IEEE INFOCOM 2021 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS) | 978-1-6654-0443-3/21/$31.00 ©2021 IEEE | DOI: 10.1109/INFOCOMWKSHPS51825.2021.9484440

An FPGA-based High-Throughput Packet Classification Architecture Supporting Dynamic Updates for Large-Scale Rule Sets

Yao Xin[1], Wenjun Li[1,2,*] , Yi Wang[1,3], and Song Yao[4]
[1]Peng Cheng Laboratory, [2]Peking University, [3]Southern University of Science and Technology, [4]New H3C, China
xiny@pcl.ac.cn, wenjunli@pku.edu.cn (*corresponding author), wangy37@sustech.edu.cn, yaosong@h3c.com

***Abstract*—A high-performance packet classification architecture based on FPGA supporting large-scale rule sets up to 100k is proposed in this poster. It supports fast dynamic rule update and tree reconstruction. The update throughput is comparable to that of classification. An efficient data structure set for decision tree is constructed to convert tree traversal to addressing process. Different levels of parallelism are fully explored with multi-core, multi-search-engine and coarse-grained pipeline. It achieves a peak throughput of more than 1000 MPPS for 10k and 1k rule set for both classification and update.**

I. Introduction

As one of the building blocks of Software-Defined Network (SDN), OpenFlow is widely used to provide dynamic traffic steering between applications on different platforms. An OpenFlow switch consists of one or more flow tables to perform packet lookups and forwarding which is essentially a multi-field packet classification problem [1]. In the last two decades, FPGA has been increasingly adopted as a favorable platform for packet classification to replace TCAM-based methods [2], [3]. Although intensive research has been conducted in this field, there is still no single hardware architecture that can meet the demands for packet classification in OpenFlow switches: 1) multi-field (not limited to typical 5 tuples) matching; 2) large-scale rule set supporting; 3) fast dynamic rule update. Nowadays, FPGA-based decision tree implementations still face great difficulty in rule update due to the notorious rule replications [2]. Furthermore, the features of unbalance and unbounded depth in decision trees cause fully pipelined designs result in a high dependence on rule set and a resource waste. As the cutting-edge FPGA-based packet classification technique, the recently proposed Bit Vector (BV) approach could achieve a good performance in both classification and update [3]. However, the scale of Vector is restricted by FPGA resources, thus only small-scale rule sets can be applied.

To address the above problems, we present a dynamically updatable hardware architecture for multi-field packet classification based on the improved version of our proposed Tabtree algorithm [4]. Not only does it achieve a high performance in classification and update, it also supports large-scale rule sets up to 100k. The maximal average throughput for classification and update is up to 1132 MPPS and 1036 MUPS respectively for 5-tuple 1k rule sets, on a state-of-the-art FPGA. The major contributions of this work are as follows:

- Suitable data structures for nodes and rules are designed which are stored in large pieces of RAMs instead of distributed small ones. This facilitates the support for large-scale rules and breaks the constraints of limited tree depth and imbalanced node distribution. Thus, the architecture is independent of rules. Moreover, the tree reconstruction is equivalent to the update of RAM data which can be easily realized.
- Fast dynamic rule update is fully supported for large-scale rule sets. It is able to create and delete tree node in real-time with the help of storage space dynamic allocation.
- Multiple parallel techniques are fully explored such as multi-core, multi-search engine, and coarse-grained pipeline to achieve a high performance.

This work is partially supported by project “PCL Future Greater-Bay Area Network Facilities for Large-scale Experiments and Applications (LZC0019)”.

II. Architecture Design

In spite of extensive research on decision-tree algorithms [5], [6], the unbalanced nodes hinder designing fully pipelined architectures on FPGA. Furthermore, only a limited number of tree levels is supported due to the explosive growth of high-level nodes. To address the above issues, two chain-table-like data structures are constructed for nodes and rules which are shown in Fig. 1. Nodes are divided into internal nodes and leaves, each one associated with a table of 134 bits (selection bit length in Tabtree is 2). Each leaf is associated with one rule subset with a number threshold of *binth*. Each rule corresponds to a 273-bit table (5-tuple, mask is transfered to ranges) which connects end to end within one subset.

These two types of tables are stored in two bulks of on-chip RAMs: node table RAM and rule table RAM. Therefore, the tree traversal to search a rule is converted to the addressing process. This design has three major advantages:

- The architecture does not depend on specific rule sets.
- The level of nodes could not be restricted.
- It facilitates real-time rule update, as rules could be traced back to upper-level nodes with cached node information.

The proposed FPGA architecture is shown in Fig. 2. The outermost architecture utilizes data parallelism by instantiating multiple cores for independent computing in parallel. Taking typical 5-tuple rule set as an example, each computing core consists of three tree structures based on the small rule fields according to Tabtree [4]: source IP address (SA), destination IP address (DA), SA & DA. The architecture for each tree structure is composed of Node Search module and Rule Operate module. The former one finds the rule subset address while the later one searches rules linearly and makes actions of search, delete, or insert according to the operation code. These two modules run independently and constitute a two-stage coarse-grained pipeline. In Node Search module and Rule Search submodule, multiple engines are implemented to speed up the search process and improve memory access efficiency. A fully pipelined design is not adopted because the storage resources are too scattered which would cause

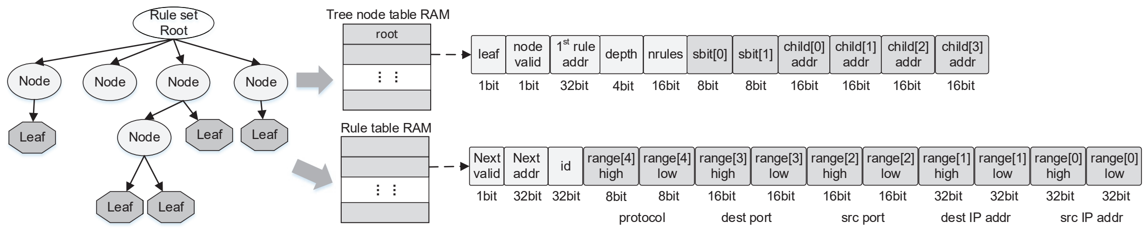

Fig. 1. The data structure design for nodes and rules.

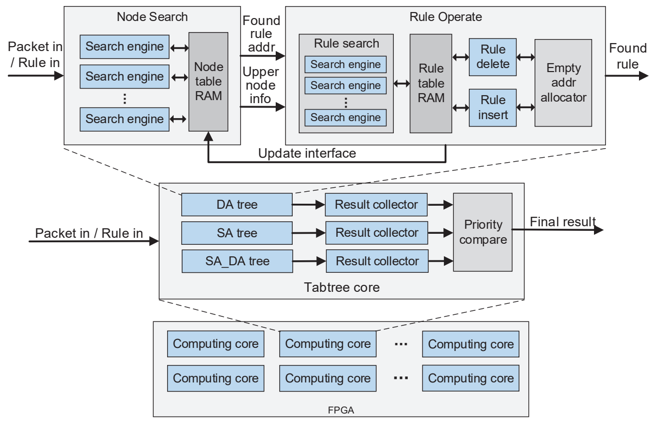

Fig. 2. The detailed hardware architecture.

TABLE I
IMPLEMENTATION RESULT

Average rule size	Core num	LUTs (1182240)	Registers (2364480)	BRAM (2160)	URAM (960)	Frequency (MHz)
100k	5	23735	33610	1030	640	235.4
32k	14	66488	94113	1344	672	222.5
10k	35	145604	211694	1260	706	200.6
1k	50	300160	539789	922	718	185.3

resources being seriously wasted, impede real-time update, and make reconstruction impossible in such configuration.

The rules to be added or deleted are processed in the same way as the packets in Node Search stage, but they are processed by rule delete and insert engines separately in Rule Operate module. The empty address allocation logic helps dynamically manage the available space in node table RAM and rule table RAM for rule update. With cached complete information from the Node Search module, upper-level nodes can be traced back to support real-time update of internal nodes, deletion and addition of leaf nodes.

III. EXPERIMENTAL RESULTS

Resource Utilization. The architecture adapting to rule sets in different sizes is implemented on Xilinx Virtex UltraScale+ VU9P FPGA, in order to make a comprehensive comparison. Table I summarizes the resource usage and maximum frequency with different core numbers. It can be noted that the memory (URAM & BRAM) is the most used FPGA resource.

Experimental Setup. Three types of rule sets are generated by ClassBench to make the performance evaluation: ACL, FW and IPC, each of which has four sizes: 1k, 10k, 32k, 100k. The small-field threshold and selection bit length in Tabtree are 12 and 2. The threshold of rule number in leaf (*binth*) is set to 6. The number of search engines in Node Search and Rule Operate module is set to 6 and 4 separately. The rules in leaves with the number more than *binth* are filtered into the big rule

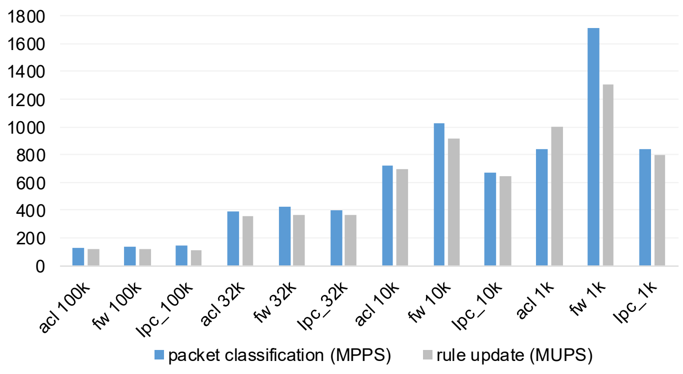

Fig. 3. The throughput for packet classification and rule update.

set which is not included in tree structures. Performance merits consist of packet classification throughput and rule update throughput in units of MPPS (Million Packets per Second) and MUPS (Million Updates per Second) respectively.

Performance Evaluation. To our knowledge, there is currently no FPGA-based architecture that supports both classification and update for large-scale rule sets to make a peer-to-peer comparison. Therefor, only our performance results are shown in Fig. 3 to be a reference for subsequent work. The average throughput of classification for 1k, 10k, 32k, 100k rule sets is 1132 MPPS, 804 MPPS, 405 MPPS, 136 MPPS, while the update throughput is 1036 MUPS, 754 MUPS, 364 MUPS, 119 MUPS respectively. Therefore, this design has achieved a high performance in both packet classification and rule update.

IV. FUTURE WORK

In order to ensure the stable performance for large-scale rule sets, rules in a leaf with the number greater than the threshold value are filtered into the set of big rules (i.e. the rules with large field range). And PSTSS adopted in original TabTree is not utilized for leaf rules. The process of big rules is not addressed hereof as they only account for a small percentage of the entire rule set. Our future work is to design efficient architectures to process big rules.

REFERENCES

[1] A. Rashelbach, O. Rottenstreich, and M. Silberstein, "A computational approach to packet classification," in *ACM SIGCOMM*, 2020.

[2] W. Jiang and V. K. Prasanna, "Scalable packet classification on fpga," *IEEE TVLSI*, 2012.

[3] Y. R. Qu and V. K. Prasanna, "High-performance and dynamically updatable packet classification engine on fpga," *IEEE TPDS*, 2016.

[4] W. Li, T. Yang, Y. Chang, T. Li, and H. Li, "Tabtree: A tss-assisted bit-selecting tree scheme for packet classification with balanced rule mapping," in *ACM/IEEE ANCS*, 2019.

[5] W. Li, T. Yang, O. Rottenstreich, X. Li, G. Xie, H. Li, B. Vamanan, D. Li, and H. Lin, "Tuple space assisted packet classification with high performance on both search and update," *IEEE JSAC*, 2020.

[6] B. Vamanan, G. Voskuilen, and T. N. Vijaykumar, "Efficuts: Optimizing packet classification for memory and throughput," in *ACM SIGCOMM*, 2010.