An Energy-efficient TCAM-based Packet Classification with Decision-tree Mapping

Zhao Ruan, Xianfeng Li , Wenjun Li
Engineering Lab on Intelligent Perception for Internet of Things (ELIP)
School of Electronic and Computer Engineering, Peking University, Shenzhen, China
zhaozhaoruan@gmail.com, lixianfeng@pkusz.edu.cn, liwenjun@sz.pku.edu.cn

Abstract—Network packet classification is a key functionality provided by modern routers enabling many new network applications such as quality of service, access control and differentiated services. Using ternary content addressable memories (TCAMs) to perform high-speed packet classification has become the de facto standard in industry. However, despite their high speed, one major drawback of TCAMs is their high power consumption. Although SmartPC, the state-of-the-art technique, was proposed to reduce power consumption by constructing a pre-classifier to activate TCAM blocks selectively, its bottom-up approach restricts its ability of grouping rules into disjoint TCAM blocks. In this paper, we propose a top-down approach for two-stage TCAM-based packet classification. The novelty of our work is the intelligent combination of software-based packet classification with TCAM-based techniques. We start by constructing a set of decision-trees for the packet classification rules, which enable the subsequent steps an excellent global view on the relationships among rules. The decision-trees are then mapped to TCAM blocks with flexible heuristics. Our top-down framework addresses the bottlenecks (the number of general rules, which have to be activated unconditionally every time) of SmartPC very effectively. Using ClassBench in our experimentations, we show that our technique is able to restrict the number of general rules to just 1% of the overall rule set. This leads to a dramatic power reduction of up to 98%, and 96% on average, which significantly outperforms SmartPC.

Keywords—Packet Classification; TCAM; Power Consumption;

I. INTRODUCTION

Packet classification is the first and foundational step in modern network routers that enables many network applications such as quality of service, traffic monitoring, securities and virtual private networks (VPNs). Because of its importance and challenge, packet classification has received extensive research efforts in the past fifteen years, and both hardware and software solutions have been investigated. Although software based solutions are desirable due to their low cost and flexibility, they fail to meet the wire speed requirement of high performance networks. As a result, hardware based solutions are still the de facto standard in industry.

A hardware based solution typically employs Ternary Content Addressable Memories (TCAMs), which stores all classification rules as an array of ternary bit strings. A few header fields of an incoming packet are compared against all the TCAM entries simultaneously to find the best matched one, and an action associated with the matched TCAM entry will be taken.

Despite their high speed, TCAMs are low in density and high in cost. More seriously, they consume extraordinary power due to the parallel activation of all memory entries. To reduce power, TCAM vendors now provide mechanisms to selectively activate and search part of the TCAM device [1]. In particular, a modern TCAM chip can be partitioned into fixed-sized blocks, and a variable number of TCAM blocks can be activated each time. In recent years, a few techniques leveraging this feature have been proposed for power reduction [1][2][3]. However, the potential of power reduction has not been fully exploited, or the performance degradation compared to traditional TCAM-bases solution is not acceptable. For example, a large number of rules in [2] are treated as general ones, and they have to be activated unconditionally for each incoming packet.

In this paper, we introduce a two-stage TCAM-based packet classification built on decision-tree mapping. The combination of software based packet classification techniques (using decision-tree) with hardware based packet classification (using TCAMs) enables us to better exploit the relationships among rules with a global view, thus it works very effectively on reducing the number of general TCAM blocks as well as the power consumption. Using Classbench [4], we evaluate the power reduction for classifiers with scales from 1,000 to 100,000 rules. The result shows that our algorithm can achieve a power reduction up to 98% and an average reduction of 96%.

The rest of the paper is organized as follows. Section II introduces the background on packet classification and the state-of-the-art TCAM-based techniques. Section III presents the technical details of our work. Section IV provides experimental results. Finally, Section V concludes the paper.

II. BACKGROUND

A. Problem of Packet Classification

Packet classification is performed using a packet classifier. A classifier or a rule set consists of a finite set of rules with priorities. A typical rule is a 5-tuple (source and destination IP addresses, source and destination port numbers, and protocol) and an associated action. Each rule field is represented as a prefix or a range, and has three kinds of matches: exact match, prefix match or range match. The packet classification is to

978-1-4799-2827-9/13/$31.00 ©2013 IEEE

TABLE I. AN EXAMPLE OF RULE

Rule	SA	DA	SP	DP	Proto	Action
0	0.0.0.0/0	15.0.0.0/8	0: 65535	13 : 13	*	action0

compare the header of an incoming packet with a set of predefined rules to find the best match. Table I shows an example 5-tuple rule. Packet classification is a much more challenging problem than route lookup, and it is easily the bottleneck of a router/switch.

B. Related Work

A large amount of previous work is based on software. Software based solutions often have three categories: tree-based implementations such as HiCuts[5], HyperCuts[6], EffiCuts[7], decomposition-based implementations[8][9] and hash-based implementations [10], which are flexible but fail to meet the wire-speed requirement enforced by many high-speed network devices.

Using TCAM to perform high speed classification now becomes the de facto industrial standard in high performance network processor. A principal drawback of TCAM is its high power consumption because of the activation of all memory entries for every packet lookup. The idea of using multiple stages to implement packet classification to reduce power consumption has been proposed [1][2][11]. The CoolCAMs algorithm concentrates on the problem of IP lookups (perform classification on destination IP address) [3].

SmartPC, a state-of-art two level TCAM-based algorithm, introduced a pre-classifier to reduce power consumption [2]. It builds its pre-classifier entries by the idea of expanding and combining original rules based on their locations in the source - destination dimension space. Starting from a rule, SmartPC expands the rectangle to cover other rules, which will be put into the same TCAM block and will be indexed by a specific entry in the pre-classifier. If the inclusion of a rule causes the number of rules in the rectangle exceeding the TCAM block size, or leads to conflictions (overlaps) with existing blocks, the rule will be simply marked as *general*, which will not be partitioned any more. An example is shown in Figure1, the rule set processed consists of 18 rules and the block size is 4. Starting from R0, SmartPC expands the first rectangle to cover R0, R1, R14, R15, then R5, R6, R7, R11, R12, R13, R16, R17 will be marked as general rules. The second block includes R2, R3, R4, R9, and R10 is marked as a general rule. In the end, 10 rules are marked as general, which the general rules account for more than 50% of the rule set.

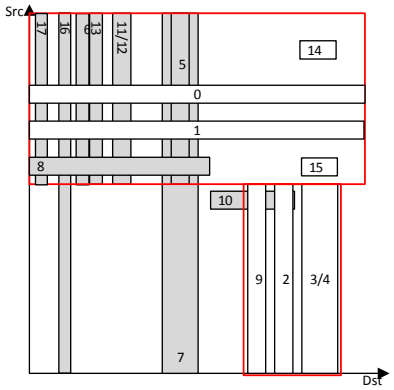

Figure 1. An example of SmartPC

Clearly, this bottom-up way of generating TCAM blocks is not an intelligent approach, as it lacks the global view on the relationships of rules. Consequently, it has little freedom of heuristics over the grouping of rules into blocks, and has no effective control on the scale of general rules. During packet classification, in addition to the TCAM block indexed by a matched pre-classifier entry, the blocks accommodating the general rules have to be search as well. Therefore, the large number of general rules in SmartPC becomes the primary contributor of TCAM power consumption. Although SmartPC reports an average reduction of 88%, for some unfavorable rule sets, the portion of rules marked as general is up to 20%, which leads to a power reduction of only 78% (recall that a TCAM consumes 30 times more power than an SRAM of equivalent size). More importantly, the lack of controls over the general rules makes it difficult for giving even a rough bounding on power reduction.

III. TECHNIQUE

From the earlier discussion, it is clear that bounding the number of general rules is vital for a two-stage TCAM-based solution using a pre-classifier. For this purpose, we introduce a framework that has a better global view on the geometric relationships among rules. This framework combines software based packet classification with hardware based techniques. In particular, we construct one or more decision-trees for rules like a traditional software-based scheme, then map the leaf nodes of trees to TCAM blocks, leaving only a small number of rules that are not in any tree as general ones. The index entries in the pre-classifier correspond to the paths leading to the tree leaf nodes. During the process of tree construction, we not only have an excellent global view on the geometric relationships among rules, but also enjoy a broad freedom on the decisions of grouping rules. The details of this framework are elaborated as follows.

A. Two-stage Packet Classification Architecture

Figure 2 gives the architecture of the two-stage classifier. With an incoming packet, parts of its header fields are looked up in the Index TCAM. Here we use SA and DA (64 bits in total) as the index fields, as it is usually easier to differentiate rules with these two fields. Then the matched Index TCAM entries will be used to activate the corresponding Data TCAM blocks. Meanwhile, the Data TCAM blocks corresponding to the general rules have to be activated as well. Finally, a best matched rule from the multiple activated blocks will be returned by the priority encoder of the Data TCAM.

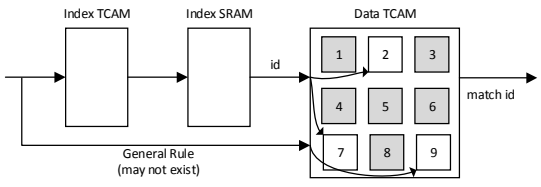

Figure 2. Packet classification architecture

B. Decision-tree Construction

Our decision-tree construction is different from traditional software-based packet classification on several important aspects. First, the objective is for efficient mapping to TCAM blocks instead of efficient tree traversing; therefore, the trees do not need to be short. Second, although rule replications are also a concern in software-based packet classification, but the reduction of rule replications is more important in our work, as the TCAM capacity is a more precious resource than commodity DRAM. Third, a leaf node in our work contains much more rules than in software-based solutions, which in turn helps reduce rule replications as the majority of rule replications happen in deep tree levels for software-based solutions. Based on these differences, our tree construction strategy is very different from traditional decision-tree constructions. The pseudo code of the high-level algorithm is sketched in Figure3.

```
construct_pre_classifier(ruleset, num_trees)
    tmp_ruleset=ruleset
    //build at most num_tree trees
    for i = 0..num_trees
        if num_rules(tmp_ruleset) ≤ blocksize || i==num_trees then
            set_general_blocks(tmp_ruleset)
            exit for loop
        tree[i].ruleset = tmp_ruleset
        build_tree(tree[i])
        tmp_ruleset = trimed_rules
        trimed_rules={ }
    end for

build_tree(root)
    bit = select_bit(root);
    cut_node(root, bit, leftchild, rightchild)
    if duptimes > spacf
        trimmed_rules = trimmed_rules ∪ trim_duprules(tree[i])
    build_tree(leftchild)
    build_tree(rightchild)
```

Figure 3. Algorithm of pre-classifier construction

We regard the two IP address fields as a 64-bit string, each bit has 3 states, "0", "1" or "*". At each tree node, we choose one bit to cut the set of rules in the node in two halves: rules with value 0 or * for that bit are assigned to the left child of current node, and rules with value 1 or * to the right child. Note that here rule replication happens for rules with value *. Since rule replications result in inefficient tree construction and more TCAM space to accommodate them, we should try to restrict rule replications to a predefined threshold. To meet this constraint, and in the meantime to produce a balanced tree, we associate every bit in the IP address filed of node[v] with a cost[i] (i□(0,1……,63)) according to the numbers of occurrences of 0/1/* in all rules at bit i.

Nevertheless, rule replications cannot be completely avoided. To restrict the replications to a certain level, we use a threshold and record each occurrence of rule replication during tree construction. Even more, we order the replicated rules in their frequencies of replications. Once the replications are over the threshold, we invoke *trim_duprules()* to trim the rule with the highest replications off the tree, and put it into *trimmed_rules*, a temporary rule set that will be used for a new tree construction in the next round.

The total number of trees that can be constructed is also restricted to a predefined number. This is because with N trees, we need to conduct parallel searches on each tree for an incoming packet, and this corresponds to searching N TCAM blocks simultaneously. The more TCAM blocks to search, the higher power consumption. Therefore, when a certain number of trees (num_trees in the pseudo code) are constructed, the rest rules in trimmed_rules will be simply treated as general rules, as SmartPC does. In our experiments, we find that the tree construction works very efficiently, and at most three trees are needed for very large rule sets.

C. Decision-tree Mapping

After the construction of trees, the next step is to map these trees as well as the general rules into the TCAM. This consists of two steps: pre-classifier formation, and TCAM block filling, both are straightforward processing.

To form the entries of the pre-classifier, which is also a small TCAM, we travel the trees from their roots to all leaf nodes, each path from a root to a leaf node corresponds to a 64-bit ternary string. This ternary string is put into a pre-classifier TCAM entry, and the rules in the leaf node will be filled into the corresponding Data TCAM block. A matching of an incoming packet on the pre-classifier entry will return the block id to activate the corresponding Data TCAM block.

Finally, the general rules (which cannot be indexed by a pre-classifier entry) will be assigned to one or more blocks in the Data TCAM. These blocks will be unconditionally activated for every incoming packet.

The power consumption of our packet classification architecture can be evaluated as follows:

$$\lceil \#pre-classifier / blksize \rceil + \#trees + \lceil \#general\_rules / blksize \rceil$$

Where *#pre-classifer* is the number of entries in the pre-classifer, *blksize* is the TCAM block size, *#trees* is the number of trees constructed, and *#general_rules* is the number of general rules that do not belong to any tree.

D. Example Analysis

To better illustrate the mechanisms and advantages of our work, we use the same example for SmartPC in Figure 1. The decision-tree constructed for the rule set is shown Figure 4 with a blocksize=4. Note here only one tree is needed, and there are three rules marked as general. The mapping to TCAM will result in a pre-classifier with four entries, and a general block is

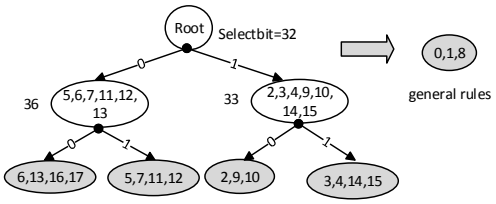

Figure 4. An example of block generation with block=4

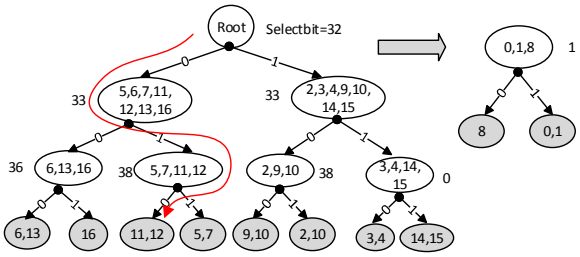

Figure 5. An example of block generation with block=2

needed to accommodate rules 0, 1, and 8. Each packet classification will activate the pre-classifier (one block), one Data TCAM block, as well as one general block. Overall, three TCAM blocks are activated in our framework, whereas five TCAM blocks are activated for SmartPC, as illustrated in last section. From this example, we can see that our top-down approach using decision-tree mapping works better than the bottom-up approach of SmartPC, which inefficiently marks too many rules as general ones.

Since this 18-rule example is trivial, and does not show the multi-tree possibility and rule replication, by reducing block size to 2, we get a result of two decision trees (but without general rules), and one rule replication (rule 10 appearing in two leaf nodes), as shown in Figure 5.

IV. Experimental Results

In this section, we implement our algorithm and evaluate its performance. The rule sets used in our experiments are generated by ClassBench [4] with characteristics representative of real-word classifiers, which provides three types of rule sets: Access Control List (Acl), Firewall (Fw) and IP Chain (Ipc). Each rule set is named by its type and size (e.g. Fw10k refers to the firewall rule set containing about 10,000 rules).

A. Reduction of General Rules

As discussed earlier, the number of general rules is a major factor affecting the performance of two-stage TCAM-based packet classifications, we evaluate our algorithm with 4 different block sizes: 32, 64, 128, and 256. Table II shows the numbers of general rules and the numbers of pre-classifier entries. Table III displays the number of trees handling the classifier corresponding to Table II and the blocks need to be searched to find a match.

TABLE II. NUMBER OF GENERAL RULES AND PRE-CLASSIFIER ENTRIES

rule_set	#general rules			#pre-classifier		
	64	128	256	64	128	256
Acl10k	0	0	0	274	128	64
Fw10k	65	59	0	192	96	48
Ipc10k	87	0	0	217	114	60
Acl100k	0	0	0	2049	1024	512
Fw100k	852	724	356	2777	1504	636
Ipc100k	183	0	0	2302	1154	584

TABLE III. NUMBER OF TREES AND SEARCHED BLOCKS

rule_set	#trees			#blocks searched		
	64	128	256	64	128	256
Acl10k	1	1	1	6	2	2
Fw10k	2	2	2	7	4	3
Ipc10k	2	2	2	7	3	3
Acl100k	1	1	1	33	9	3
Fw100k	3	3	3	61	21	8
Ipc100k	2	3	3	41	13	6

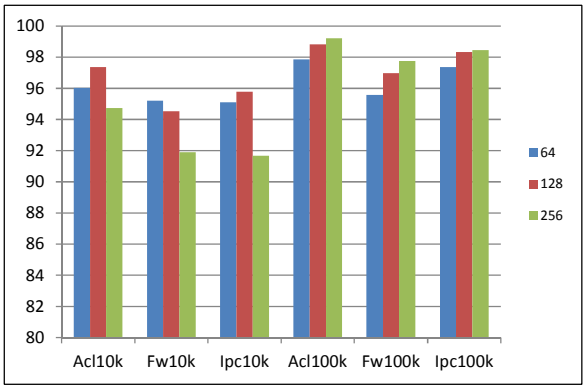

Figure 6. The percentage of power reductions

Compared with SmartPC, in which the general rules are more than 10% in most cases (and 20% in some cases), our algorithm generates an average of general rules less than 1%, which will result in a more significant saving of power consumption. This is because our top-down approach using decision-tree mapping has an excellent view of the global characteristics of rules, and more control over the grouping of rules with adaptive decision-tree construction.

Our work also has a good scalability. With the increase of rule set size, the number of trees rarely increases, therefore the number of TCAM blocks to activate grows slower than the size of rule set. Note there exists a design space exploration with different sizes of rule sets, block size, and the number of allowed trees. For example, if we continue to increase the rule set size, while keeping the block size and the maximum number of trees as constant, it may produce a very large pre-classifier. It means that even tough we may save power consumption for the Data TCAM access, the large number of pre-classifier blocks (which are always activated every time) may dominate the overall power consumption. As a result, in practical implementation, the block size as well as the maximum number of trees should grow with the rule set size to make a good balance among multiple components.

B. Power Reductions

According to the calculation method given in the last section, the percentage of power reductions is shown in Figure

6. The x-axis presents different rule sets at block sizes 64, 128, and 256. The y-axis presents the percentage of power reduction. Our scheme achieves a power reduction range from 91% to 99%, and an average reduction of 96%. It can be noted that for large rule sets, which suffer more serious power consumption problems, our work achieves more power reductions than on smaller rule sets.

The SmartPC only achieves an average reduction of 88%[2], because it has too many general rules that always need to be searched for every packet.

V. CONCLUSION

This paper presents a two-stage TCAM-based packet classification to significantly reduce the high power consumption of TCAMs. Unlike previous bottom-up work which has poor control over the grouping of rules into TCAM blocks, our work takes a top-down approach. This approach leverages decision-tree construction popular in software-based packet classification techniques. The decision-tree construction enables us a better global view of rule set characteristics, and more freedom on the control of rule grouping decisions. A direct benefit of our work is a significant reduction of general rules, which are the major source of power consumption in previous work. The experimental results show that we can achieve a reduction up to 98%, and an average 96% reduction on power, which is much better compared to SmartPC, the state-of-the-art two-stage TCAM classification work. Furthermore, our work enables a more flexible design space exploration of practical designs.

In future work, we plan to combine this framework with rule projection techniques to reduce the width of TCAMs needed for packet classification, thereby saving power both vertically and horizontally.

ACKNOWLEDGMENT

This work is supported by the grant of Shenzhen municipal government for basic research on Internet technologies (Outstanding Young Scholar, No. JC201005270274A) and Shenzhen municipal government for basic research on information technologies (No. JCYJ20130331144751105).

REFERENCES

[1] F. Zane, G. Narlikar, and A. Basu, "CoolCAMs: Power-efficient TCAMs for forwarding engines," in IEEE INFOCOM, 2003.

[2] Y. Ma and S. Banerjee, "A smart pre-classifier to reduce power consumption of TCAMs for multi-dimensional packet classification," in ACM SIGCOMM, 2012.

[3] B Vamanan, TN Vijaykumar, "TreeCAM: decoupling updates and lookups in packet classification," in ACM CoNEXT, 2011.

[4] D. E. Taylor. and J. S. Turner, "Classbench: A packet classification benchmark," in IEEE INFOCOM, 2005.

[5] P. Gupta and N. McKeown, "Packet classification using hierarchical intelligent cuttings," in IEEE HOTI, 1999.

[6] S. Singh , F Baboescu, G. Varghese, J. Wang, "Packet classification using multidimensional cutting," in ACM SIGCOMM, 2003.

[7] B. Vamanan, G. Voskuilen, and T. Vijaykumar, "EffiCuts: optimizing packet classification for memory and throughput," in ACM SIGCOMM, 2010.

[8] F. Baboescu and G. Varghese, "Scalable packet classification," in ACM SIGCOMM, 2001.

[9] P. Gupta and N. McKeown, "Packet classification on multiple fields,"in ACM SIGCOMM, 1999.

[10] V Srinivasan, S Suri, G Varghese, "Packet classification using tuple space search," in ACM SIGCOMM, 1999.

[11] E. Spitznagel, D. Taylor, and J. Turner, Packet classification using extended TCAMs," in ICNP, 2003.