Flow-Length Aware Cache Replacement Policy for Packet Processing Cache

Hayato Yamaki
Dept. of Computer and Network Engineering
The University of Electro-Communications
Chofu, Japan

***Abstract*—Recent core routers are required to process packets not only at high throughput but also with low power consumption due to the increase in the network traffic amount. Packet processing cache (PPC) is one of the effective approaches to meet the requirements. PPC enables to process a packet without accessing to a ternary content addressable memory (TCAM) by storing the TCAM lookup results of a flow in a cache. Because the cache miss rate of PPC directly impacts on the packet processing throughput and the power consumption of core routers, it is important for PPC to reduce the number of cache misses. In this study, we focus on characteristics of flows and propose an effective cache replacement policy for PPC. The proposed policy, named Hit Dominance Cache (HDC), divides the cache into two areas and assigns flows to the appropriate area to evict mice flows rapidly and to retain elephant flows preferentially. Simulation results with 15 real network traces show that HDC can reduce the number of cache misses in PPC by up to 29.1% and 12.5% on average when compared to 4-way LRU, conventionally used in PPC. Furthermore, the hardware implementation using Verilog-HDL shows that the hardware costs of HDC is comparable to those of 4-way LRU though HDC performs as if the cache was composed of 8-way set associativity. Finally, we show that HDC can achieve 503 Gbps with 88.8% energy of conventional PPC (20.5% energy of TCAM only architecture).**

***Keywords*—Router; packet processing; cache replacement**

I. INTRODUCTION

Internet traffic has increased year by year due to the popularization of internet applications which generate a large number of packets, such as file sharing, cloud services, and video streaming. Because the traffic concentrates on routers, the processing load of routers has increased and becomes a serious problem. According to The Ministry in Japan [1], [2], it is reported that the total amount of internet traffic and the power consumption of network devices will increase approximately 190 times and 10 times, respectively, in 2025 compared to 2006. The power consumption of routers is no longer negligible because it will account for several percentages of total power consumption in the world [3], [4]. Not only high throughput but also low power consumption is required for routers and especially for core routers, which handle the huge traffic close to the center of the internet.

In a core router, table lookups for packet processing is known as the main cause of both degrading the throughput and consuming the power [5]-[7]. To determine how to process a packet (i.e., where to transmit or how to filter the packet, etc.), routers are required to retrieve tables such as the routing table, the address resolution protocol (ARP) table, the access control list (ACL), and the quality of service (QoS) table. In recent core routers, these tables are stored in a ternary content addressable memory (TCAM), which is a memory specialized in high-speed data search. While the TCAM can obtain a table lookup result with one cycle, it consumes approximately 16 times as large power as a same sized static random access memory (SRAM) [8]. Due to this, it is indicated that the TCAM accounts for 40% of all power consumed in a router [9], [10]. Make matter worse, the lookup performance of the TCAM cannot reach the throughput required for future internet (i.e., more than 400 Gbps) because of the low operation frequency. Thus, improvement of the TCAM lookups is important for future core routers to achieve both high throughput and low power consumption.

As one of the solutions, optimizing the TCAM use is the most popular approach. Nawa et al. proposed a novel searching scheme for the TCAM [9]. They enabled to reduce the dynamic energy of the TCAM lookups by dividing the all TCAM entries into several groups and searching only appropriate group. Gamage et al. proposed a method for high-throughput table lookups with parallelized TCAMs [5]. The proposed method enabled to accelerate the packet processing by assigning packets to the suitable TCAMs. However, it is still difficult to achieve more than 400-Gbps throughput and SRAM-like power. Other approaches are required for further improvement.

Packet processing cache (PPC) is another approach and recently reevaluated because it can improve the table lookup performance without impeding the TCAM-based approaches and thus adopt concurrently with them. PPC includes a cache which retains the TCAM lookup results of packets and reuses them to process following packets. If the TCAM lookup result of a packet is in the cache, PPC can process the packet without accessing the TCAM using the cache. The more PPC can process packets with the cache, the larger PPC can acquire the throughput and the power reduction because of a small number of TCAM accesses. In other words, the performance of PPC depends on the cache hit/miss rate. Thus, to reduce the number of cache misses is an important issue for PPC. In this study, we investigate causes of the cache misses in PPC and propose an effective cache replacement policy for PPC to reduce the number of cache misses.

This work was supported by KAKENHI 18K18022, a research grant from The Mazda Foundation, and a research grant from Sumitomo Foundation.

The contribution of this paper is summarized as follows:

- Major Causes of the cache misses in PPC are suggested. We show that two types of flows make a large number of cache misses.
- Our simulation shows that Hit Dominance Cache (HDC), proposed in this paper, reduces the number of cache misses in PPC by up to 29.1% (12.5% on average) with comparable hardware cost to 4-way Least Recently Used (LRU), typically used in PPC.
- The performance of cache replacement policies to various types of traffic patterns is evaluated. Although the cache access patterns in PPC (i.e., behavior of packets) differ depending on the network structure, previous studies simulated with only a few network traces. This study uses 15 network traces for evaluation.
- This paper is an expansion of the paper [11], which published in the proceedings of Future of Information and Communication Conference (FICC) 2018. It newly reveals more detailed relation between flows and the PPC cache misses. In addition, the performance difference of variable HDC designs and the more detailed hardware costs are also evaluated.

The rest of this paper is organized as follows. We first show the more details of PPC in Section 2 and introduce the related works of reducing the number of cache misses in PPC in Section 3. After that, we investigate major causes of the cache misses in PPC and propose our technique HDC in Section 4. Section 5 evaluates HDC performance from the aspects of the cache miss reduction, the implementation costs, the throughput, and the energy consumption. Finally, we conclude this paper in Section 6.

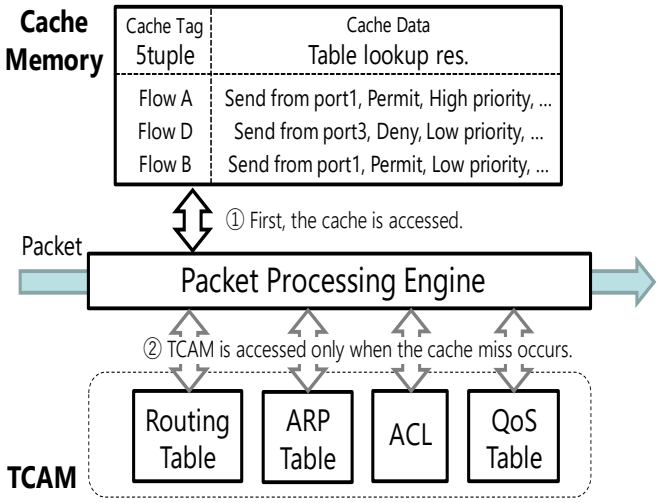

Fig. 1. Outline of packet processing by PPC.

II. Packet Processing Cache

In the routers, the TCAM lookup results depend on several fields in the packet header (e.g., IP addresses) because they are used as a key for retrieval. In particular, the five-tuple (i.e., the source and destination IP addresses, the source and destination port numbers, and the protocol number) are used in most tables in a router. Based on this fact, PPC defines packets which have the same five-tuple as a flow and stores the TCAM lookup results of the first packets of flows to a cache memory. If the TCAM lookup result of a flow is in the cache, PPC can process the subsequent packets of the flow without accessing the TCAM using the cached result and reduce the number of TCAM accesses. Because the packets of the same flow arrive in a router at short intervals [12], [13], routers can benefit from the cache. The outline of the packet processing by PPC is shown in Fig. 1. The latency and the energy consumption of the cache are considerably lower than those of the TCAM, and therefore PPC can process packets at high speed with low energy consumption if the cache hits occur.

The cache is composed of 13-byte flow information as a cache tag and 15+-byte TCAM lookup results as cache data. The cache data include 1 byte as a result of the routing table lookup (output interface number), 12 bytes as a result of the ARP table lookup (MAC address), 1 byte as a result of the ACL (filtering decision), 1 byte as a result of the QoS table (priority value). Moreover, the cache can also store other processing results such as filtering results of NIDS (Network Intrusion Detection System), encapsulation results, and encryption results by expanding the cache data field. PPC can perform many packet processing with one cache access.

In PPC, TCAM accesses are required only when the TCAM lookup results of the corresponding flows are not in the cache. For this reason, the performance of PPC depends on the cache miss rate. We define the throughput and the energy consumption of the table lookups per packet with PPC as T_{PPC} and E_{PPC}, respectively. These variables are calculated as:

$$T_{PPC} = \begin{cases} \frac{1}{l_{Cache}} \cdot 64 \text{ byte} & (l_{Cache} > l_{TCAM} \cdot m) \\ \frac{1}{l_{TCAM} \cdot m} \cdot 64 \text{ byte} & (l_{Cache} < l_{TCAM} \cdot m) \end{cases} \quad (1)$$

$$E_{PPC} = (DE_{Cache} + DE_{TCAM} \cdot n \cdot m) + (SE_{Cache} + SE_{TCAM}). \quad (2)$$

Here, l_{Cache} and l_{TCAM} represent the latencies of the cache and the TCAM, respectively. Likewise, DE_{Cache} and DE_{TCAM} represent the dynamic energy of the cache and the TCAM per access, respectively, while SE_{Cache} and SE_{TCAM} represent the static power of them. Conventionally, the impact of the static power in the memories can be ignored because the dynamic energy dominates the total energy consumed by a memory in a router [14].

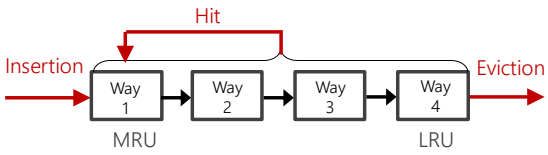

Fig. 2. Outline of LRU entry replacement.

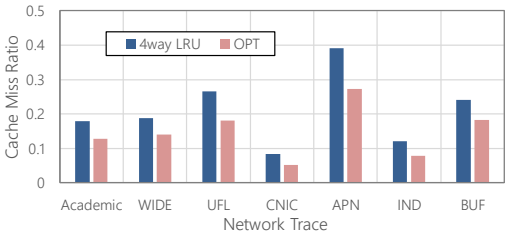

Fig. 3. Comparison of cache miss rates between LRU and OPT.

However, we consider both of energies in this paper because the static power is not negligible in case of a small cache memory. *m* represents the cache miss rate. *n* represents the number of TCAM accesses needed to process a packet. In this paper, we suppose $n = 4$ because popular routers have four types of tables as shown in Fig. 1. 64 bytes referred in (1) represents the shortest packet length. Equation (1) means that the table lookup throughput with PPC is limited by the smaller throughput of the cache or the TCAM. These equations indicate that the throughput and the power consumption with PPC are mainly determined by the cache miss rate due to the large latency and high energy consumption of the TCAM. Thus, to achieve low cache miss rate is important to improve both the throughput and the power consumption with PPC.

To increase the total number of cache entries is the most simple and effective solution to reduce the number of cache misses for many cache systems. However, for PPC, this approach leads to the increase in the cache capacity easily due to the large entry size (28 bytes per entry). As a result, it increases in both the latency and the power consumption of the cache. Unlike processor caches, the gap of the latencies between the cache and the TCAM is small. Thus, it is desirable for PPC to use a small cache memory such as level-1 (L1) caches in processors. In this paper, we aim to reduce the number of cache misses in L1-sized PPC without increasing the cache capacity by improving the use of the cache space.

Conventionally, PPC is designed by a 4-way set associative cache from the aspects of the cache miss rate and the hardware complexity [15]-[17]. It means that each cache line has four entries, and PPC can keep useful entries in the cache by applying a suitable cache replacement policy. The cache replacement policy is used to determine which entry should be replaced when the cache line is full. It is known that the cache replacement policy impacts on the cache miss rate significantly. In PPC, Least Recently Used (LRU), whose concept is to evict the entry hit oldest, is empirically used as the cache replacement policy. As shown in Fig. 2, LRU inserts a new entry into the Most Recently Used (MRU) position and evicts it from the Least Recently Used (LRU) position. In addition, when an entry is referenced, LRU shifts it to the MRU position. LRU performs good in many cache systems because it can utilize the temporal locality of data; however, it is not certain that 4-way LRU is suitable for PPC. Fig. 3 shows the difference of the cache miss rates measured by a simulation with real network traffics. The details of the simulation are described in Section 5. In this simulation, we designed 4-way LRU and optimal page replacement algorithm (OPT) [18] as the cache replacement policy in PPC. OPT is an ideal replacement policy which uses the information of all future arrived data. Thus, the performance of OPT is the best of all cache replacement policies; however, it cannot be implemented for practical use in most cases. Fig. 3 indicates that approximately 30% of all the cache misses in PPC have the opportunities to be reduced by replacing entries more effectively than LRU. In this study, in order to close the gap of the cache miss rates between LRU and OPT, we consider the effective replacement policy.

III. Related Work

Various cache replacement policies have been proposed in previous studies for many cache systems. However, they are not always effective for PPC because of the difference of the cache access patterns. In this section, we introduce a few studies focusing on the reduction in the number of cache misses in PPC and reveal the problems of the proposed methods.

Chang et al. pointed out that PPC cannot prepare a large number of cache entries due to the large tag size and proposed a method to compress the cache tag [19]. They used a 32-bit hash value calculated from the five-tuple as the cache tag instead of the 104-bit flow information. However, adding extra hardware is needed to avoid the conflicts of the hash values. Similarly, Ata et al. compressed the cache tag of PPC by using only three fields of the five-tuple: the source and destination IP addresses and the smaller number of ports [20]. However, it cannot meet demands of recent routers. For example, the QoS table requires the five-tuple to determine the QoS value. Compressing the cache tag sacrifices the information stored in the cache or requires to add extra hardware.

Li et al. discussed the appropriate cache design for PPC [17]. In order to use the cache space efficiently, they focused on three viewpoints: the cache associativity, the cache replacement policy, and the hash function. In [17], the authors concluded that a 4-way set associativity with LRU is the best design from the balance between the implementation costs and the cache hit rate. Additionally, they show that the difference of the hash functions does not impact on the cache hit rate largely. While the authors evaluated three policies (LRU, least frequently used (LFU), and round robin), other policies were not considered.

Kim et al. proposed an effective cache replacement policy for PPC [21]. They indicated that LRU was not suitable for PPC because LRU focused on only temporal locality and cannot utilize activities of networks. The proposed policy classifies the cache entries into two types: a switching entry and a non-switching entry. The switching entry is the entry hit at least once; non-switching entry is the entry never hit. The entry is replaced from the non-switching entries. Furthermore, they proposed two types of cache replacement policies called Weighted Priority LRU Scheme and L2A Cache Scheme. In

Weighted Priority LRU Scheme, the non-switching entries cannot be replaced until a threshold time passes because it is expected that the non-switching entries are referenced again in a short time. In L2A Cache Scheme, the replaced entry is decided by the amount of the timestamp values in last two packets. L2A Cache Scheme can reduce the number of cache misses compared to LRU in case that the cache size is small. However, concrete hardware requirements, such as the number of bits for storing the timestamp to the cache and the way for getting the time, were not referred. The increase in the memory costs becomes a critical problem especially in PPC.

Yamaki et al. considered the methods of denying the cache registration of one-packet flows because they have no opportunities to hit in the cache [22], [23]. They proposed several methods to specify applications which create flows composed of only one packet, such as flows created by domain name system (DNS) or by several types of network attacks, and not to store these flows in the cache. While the proposed methods can reduce the number of cache misses in PPC by approximately 8%, the misses caused by various small factors cannot be improved because the methods handle the misses of only specific applications.

IV. Cache Replacement Policy

In this paper, we focus on the cache replacement policy because it has a potential to reduce a large number of cache misses as shown in Fig. 2 with small hardware modification. In this section, we first analyze the flow behavior in PPC to reveal the cause of the cache misses in PPC. After that, we propose a novel cache replacement policy based on the above analysis.

A. Analysis of Flow Behavior in PPC

Flows composed of a few packets (referred to as mice flows) are one of the main causes of increasing the cache misses in PPC [23]. The mice flows cause cache pollution due to the occupation of entries in spite of a few cache hits. In particular, flows composed of one packet are not needed for the cache because they never hit in the cache. In [23], it was indicated that 99% of all flows are the mice flows composed of less than 10 packets. Moreover, the flows composed of one packet accounts for about half of all flows in networks. Therefore, mice flows occupy most entries in the cache and impact on the cache performance significantly. It is important for PPC to evict the mice flows from the cache rapidly.

Unlike the mice flows, there are flows composed of a large number of packets (referred to as elephant flows), such as video flows. These elephant flows are composed of more than 1,000 packets. Although the number of elephant flows are few, they have great impact on the cache miss rate because of a large number of cache references. These trends that a large number of mice flows and a few number of elephant flows account for most packets in networks is known as the elephant and mice phenomenon [24]. We analyze the behavior of the elephant flows in the cache and show the examples in Fig. 4. This graph shows logs of the cache hits and misses in elephant flows. Contrary to expectation, many packets in an elephant flow make the cache misses. It means that entries of the elephant flows are replaced repeatedly though they are referenced many times.

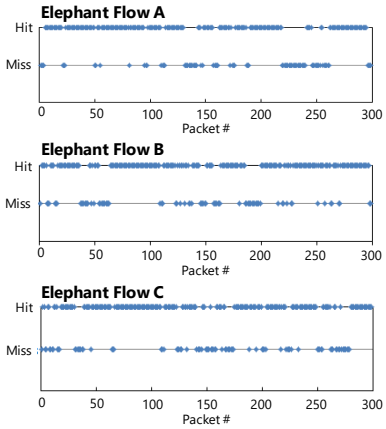

Fig. 4. Logs of the cache hits and misses in elephant flows.

We also investigate the impact of the mice flows and elephant flows on the number of cache misses. Fig. 5 shows the total amount of packets, compulsory misses, and other misses in the flows composed of the specific number of packets. Compulsory misses shown in the figure are misses caused by first packets of flows, and thus it is difficult for PPC to prevent them. In contrast, other misses shown in the figure can be reduced by retaining entries of the corresponding flows appropriately. This figure indicates that a large number of misses, especially compulsory misses, are caused by mice flows, and it pollutes the cache. In addition, we show the ratios of each cache miss to all packets in Fig. 6. As shown in Fig. 4, it also indicates that packets in elephant flows cause the cache misses at a higher rate than we expect. Thus, it has better for PPC to prevent the cache pollution caused by mice flows and to prioritize elephant flows.

B. Hit Dominance Cache

Based on above considerations, we propose Hit Dominance Cache (HDC) to prevent the cache pollution caused by mice flows and retain elephant flows preferentially. The main concept of HDC is to provide difference cache priorities to elephant flows and mice flows by assigning difference cache areas to them. Because it is difficult to identify whether a packet is belonging to an elephant flow or a mice flow accurately when the packet comes, HDC judges it from the number of cache hits. HDC prioritizes entries which hit many times as elephantish flows.

Fig. 7 shows the outline of HDC entry replacement. In HDC, the cache area is divided into two areas: the hit area and the primary area. Entries are inserted in and evicted from the LRU position of the primary area. Thereby, new entries can be evicted by one replacement at the shortest. It enables to evict the mice flows rapidly. When an entry in each area is referenced, the entry is shifted to the MRU position in each area. In addition, one notable behavior of HDC is that the entry in the primary area is swapped for the LRU position entry in the hit area if the entry is referenced threshold times (we define it as the HDC threshold). It enables to retain elephant flows preferentially because mice flows disturb entries in only the primary area. In Fig. 7, HDC is depicted as the combination of 4-way hit area and 4-way primary area; however, each area can be designed variably.

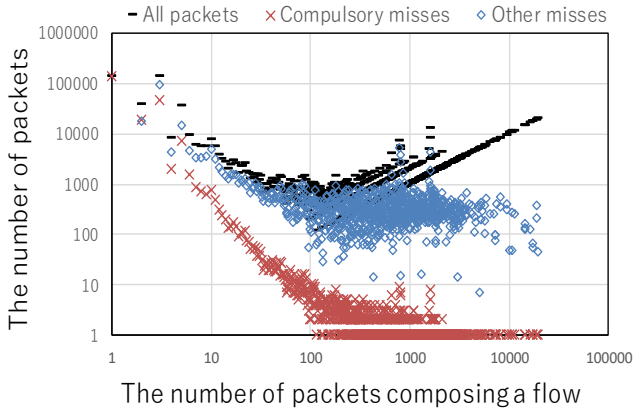

Fig. 5. Total amount of all packets, compulsory misses, and other misses in the flows composed of the specific number of packets.

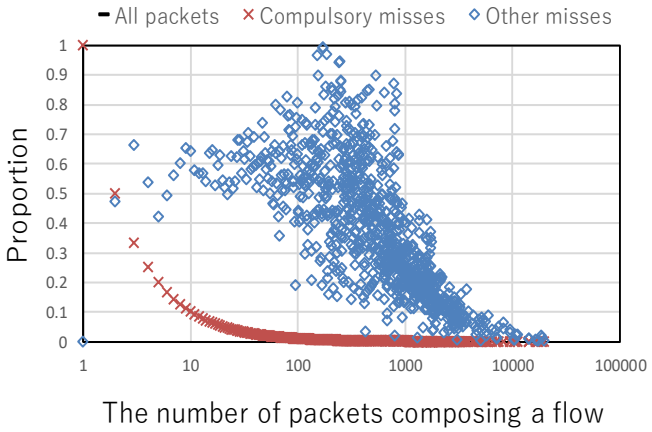

Fig. 6. Average ratios of two types of cache misses to all packets in the corresponding flow. For example, this graph shows all packets in the flows composed of one packet cause the compulsory misses.

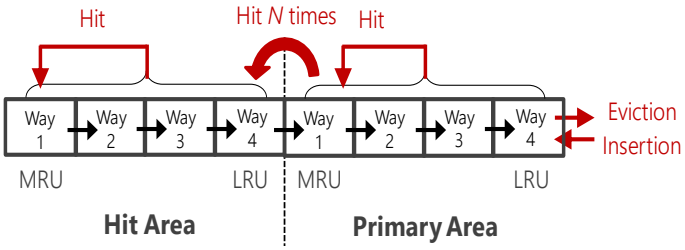

Fig. 7. Outline of HDC entry replacement.

V. Evaluation

This section evaluates the usefulness of HDC. In this paper, we simulated the packet processing with PPC including HDC using a software PPC simulator and implemented HDC using a hardware description language to evaluate HDC from following three aspects:

- Cache miss reduction
- Implementation costs
- Overall throughput and energy consumption.

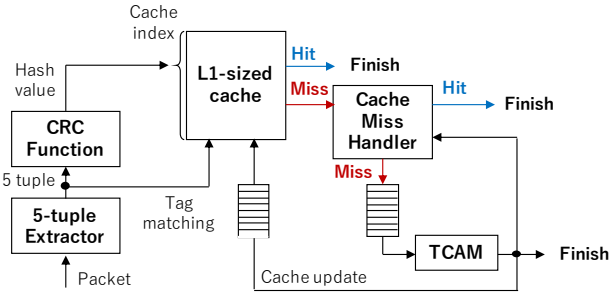

Fig. 8. Block diagram of PPC simulator.

A. Cache Miss Reduction

To measure the cache miss rate of PPC, we prepared a cycle-based PPC simulator written in C++. The block diagram of the PPC simulator is shown in Fig. 8. The simulator models the PPC behavior such as reading packets, extracting the flow information, calculating the cache index, referring to and updating the cache, and referring to the TCAM. First, pcap-format file is read, and a packet is extracted based on the timestamp value. After that, the cache index is calculated from the five-tuple of the packet using CRC hash function module. Next, the cache is referenced using the cache index. If the packet hits in the cache, the PPC simulator finishes processing the packet and extracts the next packet. On the other hand, if the packet misses in the cache, the packet is forwarded to Cache Miss Handler (CMH). CMH is a module to prevent the cache misses caused by the time-lag between the cache reference and the cache update. Because it takes a little time to insert a new entry in the cache, cache misses may be occurred if subsequent packets of the same flow come before updating the cache. CMH manages the flows just processed in the TCAM and stores the subsequent packets of the same flows until the corresponding entry is prepared in the cache. More details of CMH is described in [25]. If the packet misses in CMH, the packet is forwarded to the TCAM module. After the TCAM access latency, the PPC simulator finishes processing the packet and updates the cache.

Table I shows the parameters of the PPC simulator. The cache was estimated as an L1-sized cache. The latencies of the cache and the TCAM were set to 0.5 ns and 5 ns, respectively. Note that we assumed the sizes of CMH and queues shown in Fig. 8 were enough large, and the simulator can process packets without any packet losses. Besides, we used 15 types of network traces shown in Table II as workloads to reveal the performance of the cache replacement policies without depending on the network traffic patterns. The network traces were acquired from RIPE Network Coordination Centre [26] and Widely Integrated Distributed Environment (WIDE) [27]. Furthermore, an academic trace acquired from a core network in Japan was used as a high-bandwidth workload.

First, the suitable design for each area of HDC was evaluated. As described in Section 4, HDC can variably design the number of associativity sets in each area. We implemented 2-way hit area 4-way primary area HDC (2-4 HDC), 4-way hit area 2-way primary area HDC (4-2 HDC), and 4-way hit area 4-way primary area HDC (4-4 HDC) and compared the cache miss rates of them. In this simulation, each HDC design had

the same total number of entries and set the HDC threshold = 8. Note that 8-way design was not evaluated in this simulation because it cannot be implemented for practical use due to the high implementation costs as described later. Fig. 9 shows the cache miss rates of 4-way LRU and each HDC design. In this figure, we showed the results of only three networks (i.e., TXG, IPLS, and UFL) because the trends of all the results are mostly the same. Fig. 9 indicates that 4-4 HDC is the best design to achieve the low cache miss rate. 4-4 HDC can reduce the number of cache misses by up to 20.8% and 16.7%, when compared to 2-4 HDC and 4-2 HDC, respectively. We consider it is because the total amounts of the mice flow packets and the elephant flow packets are almost the same as shown in Fig. 5, and one to one design is fitting to split these flows symmetrically. From this result, we adopt 4-4 HDC design hereafter.

Next, the impact of the HDC threshold was evaluated. Figure 10 shows the difference of the cache miss rates among 4-way LRU and HDCs with various HDC thresholds. We set the HDC threshold to 1, 2, 4, 8, and 16 and represented them from HDC 1 to HDC 16 in the figure. As well as Fig. 9, we showed the results of only three networks because of the similar trend. This result indicates that it is the best to set the HDC threshold to eight.

TABLE I. PARAMETERS OF PPC SIMULATOR

Parameter	Value
Total cache entries	1,024 entries
Latency of the cache	0.5 ns
Latency of the TCAM	5 ns

TABLE II. DETAILS OF NETWORK TRACES

Trace	Captured date	Average # of packets [pps (packets per second)]
IND[26]	2003/1/6	15,540
BUF [26]	2003/1/18	8,380
TXG [26]	2004/3/26	12,475
APN [26]	2004/3/26	20,330
IPLS3 [26]	2004/6/1	116,778
BWY [26]	2004/10/7	17,922
COS [26]	2005/1/8	8,051
CNIC [26]	2005/3/17	28,440
MRA [26]	2005/3/21	41,372
UFL [26]	2005/3/21	50,769
FRG [26]	2006/1/10	32,955
PSC [26]	2006/2/20	26,912
PUR [26]	2006/2/20	42,515
Academic	2010/6/17	371,013
WIDE [27]	2017/4/12	58,776

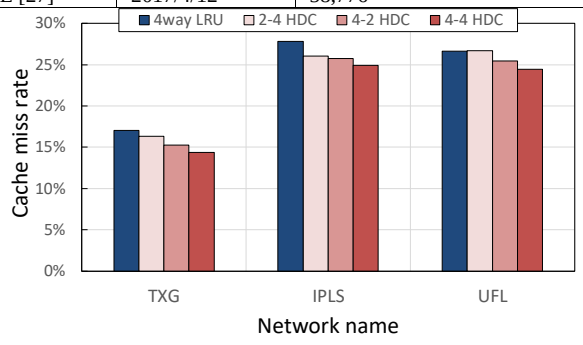

Fig. 9. Cache miss rates of various HDC designs.

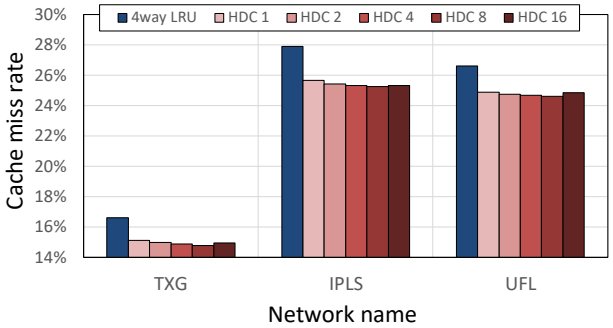

Fig. 10. Cache miss rates of various HDC thresholds.

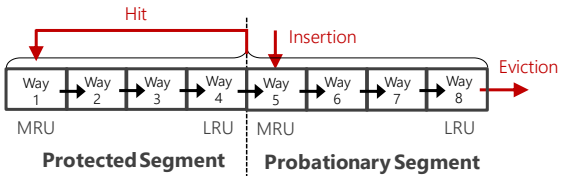

Fig. 11. Outline of SLRU entry replacement.

Finally, the cache miss reduction of HDC was evaluated. In this evaluation, 4-way LRU, 8-way LRU, OPT, and segmented LRU (SLRU) [28] were used as the cache replacement policies for comparison. SLRU is a cache replacement policy which resembles HDC and divides the cache area into two areas: the probationary segment and the protected segment. Fig. 11 shows the outline of SLRU entry replacement. In SLRU, a new entry is inserted into the MRU position of the probationary segment and evicted from the LRU position of the probationary segment. When an entry is referenced, the entry is set on the MRU position of the protected segment. Although SLRU divides the cache into two areas, it is the same as 8-way LRU whose inserted position of a new entry is changed to the middle of the 8-way entries.

Fig. 12 and 13 show the cache miss rates of various LRUs, HDC, and OPT with 15 types of network traces and the improvement ratios of them to 4-way LRU. The results showed that HDC performed the best in nine networks. HDC can reduce the cache misses by up to 29.1% (12.5% on average) compared to 4-way LRU, while SLRU can reduce the cache misses by up to 20.2% (11.1% on average) compared to 4-way LRU. SLRU performed better than HDC in six network traces; however, SLRU is not suitable for practical use because of the high implementation cost like 8-way LRU, as mentioned later. By contrast, 8-way LRU cannot achieve major improvement to 4-way LRU though 8-way LRU can achieve considerably lower cache miss rate than 4-way LRU in many other cache systems. It is because 8-way LRU cannot evict mice flows rapidly, and thus the cache entries are polluted by them. In this simulation, HDC provided up to 84.6% of the OPT performance; however, the average performance of HDC is 44.5% of the OPT performance. It means that there is still room for improvement.

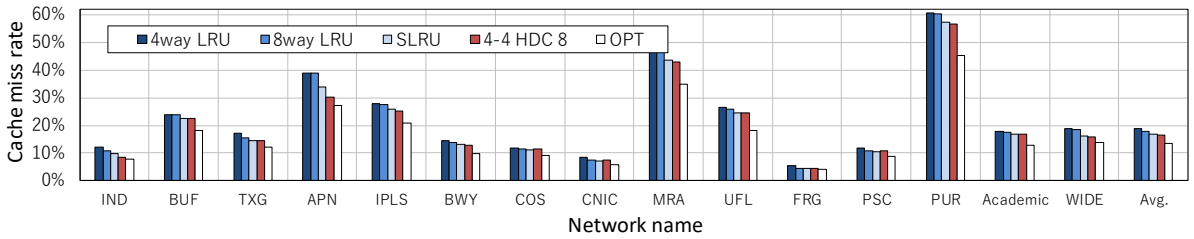

Fig. 12. Cache miss rates of various LRUs, HDC, and OPT in 15 types of network traces.

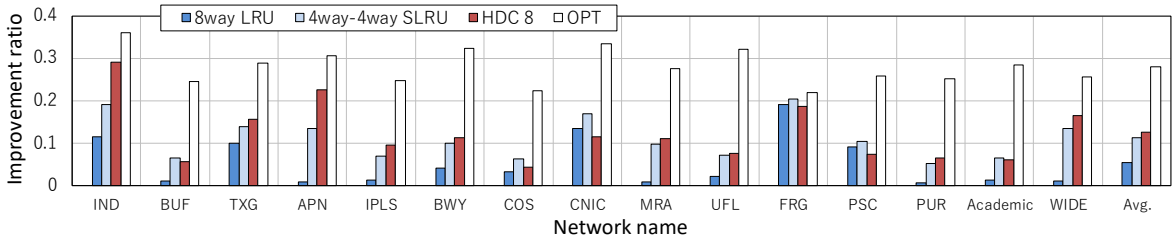

Fig. 13. Improvement ratios of the cache misses of various LRUs, HDC, and OPT to 4-way LRU.

B. *Implementation Costs*

When we consider a cache replacement policy, not only the cache miss reduction but also the implementation costs are an important issue because the cache replacement policy requires a large circuit area which cannot be ignored in some cases. In general, two modules are required to implement a cache replacement policy. One is a memory to store the replacement priorities of entries per cache line, and the other is a module to update these priorities when a cache hit or a new entry insertion occurs.

In the case of 4-way LRU, 5 bits are required per cache line to handle 24 possible combinations of the replacement priorities (because each entry is ranked from 1 to 4). On the other hand, 8-way LRU needs 16 bits per cache line to handle 40,320 possible combinations of those (because each entry is ranked from 1 to 8). Furthermore, the module to update the replacement priorities in the 8-way LRU is significant larger than those in the 4-way LRU because there are 40,320 * 7 possible patterns of the replacement priority transitions in 8-way LRU though 4-way LRU needs 24 * 3 possible patterns of the replacement priority transitions. As a result, it is not realistic to implement 8-way LRU on hardware. Similarly, SLRU cannot be implemented with realistic hardware costs.

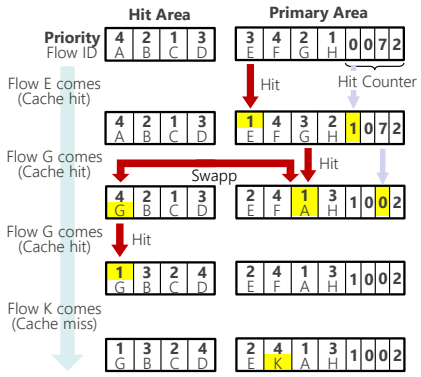

Fig. 14. An example of the method to manage the replacement priorities in 4-4 HDC 8.

Against above consideration, although HDC performs as if the cache was composed of 8-way set associativity, it enables to manage the replacement priorities in the same way as the 4-way LRU. Fig. 14 shows an example of how to manage the replacement priorities in 4-4 HDC. In order to swap the entries between the primary area and the hit area, hit counters are needed for each entry in the primary area to count the number of references. In the case of HDC 8, 3 bit * (1,024 entries / 2), namely 1.5K bits are needed as the hit counter. When a hit counter overflows, the corresponding entry is swapped for the LRU position entry in the hit area of the same cache line. At this time, the replacement priorities are updated in only the primary area because the replacement priority of the swapped entry in the hit area is not changed and keeps LRU position. Furthermore, unlike LRU, HDC does not need to update the replacement priorities when a new entry is inserted into the primary area.

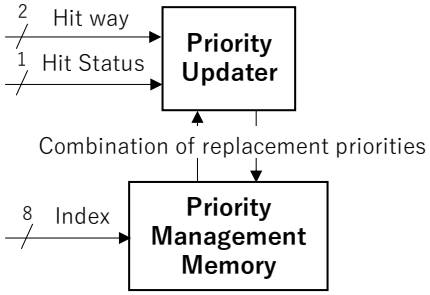

Fig. 15. Hardware architecture of general cache replacement policies.

TABLE III. SIMULATION ENVIRONMENT

Item	Tool Name
Hardware description language	Verilog-HDL
Logic simulation	Cadence NC-Verilog LDV5.7
ASIC synthesis	Synopsys Design Compiler X-2005.09
Libraries for ASIC synthesis	Free PDK OSU Library (45nm) [29]

TABLE IV. SYNTHESIS RESULTS OF 4-WAY LRU AND HDC.

	4-way LRU	HDC
Combinational circuit area	110.66 μm^2	80.332 μm^2
Memory requirement	1,280 bit	2,816 bit

TABLE V. LATENCIES AND ENERGIES OF CACHE AND TCAM

	Cache	TCAM
Latency	0.598 ns	5 ns
Dynamic energy	0.0539 nJ / access	30 nJ / access
Static power	0.0159 J/s	0.85 J/s

TABLE VI. THROUGHPUT AND ENERGY OF TABLE LOOKUPS

	Only TCAM	PPC with 4-way LRU	PPC with HDC
Throughput	102 Gbps	445 Gbps	503 Gbps
Energy per packet	124 nJ	28.6 nJ	25.4 nJ

In order to evaluate the implementation costs of HDC, the hardware implementation of HDC and LRU was simulated using a hardware description language. Fig. 15 and Table III show the hardware architecture of the cache replacement policies and the tools used for the evaluation, respectively. The priority updater shown in Fig. 14 receives the cache-hit status and the hit way number from the cache and the combination of the replacement priorities from the priority management memory when a cache hit occurs. After updating the replacement priorities, the priority updater writes them back to the priority management memory. The index of the priority management memory is the same as that of the cache in PPC.

Table IV shows the ASIC synthesis results of 4-way LRU and HDC. Note that we did not implement SLRU and 8-way LRU in this simulation because the hardware costs of them were obviously oversize for practical use. Table IV indicates that HDC can be implemented with 72.6% of the circuit area of 4-way LRU. It is because HDC does not need to update the replacement priorities when a new entry is inserted. On the other hand, the priority management memory size of HDC is 2.2 times as large as that of 4-way LRU. However, this increase is negligible because the priority management memory is small when compared to the cache. As a result, the implementation costs of HDC are comparable to 4-way LRU.

C. Overall Throughput and Energy Consumption

We finally estimated the overall throughput and the energy consumption of the table lookups. The throughput and the energy consumption with PPC can be calculated from (1) and (2), introduced in Section 2. Here, the latency, the dynamic energy, and the static power of the cache were estimated using a cache model CACTI 6.5 [30]; those of the TCAM were estimated using a TCAM power and timing model [8] (1 Mbit TCAM with 70 nm process was assumed). We show each estimated value in Table V. It indicates that both energies of the cache are remarkably smaller than those of the TCAM, and thus the cache miss of PPC significantly impacts on the table lookup performance.

Taking these estimations, we calculated the throughput and the energy of the table lookups and showed the result in Table VI. It was shown that PPC with HDC can achieve 503 Gbps throughput with 25.4 nJ energy per packet. It is 4.93 times high throughput and 20.4% energy, when compared to the conventional TCAM only architecture. In addition, when compared to PPC with 4-way LRU, PPC with HDC can improve the throughput and the energy by 13.0% and 11.2%, respectively.

VI. CONCLUSION

In this paper, an efficient cache replacement policy named Hit Dominance Cache (HDC) was proposed to reduce the number of cache misses in Packet Processing Cache (PPC) without increasing the cache size. Conventionally, Least Recently Used (LRU) is used as the cache replacement policy of PPC because it is known that LRU performs good in many cache systems. However, from the difference of the cache access patterns, LRU is not suitable for PPC. LRU cannot evict mice flows, which account for most flows in networks and make few hits in the cache, rapidly. As a result, the cache entries are polluted by the mice flows.

HDC divides the cache into two areas and assigns flows to the appropriate area depending on the number of references in the cache. HDC can evict the mice flows rapidly by inserting a new entry into the least recently used position and retain the elephant flows preferentially by shifting the entry hit many times to another area. The simulation result with 15 real network traces showed that HDC (4-way hit area, 4-way primary area, and HDC threshold = 8) can reduce the cache misses by up to 29.1% (12.5% on average) compared to 4-way LRU. Furthermore, the hardware implementation using Verilog-HDL showed that the hardware costs of HDC are comparable to those of 4-way LRU. Finally, we showed that PPC with HDC can achieve approximately 500 Gbps with 88.8% energy of conventional PPC (20.5% energy of TCAM only architecture).

REFERENCES

[1] The Ministry, “Tabulation and Estimation of Internet Traffic in Japan,” 2016, Available: http://www.soumu.go.jp/main_content/000462459.pdf. [Accessed May. 6, 2018]

[2] METI, “Green IT Initiative in Japan,” Available: http://www.meti.go.jp/english/policy/GreenITInitiativeInJapan.pdf.[Accessed May. 6, 2018]

[3] J. Fan, C. Hu, K. He, J. Jiang, and B. Liuy, “Reducing power of traffic manager in routers via dynamic on/off-chip scheduling,” 2012 Proc. IEEE INFOCOM, Orlando, FL, 2012, pp. 1925-1933.

[4] X. Zheng, X. Wang, “Comparative study of power consumption of a NetFPGA-based forwarding node in publish–subscribe Internet routing,” Computer Communications, vol. 44, 2014, pp. 36-43.

[5] S. Gamage and A. Pasqual, “High performance parallel packet Classification architecture with Popular Rule Caching,” 2012 18th IEEE Int’l. Conf. on Networks (ICON), Singapore, 2012, pp. 52-57.

[6] B. Talbot, T. Sherwood, and B. Lin, “IP caching for terabit speed routers,” Global Telecommunications Conference (GLOBECOM '99), Brazil, vol.2, 1999, pp.1565-1569.

[7] N. B. Guinde, R. Rojas-Cessa, and S. G. Ziavras, “Packet classification using rule caching,” 2013 Fourth International Conference on Information, Intelligence, Systems and Applications (IISA 2013), Piraeus, 2013, pp.1-6.

[8] B. Agrawal and T. Sherwood, "Ternary CAM Power and Delay Model: Extensions and Uses," in IEEE Trans. on Very Large Scale Integration (VLSI) Systems, vol. 16, no. 5, 2008, pp. 554-564.

[9] M. Nawa et al., "Energy-efficient high-speed search engine using a multi-dimensional TCAM architecture with parallel pipelined subdivided structure," 2016 13th IEEE Annual Consumer

Communications & Networking Conference (CCNC), Las Vegas, NV, 2016, pp. 309-314.

[10] Hewlett-Packard Development Company, “Energy Efficient Networking - Business white paper,” 2011, Available: http://h17007.www1.hp.com/docs/mark/4AA3-3866ENW.pdf. [Accessed May. 6, 2018]

[11] H. Yamaki, “Flow Characteristic-Aware Cache Replacement Policy for Packet Processing Cache,” In Proc. of Future of Information and Communication Conference (FICC 2018), Singapore, 2018, pp.1-8.

[12] G. S. Shenoy, J. Tubella, A. Gonzalez, “Exploiting temporal locality in network traffic using commodity multi-cores,” 2012 IEEE International Symposium on Performance Analysis of Systems and Software (ISPASS), 2012, pp.110-111.

[13] C. Girish and R. Govindarajan, “Improving performance of digest caches in network processors,” In Proc. of the 15th Int’l. Conf. on High performance computing (HiPC'08), India, 2008, pp.6-17.

[14] B. Agrawal and T. Sherwood, "Modeling TCAM power for next generation network devices," 2006 IEEE International Symposium on Performance Analysis of Systems and Software, 2006, pp. 120-129.

[15] C. Kim, M. Caesar, A. Gerber, and J. Rexford, “Revisiting Route Caching: The World Should Be Flat,” In Proc. of the 10th International Conference on Passive and Active Network Measurement (PAM '09), Berlin, 2009, pp.3-12.

[16] K. Y. Ho and Y. C. Chen, “Performance evaluation of ipv6 packet classfication with caching,” 2008 Third Int’l Conf. on Communications and Networking in China, Hangzhou, 2008, pp. 669-673.

[17] K. Li, F. Chang, D. Berger, F. Wu-chang, “Architectures for packet classification caching,” The 11th IEEE International Conference on Networks (ICON2003), Sydney, 2003, pp. 111-117.

[18] L. A. Belady, “A study of replacement algorithms for a virtual-storage computer.” IBM Syst. J. vol. 5, no. 2, 1966, pp. 78-101.

[19] F. Chang, W. C. Feng, and K. Li, “Efficient Packet Classification with Digest Caches,” Proc. Third Workshop Network Processors and Applications (NP-3), 2005.

[20] S. Ata, M. Murata, and H. Miyahara, “Efficient cache structures of IP routers to provide policy-based services,” IEEE Int’l. Conf. on Communications (ICC 2001), Helsinki, vol.5, 2001, pp. 1561-1565.

[21] N. Kim, S. Jean, J. Kim, and H. Yoon, ”Cache replacement schemes for data-driven label switching networks,” 2001 IEEE Workshop on High Performance Switching and Routing, Dallas, TX, 2001, pp. 223-227.

[22] H. Yamaki and H. Nishi, “An Improved Cache Mechanism for a Cache-based Network Processor,” In Proc. of the Int’l. Conf. on Parallel and Distributed Processing Techniques and Applications (PDPTA '12), Las Vegas, NV, 2012, pp. 1-7.

[23] H. Yamaki and H. Nishi, “Line Replacement Algorithm for L1-scale Packet Processing Cache,” In Adjunct Proc. of the 13th Int’l. Conf. on Mobile and Ubiquitous Systems: Computing Networking and Services (MOBIQUITOUS 2016), Hiroshima, Japan, 2016, pp. 12-17.

[24] T. Mori, M. Uchida, R. Kawahara, J. Pan, and S. Goto, “Identifying elephant flows through periodically sampled packets,” In Proceedings of the 4th ACM SIGCOMM conference on Internet measurement (IMC '04). ACM, New York, USA, 2004, pp.115-120.

[25] M. Okuno and H. Nishi, “Network-Processor Acceleration-Architecture Using Header-Learning Cache and Cache-Miss Handler,” The 8th World Multi-Conference on Systemics, Cybernetics and Informatics (SCI2004), 2004, pp. 108-113.

[26] RIPE Network Coordination Centre, “Réseaux IP Européens Network Coordination Centre RIPE NCC,” Available: http://www.ripe.net/. [Accessed May. 6, 2018]

[27] WIDE MAWI WorkingGroup,“MAWI Working Group Traffic Archive” Available: http://mawi.wide.ad.jp/mawi/. [Accessed May. 6, 2018]

[28] R. Karedla, J. S. Love, B. G. Wherry, “Caching strategies to improve disk system performance,” in Computer, vol. 27, no. 3, 1994, pp. 38-46.

[29] North Carolina State University, “FreePDK45:Contents,” Available: http://www.eda.ncsu.edu/wiki/FreePDK45:Contents. [Accessed May. 6, 2018]

[30] N. Muralimanohar et al., “Optimizing NUCA organizations and wiring alternatives for large caches with CACTI 6.0,” In Proc. of the 40th Annual IEEE/ACM Int’l. Symposium on Microarchitecture (MICRO 40), Chicago, USA, 2007, pp.3-14.