High Performance Parallel Scalable Packet Classification Architecture with Popular Rule Caching

Samoda Gamage, Ajith Pasqual
Department of Electronic and Telecommunication Engineering
University of Moratuwa
Sri Lanka
Email: {samoda, pasqual}@ent.mrt.ac.lk

Abstract—Packet Classification is the enabling function for many Internet functions like QoS and Security. In this paper we propose a Classification engine architecture which exploits parallelism to increase throughput. The architecture also make use of the Temporal locality observed in Internet traffic positively by employing Popular Rule Caching mechanism to increase the classification throughput. We also introduce Rule Splitting mechanism to increase the accuracy of the rule caching mechanism. Simulation results revealed that the architecture is capable of achieving a throughput of more than 200Gbps when lowest amount of temporal locality is present for worst case packet size of 40 bytes.

I. INTRODUCTION

With the evolution of Internet, next generation routers have to support a variety of existing and emerging network applications such as firewall functions, Quality of Service differentiation, traffic profiling and many more services. The key function that is common to all these services is the categorizing of packets according to predefined rules which consists of multiple fields corresponding to the packet header. This process is called multi field packet classification. A traffic flow is defined by five parameters, source IP address and destination IP address of layer three, Source port, destination port and protocol of layer four.

Plurality of work have been done in the area of packet classification. Most of them are decision tree based [1], [2] running on software but couldn’t keep up with the ever increasing data rates. Modern solutions of migrating software algorithms to hardware domain, specially to FPGAs, is popular [3]–[5] which claim to have throughputs ranging 20 - 100 Giga bits per second(Gbps).

Ternary Content Addressable Memory(TCAM) has become a popular solution to packet classification due to its O(1) performance and simple implementation. Even though many research have been done in this area most of them focus on the preprocessing of the rule sets to optimize TCAM space and power utilization [6]–[8]. Despite its superior performance, TCAM based solutions struggle at keeping up with higher line rates due to their low operating speeds than SRAM. DPPC-RE scheme proposed in [9] utilizes chip level parallelism of TCAM chips to process multiple packet flows simultaneously to alleviate this problem and introduce scalability. It is scalable in the sense of both throughput and number of rules. This scheme consider the temporal locality as a burden and try to alleviate it by distributing rules among the TCAMs in a way such that the incoming traffic load is evenly distributed among the TCAM chips. Drawback of this approach is that it requires the knowledge about the popularity of the classification rules i.e. the number of occurrences of a specific search rule in a given time duration and makes use of this information when distributing the rules among the TCAMs. Considerable amount of extra hardware is needed to monitor the occurrences of the search keys which would increase the complexity and monitoring and recording of these statistics would become tiresome at high speeds. But the researchers have assumed that those information are available and built their architecture upon that assumption. Also this architecture cannot tackle the rapid changing bias of the internet traffic because the table construction algorithms need considerable amount of time to learn the traffic patterns. [9] is the only classification engine architecture based on TCAMs that we could find in literature which utilizes multiple parallel TCAMs and capable of processing multiple packets simultaneously.

Many research have been done on the characteristics of Internet traffic [10], [11]. They reveal that the Internet traffic exhibit a high degree of temporal locality. Based on this observation, several flow caching mechanisms have been proposed to exploit this property to increase the classification throughput. A flow is defined as a stream of packets having the same source and destination addresses, same port numbers and protocol. [12]–[14] present flow caching architectures, cache replacement policies to be used with flow caching which claim to have good hit ratios.

With the growth of link rates, the degree of temporal locality present at flow level is has been questioned. This is due to the fact that average bandwidth of individual flows has not increased at the same rate as the aggregated line rates. Also many of the traffic flows are observed to be short lived and therefore large numbers of concurrent flows could be observed at tire 1 links, which requires large numbers of cache entries for an effective caching process. Due to these reasons flow
caching has become impractical today.

Rule Caching has been proposed as a solution where classification rules are cached instead of rules. Since several flows would be represented by a single rule, the combined temporal locality of a rule is much higher than a flow. [15] uses a concept called smart rule caching mechanism to be used with packet classification with less than four cache entries and claims that Cache miss ratios never exceed 0.5% all the time which is 2-4 orders of magnitude lower than flow cache schemes. [16] also suggest the viability of using rule caching over flow caching in order to improve PC throughput.

Even though all the above caching mechanisms are promising, they only justifies the use of Rule/Flow caching to increase the throughput. They do not propose any final classification engine architecture integrating the cache unit or do not provide any final throughput evaluation and do not discuss about the scalability or introducing parallelism to further improve PC throughput.

In this paper we propose a high throughput, parallel, Power efficient Packet Classification Engine architecture. The architecture utilizes popular rule caching mechanism and parallelly operating TCAM chips to classify multiple packets simultaneously to increase the final classification throughput which is not observable in previous TCAM based classification schemes. We do it by extending the MSMB-LPT scheme [17] which has been introduced for high speed packet forwarding, in to multi field packet classification. We also discuss sources of errors which could lead to erroneous classification and introduce Rule Splitting as a remedy to increase the reliability of the caching process.

The rest of this paper is organized as follows, section 2 describes the Classification Engine Architecture and implementation. Section 3 contains performance evaluation of the architecture and section 4 concludes the paper.

II. CLASSIFICATION ARCHITECTURE

The classification architecture consists of two stages, Stage 1 and Stage 2. Stage 1 contains several independent distributor units. Each distributor is attached with a PKM (Private Key Memory), which can be considered as a small cache unit to store the popular search rules. Stage 2 consists of the TCAM chips and a contention resolver (CR) unit attached to each TCAM chip. Number of distributor units (M) and CR units (K) is independent of each other and depend on the required throughput and rule database size respectively. Here K = 2^l, where l = (0, 1, 2, ..., 104).

Figure 1 depicts the classification engine architecture with M = 3 and K = 2. Control signals between Distributor modules and the CR modules are not shown to avoid complexity.

A. Rule Partitioning

If the number of Stage 2 TCAM chips (K) is more than one, the rule database has to be partitioned in to K independent segments. Each segment is then put inside a separate TCAM chip. Therefore some sort of rule partitioning method has to be implemented. Most suitable l bit positions are selected out of 104 positions of a rule which partition the rule set in to 2^l = K partitions of nearly equal size. If the total number of rules is N, a segment is approximately N/K in size. Our partitioning scheme is similar to the partitioning method used in [18].

B. Search Operation

When a search key is inbound, the Distributor Unit first matches it against the Private Key Memory. If a match is found the action associated with the matching rule is forwarded as the result and stage two is not consulted. If no match is found in PKM, the distributor extracts the l bit positions of the search key and decides which TCAM chip has to be consulted. There are K search request signals going from each distributor to one per each CR unit. When a distributor wants to access a certain TCAM, it asserts the relevant search request signal.

The CR unit selects which distributor is to be granted access to the TCAM associated with the CR. There would be a race condition if there are search requests from multiple distributor units. Such situation is called a contention situation. In such a situation the CR selects the distributor with the highest hold priority register value. At the same time the CR unit asserts the Busy signal to the other requesting distributor units. There are M such busy signals, one per each distributor going from a CR. Search key from selected distributor is then passed to the TCAM chip and search operation occurs. Other distributor units assert the search request signal again in the next clock cycles until the search request is fulfilled. While the busy signal is high the requesting distributors increase their hold priority register value which guarantees no starvation. After busy signal is removed and granted access to stage one, the distributor unit resets it’s hold priority register value.

C. Hardware Implementation

Figure 2 shows the operation of the Distributor Unit. The distributor unit contains the hold priority register HP, which is (M – 1) bits in width. When the busy signal is detected, at
HP unit. In [19] a CR design which corrects those flaws has been introduced.

When the stage two search operation is done the HP register is reset back to '0' again.

The Contention Resolver unit has three inputs from each and every Distributor unit $x$, namely the search request signal ($Req_x$), the $(M - 1)$ bit hold priority register $HP_x$ and the 104 bit long search key. At a contention situation the CR unit selects which distributor to be served among the requesting distributors based on hold priority register values of all the requesting distributors. The selector with the highest Hold Priority register value is granted the access to the TCAM and the other distributors are put on to hold by asserting their busy signals which make those distributors to increase their HP values.

We noticed that the Contention Resolver design in [17] fails at certain conditions. i.e. When all the $HP_x$ values are zero. In [19] a CR design which corrects those flaws has been introduced.

$$\overline{A}[M] = 1 \text{ and } \overline{A}[j] = \sum_{x=1}^{M} (HP_x[j] \cdot Req_x), j \in \{1, ..., M - 1\}$$ (1)

$$\overline{H}_x[M] = Req_x \text{ and } \overline{H}_x[j] = HP_x[j], j \in \{1, ..., M - 1\}$$ (2)

$$B_x = \sum_{j=1}^{M} (\overline{A}[j] \oplus \overline{H}_x[j]), x \in \{1, ..., M\}$$ (3)

$$S_x = \left( \prod_{y=1}^{x-1} B_y \right) \cdot B_x, x \in \{1, ..., M\}$$ (4)

In equation (1), $\overline{A}$ is assigned with the highest value of the $HP$ registers among the requesting distributors. Setting the MSB of the $H_x$ to ‘1’ in (2) if there is a request from the $x^{th}$ distributor ensures that busy signal is not set high for non requesting distributors.

Equation (3) simply sets the $B_x$ to ‘1’ if the corresponding distributor has the highest $HP_x$ value. Finally equation (4) is used for the final selection if there are more than one distributors with the highest $HP_x$ value (breaking ties). The next distributor to be served is given by \{S_M, ..., S_1\} in one hot encoded format.

D. Updating the PKM

PKM update occurs only if contentions are detected at the CR. In traditional CPU cache, the cache is updated every time a data word is fetched from the main memory. This is done in order to alleviate the bandwidth bottleneck between the CPU and main memory.

But in this architecture the PKM itself is a small TCAM unit which has the same throughput of a stage 2 TCAM. therefore there is no bandwidth bottleneck between stage 1 and stage 2. The sole purpose of introducing PKM is to minimize the contentions at TCAM units. If PKM update is performed for every stage 2 access it would ultimately decrease the throughput because of the increased number of clock cycles per lookup. Therefore it is sufficient enough to perform a PKM update only if a contention is detected.

When a contention situation occurs, a requesting distributor knows it is time to update the PKM when the hold signal is detected. It reasserts the search request signal until it is granted the permission to access the stage 2 TCAM. After the stage 2 match the distributor updates its PKM by inserting the matching rule extracted from the Stage 2. The next PKM entry to be replaced is decided by least recently used(LRU) algorithm.

E. Rule set pre-processing

1) Rule splitting: Packet classification can be viewed as locating the position of a packet in the 5-dimensional space where rules represent 5-dimensional regions. There could be overlapping rules which intersect with each other. This scenario is shown in Figure 3 for two dimensional classification with three rules R1, R2 and R3 which are in increasing order of priority.

A packet can be viewed as a point in the 5d space. A packet belonging to the intersection area of R1 and R2 matches both rules but R2’s action is performed on the packet since the priority of R2 is higher. Assume that a stage 1 distributor unit adds R1 to its PKM after a contention and R2 is not already added to PKM. Then for that distributor, all the packets belonging to the intersection area of R1 and R2 would have R1 as the match result since stage 2 is not consulted after a stage 1 match. The classification result would be erroneous.

To avoid such undesired situations we introduce rule splitting. Before populating the rules among TCAMs, overlapping rules are split such that only the highest priority rule covers
For each leaf node, if not empty (contains rules), rules inside the leaf are orthogonally projected on the axis of the second dimension(y) and resulting elementary intervals are used to construct a segment tree for dimension two. Each non-empty leaf of the first dimension segment tree points to a unique second dimension segment tree. Leaves of the second dimension tree consist of rules (rules inside the leaf of corresponding first dimension tree) which are within the elementary interval of that leaf. Resulting first dimension tree and the second dimension tree for the second leaf of the first dimension tree for the rule set shown in Figure 3 are shown in Figure 5.

Algorithm for rule splitting for two dimensions is shown in Algorithm 1. create_rule creates a new rule, Delete_rule deletes a rule. Rule_merge_y and Rule_merge_x are functions that merge several adjacent rules in to a much larger axis parallel rule along dimension 2 and dimension 1 respectively by comparing the boundary values of the newly created rules. This merging process reduces the number of new rules introduced. This algorithm can be easily extended for five dimensions. Rules above R means the rules with priority values higher than R.

Algorithm 1 Rule Splitting Algorithm

```
Rule R - \{((R_{xlo}, R_{xhi}), (R_{ylo}, R_{yhi}))\}

go to 1st dimension tree

for all leaves containing R do  " Dimension 1 search
    if Rules above R in leaf = 0 then
        create_rule((N_{xlo}, N_{xhi}), (R_{ylo}, R_{yhi}))
    else
        temp ← rules above R in leaf
        go to dimension 2 sub tree
        for all leaves containing R do  " Dimension 2
            if temp ∩ rules above R in leaf = 0 then
                create_rule((N_{xlo}, N_{xhi}), (N_{ylo}, N_{yhi}))
            end if
        end for
        Rule_merge_y
    end if
end for
Rule_merge_x
Delete_rule(R)
```

This Rule Splitting process increases the caching accuracy. Also, since the resulting rules are disjoint from each other the relative location of a rule inside a TCAM is not relevant. Therefore updating the rules at stage two becomes simple. When a new rule is added or removed, the software component re-runs the rule splitting algorithm and determines which rules to be removed and which are to be added. Next the update process is performed by changing only the required rules. Therefore only a subset of rules are removed and added without causing any large scale boundary moving.

2) Handling Ranges: Even though TCAMs perform well at prefix matching and exact matching they cannot handle ranges effectively. Breaking down ranges to prefixes has been introduced as a remedy [23].
Fig. 5. First dimension tree and second dimension tree for the second leaf node of the first dimension tree

III. PERFORMANCE EVALUATION

The theoretical throughput of the design \( R \) is given by equation (5)

\[
R \leq B\left(\frac{1}{(1 - h')P_x}\right), x \in \{1, ..., K\}
\]

where \( P_x \) is the probability of a search key destined to \( TCAM_x \) after not finding a match in the PKM. \( B \) is the maximum throughput of a single TCAM. \( h' \) is the effective hit ratio of stage one which is a function of \( M, K \) and \( h \) where \( h \) is the hit ratio of individual PKM unit. Even though the chance of having a match at stage 1 is \( h \), the effective hit ratio \( h' \) goes down when distributor units are on hold due to contentions where those distributors have no hit till the contention resolves. For a fixed \( K \), Increasing \( M \) results in increased number of contentions thus reducing the \( h' \) [17].

The maximum throughput achievable by the design is given by equation (6)

\[
R_{max} = B\left(\frac{1}{(1 - h')P_{max}}\right)
\]

Based on Equation (6) one would argue that if the rules are distributed among the TCAMs in such a way that the traffic load of each TCAM is equal, i.e. \( P_{max} = \frac{1}{K} \), the throughput would be maximum. But the probability of stage 2 traffic destined to a certain rule drops as soon as that particular rule is added as a PKM entry. Therefore the estimation of the popularity of the search keys among stage two traffic gets extremely difficult because rule caching rapidly changes the composition of stage 2 traffic.

A. Simulation Results

For the simulation purposes we developed a clock cycle accurate simulation model and used classification rules and traces generated by ClassBench [24]. Number of rules was 10K and three traffic traces were obtained for the same ruleset by varying the locality of reference by varying the values of the Pareto parameters \( a \) and \( b \) of classbench trace generator. This resulted in three traces. Note that the locality of reference is at flow level.

- Trace 1 - High locality \((a=5, b=1)\)
- Trace 2 - Medium locality \((a=5, b=0.1)\)
- Trace 3 - Low locality \((a=5, b=0.01)\)

We simulated the design with varying combinations of \( M \) and number of entries in a PKM (\( n \)) for a fixed value of \( K \). The simulation was run at 100 \( MHZ \). 100 \( MHZ \) is a possible frequency because synthesizing the distributor units and CR units in a Virtex 2Pro FPGA achieved timing closure at 125 \( MHZ \).

Figure 6 depicts the throughput of the design for \( M = (4, 8, 12, 16, 20) \), \( n = (0, 32, 64, 128, 256) \) and \( K = 4 \).

Introduction of the smallest number of PKM entries results in a huge increase in the throughput compared with the situation of no PKM present \((n = 0)\). The throughput increases with \( M \), but the difference of throughput between two consecutive values of \( M \) becomes smaller. This is due to the increased number of contentions with \( M \).

There is only a slight change in the throughput when locality is decreased at flow level. Which means that the effect on locality at rule level is minimum and proves that rule caching is far better than flow caching.

The maximum throughput of a stage 2 TCAM is 100 Million searches per second(Msps) assuming a search operation occurs every clock cycle. Therefore the maximum throughput achievable by a design which utilizes no PKM when \( K = 4 \) is 400 Msps. But in this architecture Where \( M \geq 8 \) the throughput is always more than 400 Msps for all the traces which makes clear that when PKM is present, the design exhibits an excessive speedup than the non PKM approaches [9].

For an example, when \( M = 16 \), \( n = 256 \), \( K = 4 \) and clock rate is 100 MHz this architecture is capable of achieving 630 Msps when least amount of temporal locality is present(Trace 3) which translates in to 201 Gbps for a worst case packet size of 40 bytes.

B. Power Consumption

The power consumption of a TCAM chip is considerably higher than that of a traditional RAM because in a TCAM the search operation is performed in parallel on all the entries. Due to this fact the power consumption of stage 1 can be considered negligible compared to the combined power consumption of stage 2 TCAMs. A TCAM consumes negligible amount of power when it is idle. We define TCAM utilization as the ratio between the search operations performed on stage 2 TCAMs and the total number of search operations performed. Figure 7 shows the TCAM utilization for different values of \( M \) and \( n \).

The TCAM utilization drops from 1 to less than 0.4 when PKM is introduced. In our approach state 2 TCAMs are energized only 40% of the time on average. TCAM utilization directly corresponds with the power consumption [17] which means stage 2 TCAMs consume \((40/k)\)% of the power consumed by a non cache approach with single TCAM.
with a conventional non-cache single TCAM design. A reduction in power consumption in our approach compared to that of the single TCAM is observed.

where all the searches are done on the same TCAM and it’s busy 100% of the time. Therefore, when k = 4, there is a 90% reduction in power consumption in our approach compared to a conventional non-cache single TCAM design.

IV. CONCLUSION

Explosive growth of Internet traffic and rule sets demand high performance packet classification engines. This paper tries to address this issue by introducing a parallel packet classification engine architecture utilizing parallelly operating multiple TCAM chips. The throughput is further improved by exploiting the temporal locality of the Internet traffic by employing a novel rule caching mechanism. The paper also discusses errors that could be introduced due to caching and proposes Rule Splitting as a solution. The resulting design is capable of achieving more than 200 Gbps for worst case packet sizes.

REFERENCES


