MEET-IP: Memory and Energy Efficient TCAM-based IP Lookup

Wenjun Li Xianfeng Li Hui Li*
wenjunli@pku.edu.cn lixianfeng@pkusz.edu.cn lih64@pkusz.edu.cn
School of Electronic and Computer Engineering, Peking University, P. R. China

***Abstract*—Ternary Content Addressable Memory (TCAM) is becoming very popular for designing high-throughput forwarding engines on most of today's high-end routers. Despite its capability for line-speed queries, it is very power hungry and capacity inefficient. Many recent research efforts, such as CoolCAMs and SmartPC, were proposed to reduce power consumption by constructing a pre-classifier to activate TCAM blocks selectively. However, these efforts achieve power savings at the expense of poor utilization of TCAM capacity, and the potential of power reduction is not fully exploited in many cases. In this paper, we propose MEET-IP, a memory and energy efficient two-stage scheme for TCAM based IP routing table lookup. Based on a top-down optimization of partitioning algorithm, routing tables can be evenly split into TCAM blocks without memory holes or a large amount of index TCAM. Experimental results based on real routing tables show that our design achieves more than 98% power reduction with a TCAM storage overhead of less than 3% on average.**

Keywords—IP routing table lookup; TCAM; power reduction; memory efficient

I. INTRODUCTION

IP routing table lookup is one of the critical issues in designing high-performance routers. It is a challenging problem due to following issues: 1) the size of the routing table has become more than half a million and keeps growing [1]; 2) cloud computing and network applications are pushing the line rate of core routers to 400Gbps or even higher; 3) one IP address may match multiple prefixes and the Longest Prefix Matching (LPM) should be applied; 4) the deployment of IPv6 will lead to larger routing tables with longer prefixes. With the exponential growth of the Internet, these challenges become stronger than ever. Thus, efficient and scalable solutions for IP routing table lookup are still under investigation.

Numerous IP routing table lookup techniques have been proposed in the past twenty years [2, 3]. They can be categorized broadly into two major approaches: software-based algorithmic solutions and hardware-based architectural solutions. Algorithmic solutions using ordinary memories and commercial processors have been well studied from the start because they are cheap and flexible. However, despite extensive researches, most of them are memory and performance inefficient, falling short of the needs of high-speed networks. Thus, architectural solutions using special hardware, such as TCAM and FPGA, have become the de-facto standard in industry for designing high-throughput forwarding engines on high-end routers. Of the several architectural solutions proposed, Ternary Content Addressable Memory (TCAM) is a promising device which has been widely deployed in existing network.

TCAM is a fully associative memory that allows a "don't care" state to be stored in each memory cell in addition to 0s and 1s. This feature makes it particularly attractive for packet classification and routing table lookup which require longest prefix matches [15]. When a destination address is presented to the TCAM, each TCAM entry is looked up in parallel and the longest matching address prefix is returned. Thus, a single TCAM access is sufficient to perform a routing table lookup. Moreover, TCAM-based table is much simpler to manage and update than that implemented using RAM (e.g., SRAM/DRAM) [14]. Despite their high speed and ease of management, TCAM has its own limitations with respect to IP routing table lookup [4]. First, TCAM has limited capacity and is not expected to increase dramatically in the near future. Second, TCAM chips consume large amount of power. For example, a typical 1Mb TCAM chip can consume up to 15-30 Watts of power, which is roughly 150 times more power per bit than SRAM. Third, TCAM is very expensive and its cost is a significant fraction of network device cost.

In recent years, leading TCAM vendors have provided block-based TCAM designs, where the TCAM device is partitioned into fix-sized blocks, and a subset of them can be activated for lookups as desired. This improved design provides a good substrate for potential power reductions. Many research efforts exploit this feature to optimize the power consumption of TCAM-based forwarding engines. CoolCAMs [13] proposed an architecture to split the entire routing table into multiple sub-tables or buckets, where each bucket is laid out over one or more TCAM blocks. SmartPC [16], one of the state-of-the-art works making use of blocked TCAM, introduced a pre-classifier by the idea of expanding and combining original classifier rules based on their IP address locations. However, these efforts achieve power savings at the expense of poor utilization of TCAM capacity, and the potential of power reduction is not fully exploited in many cases. Recently, GreenTCAM [24] achieves more energy reduction and solves the problem of poor utilization of TCAM capacity suffered in SmartPC, yet it cannot be well used for IP lookup.

* Hui Li is also with the Shenzhen Key Lab of Information Theory & Future Network Architecture, Future Network PKU Lab of National Major Research Infrastructure, Shenzhen Engineering Lab of Converged Networking Technology, Huawei & PKU Jointly Engineering Lab of Future Network Based on SDN and the PKU Institute of Big Data Technology, Shenzhen Graduate School, Peking University, P. R. China.

978-1-5090-2991-4/17/$31.00 ©2017 IEEE

In this paper, we propose MEET-IP, a two-stage TCAM-based scheme for IP routing table lookup. In the first stage, each incoming IP address is classified by a pre-classifier (or index), which provides information on which TCAM block needs to be activated in the next stage; in the second stage, the TCAM block from the first stage is activated and searched for a match. Generally speaking, there are three challenges for block-based TCAM scheme: 1) TCAM holes when accommodating multiple routing tables; 2) effective mapping of pre-classifier entries onto TCAM blocks; 3) quick index search mechanism during the pre-classification stage. For these purposes, we first turn the LPM problem into a point location problem through routing table projection, generating a set of address intervals. Based on these intervals, we propose a top-down partition algorithm for routing tables. Thanks to the routing table projection with top-down global views, routing prefixes can be evenly separated into blocks without huge memory holes or index TCAM entries as in CoolCAMs and SmartPC. Finally, an effective TCAM based pre-classifier is constructed for above TCAM blocks. The experimental results show that our design achieves more than 98% power reduction with an extra TCAM storage overhead of less than 3% on average, much better than CoolCAMs (92.78% with 34.5%) and SmartPC (88% with 84%). With the potential of parallelization and pipeline, one TCAM-RAM memory access can be realized in real implementations.

The rest of the paper is organized as follows. Section II briefly summarizes the related work. Section III presents the technical details of our work. Section IV provides our experimental results. Finally, Section V draws conclusions on this work.

II. Related Work

A lot of software-based algorithmic approaches have been proposed for IP lookup in the past twenty years. In general, they can be categorized into trie-based algorithms [5-11, 19], Bloom filter based algorithms [26, 27] and range-based algorithms [8, 19, 20]. However, despite extensive research, most approaches are memory and performance inefficient. Thus, architectural solutions using special hardware have been wildly studied in recent years. Based on their implementation platforms, we can categorize prior works into TCAM-based algorithms [14, 28-30], FPGA based algorithms [27, 31] and GPU-based algorithms [32, 33]. Among them, TCAM has been widely used for IP routing table lookup and packet classification because of its ability to process packet at line-speed. Moreover, TCAM-based solutions are much easier to manage than other hardware-based solutions.

Despite these advantages, this brutal force hardware solution is not only expensive and capacity inefficient, but also very power-hungry. The latest TCAM devices by leading vendors come with a power saving mechanism where a subset of its TCAM blocks can be selectively activated. A TCAM block is a contiguous, fixed-size chunk of TCAM entries, much smaller than the size of the entire TCAM. Recent research efforts exploit this feature to reduce power consumption by constructing a pre-classifier to activate TCAM blocks selectively. For the purpose of constructing more efficient pre-classifiers, we combine block-based TCAM designs with range-based routing table lookup algorithms. While some methods and concepts are not new, the novelty in MEET-IP is a subtle combination of algorithmic approaches and block-based TCAM designs, so that TCAM solutions can achieve huge power reduction without severe TCAM storage waste. Next, we give a more detailed review on a few representative energy efficient block-based TCAM techniques and related works about range-based routing table lookup algorithms.

A. Energy efficient TCAM schemes in IP lookup

CoolCAMs [13] is one of the most representative power-efficient TCAM-based routing table lookup algorithms. The key idea for the CoolCAMs architecture is to split the entire routing table into multiple sub-tables or buckets, where each bucket is laid out over one or more TCAM blocks. Two different tree-based partitioning schemes have been proposed: subtree-split and postorder-split. During the partitioning step of subtree-split algorithm, the prefix tree is traversed in post order to look for a carving node, whose count is at least half of TCAM bucket size. The drawback of subtree-split algorithm is that the smallest and largest bucket sizes vary by as much as a factor of 2, leading poor memory utilization with a lot holes in TCAM blocks. Postorder-split algorithm eliminates above TCAM holes through an even routing table partitioning, where each partitioning bucket is constructed from a collection of subtrees, rather than a single subtree as in the case of subtree-split algorithm. However, such an even partition comes at the extra cost of a larger number of entries in the index TCAM (at most W+1 index entries for the same one TCAM block, where W is the maximum prefix length in the routing table). With the continuous growth of routing table size and deployment of IPv6, the problem of index TCAM storage overhead will become more severe.

B. Energy efficient TCAM schemes in packet classification

SmartPC [16] is one of the state-of-the-art works making use of the blocked TCAM to reduce power consumption for forwarding engines. It introduced a pre-classifier for TCAM-based packet classification [18], where each incoming packet is forwarded according to a set of 5-tuple rules. The basic idea for constructing pre-classifier is to divide the whole source-destination dimensional space into some non-intersecting sub-spaces, each covering a subset of rules within it. However, its random bottom-up way of generating pre-classifier lacks the global view on the relationships of rules, leading a large number of general rules and TCAM holes. Recently, GreenTCAM [24] is proposed to solve the problem of poor utilization of TCAM capacity suffered in SmartPC. Based on the observation that most classifier rules contain at least one *small* IP address field [25], it separates the original classifier table into several sub-tables and builds pre-classifier for each sub-table. GreenTCAM achieves a 93.6% energy reduction with a TCAM storage overhead of

5.6% on average, much better than that in SmartPC (88% energy reduction with storage overhead of 84%). However, it cannot be applied to IP lookup.

C. Range-based routing table lookup algorithms

Range-based routing table lookup algorithm is one of the representative software-based solutions for routing table lookup. It stems from a well-known idea [19]: projecting a routing table onto a set of contiguous and non-overlapping address ranges or intervals, covering the entire address span of a network protocol. Then, the address interval containing the best matching prefix can be found through binary search. However, the original range-based routing table lookup algorithm suffers from prefix expansion problem, where at most $2n+1$ address intervals can be generated by projecting for n prefixes, leading bad update performance. The latest range-based routing table lookup algorithm is DXR [20], which achieves high lookup rate by taking advantage of cache efficiency. Instead of applying binary search immediately after projecting, DXR first merges neighbor address intervals resolving to the same next hop, improving the problem of prefix replication. After that, it introduced a lookup table similar to DIR-24-8 [12] to optimize the lookup performance for shorter prefixes. There are several drawbacks to this approach: 1) binary search for longer prefixes; 2) incapable to IPv6; 3) rebuild chunks from scratch for each update.

III. MEET-IP

In this section, we present MEET-IP, a memory and energy efficient two-stage scheme for TCAM based IP routing table lookup, which can also be applied to IPv6. We first transform the LPM problem to a point location problem through a routing table projection. After then, we propose a global optimal top-down partition algorithm for routing tables, which can effectively separate prefixes into TCAM blocks. Finally, a two-stage TCAM based data structure is used for routing table lookup. For the convenience of description and comparison, we introduce a sample 7-bit routing table containing 12 address prefixes, as shown in Table I, which has been used in CoolCAMs.

A. LPM problem → Point location problem

A network address prefix is commonly indicated as a pair *address/prefix length*, where only the most significant bits indicated by the prefix length are used for matching, and the rest least wildcard bits (*) are simply ignored. In essence, each address prefix corresponds to a range of values, where the lower bound and upper bound can be obtained by setting the wildcards to 0s and 1s respectively, as shown in the column of Prefix Ranges in Table I.

The range of an address prefix may be contained by the range of another address prefix, if and only if the later one is a prefix of the former one. For example, by projecting the address prefixes in Table I onto a number line, we find that P_2, P_3, P_4, P_5 and P_6 are all contained by P_1, as the prefix $0*$ of P_1 is a prefix of P_2, P_3, P_4, P_5 and P_6 respectively. Because of this property of the routing table, an IP address may match multiple prefixes. In this case, the longest prefix, which is most specific range containing the IP address, will be the best match. Algorithms finding the best match are thus called the Longest Prefix Matching (LPM) algorithms.

TABLE I. A SAMPLE ROUTING TABLE

Routing Table	Prefix Database (7 bits in length)		Next Hop
	Prefix Bit Strings	*Prefix Ranges*	
P_1	*0**	*[0, 63]*	*hop1*
P_2	*0100**	*[32,39]*	*hop2*
P_3	*01000**	*[32, 35]*	*hop3*
P_4	*0110**	*[48, 55]*	*hop4*
P_5	*01100**	*[48, 51]*	*hop5*
P_6	*01101**	*[52, 55]*	*hop6*
P_7	*10**	*[64, 95]*	*hop7*
P_8	*1101**	*[104, 111]*	*hop8*
P_9	*110100**	*[104, 105]*	*hop9*
P_{10}	*110101**	*[106, 107]*	*hop10*
P_{11}	*11011**	*[108, 111]*	*hop11*
P_{12}	*110111**	*[110, 111]*	*hop12*

With the projecting of address prefixes in Figure 1, a set of endpoints (i.e., *a, b, c…, m, n, p*) partition the whole number line into a series of contiguous and non-overlapping intervals, which are called ***address intervals*** in this paper. By introducing address intervals, the routing table lookup can be viewed as a problem of point location in a set of contiguous and non-overlapping address intervals. Therefore, given an incoming IP address, the first step is to decide the address interval where the IP address is located. Unlike the original LPM problem, there will only be a unique matching address interval for this IP address. The second step is to decide the best address prefix from all prefixes that cover the address interval obtained from the first step. For example, with an IP address of *0100001*, the address interval *[b, c]* in Figure 1 is selected, then among the three prefixes P_1, P_2 and P_3 covering *[b, c]*, the best matching prefix is P_3 as it is the most specific prefix for *[b, c]*, and in turn the most specific prefix for IP address *0100001*. The correctness of this two-step algorithm comes from an important fact: all address values (points) in an address interval share the same set of address prefixes covering each of them. That is why we can turn the LPM problem into a point location problem equivalently. Figure 2 and Table II show all address intervals and corresponding matching prefixes for routing table projection in Figure 1.

Based on this problem transformation, we then introduce an address interval based partition algorithm for routing tables, which can separate prefixes evenly into blocks without any TCAM holes (idle TCAM block entries) or huge extra index entries suffered by CoolCAMs and SmartPC. After that, we present two TCAM based pre-classifier architectures for high speed pre-classifications.

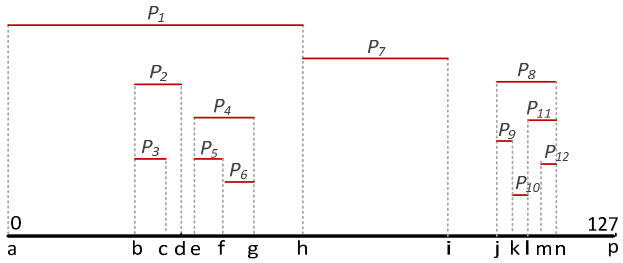

Fig. 1. Routing table projection from a geometric point of view

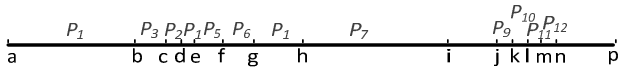

Fig. 2. LPM problem → Point location problem

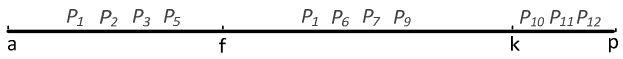

Fig. 3. Routing table partition with a TCAM block size of 4

B. *Interval based top-down partition algorithm*

One of the key issues in power efficient routing table lookup is how to separate all routing prefixes into a set of TCAM blocks and build the pre-classifiers for these blocks. Thus, in this section, we detail an address interval based top-down partition algorithm, with each partition contains as many prefixes as the TCAM block size (except possibly the last one subset).

From the above section, we know that routing table can be projected into a number line and the routing table lookup can be equivalently considered as a point location problem. From this point of view, the routing table and a set of address intervals can be equal to each other in some sense. Therefore, we propose a partition algorithm based on one-dimensional address intervals, rather than the original 32-level routing table. Intuitively, one dimensional partition problem is often easier to handle than 32-level separating problem.

More exactly, we first select the leftmost address interval in the number line (not included in any index range), and the initial sub-range corresponds to the selected address interval. Then, the sub-range continues its range expansion by moving its right end point to another right-side projection endpoint, with exactly one more address interval to be included or completely covered every time, until the subset of best matching prefixes included by current sub-range reaches the capacity limit of a TCAM block (i.e., one more included address interval may cause block overflow). When the sub-range expansion ends, it will produce a larger range, which we call the ***index range***. At the same time, a corresponding TCAM block can be allocated for the subset of prefixes that are included.

Repeat these two steps until all address intervals are included. It is easy to see that each of the generated index ranges is a combination of several consecutive and non-overlapping smaller address intervals.

TABLE II. ADDRESS INTERVALS & CORRESPONDING PREFIXES

Address Intervals	Corresponding Matching Prefixes	
	Interval Covered Prefixes	*Best Matching Prefix*
[0, 31]	P_1	P_1
[32, 35]	P_1, P_2, P_3	P_3
[36, 39]	P_1, P_2	P_2
[40, 47]	P_1	P_1
[48, 51]	P_1, P_4, P_5	P_5
[52, 55]	P_1, P_4, P_6	P_6
[56, 63]	P_1	P_1
[64, 95]	P_7	P_7
[96, 103]	*NULL*	*NULL*
[104, 105]	P_8, P_9	P_9
[106, 107]	P_8, P_{10}	P_{10}
[108, 109]	P_8, P_{11}	P_{11}
[110, 111]	P_8, P_{11}, P_{12}	P_{12}
[112, 127]	*NULL*	*NULL*

TABLE III. INDEX RANGE TABLE

Index Ranges	Included Address Intervals	TCAM Blocks
[0, 51]	*[0, 31], [32, 35], [36, 39], [40, 47], [48, 51]*	P_1, P_2, P_3, P_5
[52, 105]	*[52, 55], [56, 63], [64, 95], [96, 103], [104, 105]*	P_1, P_6, P_7, P_9
[106, 127]	*[106, 107], [108, 109], [110, 111], [112, 127]*	P_{10}, P_{11}, P_{12}

Take address intervals in Figure 2 as an example to describe the above merging method intuitively (assuming the TCAM block size is 4). First, the left-most address interval *[a, b]* is been selected, and the best matching prefix P_1 is included by the initial sub-range. In order to contain more prefixes, the sub-range *[a, b]* then moves its right end point from endpoint *b* to *c*, so that the next address interval *[b, c]* is exactly included, generating a new sub-range *[a, c]*. At this point, two address intervals are completely covered by the new sub-range, with two best matching prefixes been included (i.e., P_1, P_3). Since the number of included prefixes is still smaller than the block size, the current sub-range continues to move its right end point to produce a larger new sub-range, which may cover more not yet included address intervals. When the sub-range moves its right end point to endpoint *f*, the new sub-range *[a, f]* will include five address intervals within four unique best matching prefixes (i.e., P_1, P_2, P_3, P_5), reaching the storage capacity of a TCAM block. Thus, the first sub-range expansion process stops at endpoint *f*, and the index range *[a, f]* is generated with four best matching prefixes contained in its corresponding TCAM block.

After that, a new sub-range expansion process repeats above steps with an initial sub-range corresponding to the left-most not yet included address interval (i.e., *[f, g]*), until all remaining address intervals are included. Table III and Figure 3 show all index ranges and address intervals they include, as well as the corresponding TCAM blocks.

As can be seen from Table III, the routing table can be evenly partitioned into three TCAM blocks, with each block except the last one containing as many prefixes as the TCAM block size. This fine characteristic derives from the simple one-dimensional merging method which can include at most one prefix at a time, rather than an exponential order of prefixes in CoolCAMs. Indeed, CoolCAMs splits routing prefixes based on a 32-level bottom-up expansion method, where the range of the covering prefix is doubled with each level increment (or prefix length reduction), leading an uncontrollable covering prefix size. Assuming a TCAM block size of 4, Figure 4 plots a partition example of CoolCAMs using subtree-split algorithm. CoolCAMs first selects the leftmost bottom prefix P_3 as the initial covering index prefix, and then tries to cover more prefixes by doubling the covering range (i.e., *index 1*) each time. When *index 1* tries to cover prefix node *a* in step 4, three prefixes (i.e., P_4, P_5, P_6) should be included simultaneously, which will cause an overflow of a TCAM block. Thus, only two prefixes in step 3 (i.e., P_2, P_3) can be stored in the TCAM block, leaving remaining two entries in idle. In contrast, our proposed partition algorithm is derived from the global top-down partitioning method, with more insights into prefix relationship. As described above, all address values (points) in an address interval share the same set of address prefixes covering each of them, therefore, at most one prefix can be included by adding an address interval each time, achieving a more precise control of expansion.

It is not difficult to see that a prefix can be separated into multiple blocks, because it may span multiple index ranges (e.g., P_1 in Table III), and such duplicated prefixes may introduce additional storage overhead. Fortunately, in real routing tables, the proportion of such duplicated prefixes is almost negligible, which can be verified from our following experimental results. Essentially, this feature comes from a well-known observation about prefix length distribution, that is, there are very few routes with prefixes longer than 24-bit [10, 12, 20]. For ease of evaluation, we can define this prefix replication factor as ***blocking replication***, which can be obtained by the following formula: *the total prefix in TCAM blocks/routing table size*. Thus, the blocking replication of our sample routing table is *13/12*. More evaluation results for real routing tables can be seen in Section IV.

The pseudo code for the above routing table partition and pre-classifier construction algorithm is shown in Algorithm 1.

C. *Two-stage architecture with a TCAM based pre-classifier*

A pre-classifier can be constructed with a set of index ranges, which can be generated from above partition algorithm. For each incoming destination address, a pre-classifier is first searched for the index ID, following with a second search in the corresponding TCAM block to obtain the best matching result. Figure 5 shows the two-stage architecture of MEET-IP. Therefore, the key issue is how to perform a high-speed lookup in the pre-classification phase. Next, we describe two TCAM based techniques for high performance pre-classification.

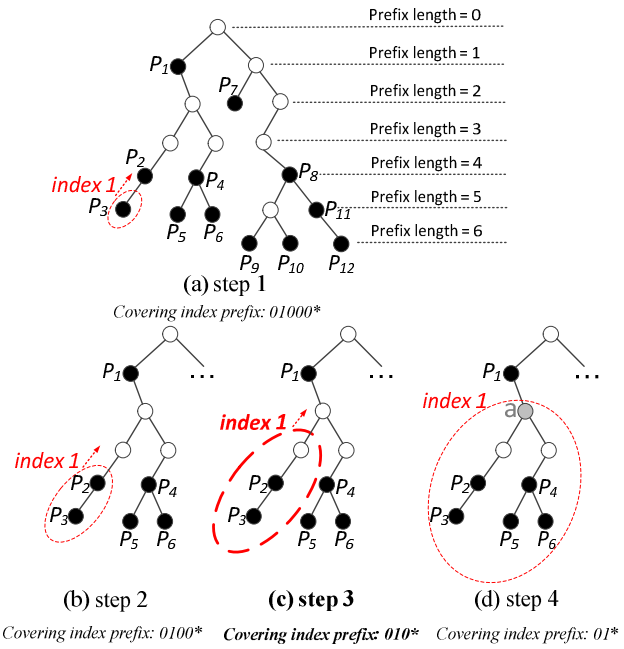

Fig. 4. A partition example of CoolCAMs (subtree-split algorithm)

Algorithm 1: BuildPreClassifier()

Input: address intervals & best matching prefixes
Output: pre-classifier & corresponding TCAM blocks

```
1: while (unused address intervals != null)
2:       covering range = leftmost unused intervals
3:       while(number of covering prefix < block size
                 and unused address intervals != null)
4:          add leftmost unused interval to covering range
5:       endwhile
6:       put covering prefixes in new TCAM block
7:       put covering range in pre-classifier
8: endwhile
```

As we know, storing ranges in TCAM relies on decomposing each individual range into multiple prefixes, which can lead to range-to-prefix blowout [22]. An effective range-to-prefix conversion scheme is first proposed in FIS [17], which is based on *elementary intervals*. In FIS, a range may be divided into multiple prefixes, and for a set of *n* ranges, the number of prefixes required in the worst case is *4n+2*. Another two state-of-the-art conversion schemes are proposed for solving the point intersection problem [21, 22], based on the concept of *Longest Common Prefix* (LCP). For *n* ranges, these LCP-based schemes can reduce prefix requirement to *2n+1* by using a 2-level TCAM-RAM architecture, where an extra comparison is needed in RAM. Based on these two concepts, we present two different TCAM based schemes for pre-classifier: 1) Range-split pre-classifier based on elementary intervals; 2) LCP-RAM pre-classifier based on longest common prefixes. Figure 5(a) and

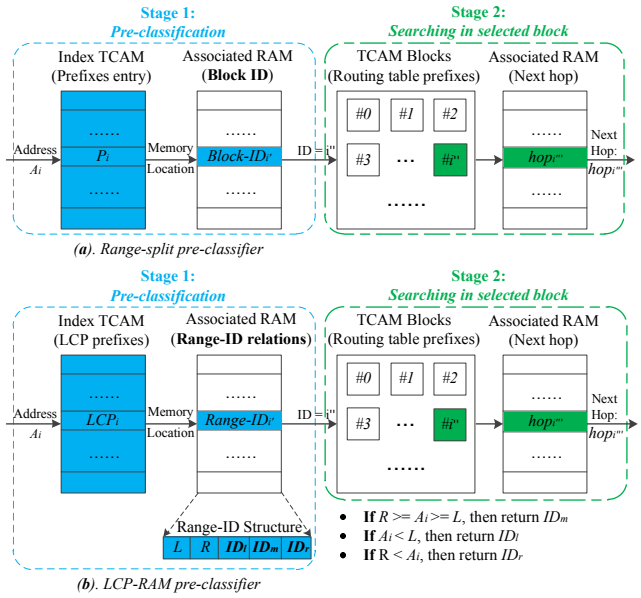

Fig. 5. The architecture of MEET-IP

(b) show architectures for these two schemes respectively. The detailed principle of range-to-prefix conversion can be found in [22].

From above routing table partitioning algorithm, it is not hard to see that there exists a desirable property for generated index ranges: they are a series of consecutive and non-overlapping ranges that cover the entire number line. Because of this fine characteristic, we draw two conclusions for our two TCAM based pre-classifiers containing n index ranges: 1) For a Range-split pre-classifier, at most *2n+1* prefixes are required in the worst case, dramatically reducing TCAM requirement as a factor of 2 compared to FIS; 2) For the LCP-RAM pre-classifier, only n prefixes are needed, following with a compare operation in the corresponding RAM. However, it is important to note that the data structure in RAM is different from previous LCP-based schemes [21, 22], which directly store 5-tuple rules or address prefixes. In contrast, the corresponding TCAM block IDs for each index range are stored in the Range-ID structure (e.g., ID_l, ID_m, ID_r).

Now, we give more searching details for each incoming destination address, based on above two-stage architectures respectively. For the Range-split pre-classifier architecture, each address is first presented to the index TCAM for retrieving best matching prefix, following with a TCAM block ID in its associated RAM. According to this ID, a second TCAM lookup is performed on its corresponding TCAM block, where the longest matching prefix can be obtained with the next hop information in its associated RAM. The only difference between the LCP-RAM and the Range-split pre-classifier is that, instead of directly acquiring TCAM block ID after the first TCAM lookup, an extra compare operation is needed in RAM for the LCP-RAM pre-classifier architecture. More details about compare operations and range-to-prefix conversions can refer to [17, 21, 22]. Apparently, each address processing in above architectures requires two TCAM-RAM accesses, twice the original TCAM solutions. However, with the potential of parallelization and pipeline, one TCAM-RAM access can be realized for above two energy-efficient architectures in real implementations.

TABLE IV. TWO CORE ROUTING TABLES

Table snapshot	Location	Date	Table size
rcc01_linx	*london*	*1/1/2012*	*396376*
oregon_linx	*oregon*	*17/12/2014*	*518231*

IV. EXPERIMENTAL RESULTS

In this section, we present some experimental results of MEET-IP using real 32-bit BGP routing tables. We start with an overview of our experimental methodology. Since the primary metrics for evaluating the performance of energy-efficient TCAM based solutions are power reductions and memory consumptions, we then evaluate power reductions and memory consumptions of our schemes separately. Due to time and length constraints, more features and evaluations of MEET-IP will be presented in future extensions, such as routing updates, scalability for IPv6.

1) Experimental Methodology

We evaluate the performance of MEET-IP using two representative routing tables obtained from *RIPE NCC* [1] and *Oregon Route Views Project* [23] respectively, which are recently used for evaluation by *SAIL* [10] and *Poptrie* [11]. Details of these two routing tables are listed in Table IV. Since the TCAM block size is an important parameter for evaluations, we show results with different block sizes to demonstrate how the performance of our scheme is affected by block size.

As mentioned in above sections, the main component of TCAM power consumptions is proportional to the number of activated TCAM blocks. For the convenient of evaluation, we employ a simple linear power model to estimate power reductions, though real reductions may be slightly different. Suppose the routing table contains N prefixes and the TCAM block size is B, the number of activated TCAM blocks for default TCAM schemes without using pre-classifier can be defined as X, where $X = \lceil N/B \rceil$. In order to forward a destination address using MEET-IP, the pre-classifier together with at most one specific TCAM block are needed to be activated. Thus, assuming that P pre-classifier entries are formed, at most $Y = \lceil P/B \rceil + 1$ TCAM blocks are activated for each routing lookup. Therefore, the percentage of ***power reduction*** with MEET-IP is $(X-Y)/X$. We use these definitions to following power reductions of MEET-IP.

Note that the primary objective of our work is to achieve power reductions without sacrificing TCAM consumption. There are two reasons for extra TCAM overhead: additional memory for building pre-classifier and multiple entries for each duplicated prefix (i.e., blocking replication). Suppose M pre-classifier entries are generated and an amount of Q

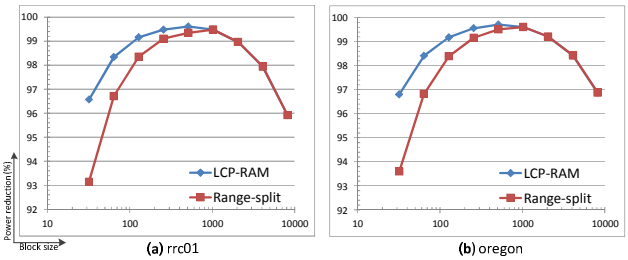

Fig. 6. The power reductions of MEET-IP

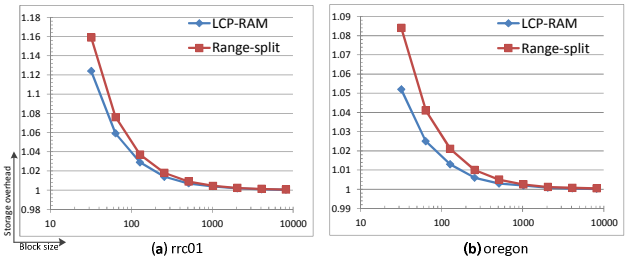

Fig. 7. The storage overhead of MEET-IP

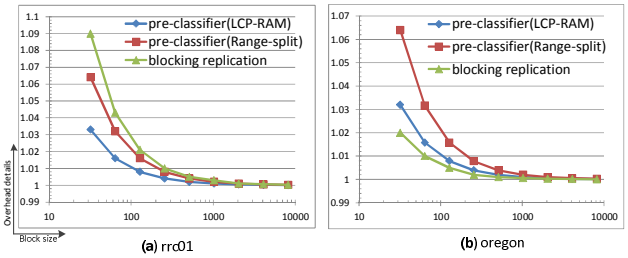

Fig. 8. More details about storage overhead

prefixes are stored in TCAM (include original prefixes and duplicated prefixes), we can calculate the ratio of ***storage overhead*** as $(M+Q)/N$. Besides, the *pre-classifier storage overhead* can be obtained by the following formula: $(Q+N)/N$. We use these definitions to following memory consumptions of MEET-IP.

2) *Power Reduction*

The reductions in power consumption using Range-split based and LCP-RAM based MEET-IP are shown in Figure 6, where x-axis represents block size and y-axis shows the percentage of power reduction. As shown in Figure 6(a) and (b), regardless of routing table sizes and types, MEET-IP achieves huge power reductions, ranging from 93.15% to 99.71%, with an average power reduction of 98.16%. We can also see that, with the increase of block size, the power reductions show a trend from rise to decline, finally reaching the same value for different schemes. The rising trend in the first stage can be explained by previous definitions of power reduction, which is defined by $(X-Y)/X$, where $X = \lceil N/B \rceil$ and $Y = \lceil P/B \rceil + 1$. With the increase of block size, $\lceil P/B \rceil$ will be a fixed value (i.e., 1) for different schemes (block size larger than the number of pre-classifier entries), leading same power reductions for different schemes.

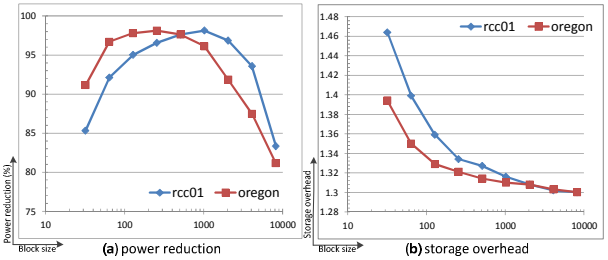

Fig. 9. The power reductions and storage overhead of CoolCAMs

3) *Storage Overhead*

The storage overhead using Range-split based and LCP-RAM based MEET-IP are shown in Figure 7, where x-axis represents block size and y-axis shows the ratio of storage overhead. We can see from Figure 7(a) and (b) that the storage overhead of MEET-IP ranges from 1.001 to 1.159, with an average storage overhead of 1.023. That is to say, only 2.3% extra TCAM overhead is needed for MEET-IP construction. Compared with the well-known evenly partition algorithm CoolCAMs (postorder-split based), MEET-IP only generate one pre-classifier entry for each TCAM block, much better than that in CoolCAMs (for above 32-bit routing tables, at most 33 pre-classifier entries are needed for the same one TCAM block). More comparisons with subtree-split based CoolCAMs and other energy-efficient TCAM designs are given in following sections. Another conclusion can be drawn from Figure 7(a) and (b): storage overhead is decline with the increase of block size.

As mentioned in above sections, two reasons may lead to extra storage overhead: pre-classifier and duplicated prefixes (i.e., blocking replication). Thus, to gain more insights about storage overhead of MEET-IP, we look into some details for these two reasons. The results are shown in Figure 8, where x-axis represents block size and y-axis shows the ratio of storage overhead. We can see that the average ratio of blocking replication and pre-classifier overhead is 1.012 and 1.011 respectively.

4) *Compare with CoolCAMs and SmartPC*

The reduction in power consumption and the ratio of storage overhead using subtree-split based CoolCAMs are shown in Figure 9(a) and (b) respectively, where x-axis represents block size and y-axis shows the results. We can see from Figure 9 that CoolCAMs reduces power consumption from 81.13% to 98.14%, with an average power reduction of 92.78%. However, the storage overhead of CoolCAMs ranges from 1.301 to 1.464, with an average storage overhead of 1.345. That is to say, CoolCAMs achieves huge power reductions with a 34.5% extra TCAM storage overhead. Besides, the experimental results given in SmartPC show that SmartPC achieves an average power reduction of 88% with an 84% extra TCAM storage overhead. Thus, these energy-efficient TCAM designs achieved significant power reduction at the expense of poor utilization of TCAM capacity.

Compared with CoolCAMs and SmartPC, MEET-IP achieves more power reductions (>98%) with much fewer TCAM storage overhead (<3%). This improvement, combined with the results in Figure 6-8, justifies our claim that careful combination of algorithmic approaches and block-based TCAM designs can achieve huge power reduction without severe TCAM storage waste.

V. CONCLUSION

TCAM is a promising device for building high-speed forwarding engines. Despite its capability for line-speed queries, this brutal force hardware is capacity inefficient and power-hungry. In this paper, we propose MEET-IP, a memory and energy efficient two-stage scheme for TCAM based IP routing table lookup. We first transform the LPM problem to a point location problem through a routing table projection method. Based on the projection, we then propose a top-down partition algorithm for routing tables, which can effectively separate prefixes into TCAM blocks. Finally, two effective TCAM based pre-classifiers are constructed for pre-classification. The novelty of our proposed MEET-IP is a subtle combination of algorithmic approaches and block-based TCAM designs, so that TCAM solutions can achieve significant power reductions without severe TCAM storage waste. The experimental results show that our design achieves more than 98% power reduction with an extra TCAM storage overhead of less than 3% on average.

ACKNOWLEDGMENT

This work is supported in part by the National Key Research and Development Program of China (No. 2016YFB0800101), by National Natural Science Foundation of China (NSFC) (No. 61671001 & No. 61521003), by Guangdong Key Program (GD2016B030305005) and by Shenzhen Research Programs (ZDSYS201603311739428, JCYJ20150629144717142, JCYJ20150331100723974 & 20140509093817684).

REFERENCES

[1] RIPE network coordination centre [on line]. Available: http://www.ripe.net.

[2] R. Miguel, B. Ernst, and D. Walid, "Survey and taxonomy of IP address lookup algorithms," *IEEE Network*, 15(2): 8-23, 2001.

[3] HJ. Chao and B. Liu, "High performance switches and routers," *John Wiley & Sons*, 2007.

[4] CR. Meiners, AX. Liu and E. Torng, "Hardware Based Packet Classification for High Speed Internet Routers," *Springer Science & Business Media*, 2010.

[5] V. Srinivasan and G. Varghese, "Faster IP lookups using controlled prefix expansion," *ACM SIGMETRICS Performance Evaluation Review*, 26(1): 1-10, 1998.

[6] S. Nilsson and G. Karlsson, "IP-address lookup using LC-tries," *IEEE Journal on selected Areas in Communications*, 17(6): 1083-1092, 1999.

[7] M.Degermark, A.Brodnik, S.Carlsson, and S.Pink, "Small forwarding tables for fast routing lookups," In *ACM SIGCOMM*, 1997.

[8] M. Waldvogel, G. Varghese, J. Turner, and B. Plattner, "Scalable high speed IP routing lookups," In *ACM SIGCOMM*, 1997.

[9] W. Eatherton, G. Varghese, and Z. Dittia, "Tree bitmap: Hardware /software IP lookups with incremental updates," *ACM SIGCOMM Computer Communication Review*, 34(2): 97-122, 2004.

[10] T. Yang, G. Xie, Y. Li, Q. Fu, AX. Liu, Q. Li, and L. Mathy, "Guarantee IP lookup performance with FIB explosion," In *ACM SIGCOMM*, 2014.

[11] H. Asai and Y. Ohara, "Poptrie: A compressed trie with population count for fast and scalable software IP routing table lookup," In *ACM SIGCOMM*, 2015.

[12] P. Gupta, S. Lin, and N. McKeown, "Routing lookups in hardware at memory access speeds," In *IEEE INFOCOM*,1998.

[13] Z. Francis, N. Girija, and B. Anindya, "CoolCAMs: Power-efficient tcams for forwarding engines," In *IEEE INFOCOM*, 2003.

[14] K. Zheng, C. Hu, H. Lu, and B. Liu, "An ultra high throughput and power efficient TCAM-based IP lookup engine," In *IEEE INFOCOM*, 2004.

[15] AJ. McAuley and P. Francis, "Fast routing table lookup using CAMs," In *IEEE INFOCOM*, 1993.

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

[17] A.Feldman and S.Muthukrishnan,"Tradeoffs for packet classification," In *IEEE INFOCOM*, 2000.

[18] D.Taylor, "Survey and taxonomy of packet classification techniques," *ACM Computing Surveys* , 37(3): 238-275, 2005.

[19] B. Lampson, V. Srinivasan, and G. Varghese, "IP lookups using multiway and multicolumn search," *IEEE/ACM Transactions On Networking*, 7(3): 324-334, 1999.

[20] Z. Marko, R. Luigi, and M. Miljenko, "DXR: Towards a billion routing lookups per second in software," *ACM SIGCOMM Computer Communication Review*, 42(5):29–36, 2012.

[21] S. Sharma and R. Panigrahy, "Sorting and searching using ternary CAMs," In *IEEE Hot Interconnects*, 2002.

[22] YH. Chang, "A 2-level TCAM architecture for ranges," *IEEE Transactions on Computers*, 55(12): 1614-1629, 2006.

[23] University of Oregon Route Views Project [on line]. Available: http://www.routeviews.org.

[24] X.Li, Y.Lin and W.Li, "GreenTCAM:A memory- and energy-efficient TCAM-based packet classification," In *International Conference on Computing, Networking and Communications (ICNC)*, 2016.

[25] W. Li and X. Li, "HybridCuts: A scheme combining decomposition and cutting for packet classification," In *IEEE Hot Interconnects*, 2013.

[26] D.Sarang, K.Praveen and D.Taylor, "Longest prefix matching using bloom filters," In *ACM SIGCOMM*, 2003.

[27] H. Lim, H. Lim, N. lee and K. Park, "On adding bloom filters to longest prefix matching algorithms," *IEEE Transactions on Computers*, 63(2): 411-423, 2014.

[28] D. Shah and P. Gupta, "Fast incremental updates on Ternary-CAMs for routing lookups and packet classification," In *IEEE Hot Interconnects*, 2000.

[29] P. Rina and S. Samar, "Reducing TCAM power consumption and increasing throughput," In *IEEE Hot Interconnects*, 2002.

[30] K. Zheng, C. Hu, H. Lu and B. Liu, "A TCAM-based distributed parallel IP lookup scheme and performance analysis," *IEEE/ACM Transactions On Networking*, 14(4): 863-875, 2006.

[31] M. Bando, Y. Lin and H. Chao, "FlashTrie: beyond 100-Gb/s IP route lookup using hash-based prefix-compressed trie," *IEEE/ACM Transactions On Networking*, 20(4): 1262-1275, 2012.

[32] S. Han, K. Jang, K. Park and S. Moon, "PacketShader: a GPU-accelerated software router," In *ACM SIGCOMM*, 2010.

[33] Y. Li, D. Zhang, AX. Liu and J.Zheng, "GAMT: a fast and scalable IP lookup engine for GPU-based software routers," In *ACM/IEEE ANCS*, 2013.