GPU によるソフトウェア OpenFlow スイッチのフロー検索高速化

川光 剣士† 廣津 登志夫†

OpenFlow スイッチでは到着したパケットに対する処理を決定するためにフローテーブルからエントリーを検索する．しかし，エントリー数が増加すると CPU がボトルネックとなり，フロー検索性能が低下する．本研究では，CPU ボトルネックとなる処理を GPU へオフロードすることで，フロー検索時の遅延解消を目指す．提案手法では，検索アルゴリズムとしてトライ木を使用し，GPU 上で並列にフロー検索を実施する．また，トライ木の実装では Double Array を用いるが，エントリー数が増加するに従い，Double Array の構築に膨大な時間がかかるため Double Array を複数個に分割し，並列に構築することを提案する．

1. はじめに

OpenFlow は近年注目されている Software Defined Network(SDN)を実現する主要技術である。OpenFlow プロトコルを使用することで OpenFlow Switch(OFS)にフロールールを指定でき，フロー単位でのパケット制御が可能となる．しかし，フローエントリー数が増加するとフロー検索時に遅延が発生し，転送性能の低下が起こるため，OFS 上でのフロー検索の高速化が望まれている．しかし，OpenFlow ではエントリー内のフィールドにおいて任意のビットマスクを指定可能であるため，従来の IP ルーティングで検討されてきた手法の適用ができず，高い性能を実現することが難しい．この問題に対して様々な研究が行われている．

以下，2 章では関連研究としてトライ木を用いたフロー検索手法について示す．3 章では，GPU を用いた OFS のフロー検索高速化についての提案手法を示す．まず，Double Array 構築を並列に行うことで実現する高速化について示し，後に GPU 内でのフロー検索の設計を示す．4 章では提案手法について評価を行った結果を示し，5 章に本稿のまとめを示す．

2. 関連研究と課題

既存研究[1]では，エントリー内の各ビットをそれぞれ一つのフィールドと考え，ビットマスクを含むフロー検索の問題を L 次元の Packet Classification として扱っている．L は検索を行うヘッダーのビット長である．この既存研究では，トライ木の上半分のみワイルドカードをあらかじめ展開することで検索時のバックトラックを削減する手法を提案し，ビットマスクを持つエントリーに対しての高速なフロー検索アルゴリズムとして有効性が証明されている．また，トライ木実装の際にはメモリ使用量を抑えるために Double Array[2]を用いた実装を行っている．しかし，フロー検索の並列化が行われていないため大量のパケット処理を行うような環境では，転送性能の低下が発生するという問題がある．また，Double Array へのエントリー挿入に膨大な時間を要するため，Double Array 構築時間の削減の必要がある．

3. 提案手法

まず，Double Array 構築時間を削減するためにエントリー挿入の並列化を検討する．Double Array 構築アルゴリズムの特徴から一つの Double Array に対して並列にエントリーを挿入することは出来ない．そこで，Double Array を複数個用意し，CPU スレッドにより並列に Double Array を構築することで構築時間の短縮を図る．このとき，Double Array を分割することでフロー検索時のバックトラックを削減することが可能であり，フロー検索時間の短縮が考えられる．また OpenFlow ではフローテーブル内で複数エントリーにマッチした際には各エントリーに割り当てられている優先

† 法政大学情報科学部
Faculty of Computer and Information Sciences, HOSEI University

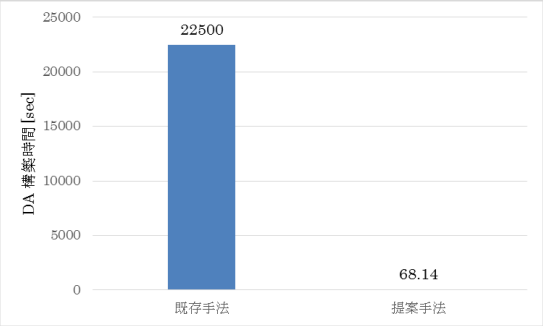

図 1 Double Array 構築時間

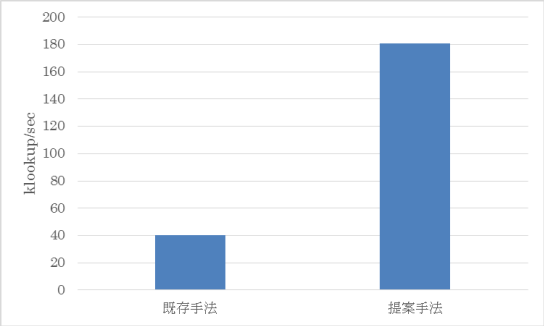

図 2 フロー検索性能

度の高いほうを選択するという仕様になっている．そこで，フローテーブル内のエントリーをあらかじめ優先度の高い順に並び替えることで，複数個存在する Double Array 全てに対してフロー検索する必要がなくなり，エントリーを発見した段階でフロー検索を終了することが可能になる．

次に，フロー検索対象であるパケットヘッダーに対して個数分 GPU スレッドを実行し，フロー検索の高速化を図る．各 GPU スレッドは，まず優先度の高いエントリーが含まれている Double Array から検索を行う．もし，該当するエントリーが発見された場合にはそれ以降の Double Array に対しての検索は行わず，該当するエントリーが発見されなかった場合には次の Double Array に対してフロー検索を行い，該当するエントリーが得られるまで Double Array に対して検索を行う．

4. 評価

ビットマスクが 100 種類である場合に対してエントリー数を変化させ，提案手法のフロー検索性能の評価を行う．ビットマスクはランダムに生成し，フローテーブル内の各エントリーはビット長が 32 ビットであるランダムな値にビットマスクを適用したものとする．評価実験を表 1 の環境で行った．

表 1 評価環境

CPU	Intel(R) Xeon(R) CPU E5520
GPU	GeForce GTX 580
メモリ	DDR3 24GB
OS	Scientific Linux 6.4 x86_64
CUDA Toolkit	6.5

まず，既存手法の Double Array 構築時間と提案手法の Double Array 構築時間の計測結果を図 1 に示す．提案手法では，CPU スレッドを 10 個使用し，並列化を行った．図 1 より提案手法が Double Array 構築時間の短縮に有効であることがわかる．

次に，既存手法と提案手法において，フローエントリー20 万件の中から該当するエントリーを検索する際の性能評価の結果を図 2 に示す．この時既存手法では，1 つの Double Array のみ使用し，提案手法と同一検索アルゴリズムを使用するためトライ木内のワイルドカードを展開しない事とする．図より既存研究と比較して提案手法を用いた場合に約 4 倍高い性能を得られることがわかる．

5. まとめ

OFS のフロー検索高速化のためにトライ木を検索アルゴリズムとして用い，GPU スレッドにより並列にフロー検索を行った．その結果，既存研究と比較して約 2 倍の性能向上が得られた．また Double Array を複数個用意し並列に構築することで既存手法と比較し，大幅に構築時間を短縮することができた．

今後は，ビットの先頭部が同一のエントリーを同じ Double Array に格納することでバックトラックの発生を削減し，フロー検索性能の向上を図る．

参 考 文 献

[1] 日比智也，中島佳宏，高橋宏和，尾花和昭,“ソフトウェア OpenFlow スイッチにおける Bitmask 対応の高速なフロー検索手法の検討，”情報処理学会研究報告，2014-OS-128(12)，pp.1-7，Feb 2014 年.

[2] J. -I. AOE, “An efficient digital search algorithm by using a double-array structure , ”IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL.15, NO.9, 1989

[3] S. Han, K. Jang, K. Park and S. Moon, “PacketShader: a GPU-Accelerated Software Router, ”SIGCOMM ’10 Proceeding of the ACM SIGCOMM 2010 conference, pp.195-206, 2010