学習型インデックスを応用した高速なパケット転送

Forwarding Using Learned Index Structures

小泉 佑揮（Yuki Koizumi, Ph. D.）
大阪大学 大学院情報科学研究科 准教授
(Associate Professor, Graduate School of Information Science and Technology, Osaka University)
ACM IEEE 電子情報通信学会
研究専門分野：情報ネットワーク ネットワークセキュリティ

あらまし

パケット転送は、宛先情報を有するテーブル（FIB: Forwarding Information Base）から、パケットごとに宛先を検索する処理であり、ルーターの重要な役割の一つである。一方で、最近、データベース分野で、学習型インデックスと呼ばれる機械学習に基づく新たなデータ構造が提唱され、そのコンパクトさと検索の高速性から様々な応用が期待されている。しかし、学習型インデックスを FIB 検索に用いるためには、さらなる高速化に加えて機械学習に起因する検索速度のばらつきを抑制する必要がある。本研究では、学習型インデックスの本質が区分線形関数による回帰であることに着目して学習型インデックスを FIB に最適化することで、FIB 用に提唱された最新のデータ構造と同等の検索速度を達成し、かつアドレスのプレフィクス長に依存しない一定の検索速度を実現する。

１．研究の目的

IP（Internet protocol）におけるパケット転送は、各ルーターが、受信したパケットについて、FIB と呼ばれるテーブルに格納されたパケットの転送情報を検索し、パケットの宛先を決定し、決定した宛先情報にしたがってパケットを転送する処理である。FIB の検索処理は、パケット転送の根幹をなす技術であり、その高速化がルーターの高速化に直結することから、FIB 検索の高速化に関する数多くの研究がなされてきた[1, 2, 3, 4]。

一方で、データベース分野において、学習型インデックス（learned index structure）と呼ばれる、機械学習に基づく新たなデータ構造が提案された [5]。学習型インデックスは、キー・バリュー型データベース（KVS: key-value store）において、キーとそれに対応するエントリの格納位置の関係を機械学習で再現するインデックス用のデータ構造である。与えられたキーに対して、対応エントリの格納位置を機械学習で予測し、予測された位置の周辺を局所的に探索することでエントリを発見する。学習型インデックスは、コンパクト性と高速性を両立するデータ構造として期待されている。

本研究は、学習型インデックスの提唱をふまえて、パケット転送に関するデータ構造とアルゴリズムに関する研究に再訪し、学習型インデックスの FIB への適用を目指す。学習型インデックスを用いた FIB を学習型 FIB と呼ぶ。このとき、学習型 FIB を実現する上で、機械学習によるエントリ格納位置の予測に要する時間と、機械学習によるエントリ格納位置の予測誤差を訂正するための局所探索に要する時間の削減が課題となる。これに対して、ニューラルネットワークをベースとする回帰は、区分線形関数による回帰に帰着する点に着目し、最悪時の予測誤差を保証し誤差保証のための局所探索アルゴリズムを最適化する。これにより、既存の FIB 用のデータ構造よりもコンパクトでありながら、同程度の検索速度を実現する。さらに、宛先のプレフィクス長に依存せずに一定時間での FIB 検索を実現する。

２．研究の背景

IP パケット転送は、受信パケットの宛先 IP アドレスをキーとして、FIB からパケットの転送情報が格納されたエントリを検索する処理である。FIB 上のエントリは、Classless Inter-Domain Routing (CIDR)を前提として、集約されている [6] ため、FIB の検索には最長一致検索が用いられる。つまり、ルーターは、パケットを受信するごとに、宛先 IP アドレスと最も長く一致する IP プレフィクスをキーとするエントリを

学習型インデックスを応用した高速なパケット転送

Forwarding Using Learned Index Structures

FIB から探索し、パケットの宛先を決定する。最長一致検索の処理は、単純な厳密一致検索処理よりも複雑であることから、この処理の高速化がパケット転送の高速化の鍵であり、最長一致検索を高速およびコンパクトに実現するFIBのデータ構造や検索アルゴリズムに関する研究［1, 2, 3]、あるいは、最長一致検索をサポートするハードウェア技術を応用する研究 ［4］がなされた。

FIB のデータ構造や検索アルゴリズムに注目すると、高速な検索を実現するFIBのデータ構造を実現するためのアプローチは大きく2つに大別できる。それは、第一に、検索速度自体が高速であること、第二に、SRAM（CPU キャッシュ）など小容量ながら高速なアクセスが可能なメモリに搭載可能なコンパクトさを有することである。これらの観点から、多くの研究が、高速な検索とコンパクト差を両立できるトライと呼ばれる木ベースのデータ構造を用いている［1, 3]。

一方で、パケット転送の研究分野とは独立に、データベース分野において、Kraska らによって、機械学習を応用したデータベース用のインデックス構造である学習型インデックス ［5］ が提案された。インデックス構造とは、KVS 型のデータベースにおいて、あるキーとそのキーに対応するエントリーの格納位置を保存するデータ構造であり、データベース分野ではB木やハッシュテーブルなどが用いられる。これに対して、学習型インデックスではキーと対応するエントリーの格納位置の関係性を機械学習の回帰モデルで再現するアプローチである。学習型インデックス構築時は、キーと対応するエントリの格納位置の関係を機械学習の回帰モデルで学習する。検索時は、回帰モデルを用いてキーに対するエントリが格納されている位置を予測し、予測誤差を保証するために予測された位置の周辺を局所的に探索して真のエントリ格納位置を探索する。学習型インデックスは、コンパクトながら、近年の機械学習向けの計算高速化技術を応用することで高速にエントリの検索が可能であることから、様々な応用が期待されている。ネットワークの分野においては、Rashelbach ら［7］が、Software-defined networking (SDN) におけるフローテーブルの検索に応用するなど、データベース分野以外でも注目されている。

本研究では、前述の通り、学習型インデックスをFIBに応用し、既存のFIB用のデータ構造よりもコンパクトながら、同程度の高速な検索を実現することである。このためには、機械学習によるエントリ格納位置の予測に要する時間と、機械学習によるエントリ格納位置の予測誤差を訂正するための局所探索に要する時間の削減が課題となる。これらの課題に対して、Rashelbach ら［7］は、高速な学習型FIBの計算に向けて、ニューラルネットワークの計算の最適化、計算機のニューラルネットワーク計算最適化技術の利用などに加えて、Rectified linear function (ReLU) 関数を用いたニューラルネットワークが、区分線形関数になることを応用した学習の最適化技術を提案している。

３．研究の方法

3.1. 学習型インデックス

はじめに、本章では、学習型インデックスの詳細を説明する。学習型インデックスは、キーとそれに対応する値の格納位置の関係を機械学習の回帰モデルとして再現するデータ構造である。その概要を図1に示す。

学習型インデックスの構築時は、KVS に格納するエントリーをそのキーの昇順（あるいは、降順）にソートして格納する。このとき、エントリの格納位置はキーに対する単調増加関数と見なすことができる。これを、ターゲット関数と呼ぶ。ターゲット関数を回帰モデルとして学習する。

構築した学習型インデックスを用いてエントリを検索するときは、まず、検索するキーに対するエントリの格納位置を回帰モデルで予測する。回帰モデルによる予測には誤差が生じるため、予測された位置の周囲から真のキーの格納位置を探索する。ここで、誤差は、予測された位置と真の格納位置との差とする。探索には、線形探索に加えて指数探索などが利用できる。以降、2 つのステップをそれぞれ予測フェーズと探索フェーズと呼称する。

回帰モデルによる誤差が大きいと、誤差の訂正のための探索フェーズに要する時間が増加する。したがって、探索フェーズの時間を削減するためには、誤差の縮小が課題である。層数やニューロン数の増やし高精度な回帰モデルを作成することで、誤差を縮小するこ

とは可能である。一方で、層数やニューロン数の増加にともない、予測フェーズに要する時間が増加する。この課題に対して、Kraska ら［5]は、予測の精度向上のために単一の大きなニューラルネットワークを使うのではなく、図 1 に示すように、ニューラルネットワークのモデルを階層的に連結することを提案した。各ステージのモデルは次のステージで利用するモデルを選択し、最終ステージのモデルが位置を予測する。最終ステージのモデルをエキスパートモデルと呼ぶ。それぞれのエキスパートモデルが回帰を担当する範囲を限定することで、層数やニューロン数の少ないニューラルネットワークで精度を向上させる。

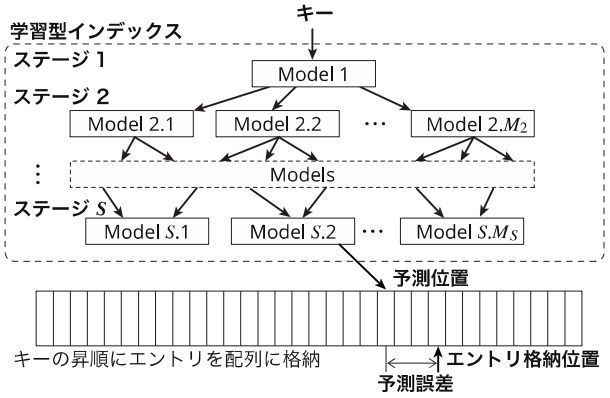

図 1　階層型学習型インデックス

3.2. 学習型 FIB の高速化方針

本研究では、予測フェーズと探索フェーズの時間短縮のため、Rashelbach ら［7］が提唱した知見、つまり、ニューラルネットワークをベースとする回帰が区分線形関数による回帰に帰着する点を応用する。具体的には、学習型インデックスの生成に、ニューラルネットワークの学習アルゴリズムを用いずに、予測誤差が事前に指定した規定値以内となるように区分線形関数による近似回帰を設計することで、最悪時の予測誤差を保証する。予測誤差が規定値以内であることを前提に、以下の通りに予測フェーズと探索フェーズの時間を短縮する。

- **予測フェーズ：**予測誤差が規定値以内である回帰モデルを構成できることを前提に、階層学習型インデックスの上位ステージのニューラルネットワーク計算を単純なテーブル検索に置換する。これにより、ニューラルネットワーク数、ニューロン数、あるいは、ニューラルネットワークの層数などを増加させることなく、予測フェーズの時間を短縮する。
- **探索フェーズ：**探索フェーズの時間は、予測誤差が小さく短い時間で探索が終了できることに加えて、計算機上の分岐予測ミスによる計算時間の増加を解決する必要がある。事前実験で、探索フェーズの計算時間のうち、44.4%の時間を CPU における分岐予測のミスによるパイプラインストールに費やされていることが分かった。これに対して、予測フェーズにおける予測誤差が事前設計した規定値以内であり、さらに、その規定値が十分に小さい値であることを保証できることを前提に、局所探索のアルゴリズムに単純な線形探索を用いながらも、ループの展開と条件文を排除し、分岐命令を用いないことで、分岐予測によるパイプラインストールを無くし高速な検索を実現する。
- **実装による高速化：**上記の方針に加えて、最新の CPU アーキテクチャを想定した実装により学習型インデックスの計算をさらに高速化する。具体的には、Single instruction multiple data（SIMD）、Advanced vector extensions（AVX）命令セットの活用、積和演算の活用、CPU キャッシュミスを削減する実装法により高速化する。このため、学習型インデックスの回帰モデルとして、SIMD/AVX 命令と積和命令で高速化が可能なニューラルネットワークを採用する。

3.3. 学習型 FIB の構築法

本章では、この学習型 FIB の構築法を議論する。詳細な構築法を議論することは、本稿の趣旨から逸脱するため、本稿では重要な箇所のみを議論する。具体的な設計法については文献［9］に譲る。

図 2 に上記の方針に沿って設計した学習型 FIB の

学習型インデックスを応用した高速なパケット転送

Forwarding Using Learned Index Structures

全体像を示す。学習型インデックスの構成は Kraska ら [5] の階層型の構成を採用するが、多段のニューラルネットワークによる計算時間の増加を防ぐために、トータルで 2 ステージに限定し、さらに、第一ステージにはニューラルネットワークではなく、テーブル参照によりエキスパートモデルを選択できるようにする。このテーブルは、入力キーである IP アドレスと、その IP アドレスに対応するエントリが格納された範囲を担当する回帰モデルの ID が記入されている。ただし、IP アドレスに対して上記の対応を保存できないため、IP アドレスの空間を2^nの範囲に分割し、回帰モデルが担当する範囲をその分割した範囲に限定させる。これにより、IP アドレスの上位nビットをテーブルのキーとして用いることができ、省スペースで高速にエキスパートモデルが選択できる。

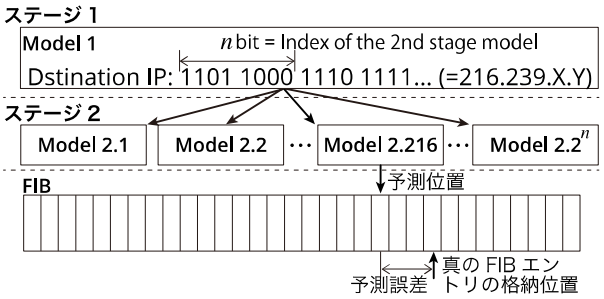

図 2　学習型 FIB の概要

続いて、ステージ 2 の回帰モデルの構築法を議論する。Kraska ら [5] は、図 3 の上部に示すように、ターゲット関数をニューラルネットワークで学習した。これに対して、我々は、Kraska らのオリジナルの学習型インデックス構成法とは逆に、先にターゲット関数を区分線形関数で近似し、その後にその区分線形関数と等価なニューラルネットワークを計算する方法を採用した。これにより、最大誤差を事前指定した閾値以内となるように区分線形関数を設計することができ、前章で示した通り、予測フェーズと探索フェーズの実装を高速化することができる。

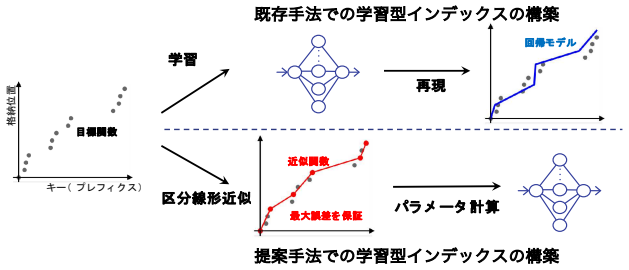

図 3 学習型インデックスと学習型 FIB の構築の流れ

具体的な学習型 FIB の構成法は次の通りである。また、図 4 に概要を示す。

A) IP プレフィクスと FIB エントリの格納位置の関係であるターゲット関数を導出する
B) ターゲット関数を区分線形関数で近似する
C) 導出した区分線形関数をニューラルネットワークで表現できるように分割する
D) 分割した区分線形関数と一致するニューラルネットワークの係数を導出する

区分線形関数の設計については、動的計画法を用いた設計 [9]、あるいは、Shrinking Cone を用いた設計 [10] などが利用できる。その詳細な説明は、割愛する。

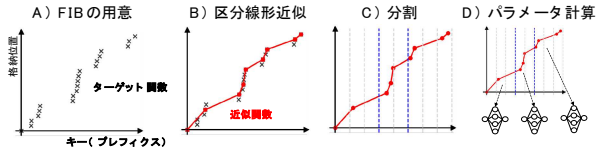

図 4　学習型 FIB の構築法

3.4. ニューラルネットワークと区分線形関数

回帰モデルが、ReLU 活性化関数を持つニューラルネットワークで構成されている場合、その回帰モデルは区分線形関数に帰着する [7, 9]。隠れ層が増加すると計算速度が低下するため、本稿では隠れ層 1 層のみで構成されるニューラルネットワークを考える。隠れ層のニューロン数は m とする。学習型 FIB では、入力は IP アドレス、出力は FIB エントリの格納位置であるので、入力および出力は 1 次元である。さらに、回帰モデルの出力層の活性化関数は恒等関数とする。この場合、このニューラルネットワークは、m 個の区分を持つ区分線形関数となる。区分線形関数になるこ

との証明は、[9] で議論している。

3.5. 区分線形関数の分割

設計した区分線形関数は、細かいサブ関数に分割する必要がある。それぞれのサブ関数が、前述した m ニューロンのニューラルネットワークに対応するため、それぞれのサブ関数が m の線形区分を持つように分割したい。しかし、IP プレフィクスの先頭 n ビットを、それぞれのニューラルネットワークへのインデックスとして用いるため、単純に区分線形関数を前方から順に m 区分ごとに分割することはできない。これに対して、キーの範囲を 2 のべき乗数で等分割し、各区間に含まれる区分の数が m を超えない範囲で隣接する区分線形関数を結合する。

この方法で分割した区分線形関数に対応するニューラルネットワークのウェイトおよびバイアスの決定方法については、[9] で議論している。

3.6. 学習型 FIB の検索アルゴリズム

学習型 FIB の検索アルゴリズムを図 5 の Algorithm 1 にまとめる。予測フェーズは、CPU の SIMD/AVX 命令と積和命令を用いて実装できる。探索フェーズについては、指数探索よりも時間計算量は多いものの、ループと分岐を用いることなく実装できる線形探索を採用する。予測フェーズの予測位置をp、最大予測誤差をε、入力 IP アドレスをx、位置pの FIB エントリの IP プレフィクスをx_pとすると、真の FIB エントリの格納位置lを探索するのは、$[p-\varepsilon, p+\varepsilon]$の範囲で、$x \leq x_i$となる FIB エントリの数を数えることに他ならない。したがって、探索フェーズは、線形探索を用いて、$l = p - \varepsilon + \sum_{i \in [p-\varepsilon, p+\varepsilon]} cmp(x \leq x_i)$ で求めることができる。ここで、$cmp(x \leq x_i)$ は、$x \leq x_i$が真であれば 1、偽であれば 0 をとる関数である。右辺第二項は、ループの展開と SIMD 化された cmp 命令である vpcmp 命令を用いることで、Algorithm 1 の 8-9 行目に示した通り、ループを用いることなく計算できる。

Algorithm 1: FIB 検索アルゴリズム

Input: x: Address (32-bit unsigned integer)
Output: h: Next hop information
// **— Prediction phase —**
// **— Learned index 1st stage —**
1 $r \leftarrow \mathrm{srl}(x, 32 - m)$ // Shift (slr: shift right logical)
2 $i \leftarrow \mathrm{table}[r]$ // Get model index
3 $\boldsymbol{w}_1, \boldsymbol{b}_1, \boldsymbol{w}_2, b_2 \leftarrow \mathrm{model}[i]$ // Get weights and biases
// **— Learned index 2nd stage —**
// Predict the position of x via a neural network
4 $\boldsymbol{z} \leftarrow \mathrm{vfmadd}(\boldsymbol{w}_1, x, \boldsymbol{b}_1)$ // $\boldsymbol{z} \leftarrow \boldsymbol{w}_1 \times x + \boldsymbol{b}_1$
5 $\boldsymbol{z} \leftarrow \mathrm{vmax}(\boldsymbol{z}, 0)$ // $\boldsymbol{z} \leftarrow \mathrm{ReLU}(\boldsymbol{z})$
6 $\boldsymbol{z} \leftarrow \mathrm{vmul}(\boldsymbol{w}_2, \boldsymbol{z})$ // Element-wise $\boldsymbol{w}_2 \times \boldsymbol{z}$
7 $y \leftarrow \mathrm{accumulate}(\boldsymbol{z}) + b_2$ // $y \leftarrow \sum \boldsymbol{z} + b_2$
// **— Local search phase —**
8 $l \leftarrow y - \varepsilon$ // Search the range $[y - \varepsilon, y + \varepsilon]$ for x
// Set the results of $\boldsymbol{k}[l : l + 2\varepsilon] \leqq x$ to $\boldsymbol{z}$
9 $\boldsymbol{z} \leftarrow \mathrm{vpcmp}(\boldsymbol{k}[l : l + 2\varepsilon], x)$
10 $l \leftarrow l + \mathrm{accumulate}(\boldsymbol{z})$
11 $h \leftarrow \mathrm{nexthop}[l]$

図 5　学習型 FIB 検索アルゴリズム

3.7. 評価結果

本章では、学習型 FIB の性能を他の最先端の FIB 用データ構造と比較する。比較対象としては、本研究の遂行時点で、世界最速の FIB 用のデータ構造である Poptrie [3] を用いる。Poptrie は、トライベースのデータ構造である。

比較の指標には、IP アドレス 1 つの転送先の検索に要する平均時間を用いる。時間の計測には、CPU サイクル数を用いた。学習型 FIB を構築するための最大誤差は 32 とした。FIB の構築には、オレゴン大学 Route Views Project [11] で測定された BGP routing information base (RIB)の 2019 年 11 月 20 日のスナップショットを用いる。FIB のネクストホップの情報は、CAIDA [12] で公開されている AS マップを用いて作成した。

まずメモリサイズについて、学習型 FIB が 1.83 Mbytes、Poptrie が 1.98 Mbytes であり、Poptrie よりも小さかった。

次に計算速度について、図 6 と図 7 はマッチしたプレフィックス長ごとの計算時間を示す。 各ひげの上/下端は 5/95 パーセンタイル、箱の上/下端は第一/三四分位、箱内部の棒は中央値をそれぞれ表す。

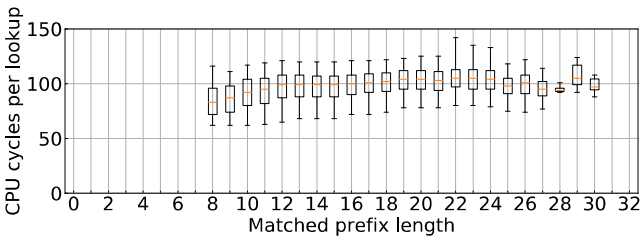

図 6 学習型 FIB の検索時間

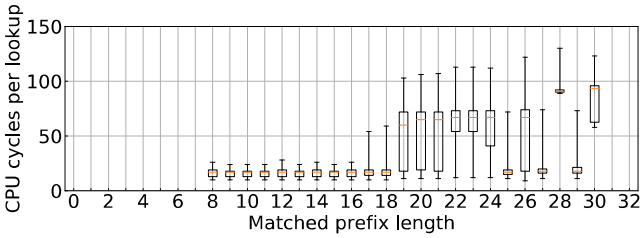

図 7 Poptrie の検索時間

図から分かるように、学習型 FIB は Poptrie よりも遅いものの、比較可能なレベルの検索速度を実現できている。さらに、学習型 FIB はどのプレフィクス長についても一定の計算時間である一方、Poptrie FIB はプレフィクス長によって計算時間に差がある。これは Poptrie がプレフィクス長 18 以下の場合はテーブル検索のみで探索が終了するため高速であることと、プレフィクス長 18 より大きい場合は木を探索するため、木の深さに依存した時間がかかることが理由である。この結果は、将来的に 24 ビット程度の長いの IP プレフィクスが増加した場合、あるいは IPv6 などプレフィクス長が長い場合に、学習型 FIB が有利になる可能性があることを示唆している。

４．将来展望

前章の評価結果で議論した通り、トライに基づくデータ構造では、エントリの検索に木を辿る操作が必要であることに起因して、プレフィクス長に依存して計算時間が延びる傾向にある。これに対して、提案手法は、プレフィクス長に依らず計算時間が一定である。ここから、2 つの展望が見える。

第一に、将来的に 24 ビット程度の長いの IP プレフィクスが増加した場合、あるいは IPv6 などプレフィクス長が長い場合に、学習型 FIB が有利になる可能性があることを示唆している。第二に、現在の学習型 FIB では全ての IP アドレスに対して、第一ステージのテーブル検索と第二ステージのニューラルネットワークの計算を課している。これを、Poptrie と同様に、一部の IP アドレスについてはテーブル検索処理のみで学習型 FIB の検索処理を簡潔させることで、Poptrie と同程度の平均計算速度を実現できる可能性がある。

おわりに

本研究では、学習型インデックスと呼ばれるデータベース向けのインデックス構造の提唱に端を発して、IP パケット転送における FIB 検索のデータ構造とアルゴリズムの課題に再訪した。学習型インデックス中のニューラルネットワークによる回帰を区分線形関数による近似を用いて設計することで、学習型インデックスによる予測と局所探索フェーズの高速化を実現した。

参考文献

[1] S. Nilsson and G. Karlsson, "IP-address lookup using LC-tries," IEEE Journal on Selected Areas in Communications, vol.17, no.6, pp. 1083-1092, June 1999.

[2] S. Dharmapurikar, P. Krishnamurthy, and D.E. Taylor, "Longest prefix matching using bloom filters," IEEE/ACM Transactions on Networking, vol.14, pp. 397-409, April 2006.

[3] H. Asai and Y. Ohara, "Poptrie: A compressed trie with population count for fast and scalable software IP routing table lookup," in Proceedings of ACM SIGCOMM, pp. 57-70, Aug. 2015.

[4] W. Jiang, Q. Wang and V. K. Prasanna, "Beyond TCAMs: An SRAM-based parallel multi-pipeline architecture for terabit IP lookup," in Proceedings of IEEE INFOCOM, Apr. 2018.

[5] T. Kraska, A. Beutel, E.H. Chi, J. Dean, and N. Polyzotis, "The case for learned index structures," in Proceedings of ACM SIGMOD, pp. 489-504, June 2018.

[6] B. Zhang, L. Wang, X. Zhao, Y. Liu, and L.

Zhang, "FIB aggregation," Internet-Draft draft-zhang-fibaggregation-02, Internet Engineering Task Force, Oct. 2009.
[7] A. Rashelbach, O. Rottenstreich, and M. Silberstein, "A Computational Approach to Packet Classification," in Proceedings of ACM SIGSOMM, Aug. 2020.
[8] S. Higuchi, J. Takemasa, Y. Koizumi, A. Tagami, and T. Hasegawa, "Feasibility of Longest Prefix Matching Using Learned Index Structures," ACM SIGMETRICS Performance Evaluation Review, May. 2021.
[9] S. Higuchi, Y. Koizumi, J. Takemasa, A. Tagami, and T. Hasegawa, "Learned FIB: Fast IP Forwarding without Longest Prefix Matching," in Proceedings of IEEE ICNP, Nov. 2021.
[10] A. Galakatos, M. Markovitch, C. Binnig, R. Fonseca, and T. Kraska, "FITing-Tree: A Data-Aware Index Structure," in Proceedings of ACM SIGMOD/PODS, June 2019.
[11] University of Oregon Route Views Project. http://www.routeviews.org/routeviews/
[12] CAIDA. https://www.caida.org/home/

この研究は、令和元年度ＳＣＡＴ研究助成の対象として採用され、令和２～３年度に実施されたものです。