台の FSO デバイス（手前と奥）による FSO 通信

3.2　ネットワークトポロジの動的再構成

ラック内、ラック間共にネットワーク性能を強化するために効果的にトポロジを再構成するためには再構成手法が重要となる。再構成手法としては文献 [5] で使用されていた手法と同様のクラスカル法に基づいたトポロジの算出を行う。一定時間内の各ノード間の通信量を記録しておき、その通信量をクラスカル法における重みとして扱う。ラック内においては各リソースの 1 つのデバイスを 1 ノード、ラック間においては各ラックに存在する ToR スイッチを 1 ノードとして扱い、ネットワークの再構成は一定時間ごとに行う。ネットワーク再構成を行う一定時間の間、ラック内、ラック間それぞれにおいて各ノードに存在するポートがすべて埋まるまでクラスカル法によるエッジの追加を繰り返す。これにより、生成されるトポロジは以下の特徴を有する。

- 全てのノードが接続される
- 全てのポートが使用される
- 通信需要が多いノード間に優先的にリンクが割り当てられる

クラスカル法を用いた再構成アルゴリズムの詳細を Algorithm 1 に示す。このアルゴリズムでは各ノードのペアごとに通信量をリストとして保持し、リストを更新していくことで通信量の多いノードのペアに優先的にエッジが構成されるようにする。

2 行目の node[NODESIZE] はネットワークトポロジ再構成の対象となるノードを構造体配列で管理する。ラック内、ラック間どちらのネットワークにおいてもノードはスイッチの役割を担っていて、持っているポートの数がグラフにおいてはノードから出るエッジの本数となる。構造体の要素としては接続されているノード番号と node[i].treenum という要素を持つ。node[i].treenum は、現在着目しているノードをノード i としたときそのノード i がどの木に属しているかを表すメンバ変数である。このアルゴリズムにおいては、クラスカル法で最小全域木を作る際、ノード i が既に同じ木に属しているノードか否かを判定するために使用する。3 行目の node_pairlist は全ノードを 2 つずつのペアにしたときのペアのリストである。node_pairlist にはペアとなる 2 つのノード番号と、再構成の間隔となる一定時間内のその 2 つのノードの間の通信量を要素として持っている。なお、Algorithm 1 では省略したが node_pairlist は通信量によって降順にソートされている。

4〜21 行目がアルゴリズムの本体となっていて、node_pairlist の要素が全て無くなるまで繰り返される。5〜7 行目はすべてのノードの treenum をノードごとに異なる値で初期化する。treenum の値が同じノードは接続され同じ木に属していることを表すので、初期化された段階ではすべてのノードは別々の木に属していることになる。8 行目の Set_liststart() 関数では tmp_list に node_pairlist の一番最初の要素を設定する。tmp_list は node_list の内の 1 つの要素を表すポインタである。9〜20 行目の処理は tmp_list が node_list の最後の要素にたどり着くまで繰り返される。10 行目の Check_Usedport() 関数は、tmp_list で表されるペアの 2 ノードについて、それぞれポートが空いているかを確認する関数であり、ペアのどちらのノードについても空きのポートがあった場合、true の値を返す。11 行目の Remove_node_pairlist() 関数は、10 行目の条件判定の結果、ペアのノードのどちらか片方にでもポートの空きがなかった場合、これ以上エッジを追加できないため、このペアをリストから削除する関数である。12 行目の Check_Treenum() 関数は、tmp_list で表されるペアの 2 つのノードが同じ木に属しているのかを判定する関数である。同じ木に属していた場合は true の値を返し、処理は 13 行目に遷移する。13 行目の Move_nextlist() 関数は tmp_list を node_pairlist における次の要素に進めて設定する関数である。12 行目の判定の結果、tmp_list で表されるペアの 2 つのノードが同じ木に属していない場合、15〜20 行目の処理を行う。15 行目の Configure_link() 関数では、tmp_list で表されるノードのペアの間にエッジを構成し、node 配列の対応する要素に 2 つのノードが接続されたことを記録する。16 行目の Add_Useport() 関数では、15 行目で接続したノードのペアに対してそれぞれ接続に 1 ポートづつ使用するため、node 配列の対

応する要素に対してポートが使用されたことを記録する。17 行目 Set_sametreenum() では、15 行目で接続したノードのペアに対して、2 つのノードが接続されたことによりそれらが同じ木に属しているため、treenum の値を同一のものにする。このとき、それぞれのノードが属していたツリーのすべてのノードに対して treenum の値は同一のものにする。18 行目ではリストを次に進める処理と、リストの内容を更新するため、tmp_list の内容を tmp2_list にコピーする。19 行目では tmp_list をリストの次の要素に進める。20 行目の Insert_afterlist() 関数では、エッジを追加したペアのリスト (tmp2_list) をリスト内で一定の間隔分後ろに挿入する。この操作によりすでにエッジとして構成されたペアに関しては、クラスカル法を繰り返す次のステップにおいては優先度が下がることになる。以上の 4〜23 行目の処理を、11 行目の Remove_node_pairlist() によってリストの全ての要素が取り除かれるまで（全てのノードの全てのポートが接続に使用されるまで）繰り返すことで完了する。

Algorithm 1 クラスカル法を用いたネットワークトポロジ再構成アルゴリズム

```
 1: NODESIZE //トポロジの対象となるノードの数
 2: node[NODESIZE] //ノードの情報を管理する構造体配列
 3: node_pairlist //ノードのペアの数存在するリスト構造
 4: while node_pairlist ≠ φ do
 5:    for i = 0 to NODESIZE do
 6:       node[i].treenum ← i
 7:    end for
 8:    Set_liststart(tmp_list, node_pairlist)
 9:    while tmp_list ≠ node_pairlist.end do
10:       if Check_Usedport(tmp_list, node[]) ≠ true then
11:          Remove_node_pairlist(tmp_list, node_pairlist)
12:       else if Check_Treenum(tmp_list, node[]) = true then
13:          Move_nextlist(tmp_list, node_pairlist)
14:       else
15:          Configure_link(tmp_list, node[])
16:          Add_Usedport(tmp_list, node[])
17:          Set_sametreenum(tmp_list, node[])
18:          tmp2_list ← tmp_list
19:          Move_nextlist(tmp_list, node_pairlist)
20:          Insert_afterlist(tmp2_list, node_pairlist)
21:       end if
22:    end while
23: end while
```

この Algorithm 1 のクラスカル法を用いたトポロジ再構成アルゴリズムをラック内、ラック間の 2 層に分けて適用する。また、ラック内のネットワーク再構成については Algorithm 1 に示した処理の前に、ラック外への通信のために使用される特別なリンク（アップリンクと呼ぶ）を設定する処理が存在する。このアップリンクに関しても通信量をリストとして管理し、優先度の高いノードに設定するといった処理をする。ラック内のネットワーク再構成では、まずこのアップリンクを配置する処理を行い、アップリンクへの通信量を考慮した通信量をリストを算出してから Algorithm 1 の処理を行う。

4. 評価

ラック内＋ラック間のネットワークの再構成による効果を評価する。評価にはネットワークの再構成アルゴリズムを組み込んだネットワークシミュレータを用いる。従来のデータセンターネットワークは、一通りの計算リソースが揃った計算機をネットワーク接続の対象としているが、本研究では粒度の細かい計算コンポーネント同士を接続するネットワークを対象とする。そこで本研究では、実機による計算コンポーネント間の通信トレースをシミュレータのパラメータとして用いることで、計算コンポーネント間ネットワークの通信を想定する。

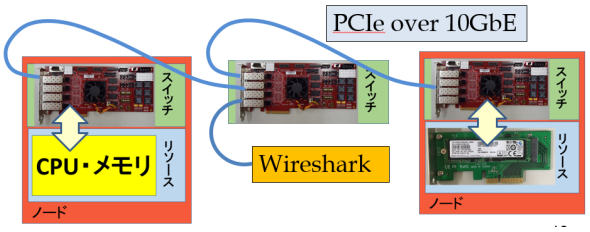


図 4　FPGA スイッチを介して計算コンポーネント間通信のトレース取得

4.1 実機による通信トレースの取得

ネットワークシミュレータで使用する通信トレースを実機によって取得する。計算コンポーネント間ネットワークのプロトタイプとして図 4 のような環境を構築した。簡易的な 10GbE スイッチとして NetFPGA-10G と呼ばれる 10GbE インタフェースを 4 個有する FPGA ボード [11] に Reference Switch Learning Lite と呼ばれる 10GbE スイッチ機能を焼き込んで使用した。Reference Switch Learning Lite には受信キュー、アービタ、16 エントリの MAC アドレステーブルを有しており、シンプルな 10GbE スイッチとして動作する。そこで、中間に位置する FPGA には他の 10GbE インタフェースを 3 個の通信情報を 1 つの 10GbE インタフェースに複製して流す機能を加えたものを使用し、図 4 のように Wireshark を用いてコンポーネント間の通信パケットを取得する。ここでは SSD デバイスとして Samsung MZ-HPU512T/004、GPU デバイスとして NVIDIA GeForce GTX980Ti を用いた。

4.2 アプリケーションプログラム

トレースを取得するアプリケーションプログラムとして今回 3 種類のプログラムを用意した。

(1) MongoDB：ホスト-SSD 間通信を想定した MongoDB を用いたアプリケーション。SSD 上に存在する 1000 文字の文字列を要素とする 1 億件のデータベースに対してランダムにアクセスする読み出しクエリを 10 万回実行する。

(2) GPU 文字列比較：ホスト-GPU 間通信を想定した CUDA プログラムを用いたアプリケーション。多数の 6 文字のワードから成る 512MB のテキストから 6 文字のパターンを見つけ出すという文字列完全一致探索を行う。512MB 分のテキストを GPU に転送し、完全一致探索を行う処理を 10 回実行する。

(3) Chainer：ホスト-GPU 間通信を想定した、深層学習フレームワーク Chainer を用いたアプリケーション。GPU 上で、Chainer によって mnist と呼ばれる基本的な手書き文字認識の学習を行う。

4.3 シミュレーション評価

4.3.1 シミュレーション環境

ここでは 3.2 節で議論した再構成アルゴリズムを実装した C 言語で作成したネットワークシミュレータを用いて、ラック間をまたぐ RSA を構成するネットワークにおける FSO によるネットワークトポロジ再構成の効果を評価する。主にネットワークのスループットを算出し、一部の評価ではパケット転送に要したホップ数を算出する。本論文で対象とするベーストポロジは、

各計算コンポーネントを単一ノードとしたとき、2 階層の 3D トーラストポロジである。それら計算コンポーネントノードがラック内に 64 ノード、それらが格納されたラックが 64 ラックあるとした、全 4096 ノードのネットワークとなっている。また、今回ラック間をまたぐコンポーネント間通信に注目するため、基本的には 2.1 節で示したような 1 つのラックに 1 種類のデバイスが格納されているような構成を想定する。FSO デバイスはラックの上部に設置され、弓形の FSO 端末のフロアレイアウト [12] を想定し、他の端末による遮断なしに任意のラック間に FSO リンクが構築できるものとする。

以下にシミュレータの動作内容を記す。シミュレータは 1 サイクル 25ns の精度で動作し、ネットワーク上で 4.1 節で取得したトレースログファイルに基づいたパケット群を転送する。これは 4.1 節で構成した計算コンポーネント間ネットワークのプロトタイプにおいて使用した FPGA スイッチが 160MHz、1 フリット=32Byte で転送することに基づいて設定したものである。したがって、シミュレータに流す通信トレースについても 32Byte を 1 フリットとして換算する。パケットは通信トレースごとにまとまりとして扱い、ランダムにメインマシンとなる CPU ノードと、ランダムまたは指定の SSD もしくは GPU ノードとの間の通信として発生する。シミュレーションは 4 億サイクル分行い、ネットワークの再構成は 4000 万サイクルごとに実行する。再構成のオーバーヘッドは 40 万サイクル (10ms) とする。これは、文献 [13] で示されている光ビーコントラッキングシステムではアングル内のアラインメントを 10ms で達成することができるという報告に基づくものである。このとき、ラック間通信においては FSO デバイスによる FSO リンクの再構成を行うが、元の 3D トーラスを構成するすべてのリンクが FSO リンク（再構成可能リンク）でない場合を想定し、FSO リンクの割合を制限する条件を設けた。具体的にはラック間の全リンクの内 0%(ラック間の再構成なし)、50%、100%と再構成可能リンクの割合を変化させて評価を行う。また、ラック内、ラック間ともにトポロジの再構成を全く行わない素のままの 3D トーラスとして「再構成無し」という条件においても評価を行い、トポロジ再構成の効果を評価する。

4.3.2 シミュレーション評価結果

上記のシミュレータを用いたシミュレーション結果を以下に示す。評価はネットワーク全体のスループットを評価した（図 5〜8）。横軸はネットワークに注入したパケットの注入量を、1 フリット (flit)=32Byte として換算したものをシミュレーションのサイクル数 (cycle)、対象ネットワークのノード数 (node) で割ったものとして表している。縦軸は注入したパケットのうち受理できたパケット量を横軸同様の単位として換算して表している。評価のパラメータとしては、大きく分けて 4.1 節で示した 3 種類のトレースを使用し、MongoDB のトレースを使用した評価においてはその他にも以下に記すようにパラメータを変更して評価を行う。4.1 節で記したプログラムのトレースはそれぞれ実行時間が異なる。ネットワークの再構成は一定時間間隔で行うため、通信の時間的な偏りが再構成の効果に影響する。従って各トレースの 1 回辺りの実行時間を揃えることで各トレースの評価の条件を揃えることにする。

(1) MongoDB トレースを用いた評価（特殊条件無し）: ホスト-SSD 間通信を想定した評価で、41.3s 分のトレースの内、中間時間の 3s 分のトレースを抽出してシミュレータに流す。評価結果を図 5 に示す。全く再構成を行わない 3D トーラスと比較して、既存手法であるラック内のみの再構成（0%）では 5.4%のスループット改善なのに対し、FSO リンクによるラック間のトポロジの再構成を導入した場合、再構成可能リンクの割合が 100%の時最大で 25.9%の改善となった。また、再構成可能リンクの割合が 50%の場合においても 21.9%の改善を達成することができた。

(2) MongoDB トレースを用いた評価（SSD へのアクセスに偏りがある場合）: (1) と同様のトレースを流す。ホストとなるノードから SSD へのアクセスが正規分布に従い特定の SSD へ偏っている場合において評価を行う。正規分布のパラメータは平均 672、標準偏差 180 とした。図 6 に評価結果を示す。既存手法であるラック内のみの再構成 (0%) では 7.3%の改善、再構成可能リンクの割合が 100%のとき最大で 22.7%の改善となった。

(3) GPU 文字列比較トレースを用いた評価 : ホスト-GPU 間通信を想定した評価で、63.3s 分のトレースの内、中間時間の 3s 分のトレースを抽出してシミュレータに流す。評価結果を図 7 に示す。この条件においてはネットワーク再構成の効果はほとんど見られなかった。これはアプリケーションにおける通信密度の影響によるものである。改善の効果が見られた MongoDB トレースの 1s あたりの通信量を計算すると 3,546,740flit/s であるのに対して、GPU 文字列比較トレースの 1s あたりの通信量は 111,362flit/s となっており、通信量が約 1/32 となっていた。GPU で文字列比較を行うアプリケーションでは、実行時間に対してホスト-GPU 間の通信量が少ないため、ネットワーク内の通信に注目した本研究においてはあまり効果が見られないワークロードとなった。

(4) Chainer トレースを用いた評価 : ホスト-GPU 間通信を想定した評価で、34.9s 分のトレースの内、中間時間の 3s 分のトレースを抽出してシミュレータに流す。評価結果を図 8 に示す。こちら評価 (4) と同様に GPU を用いたアプリケーションにおける評価ではあるが、トポロジ再構成の効果は確認できた。Chainer トレースの通信量を計算してみると 2,140,226flit/s となっており、MongoDB トレースに比べると少ないものの、ある程度の通信が発生しているため、トポロジ再構成による改善が達成できた。

4.3.3 ホップ数の評価

評価条件 (1) の MongoDB トレースを用いたシミュレーションにおいて、トポロジ再構成の効果としてホップ数の評価を行う。図 9 の横軸は前節のスループット評価のグラフと同様にパケットの注入量を、縦軸は受理されたパケット毎に転送されるのにかかったネットワーク上でのホップ数の平均を表している。

図 9 を見ると、再構成可能リンクの割合が大きくなるほど平均ホップ数が削減されていることが分かる。図 9 の横軸は図 5 と同様の値でパケットの注入量を表しているため、グラフの左側の注入量が小さいときの評価結果に注目すると、図 5 のスループットの評価ではほとんどネットワーク再構成の効果が見られないのに対して、図 9 の平均ホップ数の評価では平均ホップ数の削減が確認できる。これは、パケットの注入量が小さくスループットとしてはネットワーク再構成の効果が無い場合においても、ホップ数の削減という形でネットワーク性能の改善が達成されていると言える。

5. まとめ

本論文では、RSA におけるラック間をまたぐハードウェアコンポーネント間通信に FSO リンクを導入し、ラック間トポロジを動的に再構成することでコンポーネント間ネットワークを強化する手法を提案した。その効果を検証するために、まず、商用の PCI-Express over 10GbE 技術を用いてラック間をまたぐ RSA 環境を構築し、その環境で動作する 1)MongoDB、2)GPU

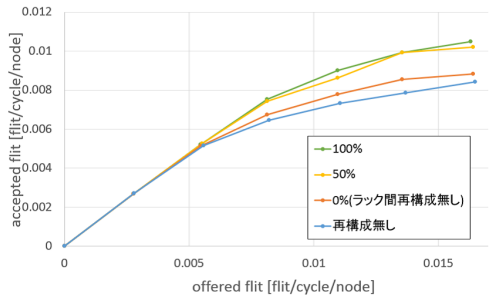


図 5　MongoDB トレースを用いた評価（特殊条件無し）

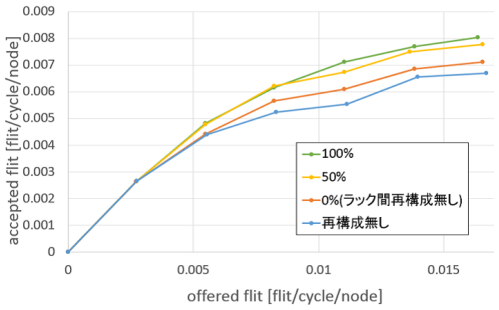


図 6　MongoDB トレースを用いた評価（SSD へのアクセスに偏りがある場合）

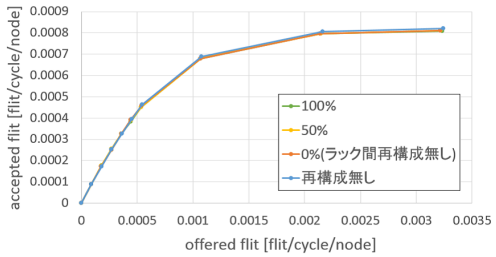


図 7　GPU 文字列比較トレースを用いた評価

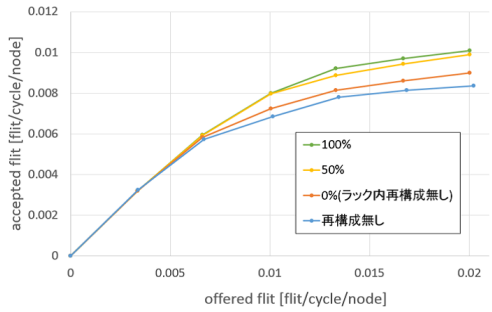


図 8　Chainer トレースを用いた評価

による文字列比較、3)Chainer を用いたものをアプリケーションとして、実行における通信トレースを取得した。取得したトレースを基にネットワークシミュレータを用いた評価を行い、計算コンポーネント間通信に FSO リンクを用いたトポロジ再構成の効果を評価した。その結果、ネットワーク全体のスループットとして、静的トポロジに比べて最大 25.9%の改善（ラック内のみを再構成する既存手法では 5.4%）が確認できた。この効果はラック間の再構成可能リンクの割合を 50%と制限した場合においても 21.9%の改善を確認できた。また、ネットワーク

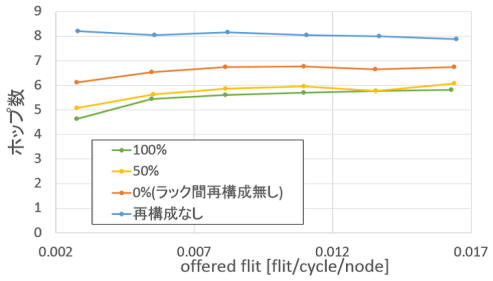


図 9　MongoDB トレースを用いたときのホップ数の評価

全体の通信量が少なく、スループットにおける改善が見込めない場合においてもホップ数削減の効果が確認できた。

謝 辞　本研究開発は総務省 SCOPE(受付番号 152103004) の委託を受けたものです。

参考文献

[1] 尾崎友哉，鯉渕道紘，天野英晴，松谷宏紀：光無線リンクとオンオフ制御による省電力ネットワーク設計手法，電子情報通信学会論文誌，Vol. J98-D, No. 6, pp. 1005–1018 (2015).

[2] Fujiwara, I., Koibuchi, M., Ozaki, T., Matsutani, H. and Casanova, H.: Augmenting Low-latency HPC Network with Free-space Optical Links, *Proceedings of the IEEE International Symposium on High-Performance Computer Architecture (HPCA'15)*, pp. 390–401 (2015).

[3] Kyathsandra, J. and Dahlen, E.: Intel Rack Scale Architecture Overview, *Interop* (2013).

[4] Zang, D., Cao, Z., Liu, X., Wang, L., Wang, Z. and Sun, N.: PROP: Using PCIe-Based RDMA to Accelerate Rack-Scale Communications in Data Centers, *Proceedings of the International Conference on Parallel and Distributed Systems (ICPADS'15)*, pp. 465–472 (2015).

[5] Legtchenko, S., Chen, N., Cletheroe, D., Rowstron, A., Williams, H. and Zhao, X.: XFabric: A Reconfigurable In-Rack Network for Rack-Scale Computers, *Proceedings of the USENIX Symposium on Networked Systems Design and Implementation (NSDI'16)*, pp. 15–29 (2016).

[6] Suzuki, J., Hayashi, Y., Kan, M., Miyakawa, S. and Yoshikawa, T.: End-to-End Adaptive Packet Aggregation for High-Throughput I/O Bus Network Using Ethernet, *Proceedings of the IEEE International Symposium on High-Performance Interconnects (Hot Interconnects 22)*, pp. 17–24 (2014).

[7] 光無線通信システム推進協議会：`http://j-photonics.org/icsa/`.

[8] Hamedazimi, N., Qazi, Z., Gupta, H., Sekar, V., Das, S., Shah, H. and Tanwer, A.: FireFly: A Reconfigurable Wireless Datacenter Fabric using Free-space Optics , *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM'14)*, pp. 319–330 (2014).

[9] IEEE Standards Association: IEEE Std 802.11ad — Enhancements for Very High Throughput in the 60 GHz Band (2012).

[10] Halperin, D., Kandula, S., Padhye, J., Bahl, P. and Wetherall, D.: Augmenting Data Center Networks with Multi-Gigabit Wireless Links, *Proceedings of the ACM Special Interest Group on Data Communication (SIGCOMM'11)*, pp. 38–49 (2011).

[11] NetFPGA Probject: `http://netfpga.org/`.

[12] 藤原一毅，Fichet, A.，鯉渕道紘：マシンルームにおける空間光通信端末のレイアウト解析，情報処理学会研究報告 2014-HPC-144, no.15，pp. 1–7 (2014).

[13] Arimoto, Y.: Near field laser transmission with bidirectional beacon tracking for Tbps class wireless communications, *Proceedings of the SPIE Free-Space Laser Communnication Technologies*, No. 7587, pp. 1–8 (2010).