Multipath TCP を用いたデータセンターネットワークの改善

藤居 翔吾† 田崎 創‡ 関谷 勇司‡
† 東京大学大学院工学系研究科
‡ 東京大学情報基盤センター

クラウド型のサービスの性質により, 今日のデータセンターではデータセンター内のトラフィックが増大しており, 既存の TCP を拡張した Multipath TCP(MPTCP) により, 通信性能向上を目指す取り組みがされている. しかし, 様々なニーズを抱えるトラフィックが混在している中で, レイテンシ志向なサイズの小さいフローに対し, MPTCP が性能を劣化させる問題が報告されている. そこで本論文では, この問題に対し, 並列分散処理アプリケーションが稼働しているクラスター PC のトラフィックを観測する事で, 単一 NIC(Network Interface Card) への通信集中による遅延の影響がある事を示し, MPTCP によるデータセンターモデルのように複数の NIC を持つノードが存在する環境で, 通信経路を切り替えることにより NIC の負荷を分散させる手法を提案した. この手法による効果について, 中継スイッチとエンドノードに対してスループット, フロー完結時間の二つのメトリックを用いた検証を行い, 改善手法が与える影響を考察する.

Improving the datacenterr network with Multipath TCP

Shogo Fujii† Hajime Tazaki‡ Yuji Sekiya‡
†The University of Tokyo, Graduate School of Engineering
‡The University of Tokyo, Information Technology Center

As increasing the amount of traffic in a datacenter by cloud service, the effective network for utilization of massive computer clusters has been studied. Recently, Multipath TCP(MPTCP) has been tackled this problem. MPTCP can achieve the effective consumptions of the resources with multipath, but a researcher reported MPTCP causes the delay of flow completion for short flows. In this paper, I presented measurements of the distribute processing cluster PCs and reveal impairments that lead to that latencies, rooted in single NIC(Network Interface Card) with intensive load traffic, and proposed the method balancing the load for multiple NIC, in such a MPTCP datacenter model. I verified the effect of the method for switch and end-node with two metrics, FCT(Flow completion time) of short flows and throughput of backgroung flows, and considered the effects of the load balancing.

1 導入

今日の一般家庭のインターネット接続環境がギガビット級の速度に達しようとしている中, 多様の端末がインターネットに接続できるようになり, 大量かつ多種多様なデータの取得が可能となった. 特にトラフィックデータ量の増加傾向は顕著で, 18~24 ヶ月単位で総データ容量が 2 倍になるという予測がされている [1]. また Facebook では, 300 ペタバイト以上のデータ量を保有しており, 1 日あたりに 1 ペタバイトのデータを解析している [5]. このように近年では, ビッグデータの活用が着目され, 例えばウェブ検索エンジン, SNS(Social Networking Service) などのデータセンターを用いたクラウド型サービスにおいて, リアルタイムに近いレ

スポンスを返すような場面において使われ始めている. そのようなクラウドサービスには近年, より高いユーザーエクスペリエンスが要求されており, Amazon では 100[ms] の遅延により売り上げが 1% 下がる, といった報告 [2] があるように, 例えば e コマースサイトでの商品購入や, インターネット広告のコンバージョンのようなユーザの意思決定へのレスポンス遅延の影響は深刻な問題となっている [3]. そのため, 大規模データをより高速に処理することが求められており, データセンターではサーバ運用台数が増加の一途を辿っている. そうした中で, 可用性, 計算性能, 低コストの三つの要件がデータセンターの抱える課題となっている [4]. 特に計算性能について, 大量の計算機資源から最大限の性能を引き出すためには, 従来の仕組みではデータセンター内トラフィックに対して一部の資源にトラフィックが集中する問題に対応できないため, 計算機資源を有効活用するための研究が盛んに行われている [6–14]. そのようなスケーラビリティ拡大には, ネットワークトポロジー, アプリケーション, プロトコルに対する三つのアプローチがある.

ネットワークトポロジーを改良するアプローチでは, 従来の単純な階層構造では, データセンター内で発生するトラフィックに対して帯域が最大限割り当てられない [8]. そのため, 近年ではそのようなトラフィックに対してスイッチを多段に構成することで帯域を有効利用するトポロジーが提案されている [8–10].

大量データの処理速度を改良するアプローチでは, 分散処理のために partition-aggregate 計算モデルが提案されている. MapReduce [6], Dryad [7] 等の並列分散処理フレームワークは, この計算モデルに従っており, 今日の大規模クラウドサービスにおいては, 必要不可欠である.

プロトコルを改良するアプローチでは, 従来の TCP を拡張した Multipath TCP (MPTCP) [16] をデータセンターネットワークに用いる提案がされている [8–10]. MPTCP を用いることにより, OS の制御によって複数の NIC(Network Interface Card), 複数の経路を同時に利用し, スループットを向上させることが期待されている.

しかし, 並列分散処理フレームワークを用いることで, フローサイズの小さい大量のクエリーが発生し, MPTCP は, フローサイズの小さいトラフィック (ショートフロー) に対しては, TCP よりも性能が劣化する問題が報告された [12].

このような背景から, 大規模データセンターネットワークへの MPTCP 適用時のショートフロー遅延の問題は並列分散処理アプリケーションの性能の面で深刻な問題である. そこで本論文では, データセンターにおけるショートフロー遅延の問題を解決する為に, 二つの検証を行った.

一つは, 実環境の並列分散処理アプリケーションが稼働しているクラスター PC を用いて, 実トラフィックの観測, 解析を行った. クラスター PC がジョブを実行していない定常状態時と並列分散処理を実行している時の二種類のトラフィックを解析することで, 並列分散処理アプリケーションが生成するトラフィックパターンの特徴および, 汎用的なネットワーク機器を用いた低コストなデータセンターにおける要求案件を検証した. そして, ショートフロー遅延が生じる状況と既存のネットワーク設計の背景について, ボトルネックとなりうるネットワーク環境を検討し, その原因を示した.

二つ目は, MPTCP を用いたデータセンターネットワークモデルにおける, 複数の NIC, 複数の経路を利用した経路切り替えによる改善手法を提案する. これは, 複数の経路を利用しスイッチやエンドノードの持つ複数の NIC へと負荷を分散させることで, 単一の NIC に負荷が集中することによる, キューイングやプロトコル処理の遅延を抑える事ができる, という仮定

に基づき, 提案する手法である. この手法により, 並列分散処理におけるショートフローのような, 低レイテンシが求められる通信においては, 他のフローの影響を受けずにすぐに割込み処理が行われ, 受信処理の CPU 負荷の分散を期待し, その効果をシミュレーション, 実環境での実験を用いて検証した. そして今後の課題として, 経路切り替えのアルゴリズム検討する.

2 関連研究

本章では, これまでに報告されている複数経路利用によるフロー完結時間短縮化技術について簡潔に述べ, その優位性や問題点を示す.

2010 年に Alizadeh らによって, データセンターネットワーク特有のトラフィックパターンに特化して, パラメータを決定するアルゴリズムが提案された [11]. データセンター特有のバースト性のあるトラフィックが引き起こす問題点として, キューの生成によるキューイング遅延, キュー溢れによるパケットロス, スイッチのバッファに掛かる負荷がある. これらの問題に対し, 経由するスイッチにおいて, ECN(Explicit Congestion Notification) によってエンドホストに輻輳を通知し, キューの大部分が占有される前にウィンドウサイズを動的に変化させ, キューサイズを小さく保つアルゴリズムによって, 大部分のキューの伝送時間を短縮することを可能にした. しかし, 大規模計算資源を想定したトポロジーにおける検証がされておらず, また各ネットワークデバイスに細かなチューニングを必要とするため, 大規模データセンターでは運用面での問題がある. さらに, サイズの大きいフローの割合の大きいトラフィックの中では, その影響によりウィンドウサイズを小さくする制御が働くため, サイズの小さいフローの完結時間への効果が小さい, と報告されている [14].

2011 年に Costin らによって, MPTCP を用いたデータセンターネットワークモデルが提案された [12]. 近年の大規模計算資源を有効活用するために提案されたネットワークトポロジーでは, 高性能なデバイスや特殊な機器を必要とせず, 汎用的なネットワーク機器のみを用いてデータセンター内のエンドノード同士の通信に経路が複数用意されている. 既存の取り組みでは, 通信に使わない経路をセカンダリ経路として利用することで, 耐障害性を持たせていたのに対し, 提案されたデータセンターモデルでは, MPTCP を用い複数経路を同時に利用する事で, 耐障害性を保ちながら, 帯域を最大限利用する事を可能にした. また, 様々なトポロジーに MPTCP を適用することで, 従来の TCP よりも高いスループットが出せることを示した. しかし, MPTCP と Single path-TCP が混在する環境において, Single path-TCP で行われるサイズの小さいフロー ($\leq 70KB$) のフロー完結時間に着目すると, 従来の Single path-TCP のみのネットワーク環境よりも時間がかかるという問題点があった.

2012 年に Zars らによって, 複数レイヤー間でトラフィックを監視し, しきい値を設定することによるフロー完結時間の短縮化技術を提案した [13]. 今日のデータセンター内ネットワークのような, サイズの異なるフローが混在するネットワークにおいては, サイズが小さいフローがサイズの大きいフローに圧迫され, 伝送遅延が大きくなる問題があったが, この提案手法では, データリンク層からアプリケーション層までの各層が, 相互にトラフィックを監視する機能をスイッチに実装し, 優先度をつけ, バッファサイズを調整することで, フロー完結時間の劣化を抑えることを可能にした. しかし, 実験では Click [15] を用いて実装を行っており, 現実世界での全てのネットワーク機器の置き換えが必要となるので, 実現は難しい.

以上で述べたことをまとめると, 近年のデータセンターネットワークに対して, 以下のような要件が考えられる.

- 大規模計算機を有効活用するトポロジーの利用
- 分散処理の際に発生する大量のサイズの小さいフローの送信時間の短縮
- 特殊な実装やデバイスを用いず, シームレスな運用の実現

3 データセンターネットワーク

本章では, データセンターネットワークを構成する技術に関して, その概要を述べる.

3.1 Multipath TCP

MPTCP は, 一つの経路でデータ転送する TCP を拡張し, 複数のインタフェース, 複数のポートを用いてデータ転送をするプロトコルである [16]. クライアントが複数の IP アドレスを持っていた場合, 新たにサブフロー[*1]のコネクションが確立される. 追加されたサブフローは, インターフェース毎に異なる IP アドレスの組み合わせで通信を行う. ルーティングに関しては, 複数の宛先, 送信元 IP アドレスのペアからそれぞれ経路決定される. このように, アプリケーション層より下のレイヤーのみで複数の経路を使ってデータ転送を行うため, アプリケーション側が MPTCP での通信を意識することなくデータ転送ができる.

さらに MPTCP では, TCP と同様に AIMD(additive-increase and multiplicative-decrease) による輻輳制御がサブフロー単位で行われ, 経路状態に合わせて輻輳制御をする [17].

3.2 FatTree トポロジーと MPTCP によるデータセンターモデル

この節では, データセンターを構成する要素について述べる.

3.2.1 トポロジー

従来のデータセンターモデルでは, Host が Edge スイッチにつながり, これらのスイッチが Aggregation スイッチに集約され, core スイッチに接続するといったように, 階層的に二分木トポロジーを形成していた [8]. このような単純な階層構造を持つトポロジーは, トラフィックの大部分がデータセンター外の通信には有効であった. しかし, 今日のようなデータセンター内で生じるトラフィックが大半を占める場合, 上の階層にある広帯域の経路を使わない通信が増え, 帯域の割当が不適切である. このような, データセンター内のトラフィックが主であれば, 階層型のトポロジーはボトルネックを引き起こす可能性がある. 近年の研究 [8–10] では, トラフィックがデータセンター内に集中した時の問題を, 物理的なアプローチとして, トポロジーを工夫する事で解消を試みている.

図 1 のように, FatTree [8] では, Core スイッチを複数用いる事で, 物理パスの最大帯域を供給する. また, 比較的狭い帯域の経路と汎用的な性能のスイッチを多数用い, データセンター内のエンドノード同士の通信で複数の経路を実現する事で, 冗長化, ネットワーク資源の効率的な利用, 低コストを実現している.

しかしこのような密な配置により, 複数の経路が形成され, ルーティングをどのように決定すべきかという問題も生じる事となる [12]. 例えば図 1 のような FatTree トポロジーでは, 4 通りの経路が考えられる. これら複数の経路をリンクエラー時の冗長性を持たせる目的だけでなく, 性能向上に活用することが求められている.

[*1] 複数の TCP コネクションの内, ある一つのコネクションにおけるフロー

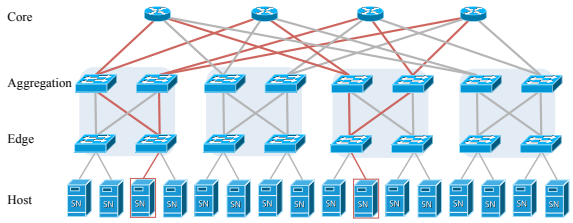

図 1　Fattree トポロジー

4　データセンター内トラフィック

4.1　データセンターにおけるトラフィックシナリオ

大量の計算機資源を有効活用するためには, 並列並列分散処理フレームワークを用いられ, 一般的に図 2 に示すように partition-aggregate 構造をとる. 並列並列分散処理フレームワークでは, 多数の処理ノードと分散処理の制御をする管理ノードから構成されており, 管理ノードからクエリーが発行され, 処理ノードがそれを受け取り, レスポンスを返す. このとき, トラフィックパターンが (1)*Query traffic*, (2)*Short message traffic*, (3)*Backgroung traffic* の 3 つに分類される [11].

Query traffic. Query traffic とは, 大規模計算処理を分割して並列処理を開始する際に, aggregator ノードから処理ノードへ具体的な処理を割り当てるためのトラフィックである. Query traffic の特徴は, 非常に小さいフローサイズ (2KB~20KB) で, フローの役割上, 処理全体の遅延に非常に強く影響を及ぼす事である. そのため, アプリケーション性能を考慮すると, 低レイテンシでの通信が求められている. また並列分散処理システムの構成上, Query traffic は ms~ μs 単位で Query が生成され, バースト性があるといえる [11].

Short message traffic. Short message traffic とは, 処理ノードの動作を制御するためのトラフィックである. Short message traffic の特徴は, フローサイズは 50KB~1MB で, Query traffic と同様に処理全体の遅延に影響を及ぼすという事である. しかし, Querry traffic ほどのフロー数は生成されず, 生成時間間隔も秒単位である.

Backgroung traffic. Backgroung traffic は, 各処理ノードへ更新データを送信するトラフィックである. Backgroung traffic の特徴は, フローサイズが 1MB~50MB と大きいことにある. さらに, その生成時間間隔は大きい. また, Backgroung traffic での更新データは, 処理精度の向上に寄与するが, 処理に必須ではないので, 処理全体の遅延にはつながらない.

つまり, 分散処理開始時に生成される Query traffic が遅延すると, 処理全体に対し遅延を引き起こすので, Query traffic のフロー完結時間は極めて重要なメトリックである.

また, Alizadeh らは, 実際のデータセンターのトラフィックでは, レイテンシ志向なショートフローとスループット志向なロングフロー, そしてバースト性のある Query traffic が混在していると報告している. さらに, Background traffic が発生する数は少ないが, 全体の通信量の大部分が Backgroung traffic によって占められていると報告しており, 経路全体へ影響を及ぼす可能性があることを示している [19].

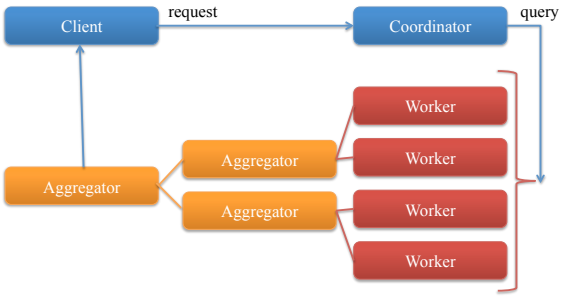

図 2　並列分散システムにおける Partition-aggregate モデル

4.2 並列分散処理アプリケーションが生成するトラフィック

次に, クラウド型サービスを想定したトラフィックの一例として並列分散処理アプリケーションを用いた二種類のトラフィックの測定結果を示す. 測定結果からトラフィックの特徴を示す事で, なぜ Single-path TCP では性能が出せないのか, なぜ複数の NIC, 経路を持つ MPTCP によるデータセンターモデルが必要なのか, 提案手法の需要を裏付けるものとなる.

測定環境としては, 管理ノード 1 台 (10.10.255.10), 処理ノード 10 台の計 11 台のクラスター PC を用いた. 管理ノードは 10Gbps イーサネットリンクで Top of Rack(ToR) スイッチに接続されている.

このクラスター PC で Presto [5] によりインタラクティブなレスポンスを返す, 分散 SQL データベースを実現しており, 4.1 節で示した三種類のトラフィックが混在している. トラフィックの測定には, 管理ノードのインターフェースを用いて, tcpdump によるパケットレベルの測定を行った.

定常状態. 管理ノードに対し, ジョブ命令を一切与えていない中で約 10 時間程度トラフィックを測定した. 図 3 に定常時のフローサイズの確率分布を示す. この分布が示すように, ショートフローの数が全体のトラフィックの大部分を占めている. 実際, 80% 以上のフローが 10KB 以下であった. 一方で, 通信量に着目すると, フロー数は比較的少ないがフローサイズの大きいトラフィックが大半を占めている.

次に, 図 4 に管理ノードへのトラフィックの影響を示す. この分布が示すように, 各処理ノードから管理ノードへのトラフィックの割合が大きく, それぞれフローサイズも大きい. 一方で, 管理ノードから各処理ノードへのトラフィックについては, 比較的フローサイズの小さいトラフィックの割合が大きい.

さらに, 図 5 に時間毎の同時接続数の分布を

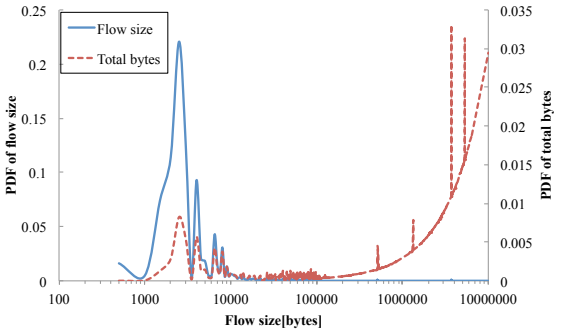

図 3 Presto クラスタの定常時のトラフィック分布

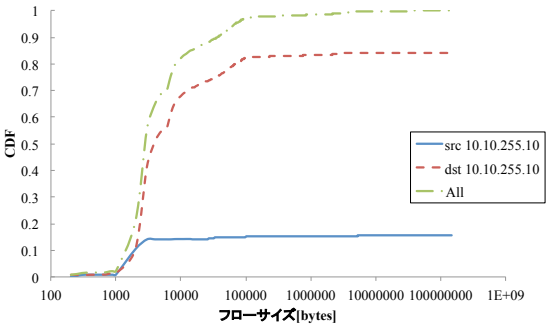

図 4 管理ノードから見た定常時のトラフィック累積分布

示す. この分布が示すように, 各処理ノードから管理ノードへのトラフィックの同時接続数が多く, 積極的に通信が行われている. また, 短い通信時間でスパイク性のある中で, 長時間通信を行うフローが固定的に存在している.

並列分散処理実行時. 管理ノードに対し, 約 1 分間程度で完了する SQL ジョブを与えた中でジョブが完遂するまでの間トラフィック測定を行った. SQL ジョブには, “$select * from$ テーブル $where$ 条件”を実行し, 全ての処理ノードがジョブを与えられるようにした. 図 6 にジョブ実行時のフローサイズの確率分布を示す. この分布が示すように, ショートフローの数が全体のトラフィックの大半を占めるが, 定常状態と比べると, 全体的にフローサ

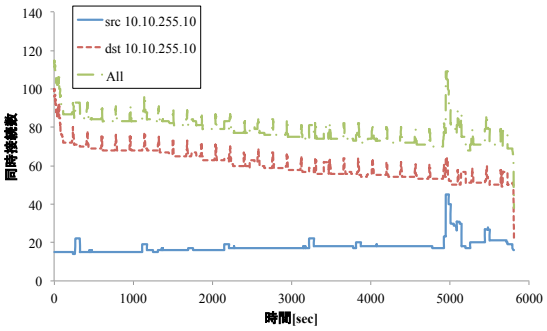

図 5　定常時トラフィック:同時接続数の分布

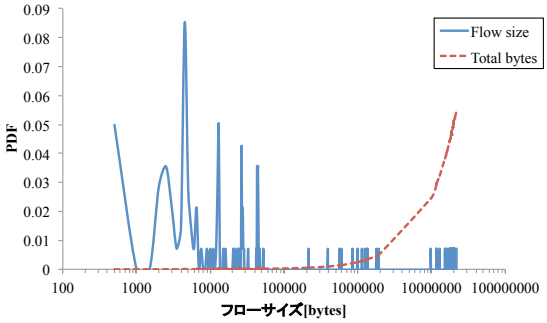

図 6　Presto クラスタのジョブ実行時のトラフィック分布

イズ大きいトラフィックが増えている．実際，80% 以上のフローが 110KB 以下であったように，ショートフローの割合が小さくなった．同様に通信量に着目すると，フロー数は比較的少ないがフローサイズの大きいトラフィックが大半を占めるという事が分かる．

次に，図 7 に管理ノードへのトラフィックの影響を示す．この分布が示すように，各処理ノードから管理ノードへのトラフィックの割合が大きく，フローサイズは小さいものが多いことが分かる．しかし，図 4 の定常時のトラフィックと比べると，管理ノードから各処理ノードへのトラフィックの割合が大きくなっている．

次にに，図 8 に時間毎の同時接続数の分布を示す．この分布が示すように，ジョブ実行中は全体的にフローの数は増え，とりわけ管理ノードから各処理ノードへのトラフィックの割合が大きくなっている．さらに，ジョブ終了後も同時接続数が大きく変化していないことから，長時間通信を行うフローが固定的に存在している．また，各処理ノードから管理ノードへのトラフィックに着目すると，ジョブ開始時に接続数が大きく増えている事から，バースト性があるトラフィックであるといえる．

これらの分布から，クラウド型サービスを想定したトラフィックの一例として並列分散処理アプリケーションを実行した際の特徴として以

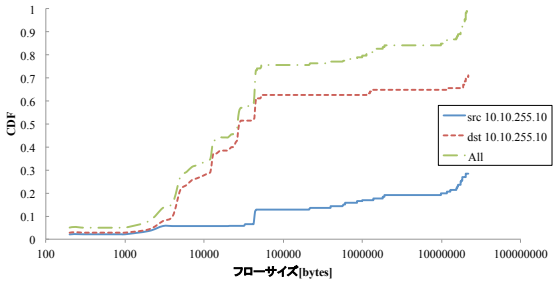

図 7　管理ノードから見たジョブ実行時のトラフィック累積分布

下の事が述べられる．

- 定常時もジョブ実行時も同様に，管理ノードへ送信されるトラフィック量は多い
- 長い時間通信を行うフローが固定的に存在している
- ジョブ実行時の処理ノードから管理ノードへのトラフィックには，フローサイズも小さく，バースト性がある

これらの特徴から，管理ノードへのトラフィックが集中する問題，ショートフローのバースト性の問題，そして長時間通信を行う Backgroung traffic の問題が生じていると考えられる．従って，大きく二つの管理ノードに対するトラフィックパターンを検討する必要がある．

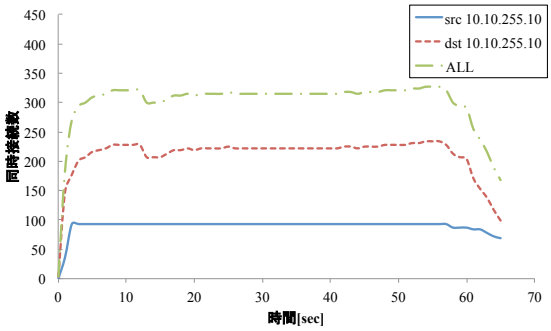

図 8　ジョブ実行時トラフィック:同時接続数の分布

1. ジョブ開始時のバースト性のあるショートフロートラフィック
2. アプリケーション性能に直接影響しない Backgroung traffic が通信している中で, 低レイテンシ通信が求められているショートフローの通信

次に, これらのトラフィックパターンが引き起こす可能性のある二ヶ所のボトルネックについて検討する.

4.3　スイッチ - 性能障害

現在のスイッチ機器では複数のフローを多重に扱うための共有メモリを持つ. そして共有メモリプールから MMU(Memory Management Unit) によって各インターフェースが利用できるメモリ量を動的に割り当てる事で, 複数の通信を公平に処理する事を目指す. しかし, 比較的安価なスイッチでは制御できるメモリ量が制限されているため, 様々な性能障害を引き起こす [27].

4.3.1　Incast

図 9(a) に示すように, 短期間に一つのインターフェースへとフローが集中した場合, 用意されているバッファを使い果たし, 最悪の場合パケットロスを引き起こす. これは, 4.1 小節で示した partition-aggregate 構造によるもので, リクエストを受けた処理ノードが同期して一斉にレスポンスを返すことにより, そのレスポンスを集約して受け取るノードが接続しているスイッチでのポートのキューサイズが大きくなる. こうした問題に対して, アプリケーションレベルにおいては二つのアプローチがある. 一つは, レスポンスのサイズを意図的に小さくし, スイッチバッファの圧迫を抑えることである. もう一つは, それぞれのリクエストにジッタを混ぜる事で, レスポンスを同期させないことである [28]. さらに, パケットロスを生じた際へのアプローチとしては, RTO_{min} を小さくする事でパケットロスの影響を抑える事ができる.

4.3.2　Queue buildup

4.1 小節で示したように, 並列分散処理のレスポンスには直接影響しない Backgroung traffic は, スイッチバッファにパケットロスを引き起こすほどの影響を及ぼし, そのポートがボトルネックとなる可能性がある. 図 9(b) に示すように, Backgroung traffic と Query traffic が同じポートを利用する場合に, サイズの大きいフローによるショートフローのキューイング遅延が生じる, Queue buildup 問題がある. この障害は, パケットロスは生じないため RTO_{min} による改善はできない. さらに, 多数の同期された Query traffic も必要としない. そのため, キューイング遅延の問題に対する唯一の解決策は, キューサイズをなるべく小さく保つことにより, キューに溜まったパケットを素早く排出する事である.

4.4　エンドノード - ネットワーク I/O

今日の GbE(Gigabit Ethernet) 通信において, 割込み処理は大きなボトルネック要因の一つである. 例えば, 1GbE において 64 バイトフレームの最大受信可能数は, 毎秒約 150 万であり, 1 パケット受信する度に割り込み処理を行っていては, CPU リソースが枯渇する. そのため, 割込み処理の回数を抑えることが必要であるが, その分レイテンシが上がる可能性があり, 互いのトレードオフを適切に対処し高い性能を得る必要がある. また, 今日の多くの CPU

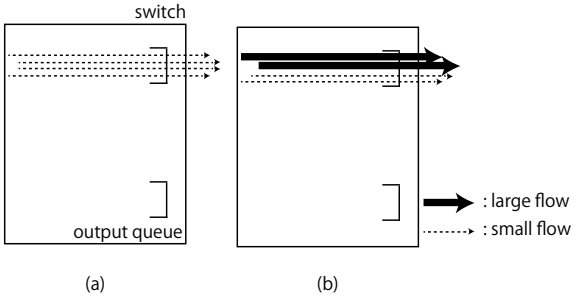

図 9　Two ways in which flows interact on a multi-ported switch resulting in performance problems.

はマルチコアであり, CPU リソースを効率的に利用する事が求められている.

4.4.1　割込み処理

パケット受信の際の NIC によるハードウェア割込みは, 即座に受信処理を行う事ができ, キューイングの遅延を小さくする事ができる. しかし, 割込み処理が増えれば, その分オーバヘッドが大きくなり, 結果的に OS の性能が劣化する. 割込み処理を扱う代表的な仕組みとして, ポーリング, interrupt coalescing がある.

ポーリングは NIC の割り込みを使わず, タイマーを使って定期的に NIC の受信キューを監視することで, 割り込み負荷を軽減するソフトウェア技術である. しかし, NIC でパケットを受けてから即座に処理できない為, 遅延が発生する場合がある. 現在の Linux カーネルにおいては, NAPI により, 通信量が多く高負荷時のみはポーリングが作用する [20]

interrupt coalescing は, 複数のパケット, あるいは一定期間待ってからをまとめて一度で割り込ませる事で, 割込み回数を減らすハードウェア技術である. しかし, ポーリングと同様, 即座に処理できない為, 遅延が発生する場合がある.

4.4.2　プロトコル処理

マルチコア環境においても基本的には一つの NIC の受信処理は 1 つの CPU でしか行えない. そのため, ハードウェアへのアプローチとして, 1 つの NIC に複数の受信キューを持たせて, 受信処理をそれぞれの CPU へ分散させている, Receive Side Scaling(RSS) がある [21]. しかし, 一般に複数受信キューを持ち, RSS 機能がある NIC は高価である. そのため, 一つしか受信キューを持たない NIC であっても, 複数の CPU を分散させるソフトウェア技術として, RPS(Receive Packet Sterring) がある [23]. しかし RPS では, プロトコル処理とアプリケーション処理の CPU が異なる場合が生じ, その問題を最適化したのが RFS(Receive Flow Sterring) がある [22]. これらの技術により, CPU の複数のコアをより効率良く利用する事ができる.

また, プロトコル処理やアプリケーションでの処理については, RFS 等で複数の CPU へと分散させる事ができるが, その際の割込み処理についてはオーバヘッドが生じる可能性がある.

5　検証実験

これまでの研究において報告された MPTCP によるショートフロー性能劣化の問題 [12] を受け, その再現実験を行うことにより, 原因を解析し, 二つの要因を明らかにした [26]. 一つ目は, MPTCP は TCP よりも多くのトラフィックを排出し, より多くの NIC インタフェースを利用する事で, 中継スイッチにおいてショートフローが利用するインタフェースと競合し, スイッチでの遅延, パケットロスが生じるということ二つ目は, ショートフロートラフィックのバースト性の問題により, エンドノードで単一の NIC に負荷が集中し, 受信処理の遅延が生じたということこれらの結果を受け, 低レイテンシでの通信が求められるショートフローに対して, MPTCP によるバックグラウンドトラフィックが利用しているインタフェースを回避し, 比較的輻輳が起こっていな

い経路を適切に選ぶ事で, 単一 NIC への通信負荷の問題は解消され, ショートフローのフロー完結時間 (FCT) が改善できるのではないかと, 仮説を立てた. この章では, シミュレーション, 実機での実験を用いて, その仮説の検証, また複数の NIC, 複数の経路を利用した経路切り替えによる改善手法の効果の検証を行う. 具体的には, 中継スイッチとエンドノードへのそれぞれの単一 NIC の負荷について, 複数の NIC を用いて分散させ, その効果を検証する.

5.1 中継スイッチへの NIC 負荷シミュレーション実験

5.1.1 シミュレーション環境

ネットワークトポロジーには, FatTree トポロジーを部分的に抽出した 2:1 にオーバーサブスクリプションされたトポロジーを用いる. 図 10 に, シミュレーションでトポロジーを示す.

ベンチマークトラフィックについては, 二つのペアに対してホスト同士の 1 対 1 通信を用いている. 一方のペアに対しては, シミュレーションを実行している間, 継続してデータ転送 (バックグラウンドトラフィック) を行う. 他方のペアに対しては, TCP による 70Kbyte のデータ転送 (ショートフロー) を毎 50[ms] の一様生起させ, 転送完了までにがかかった時間 FCT(Flow Completion Time) を計測している. ショートフローのルーティングに関しては, 3 つのパターンを用いて中継スイッチへのインターフェースへの負荷の影響を検証する.

今回の実験には ns-3 Direct Code Execution [18] を用いた. 表 1 に再現シミュレーション環境に対する各パラメータをまとめる.

Parameter	Value
Nodes	4
Link edge-edge	200Mbps
Link edge-host	100Mbps
RTT	0.5ms
Receive buffer size	500KB

表 1 シミュレーション環境パラメータ

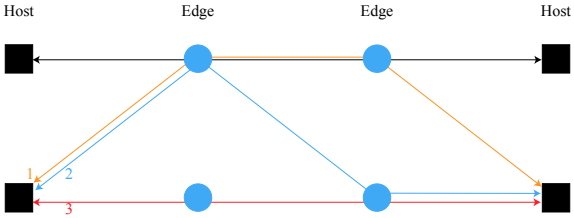

図 10 中継スイッチへの NIC 負荷シミュレーション実験トポロジー

5.1.2 設定パラメータに対する有効性の検証

伝搬遅延については RTT(Round Trip Time) として, 0.5[ms] に設定した. これは, 一般的なデータセンター内の RTT が 1[ms] 以下であるためである [25].

バッファサイズについては, 以下の帯域幅遅延積 (BDP) の式から, 200Mbps を最大限利用できるだけの値を設定した.

$$BDP[\text{byte}] = \text{帯域幅 [bps]} \times RTT \div 8 \quad (1)$$

各帯域については, 4 のノードを使って輻輳を引き起こす現象を再現するために, 実際のデータセンターのような広帯域のネットワークと比べ, 狭い帯域を設定した.

5.2 検証結果

図 11 に上記の実験環境での結果として, 70KB のショートフローの FCT とバックグラウンドフローの経路利用率を示す. この結果から, ショートフローの通信が, 中継スイッチにおいて, バックグラウンドフローと経路およびインターフェースを共有した事による影響で, FCT の分散が大きくなっている事が分かる. 一方で, 同じスイッチは利用する事になったもののインタフェースは共有しなかったトラフィックや, バックグラウンドフローとは独立に通信を行っていたトラフィックに対しては, 何も影響がなかった. これは, 単一 NIC に対して二つのトラフィックが集中したことによる遅延, あるいは負荷分散の影響であると考えられる. この影響からバックグラウンドフローに

対しても, スループットが下がっている事が分かる.

これらの事から, アプリケーション性能に直接影響しない Backgroung traffic が通信している中で, 低レイテンシ通信が求められているショートフローの通信をする際, 中継スイッチでの利用するインタフェースが競合する場合, 単一の NIC に対しトラフィックが集中する事で, 受信処理の割込みのオーバヘッドや, プロトコル処理の遅延の影響が生じたと考えられる. よって, 中継スイッチにおいて複数の NIC に対し, トラフィックを分散させる事は, レイテンシ志向のショートフローに対しても, スループット志向のバックグラウンドフローに対しても有効であるといえる.

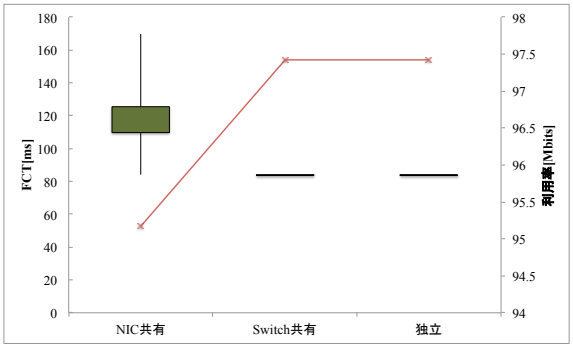

図 11　70kb ベンチマークトラフィックに対するフロー完結時間とリンク利用率

5.3　エンドノードへの NIC 負荷実機実験

5.3.1　実験環境

複数の NIC を用いた負荷分散手法のバースト性のあるショートフロートラフィックに対する影響を検証する. ネットワークトポロジーには, 図??に示すように, 二つの NIC を持った二つエンドノード同士を L2 スイッチを介してそれぞれの NIC 毎に接続した. ルーティングに関しては, それぞれの対をなす NIC 同士が通信を行う. ベンチマークトラフィックについては, ホスト同士の 1 対 1 通信を用い, ショートフローとバックグラウンドフローを通信させる. バックグラウンドフローについては, ショートフローが通信している NIC ペアと同じものを使って共有して通信させるパターンとショートフローが通信している NIC ペアとは異なるペアの NIC を用いて通信を行うパターンの 2 パターンについて検証する. ショートフローは, TCP による 10Kbyte のデータ転送を毎 10ms, 30 フローを同時に一様生起させ, 転送完了までにがかかった時間 FCT(Flow Completion Time) を計測している. バックグラウンドトラフィックは, シミュレーションを実行している間, 継続してデータ転送を行う.

図??に, シミュレーションで用いたトポロジーを示す. 表 2 に実験環境に対する各パラメータをまとめる.

Parameter	Value
OS	Linux 3.13.0
リンク経路	1Gbps
RTT	0.2ms
MAX receive buffer size	6144KB
MAX ring buffer	4096

表 2　実験環境パラメータ

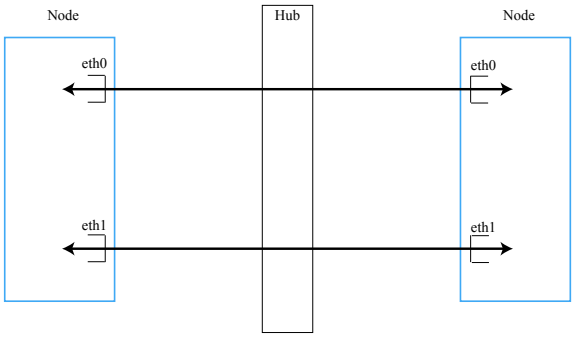

図 12　エンドノードへの NIC 負荷実機実験トポロジー

5.4　検証結果

図 13, 図 14 に上記の実験環境での結果として, 10KB のショートフローの FCT とそれぞれの経路の利用率を示す.　この結果から,

ショートフローの通信が, エンドノード間通信において, バックグラウンドフローと経路およびインターフェースを共有した事による影響で, FCT の分散が大きくなっている事が分かる. 一方で, 経路, インタフェースは競合しなかったものの, バックグラウンドフローとショートフローが同時に通信を行ったことで, FCT 下位 25% のフローについては遅延が生じ, 分散が大きくなった. これは, 単一 NIC に対して二つのトラフィックが集中したことによる負荷分散の効果が得られたが, プロトコル処理以降の部分で, 複数のフローが同時に通信を行った事に対するオーバヘッドが生じたと考えられる. またバックグラウンドフローに着目すると, インターフェースを共有した場合においては, ショートフローだけでなくバックグラウンドフローにも遅延が生じ, スループットが低下している.

これらの事から, アプリケーション性能に直接影響しない Backgroung traffic が通信している中で, バースト性のあるショートフロートラフィックの通信をする際, エンドノードに対して, 利用するインタフェースが競合する場合, 単一の NIC に対しトラフィックが集中する事で, 受信処理の割込みのオーバヘッドや, プロトコル処理の遅延の影響があると考えられる. よって, エンドノードにおいて複数の NIC に対し, トラフィックを分散させる事は, バースト性のあるショートフローに対しても, スループット志向のバックグラウンドフローに対しても有効であるといえる.

5.5 考察

これらの解析結果から, エンドノード, スイッチに対する機能障害が引き起こる要因について述べ, 今後の改善手法の検討を行う.

大量の計算機資源をいかに効率的に利用するか, という課題を今日のデータセンターは抱えており, Hadoop のような並列分散処理アプリケーションを用いる事が一般的である. 今の

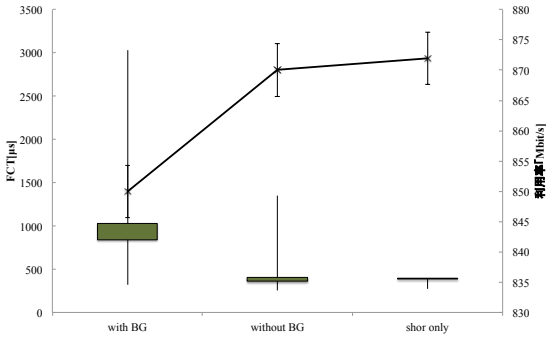

図 13　10kb ベンチマークトラフィックに対するフロー完結時間とリンク利用率 [eth0]

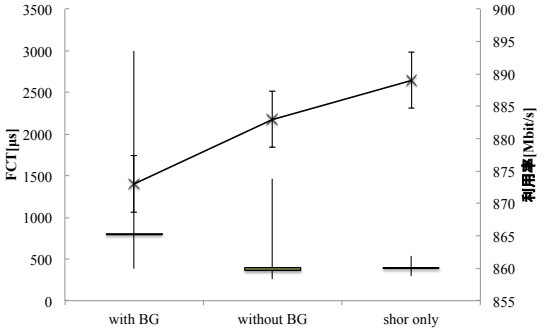

図 14　10kb ベンチマークトラフィックに対するフロー完結時間とリンク利用率 [eth1]

並列分散処理システムが partition-aggrigation 構造である以上, 管理ノードや多段のクラスター構成であればアグリゲーターノードに対して, 処理ノードからのトラフィックが集中する問題は発生する. その結果, **Queue buildup** や **Incast** のような単一 NIC への負荷集中の問題が中継スイッチやエンドノードに対して生じ, 並列分散処理の性能が劣化する.

こうした遅延の影響を軽減する為には, 混雑時の通信量を抑える制御を行う, あるいは混雑時にも空いているリソースを効率良く利用する事が必要である. MPTCP のような, 既存の計算機資源に対して複数 NIC を刺して性能向上を目指すことで, 複数のフローからデータを

受信する際に異なる物理ポートを利用する事で, マルチコアを持つ CPU の効率的な利用につなげられると期待している. すなわち, 複数の NIC に対して通信を分散させるように, トラフィックを制御する事で, 例えばレイテンシ志向なショートフローとスループット志向なバックグラウンドフローのような役割の異なるトラフィックを共存させ, 最適な通信の実現が可能であることを本論文では示した. そのような複数の NIC が介在するネットワークの中で, トラフィックをどのように制御するかという点については, スイッチやエンドノードの OS スタック等のどこで制御をするか, またどのようなアルゴリズムでそれを実現するかという事を検討する必要がある.

6 あとがき

本論文では, データセンターにおけるショートフロー遅延の問題を解決する為に, 二つの検証を行った.

一つは, 実環境の並列分散処理アプリケーションが稼働しているクラスター PC を用いて, 実トラフィックの観測, 解析を行った. クラスター PC がジョブを実行していない定常状態時と並列分散処理を実行している時の二種類のトラフィックを解析することで, 並列分散処理アプリケーションが生成するトラフィックパターンの特徴および, 汎用的なネットワーク機器を用いた低コストなデータセンターにおける要求案件を検証した. そして, ショートフロー遅延が生じる状況と既存のネットワーク設計の背景について, ボトルネックとなりうるネットワーク環境を検討し, その原因を示した.

二つ目は, MPTCP を用いたデータセンターネットワークモデルにおける, 複数の NIC, 複数の経路を利用した経路切り替えによる改善手法を提案する. これは, 複数の経路を利用しスイッチやエンドノードの持つ複数の NIC へと負荷を分散させることで, 単一の NIC に負荷が集中することによる, キューイングやプロトコル処理の遅延を抑える事ができる, という仮定に基づき, 提案する手法である. この手法により, 並列分散処理におけるショートフローのような, 低レイテンシが求められる通信においては, 他のフローの影響を受けずにすぐに割込み処理が行われ, 受信処理の CPU 負荷の分散を期待し, その効果をシミュレーション, 実環境での実験を用いて検証した. そして今後の課題として, 経路切り替えのアルゴリズム検討する.

本論文では, サイズが大きく長時間通信を行うバックグラウンドトラフィックとサイズが小さくフロー完結時間を短く抑えたいショートフローが混在する状況において, フローサイズの小さいトラフィックに対しては, 従来の TCP よりもデータ転送に時間がかかるという問題に対して, 改善手法の効果を検証した. また, 実環境の並列分散処理アプリケーションが稼働しているクラスター PC を用いて, ジョブを実行していない定常状態時と並列分散処理を実行している時の二種類のトラフィックを解析を行い, 単一の NIC に対して通信負荷が集中する問題がある事を示し, 通信経路を切り替える事により, 複数の NIC に対し負荷分散するという改善手法を提案した. この改善手法により, 他のフローの影響を受けずにすぐに割込み処理が行われ, レイテンシ志向なショートフローへの受信処理の遅延を軽減する事ができた. また, スループット志向であるバックグラウンドフローに対しても, 通信性能を向上させる事ができた.

今後はネットワークの通信状況を考慮しながら, 通信経路を切り替えるアルゴリズムの検討を行う.

参考文献

[1] 日本アイ・ビー・エム株式会社. IBM 第 1 章大容量データのバックアップ, http://www-06.ibm.com/systems/jp/storage/column/backup/01.html

[2] Jim Liddle. Amazon found every 100ms of latency cost them 1% in sales, August 2008. http://blog.gigaspaces.com/amazon-found-every-100ms-of-latency-cost-them-1-in-sales/

[3] R. Kohavi et al. Practical Guide to Controlled Experiments on theWeb: Listen to Your Customers not to the HiPPO. KDD, 2007.
[4] J. Hamilton. On designing and deploying Internet-scale services. In USENIX LISA, 2007.
[5] Facebook. Presto: Interacting with petabytes of data at Facebook, https://www.facebook.com/notes/facebook-engineering/presto-interacting-with-petabytes-of-data-at-facebook/10151786197628920
[6] Dean, Jeffrey, and Sanjay Ghemawat. "MapReduce: simplified data processing on large clusters." Communications of the ACM 51.1 (2008): 107-113.
[7] Isard, Michael, et al. "Dryad: distributed data-parallel programs from sequential building blocks." ACM SIGOPS Operating Systems Review 41.3 (2007): 59-72.
[8] Al-Fares, Mohammad, Alexander Loukissas, and Amin Vahdat. "A scalable, commodity data center network architecture." ACM SIGCOMM Computer Communication Review. Vol. 38. No. 4. ACM, 2008.
[9] Guo, Chuanxiong, et al. "BCube: a high performance, server-centric network architecture for modular data centers." ACM SIGCOMM Computer Communication Review 39.4 (2009): 63-74.
[10] Greenberg, Albert, et al. "VL2: a scalable and flexible data center network." ACM SIGCOMM Computer Communication Review. Vol. 39. No. 4. ACM, 2009.
[11] Alizadeh, Mohammad, et al. "Data center tcp (dctcp)." ACM SIGCOMM Computer Communication Review 40.4 (2010): 63-74.
[12] Raiciu, Costin, et al. "Improving datacenter performance and robustness with multipath TCP." ACM SIGCOMM Computer Communication Review. Vol. 41. No. 4. ACM, 2011.
[13] Zats, David, et al. "DeTail: Reducing the flow completion time tail in datacenter networks." ACM SIGCOMM Computer Communication Review 42.4 (2012): 139-150.
[14] Alizadeh, Mohammad, et al. "pfabric: Minimal near-optimal datacenter transport." Proceedings of the ACM SIGCOMM 2013 conference on SIGCOMM. ACM, 2013.
[15] Kohler, Eddie, et al. "The Click modular router." ACM Transactions on Computer Systems (TOCS) 18.3 (2000): 263-297.
[16] Ford, Alan, et al. TCP Extensions for Multipath Operation with Multiple Addresses: draft-ietf-mptcp-multiaddressed-03. No. Internet draft (draft-ietf-mptcp-multiaddressed-07). Roke Manor, 2011.
[17] Raiciu, C., M. Handley, and D. Wischik. "Coupled congestion control for multipath transport protocols." draft-ietf-mptcp-congestion-01 (work in progress) (2011).
[18] Inria「DCE - GETTING STARTED Direct Code Execution」http://www.nsnam.org/~thehajime/ns-3-dce-doc/getting-started.html
[19] Benson, Theophilus, Aditya Akella, and David A. Maltz. "Network traffic characteristics of data centers in the wild." Proceedings of the 10th ACM SIGCOMM conference on Internet measurement. ACM, 2010.
[20] J. Salim, When NAPI Comes to Town, Proceedings of Linux 2005 Conference, UK, August 2005.
[21] Microsoft corporation. scalable networking with rss, 2005.
[22] Herbert, T. rfs: receive flow steering, september 2010. http://lwn.net/Articles/381955/.
[23] Herbert, T. rps: receive packet steering, september 2010. http://lwn.net/Articles/361440/.
[24] ip networking lab「MultiPath TCP - Linux Kernel implementation」http://mptcp.info.ucl.ac.be/
[25] Vasudevan, Vijay, et al. "Safe and effective fine-grained TCP retransmissions for datacenter communication." ACM SIGCOMM Computer Communication Review. Vol. 39. No. 4. ACM, 2009.
[26] 藤居 翔吾, 田崎 創, 関谷 勇司, "MultiPath TCP 適用時のデータセンターネットワークでのフローサイズが与える影響に関する一考察", 電子情報通信学会, 信学技法, vol. 113, no. 364, IA2013-65, pp. 47-52, 2013.
[27] P. Agarwal, B. Kwan, and L. Ashvin. Flexible buffer allocation entities for traffic aggregate containment. US Patent 20090207848, August 2009.
[28] S. Floyd and V. Jacobson. The synchronization of periodic routing messages. IEEE/ACM ToN, 1994.
[29] A. Walid, et al. Balanced Linked Adaptation Congestion Control Algorithm for MPTCP draft-walid-mptcp-congestion-control-00, 2014.