http://ansinet.com/itj

ITJ

ISSN 1812-5638

INFORMATION TECHNOLOGY JOURNAL

ANSInet
Asian Network for Scientific Information
308 Lasani Town, Sargodha Road, Faisalabad - Pakistan

Information Technology Journal 10 (10): 1964-1970, 2011
ISSN 1812-5638 / DOI: 10.3923/itj.2011.1964.1970

A Fast and Power Efficient Updating Algorithm for Partitioned TCAMs

Yonglin Wang, Yaping Lin, Shengye Huang, Gang Wang and Rui Li
School of Information Science and Engineering, Hunan University, No. 252,
Lushan South Road, Changsha, 410082, China

Abstract: Routing update is a severe problem for Ternary Content-Addressable Memories (TCAMs) based systems. This study is focused on designing a power efficient and fast updating algorithm for the partitioned TCAMs in IP forwarding systems. TCAMs are usually partitioned into individual buckets for their high power consumption. Due to routing rules change quickly, keeping the latest routing table may cause the partitioned TCAM buckets to overflow and bring additional power consumption. In order to improve the performance of the routing update, an efficient updating algorithm for the partitioned TCAMs is proposed in this study. In the algorithm, free space added for update traces is provided according to the distribution of the routing rules. The problem of buckets overflow is solved by dynamically adjusting the free space. Furthermore, the update speed is accelerated through arranging the free space of each partitioned bucket. Compared with the algorithm which increase the partitioned buckets by k (k is a fixed constant) times, the experimental results show the proposed algorithm is faster and more power efficient.

Key words: Routing table, update, low power, table partition, ternary content-addressable memory (TCAM)

INTRODUCTION

Ternary Content-Addressable Memories (TCAMs) allow three states 0, 1 and * to be stored in each memory cell (Gupta and McKeown, 2000). TCAMs are widely used in IP forwarding engines for their parallelism which allows all TCAM entries to be searched through a single lookup (Song *et al*., 2009). However, power consumption of TCAMs is very high for their parallelism. Researchers have developed trie-based architectures to resolve this problem. The basic idea of their algorithms is using a binary trie to construct the routing table, then partition the constructed trie into individual buckets. Each lookup searches one of the buckets so as to reduce the number of the searched TCAM entries.

Although the trie-based partition schemes could reduce the power consumption effectively, the schemes did not consider real-time updates of the routing table. In practice, the routing rules are dynamic and there are almost 100 to 1000 prefixes updated to the routers per second (Yi-Mao *et al*., 2009). In the update process, routing rules can be inserted into the routing table or deleted from it (Korde and Khanale, 2011; Alsaade, 2010; Weiping and Jianyu, 2008; Rupsys *et al*., 2011). Inserting rules may cause some buckets to overflow which will slow-down routing forwarding speed and bring serious network congestion. Meanwhile, Power consumption caused by the update rules should be considered. Now-a-days the applicability of the partition TCAM schemes is limited by the routing updates. Therefore, routing update is a very complex and important part of the partitioned TCAM schemes.

Previous researchers improved the update performance by managing the added free TCAMs. Hae-Jin *et al*. (2006) and Reddy (2010) developed updating algorithms for TCAM-based routing systems. These algorithms were proposed for the small routing table systems. Besides, the authors supposed the free TCAMs size can be infinite. Therefore, the two algorithms do not suit the partition TCAM schemes. The Centralized Sorting Prefix (CSP) algorithm was proposed by Shah and Gupta (2002), researchers put free space at the bottom of TCAMs. However, the worst-case time complexity of the algorithm is very slow. Weidong *et al*. (2004) proposed a Chain-ancestor ordering constraint optimal algorithm (CAO_OPT) which allocated the free space in the middle of TCAMs. CAO_OPT converted the routing table into trie structure and only sorted the prefixes on the same chain (the path from the root to a leaf node). Compared to the CSP algorithm, the CAO_OPT algorithm reduced many unnecessary memory movements. It improved the update speed greatly. However, all the two algorithms did not

Corresponding Author: Yaping Lin, School of Information Science and Engineering, Hunan University, No. 252,
Lushan South Road, Changsha, 410082, China Tel: 86-731-88821932 Fax: 86-731-88821977

consider the extra energy consumption caused by the added free TCAMs and sizes of their free TCAMs were very large. Therefore, they do not suit the partitioned TCAM schemes. Lu and Sahni (2010) proposed a batch update model for the partitioned TCAM schemes. In the batch update model, update rules were batched at regular intervals. A new routing table was constructed at the end of the batch interval. The main drawback of this algorithm is that the routing rules are not real-time. Zane *et al.* (2003) proposed an updating algorithm for the partitioned TCAM schemes. They just increased the bucket size by k times to avoid repartition, k was a fixed constant. Meanwhile, the researchers did not present a method for computing k. However, performance of the algorithm needed to be improved. For example, all the N TCAM entries are partitioned into 8 buckets. Applying this algorithm, all the buckets size are increased by 10 times to afford additional free space. Then at least 5N/4 entries are searched during a single lookup while N/8 TCAM entries need to be searched before applying the update algorithm. This updating algorithm also has fatal defects. The power consumption caused by added free TCAMs is very high when the bucket size is large and when the bucket size is small, the update traces may cause repeated repartition. From the above, routing update algorithm of the partition TCAM schemes should minimize the added free TCAMs size. Meanwhile, enough free TCAMs must be afforded to provide space for the inserted update rules.

How to update the routing rules of partitioned TCAMs is a very complex task because both the power consumption and the update speed have to be considered. There are three key challenges in updating the routing table of the partitioned TCAM schemes. Firstly, it is very difficult to reduce the free TCAMs because the newly inserted rules require plentiful free space (Gupta and Agrawal, 2009; Hsieh and Chen, 2011). Routing update process includes inserting and updating rules. Plenty free TCAMs should be prepared for the new rules. However, the numerous added free TCAMs may bring additional heavy burden to the routing systems. Therefore, it is hard to equip applicable free TCAMs. Secondly, it is a challenge to minimize the repartition operations of the entire routing table. Updating rules may continually happen in a certain bucket and eventually cause the bucket to overflow, requiring a repartition of the entire routing table (Elizabeth Shanthi and Nadarajan, 2009; Alfantookh *et al.*, 2008; Zhang *et al.*, 2009). Repartitions of routing table may slow down the routing speed. Infinite free TCAMs can be added to avoid repartition operations. However, this is impractical and also goes against the first item. Thirdly, it is difficult to accelerate the update speed. The lookup operation in routing table uses the Longest Prefix Matching (LPM) mechanism to decide the next hop (Kai *et al.*, 2004; Agrawal and Sherwood, 2008). And the routing rules stored in partitioned buckets are in a descending order of prefix lengths. The order must be kept during the update process. This limitation brings immense memory movements which can slow down the update speed.

This study proposed a fast and power efficient updating algorithm for the partitioned TCAM schemes. The proposed algorithm overcomes the three dilemmas mentioned above. Using the techniques by Taylor and Turner (2007), simulated routing rules and related update traces were generated to evaluate the algorithm. The idea of the proposed algorithm is to provide free TCAMs for update traces and set the free space size of every partitioned TCAM bucket according to the distribution of the routing rules. Update rules usually distribute nearby in the routing trie and the update rules assigned to each bucket are uneven. To reduce power consumption of the added free space, the free TCAMs size of the buckets which have fewer routing prefixes is cut down. Actually, routing updates mainly take place in a few partitioned buckets. And the proposed updating algorithm could save the free TCAMs commendably. Learning from the TCP congestion control, a scheme which aims to avoid frequent repartitions is proposed. Free TCAMs of the partitioned buckets are dynamically adjusted during the update process. In order to avoid frequent repartitions of the routing table, free TCAMs are increased while the update traces concentrate together. When the update rules withdraw, buckets with overmuch free TCAMs are cut down to reduce the power consumption. The scheme could prevent frequent repartitions well and outperformed other algorithms in the experiments. Through arranging the free space, the memory movements caused by the update traces and the update speed have been improved greatly. Free TCAMs of every partitioned bucket are distributed according to the rules distribution. Since the routing rules are in order of decreasing prefix lengths and rules of the same length can be in any order, rules of each prefix length are allocated corresponding free TCAMs. The results show that memory movements can be significantly reduced. The proposed algorithm can be easily implemented in the routing systems and does not require any modification to the hardware.

BACKGROUND OF PARTITIONED TCAMS

High power consumption is a major drawback of TCAMs, since TCAMs look up all the entries in parallel

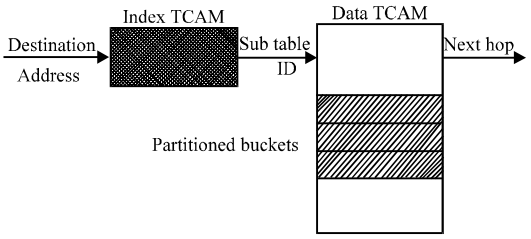

Fig. 1: Trie-based power reduction architecture

(Lu and Sahni, 2010; Zane *et al*., 2003; Weirong and Prasanna, 2009; Wencheng and Sahni, 2008). Lu and Sahni (2010) and Zane *et al*. (2003), proposed trie-based architecture to improve the power consumption. Their basic idea is to partition the TCAM entries into individual buckets and search only one bucket in a single lookup. The trie-based architecture is shown in Fig. 1.

In Fig. 1, routing rules are stored in the data TCAM and the data TCAM is partitioned into independent buckets. The ID of each partitioned bucket is stored in the index TCAM. A destination address is first presented to the index TCAM to find its matching TCAM bucket and then search the matched bucket to decide its next hop.

Subtree-split algorithm proposed by Zane *et al*. (2003) is a fundamental trie-based partition scheme. Routing rules in data TCAM are firstly converted into 1-bit trie structure. Researchers set b to be the maximum size of a partitioned bucket. For any node v, the number of routing prefixes of the subtree rooted at v is defined as the count of v. Carving nodes of the entire constructed trie are searched in post order. Carving nodes are the nodes with counts in the range [[b/2]2] and counts of their parents larger than b. Finally, the prefixes of the subtree rooted at the carving nodes are divided into partitioned buckets in post order.

In Fig. 1, frequent repartitions of the data TCAM during the update process should be avoided. Besides, we should reduce power consumption caused by the added free TCAMs to follow the purpose of the partition schemes.

ROUTING RULES UPDATING ALGORITHM

Since the distribution of new update rules in the binary trie construction is concentrated (Zane *et al*., 2003). The added free space is dynamically adjusted according to the arrived update rules and the distribution of the routing rules.

Initial size: In TCAM-based IP forwarding systems, researchers provided free space for the new inserted update rules. For the partitioned TCAM schemes, routing updates take place mainly in a few buckets (Zane *et al*., 2003). For that reason, these buckets need larger free space and the other buckets need less. In the long term, distribution of the update rules is consistent with the routing rules. Consequently, the free TCAMs are added according to the rules of each partitioned bucket in addition to the routing table.

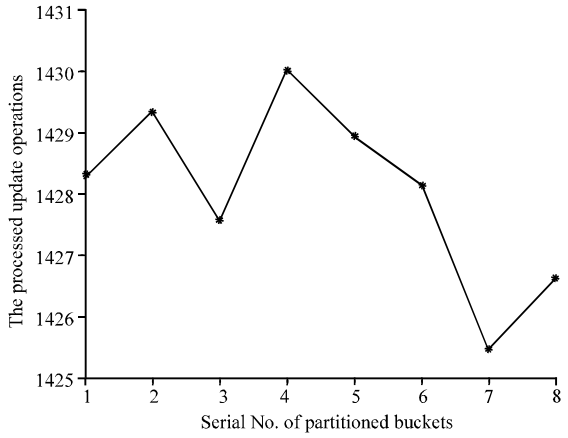

Fig. 2: Update operations divided by C^i

For the N prefixes stored in the data TCAM, prefixes of length j are referred as p_j. The prefixes of length j stored in bucket i are referred as α_j^i. P_j/N denotes the distribution of the routing rules. Because update rules are also related with the number of prefixes lengths of each partitioned bucket, so the number of lengths of bucket i is recorded by C^i. More different prefixes lengths of a partitioned bucket mean more update rules will take place in the bucket. This was verified in the experiment. The inserting and deleting operations of each partitioned bucket are counted during the update process. Meanwhile, the number of the rules lengths of each bucket is computed. In order to reflect the effect of different lengths number to the update operations, the total update operations of bucket i are divided by C^i. In the experiment, the routing rules are partitioned into 8 buckets. The related experimental results are presented in Fig. 2. As shown in the figure, errors between the 8 buckets are less than 0.35%. This suggests that update operations are proportional to C^i. As shown in Eq. 1, S^i is used to compute the initial size of the free space of bucket I:

$$S^i = \sum_{j=0}^{j=32} \alpha_j^i * (p_j / N) * C^i \quad (1)$$

The computing method takes both the distribution of the routing rules and the rules of each bucket into

consideration. Its basic idea is preliminarily estimating the distribution of update rules in order to add the free space more efficiently.

Solution for buckets overflows: Buckets overflow is a severe problem for routing update of partitioned TCAM schemes. It may bring frequent repartition operations that can slow down the packets forwarding speed greatly. Zane *et al*. (2003) developed a way to avoid frequent repartitions while buckets overflowed. But the algorithm is not efficient when update rules concentrate together. Inspired by the TCP congestion control, an algorithm is proposed in the study to solve this problem. Pseudo code of the solution for buckets overflow is shown in Algorithm.

Algorithm: Solution for buckets overflow

```
If bucket i overflow then
put the new arrival rules of bucket i to its neighbors.
if the neighboring buckets of bucket i become full then
repartition the entire routing table;
double the free space size of bucket i;
      for the latest un-update buckets && size of their free space exceed the
initial size do
        reduce the free space size by half.
   end for
end if
end if
```

The free space size is dynamically adjusted in consideration of the suddenness and randomness of the update rules, since the update floods usually hit the same TCAM bucket in a short time (Zane *et al*., 2003). On a repartition, the free space size is doubled to absorb more subsequent prefix additions. According to the experimental results and the experimental data by Zane *et al*. (2003), the number of repartition operations becomes very small when the free space size is about nine times of the bucket size. The algorithm proposed in this study can increase the buckets size by ten times no more than three repartition operations. Therefore, double the free space can meet the update demands. The free space size of the latest un-update buckets is reduced on a repartition to cut down the power consumption.

Location of free space: Routing table lookup operation searches the longest prefix of the matched rules. Prefixes stored in partitioned buckets are in order of descending prefix lengths (Akhbarizadeh and Nourani, 2005). The inserting and deleting operations caused by update traces must keep the decreasing order. This may bring enormous memory movements. Since prefixes of the same length can be located in an arbitrary order, the update speed is accelerated through optimizing the location of the free space. For each partitioned bucket, rules of the same length are allocated together and the free space is

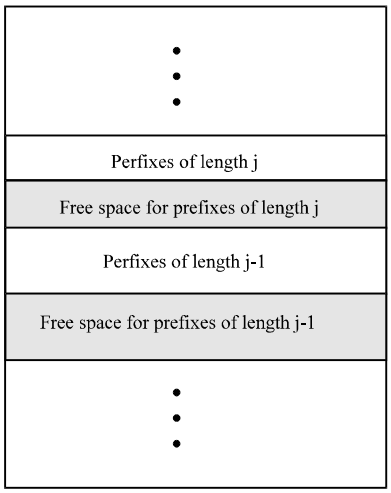

Fig. 3: Free space distribution of a partitioned bucket

assigned according to the proportion of the rules of each length. For bucket i, the proportion of prefixes of length j is referred as β^i_j. The free space assigned to prefixes of length j is referred as f^i_j:

$$f^i_j = \left| s^i * \beta^i_j \right| \quad (2)$$

The assigned free space is put at the bottom of the related rules and the free space distribution of an arbitrary partitioned bucket is presented in Fig. 3. In Fig. 3, prefixes stored in this bucket are in order of decreasing prefix lengths. Free space is allocated at the bottom of rules of each length. The inserted new rules ask free space from the empty entries below them. If the related free space is full, use the other free space until the added free entries for the bucket are all used. Compared with the algorithm that puts all the free space at the bottom of each bucket, the algorithm proposed in this study can avoid causing the movements of many irrelevant rules.

EXPERIMENTAL RESULTS

The experiments are conducted on a desktop PC running Windows XP SP3 with Duo 3.0 GHz Inter Core 2 processor and 2 G memory. 61176 routing rules and 305880 related update rules are generated from ClassBench (Taylor and Turner, 2007) tool for evaluation. ClassBench includes a suite of tools to produce synthetic filter sets and exercise traces. It is widely used for benchmarking packet classification devices and algorithms. To evaluate the algorithm of this study, the total size of the added free

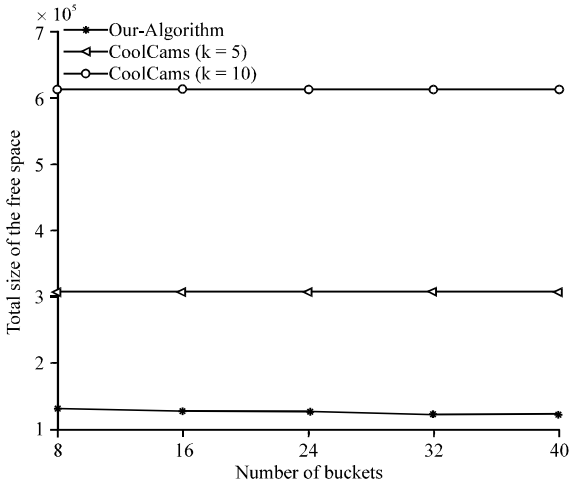

Fig. 4: Size of added free TCAMs

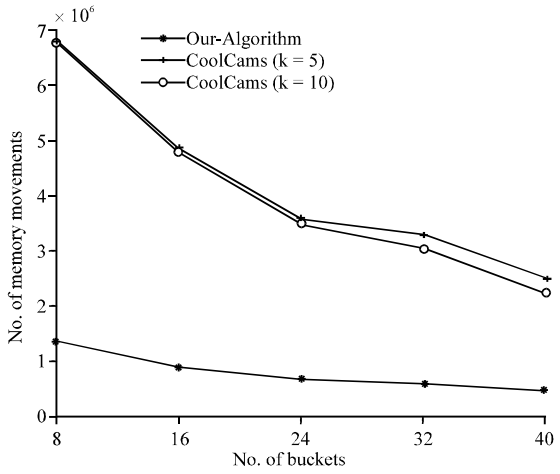

Fig. 5: Number of memory movements caused by inserting and deleting operations

space, repartition operations and memory movements are tested during the update process. All of the experiments are based on the subtree-split partition scheme proposed in CoolCams (Zane *et al*., 2003). The experimental results are also compared with the updating algorithm proposed in CoolCams. Referred to the update experiments of CoolCams, the constant coefficient k is set to 5 and 10 in the experiments.

Efficiency: For routing updates of partitioned TCAM schemes, the free space should be afforded as few as possible to reduce the power consumption. Here, size of the added free space is compared with the other two cases. Results are presented in Fig. 4, x-axis denotes the number of the partitioned buckets and y-axis denotes the total size of the added free space. As the experimental results show, the added free space of the proposed algorithm is the fewest. The additional free space caused by repartitions is very few compared to the total size. Therefore, it is much more power efficient and can also save TCAM devices compared to the other two algorithms.

In order to reduce the memory movements caused by inserting and deleting operations, the free space is arranged to optimize the update performance. The results of the memory movements are presented in Fig. 5. In comparison, the algorithm proposed in this study is much better than the other two methods. It can reduce the movements by about 5 times compared to the CoolCams (k = 5). The tiny differences between CoolCams (k = 5) and CoolCams (k = 10) owing to the buckets overflow. Through reducing movements of the irrelevant rules, algorithm of this study can immensely accelerate the update speed.

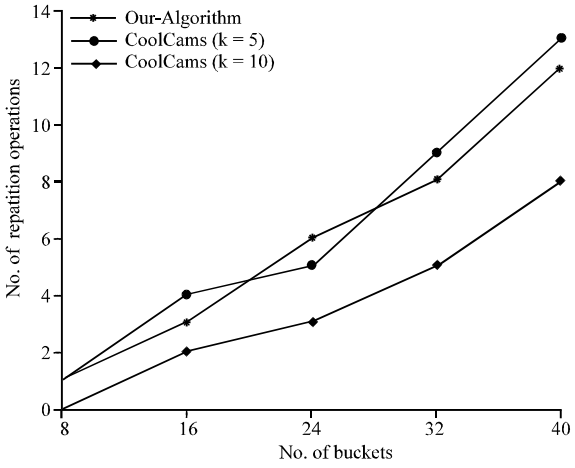

Fig. 6: Number of repartition operations during the update process

Effectiveness: During the update process, inserting rules (prefixes) may cause a partitioned bucket to overflow, requiring a repartition of the entire routing table stored in the data TCAM. This can badly damage the routing speed. Therefore, the repartition operations should be minimized. Figure 6 shows the number of repartitions for the three cases. As it shows, the results of the algorithm proposed in this study are close to the CoolCams (k = 5). When the number of partitioned buckets is less than 24, the repartition operations are less than 6. This is very small compared with the 305880 update rules. Therefore, it is suitable for the partitioned TCAM schemes.

CONCLUSION

In this study, a fast and power efficient updating algorithm is developed for the partitioned TCAM schemes. There are three main contributions in this study. Firstly, a method for computing the free space of each partitioned bucket is developed. The algorithm can add the free space reasonably and it is very power efficient. Secondly, a solution for buckets overflow is developed in the study. Frequent repartitions can be avoided when the update floods continually hit a single bucket. Thirdly, the memory movements caused by the update floods is reduced through arranging the free space of the each partitioned bucket. The experimental results demonstrated the proposed updating algorithm is appropriate for the partitioned TCAM schemes.

ACKNOWLEDGMENT

This study was partly supported by National Natural Science Foundation of China under Grant No. 60973031 (2010.1-2012.12, Hunan University).

REFERENCES

Agrawal, B. and T. Sherwood, 2008. Ternary CAM power and delay model: Extensions and uses. IEEE Trans. VLSI Syst., 16: 554-564.

Akhbarizadeh, M.J. and M. Nourani, 2005. Hardware-based IP routing using partitioned lookup table. IEEE/ACM Trans. Network., 13: 769-781.

Alfantookh, A.A., K.M. Al-Hazmi and S.H. Bakry, 2008. Assessment indicators for information technology in higher education institutions: A STOPE approach. Asian J. Inform. Manage., 2: 1-13.

Alsaade, F., 2010. Symmetric crypto-graphical model. Trends Applied Sci. Res., 5: 146-151.

Elizabeth Shanthi, I. and R. Nadarajan, 2009. An index structure for fast query retrieval in object oriented databases using signature weight declustering. Inform. Technol. J., 8: 275-283.

Gupta, P. and N. McKeown, 2000. Classifying packets with hierarchical intelligent cuttings. IEEE Micro, 20: 34-41.

Gupta, R.K. and D.P. Agrawal, 2009. Improving the performance of association rule mining algorithms by filtering insignificant transactions dynamically. Asian J. Inform. Manage., 3: 7-17.

Hae-Jin, J., S. Il-Seop, Taek-Geun and L. Yoo-Kyoung, 2006. A multi-dimension rule update in a TCAM-based high-performance network security system. Proceedings of the 20th International Conference on Advanced Information Networking and Applications, April 18-20, Vienna, Austria, pp: 62-66.

Hsieh, Y.C. and S.H. Chen, 2011. An empirical study of technological innovation, organizational structure and new product development of the high-tech industry. Inform. Technol. J., 10: 1484-1497.

Kai, Z., H. Chengchen, L. Hongbin and L. Bin, 2004. An ultra high throughput and power efficient TCAM-based IP lookup engine. Proceedings of the 23rd Annual Joint Conference of the IEEE Computer and Communications Societies, March 7-11, Hong Kong, pp: 1984-1994.

Korde, P.S. and P.B. Khanale, 2011. Cache oblivious matrix multiplication algorithm using sequential access processing. Res. J. Inform. Technol., 3: 61-67.

Lu, W. and S. Sahni, 2010. Low-power TCAMs for very large forwarding tables. IEEE/ACM Trans. Networking, 18: 948-959.

Reddy, P.M., 2010. Fast updating algorithm for TCAMs using prefix distribution prediction. Proceedings of the 2010 International Conference on Electronics and Information Engineering, Aug. 1-3, Kyoto, pp: 400-404.

Rupsys, P., E. Bartkevicius and E. Petrauskas, 2011. A univariate stochastic gompertz model for tree diameter modeling. Trends Applied Sci. Res., 6: 134-153.

Shah, D. and P. Gupta, 2002. Fast updating algorithms for TCAM. IEEE Micro, 21: 36-47.

Song, H., M.S. Kodialam, F. Hao and T.V. Lakshman, 2009. Scalable IP lookups using shape graphs. Proceedings of the 17th IEEE International Conference on Network Protocols, Oct. 13-16, Princeton, NJ., USA., pp: 73-82.

Taylor, D.E. and J.S. Turner, 2007. ClassBench: A packet classification benchmark. IEEE/ACM Trans. Network., 15: 499-511.

Weidong, W., S. Bingxin and W. Feng, 2004. Fast updating scheme of forwarding tables on TCAM. Proceedings of the 12th Annual International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunications Systems, Oct. 4-8, USA., pp: 522-527.

Weiping, H. and X. Jianyu, 2008. A new data structure and algorithm for static slicing concurrent programs. Inform. Technol. J., 7: 253-260.

Weirong, J. and V.K. Prasanna, 2009. Reducing dynamic power dissipation in pipelined forwarding engines. Proceedings of the IEEE International Conference on Computer Design, Oct. 4-7, Lake Tahoe, CA., pp: 144-149.

Wencheng, L. and S. Sahni, 2008. Low power TCAMs for very large forwarding tables. Proceedings of the 27th Conference on Computer Communications, April 13-18, Phoenix, AZ., pp: 316-320.

Yi-Mao, H., C. Ming-Jen, H. Yu-Jen, S. Hui-Kai and C. Yuan-Sun, 2009. A fast update scheme for TCAM-based IPv6 routing lookup architecture. Proceedings of the 15th Asia-Pacific Conference on Communications, Oct. 8-10, Shanghai, pp: 857-861.

Zane, F., N. Girija and A. Basu, 2003. Coolcams: Power-efficient TCAMs for forwarding engines. Proceedings of the 22nd Annual Joint Conference of the IEEE Computer and Communications Societies, March 30-April 3, San Franciso, CA., USA., pp: 42-52.

Zhang, P., F. Wang, Z. Xu, S. Diouba and L. Tu, 2009. Opportunistic distributed space-time coding with semi-distributed relay-selection method. Res. J. Inform. Technol., 1: 41-50.