TSINGHUA SCIENCE AND TECHNOLOGY
ISSN 1007-0214 14/15 pp432-439
Volume 16, Number 4, August 2011

High-Performance Packet Classification on Multi-Core Network Processing Platforms*

QI Yaxuan (亓亚烜)[1,2], XUE Yibo (薛一波)[2,3], LI Jun (李 军)[2,3,**]

1. Department of Automation, Tsinghua University, Beijing 100084, China;
2. Research Institute of Information Technology, Tsinghua University, Beijing 100084, China;
3. Tsinghua National Laboratory for Information Science and Technology, Beijing 100084, China

Abstract: Packet classification is crucial to the implementation of advanced network services that require the capability to distinguish traffic in different flows, such as access control in firewalls and protocol analysis in intrusion detection systems. This paper proposes a novel packet classification algorithm optimized for multi-core network processors. The proposed algorithm, AggreCuts, has an explicit worst-case search time with modest memory usage. The data structure of AggreCuts is flexible and well-adapted to different types of multi-core platforms. The algorithm on both Intel IXP2850 32-bit and Cavium OCTEON3860 64-bit multi-core platforms was implemented to evaluate the performance of AggreCuts. The experimental results show that AggreCuts outperforms the best-known existing algorithm in terms of memory usage and classification speed.

Key words: packet classification; multi-core; performance evaluation

Introduction

Keeping network operation and information exchange efficient and secure is highly desired. The resulting network services, such as policy-based routing, service differentiation, access control, and load balancing, require the discrimination of packets based on the multiple fields of packet headers. This process is called packet classification. To reach multi-Gbps packet classification speed, there are currently two types of solutions: software-based solutions on general-purpose processors and hardware-based solutions on ASIC/FPGA or Ternary CAMs. However, existing solutions have inherent limitations.

- **Software-based packet classification** Software-based algorithmic solutions embrace the practice of leveraging the statistical structure of classification rule sets to improve average performance. However, due to the bottleneck of computation capacity and memory hierarchy of general-purpose CPUs, software solutions implemented on this platform cannot meet the line rate packet classification requirement of high-end products[1].
- **Hardware-based packet classification** Hardware-based solutions trade programmability for processing speed, and they achieve extremely high packet classification speeds by using proprietary hardware. However, the use of application-specific hardware, such as Ternary CAMs, requires too much power and board area to support a large number of rules. Therefore, hardware-based solutions usually mean higher production costs and a longer time-to-market[2].

Thus, the challenge of combining intelligent software algorithms and flexible hardware platforms to

Received: 2011-01-05; revised: 2011-06-07
* Supported by the National High-Tech Research and Development (863) Program of China (No. 2007AA01Z468)
** To whom correspondence should be addressed.
E-mail: junl@tsinghua.edu.cn; Tel: 86-10-62796400

minimize the unfavorable characteristics of existing solutions is motivating current research efforts. As an emerging class of programmable processors highly optimized for fast packet processing operations, multi-core network processors deliver hardware-level performance to software-programmable systems[3-5]. In this paper, we propose a packet classification algorithm optimized for multi-core network processing platforms to achieve near line rate packet classification performance. The main contributions of this paper are as follows.

• **An efficient packet classification algorithm** The proposed AggreCuts algorithm has explicit worst-case classification speed with modest memory usage[6]. We developed a controllable space partition strategy to limit the worst-case memory access times and a bitmap aggregation technique to reduce memory storage. Compared to the best-known existing algorithm, AggreCuts requires 85% less memory access time and uses 90% less memory storage.

• **System-level implementation and evaluation** The AggreCuts algorithm was implemented on two different state-of-the-art multi-core network processing platforms. In our evaluation on real systems, AggreCuts outperformed the best-known existing algorithm with near line rate throughput on both Intel IXP2850[3] and Cavium OCTEON3860 multi-core platforms[4].

1 Background

Generic packet classification classifies a packet according to the multiple fields of its header. Based on certain specifications on the F fields of the packet header, each rule specifies a flow to which a packet belongs. Each rule has F components, and the i-th component of a rule R, referred to as $R[i]$, is a prefix or range match expression on the i-th field of the packet header. A packet P is said to match a particular rule R if $\forall i$, the i-th field of the header of P, satisfies the expression $R[i]$. If a packet P matches multiple rules, the matching rule with the highest priority is returned[6].

Packet classification can be viewed as a point location problem in a multi-dimensional search space[7]. It has been proven that the best bounds for point location in N non-overlapping F-dimensional hyper-rectangles are $\Theta(N)$ storage with $\Theta(\log^{F-1} N)$ search time or $\Theta(N^F)$ storage with $\Theta(\log N)$ search time[8]. Although the theoretical bounds make it impossible to design a single algorithm that performs well for all cases, real-life rule sets have inherent characteristics that can be exploited to reduce the complexity in both search time and storage space. Existing algorithmic solutions for packet classification can be categorized based on two classification strategies[6]:

• **Field-independent search** RFC[9] and HSM[10] perform independent parallel searches on indexed tables; the results of the table searches are combined in multiple phases to generate the final classification result. All the entries of a lookup table are stored consecutively in memory. The indices of a table are obtained by space mapping and each entry corresponds to a particular sub-space and stores the search result at the current stage. Algorithms using parallel search are very fast in term of classification speed, but they may require exponentially large memories to store the cross-producting tables.

• **Field-dependent search** HiCuts[11] and HyperCuts[12] are examples of algorithms that employ field-dependent searches, i.e., the search results obtained along the fields that have already been searched influence the way in which subsequent fields are searched. Field-dependent algorithms are often based on a decision tree data structure with linear searches at leaf-nodes, and they are commonly considered to be more efficient in terms of memory use compared to field-independent search algorithms. However, these algorithms cannot guarantee explicit worst-case search time due to the uncontrollable decision tree depth and the number of linear searches, and thus cannot provide a stable worst-case classification speed for different rule sets.

In practice, it is difficult for existing network processing platforms to satisfy the excessive memory requirement of field-independent algorithms for large rule sets. In contrast, algorithms that use field-dependent searches are more flexible with regard to optimization for network processor implementation. Accordingly, the presented AggreCuts algorithm is based on field-dependent search. Unlike existing solutions, AggreCuts uses a controllable space partition strategy to limit the worst-case memory access times and employs aggregated bitmaps to reduce memory usage. Therefore, AggreCuts achieves both a deterministic worst-case bound for search speed and small memory usage.

2 AggreCuts Algorithm

Field-independent search algorithms on multi-core network processing platforms cannot achieve deterministic worst-case classification speed with modest memory usage[13]. AggreCuts improves on existing algorithms by using the following strategies:

(1) Fixing the number of cuttings at internal nodes If the number of cuttings is fixed to 2^w (w is a constant referred to as stride), the current search space is then always segmented into 2^w sub-spaces at each internal node. This guarantees a worst-case bound of $\Theta(W/w)$, where W is the bit width of the packet header.

(2) Aggregating consecutive sub-spaces Because consecutive sub-spaces are very likely to have the same sub-set of rules, we can compress these sub-spaces and hence reduce the number of next-node pointers.

(3) Eliminating the next-node pointer array Because the number of next-node pointers has been reduced by space aggregation, and because the size of a next-node pointer is comparable to the size of a child node, we can eliminate the pointer array by replicating the child nodes.

Consider the common 5-tuple packet classification problem, where W=104. If w is fixed to 8, the worst-case depth of the decision tree is no greater than 104/8=13. Although w=8 means that at each internal node there are 256 next-node pointers, after space aggregation the number of sub-spaces can be significantly reduced (less than 10 in most cases). So we can eliminate the 256 next-node pointers by directly linking the current node with a small number of child nodes stored in continuous memory space. Figures 1 and 2 are examples of the HiCuts algorithm and the proposed AggreCuts algorithm, respectively.

To effectively reduce the memory usage, the bitmap technique is used to aggregate consecutive sub-spaces. After each cutting, the current search space is partitioned into 2^w sub-spaces. Consecutive sub-spaces containing the same rules are then aggregated to M ($M \leqslant 2^w$) aggregated sub-spaces. This space aggregation is represented by a 2^w-bit space aggregation bitmap (SAB): (1) set the first bit as '1'; (2) if the next sub-space is merged into the previous one, set it as '0'; (3) else, set the next bit as '1'. After setting the SAB, M child nodes are created in continuous memory space (each node has an identical size), and the memory address of the first child node (denoted as addr) is stored in its parent node. During classification, if the packet falls in the m-th sub-space ($1 \leqslant m \leqslant M$), we can locate the address of the corresponding child node as follows: (1) count the number of '1's in the first m bits of the SAB, denoted as space_id; (2) compute the address by addr + space_id*sizeof(node).

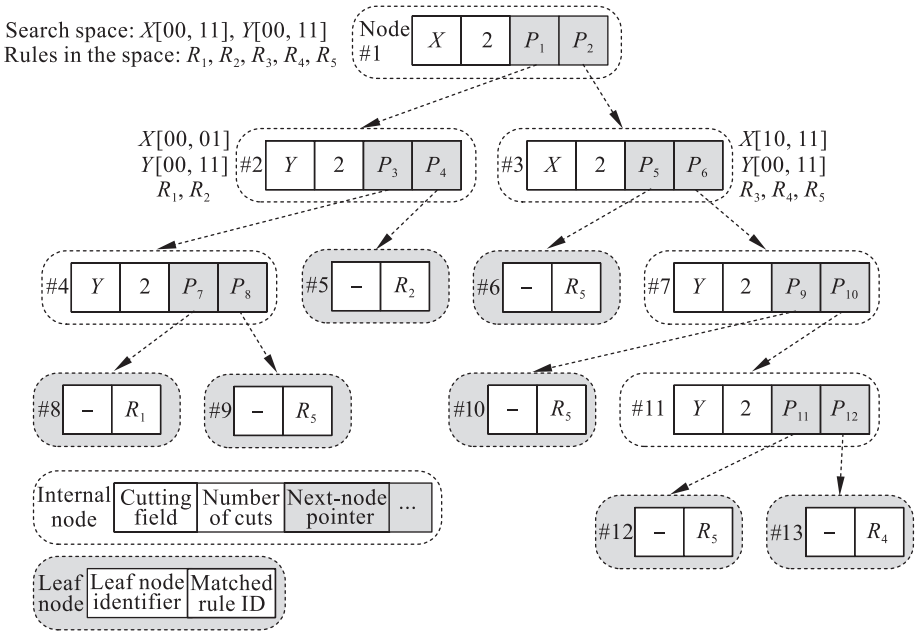

Fig. 1 HiCuts example

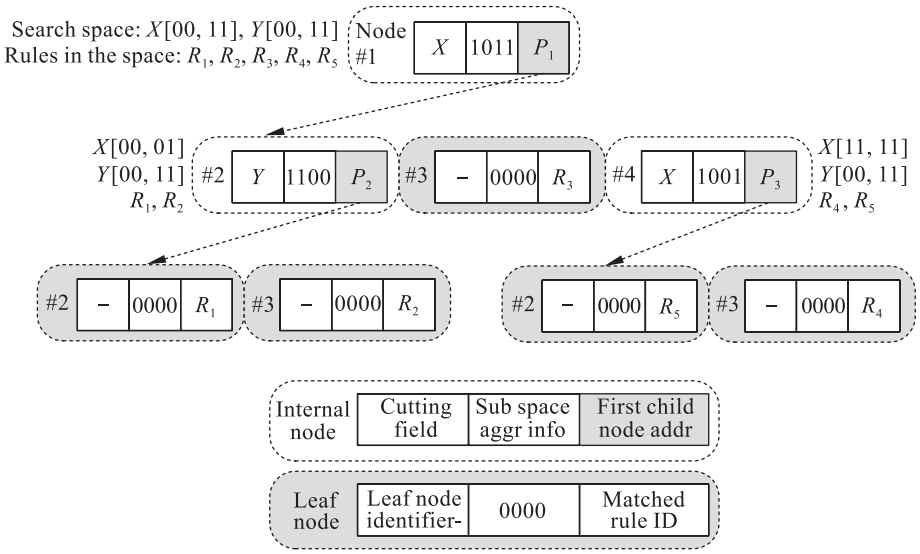

Fig. 2 AggreCuts example

With the SAB, the memory-consuming next-node pointer array is removed from the node structure, leaving only a bitmap with a single first-child pointer. However, to achieve fast classification speed, the fixed number of cuttings is large, which will still result in a large node size. For example, assume w=8, the size of the SAB is thus 2^8=256 bits, or 32 bytes. Accordingly, at each internal node, we need to load 32 bytes from memory. Since the tree depth is 104/8=13, the overall memory access time required to classify a single packet is more than 400 bytes, which might be too much for a practical memory bandwidth budget to reach multi-Gbps[1]. In addition, counting the '1's in all the 256 bits is also a heavy burden that may also reduce the processing performance.

Fortunately, in our experiments on a variety of real-life rule sets, we have found that the number of child nodes of a given internal tree node is usually very small. For example, with 256 cuttings at each internal node, the average number of child nodes is less than 10, i.e., M<10 in most cases. This observation indicates that the number of bits set as '1' in the SAB is small, and this motivates us to compress the SAB to further reduce the number of memory accesses.

Figure 3 illustrates how to use the hierarchical space aggregation bitmap (HSAB) to further compress the data structure of internal nodes. Define the sizes of the SAB and HSAB as 2^w and 2^v, respectively, and let $u=w-v$. To compress the 2^w bit SAB: (1) divide the 2^w SAB into 2^v consecutive sub-bitmaps; (2) set the first bits in the HSAB as '1'; (3) set the i-th bit of the HSAB as '1' if the i-th sub-bitmap of the SAB contains at least one '1'; (4) else, set the i-th bit of the HSAB as '0'.

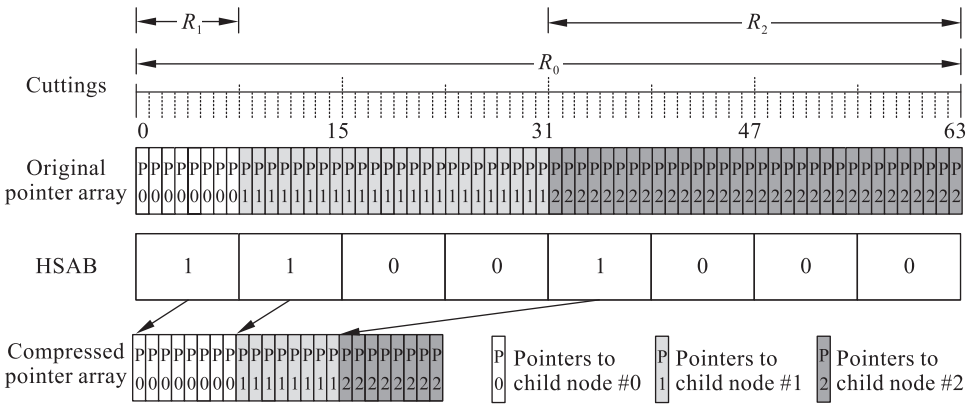

Fig. 3 Hierarchical space aggregation

A bit set in an HSAB indicates that a different sequence of the consecutively identical sub-array of pointers starts at the corresponding position. At the same time, whenever a bit is set, its corresponding

sub-array of pointers is appended in the compressed pointer array (CPA). Accordingly, the n-th pointer in the original point array can be located as follows: (1) extract the higher v bits of n to get a v-bit value m; (2) extract the lower u bits of n to form a u-bit value j; (3) add 0-m bits of the HSAB to get a sub-array index i; (4) use $((i<<u)+j)$ as the index to load the corresponding pointer from the CPA. The flowchart of AggreCuts is shown in Fig. 4. The 32-bit and 64-bit data structures of AggreCuts are shown in Fig. 5.

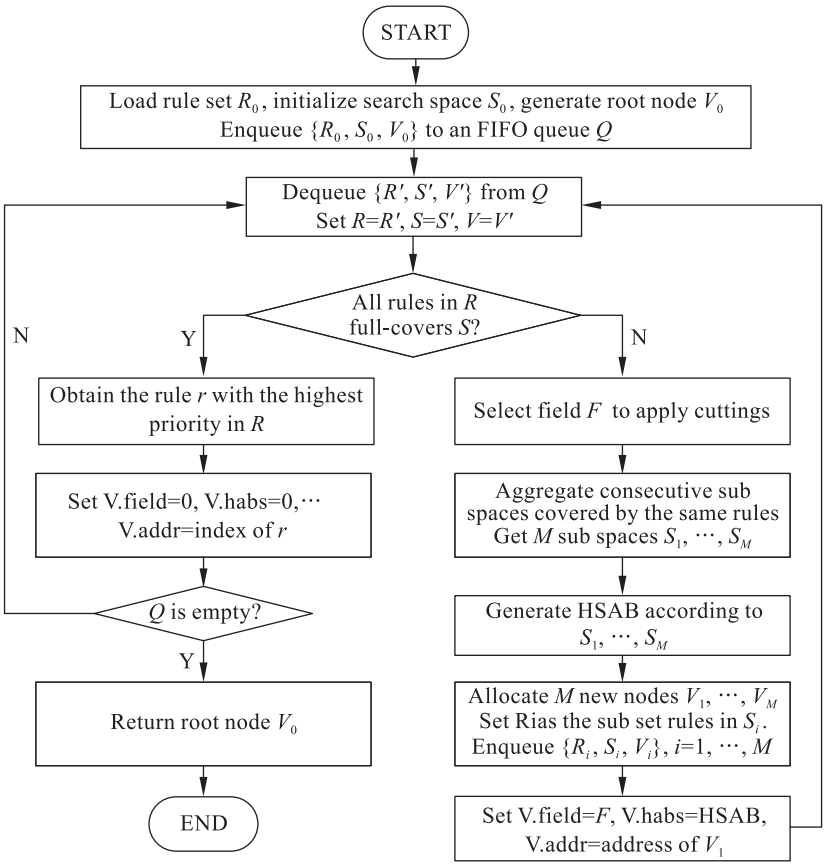

Fig. 4 Flowchart of AggreCuts

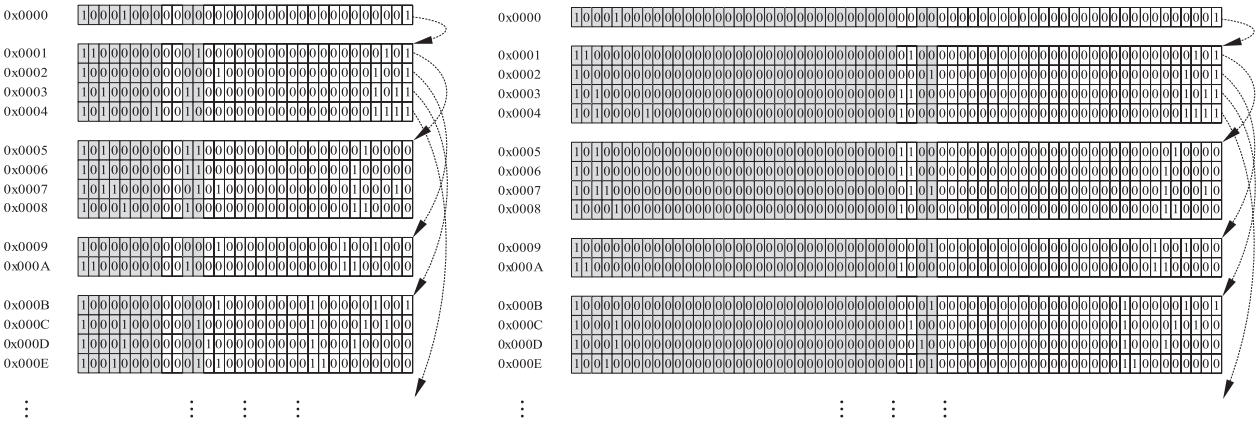

Fig. 5 32-bit and 64-bit implementation of AggreCuts

3 Performance Evaluation

We evaluated the proposed AggreCuts algorithm with real-life rules. The performance of AggreCuts was then compared to the performance of one of the most popular algorithms, HiCuts[11], through both software simulation and hardware tests.

3.1 Data sets and testbeds

Our research focuses on real-life rule sets because experimental results on these rule sets are more convincing than those obtained on synthetic rules. We evaluated all the packet classification algorithms on real-life firewall and core router rule sets. The rule sets are SET01, SET02,…, SET07[14]. The largest real-life rule set (SET07) contains 1945 rules. All rules are 5-dimensional with 32-bit source/destination IP addresses, 16-bit source/destination port numbers, and 8-bit transport layer protocols. Table 1 shows the characteristics of these rule sets. Input packets are set to match the worst-case search condition.

Table 1 7 real-life rule sets in our test

Name	Number of rules	Length of prefix
SET01	68	30-32 for destination IP 0 and 16 for source IP
SET02	136	
SET03	340	
SET04	500	Mainly 32 for destination IP Vary from 0-32 for source IP
SET05	1000	
SET06	1530	
SET07	1945	

Our testbeds are 32-bit Intel IXP2850[3] and 64-bit Cavium OCTEON3860[4] multi-core network processing platforms. The IXP2850 architecture (shown in Fig. 6) uses multiple processing micro-engines (ME) where each ME has 8 hardware threads. All MEs work in parallel running at 1.4 GHz. The IXP2850 has 4 channels of QDR SRAM running at 233 MHz and 3 channels of RDRAM running at 127.3 MHz. The IXP2850 also has flexible 32-bit media switch interfaces. Each interface is configurable as media standard SPI-4 or CSIX-L1 interfaces[6].

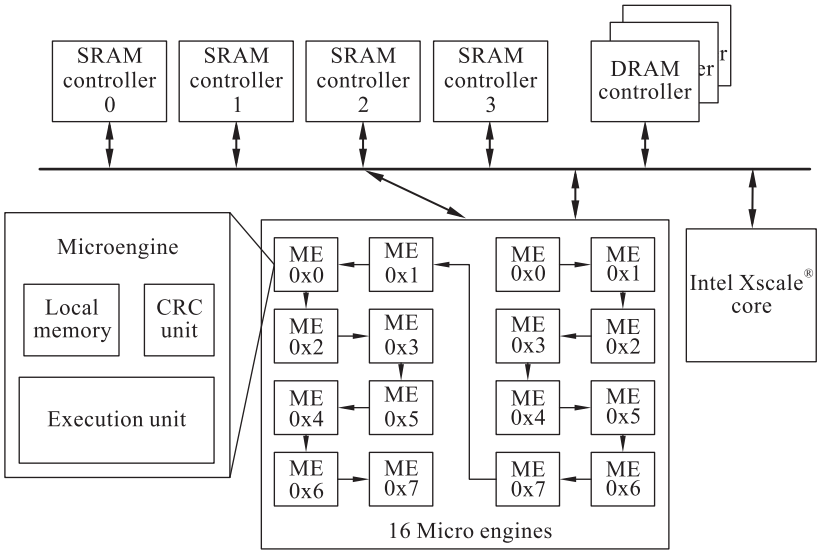

Fig. 6 Intel IXP2850 architecture

The architecture of the OCTEON3860 is shown in Fig. 7. The OCTEON3860 has 16 MIPS cores running at 500 MHz, and its network interfaces consist of eight 1 Gbps RGMII ports. Memory hierarchy includes 1 MB shared L2 cache, 2 GB DDR2 SDRAM, and 8x16 MB RLDRAM. The PIP unit receives packets from the network; then the POW unit schedules packets (as a work) to different cores for packet processing; and finally packets are sent out from the PKO unit.

3.2 Software simulations

In our software simulations, we evaluated the worst-case memory access times and the memory usage of AggreCuts. From Fig. 8, we can see that the worst-case memory accesses of AggreCuts is less than 20% of that of HiCuts. This is because the worst-case tree depth of HiCuts depends on the data structure of the rule set, while that of AggreCuts is set-independent due to the fixed stride cutting scheme. Such a definite worst-case memory access is expected to guarantee stable performance of high-speed flow classification on the network processor.

The memory usages of AggreCuts and HiCuts are shown in Fig. 9. It can be seen that AggreCuts uses significantly less memory than HiCuts. Specifically, AggreCuts requires less than 5.3 MB of memory space,

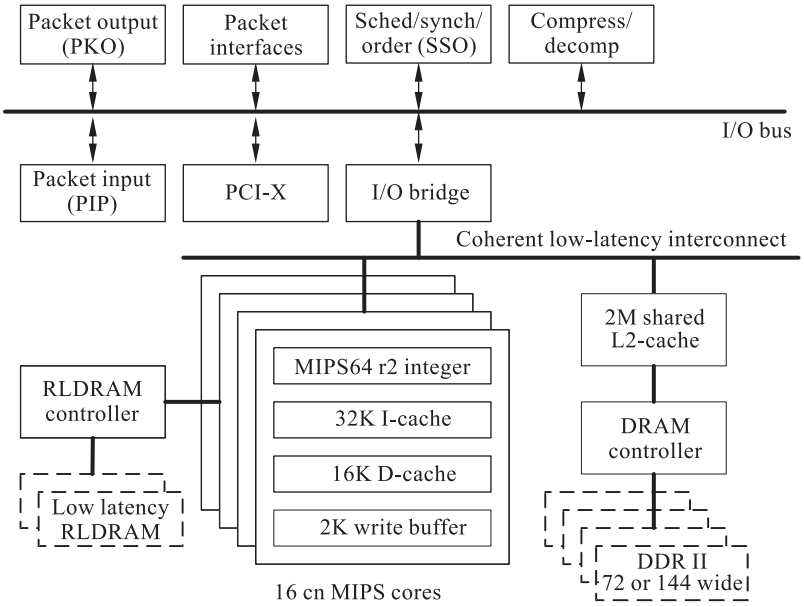

Fig. 7 Cavium OCTEON3860 architecture

Fig. 8 Memory access comparison

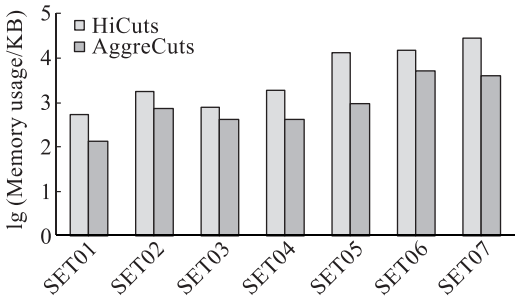

Fig. 9 Memory usage comparison

which is smaller than the size of a single SRAM chip on the IXP2850 network processor (there are three 8 MB SRAM chips on the IXP2850). In comparison, the memory usage of HiCuts on SET07 is larger than 28 MB, which exceeds the total memory size of all three SRAM chips.

3.3 Hardware performance

Figures 10 and 11 show the throughput of the AggreCuts algorithm on two different multi-core platforms. For the IXP2850 platform, we used 64-byte Ethernet packets as the input traffic and set each packet to match the longest tree path (i.e., each packet will incur the worst-case memory access). From Fig. 10, we can see that the AggreCuts algorithm achieves over 8.8 Gbps throughput on the IXP2850 platform (100% throughput is 10 Gbps). In contrast, the HiCuts algorithm achieves less than 2 Gbps throughput.

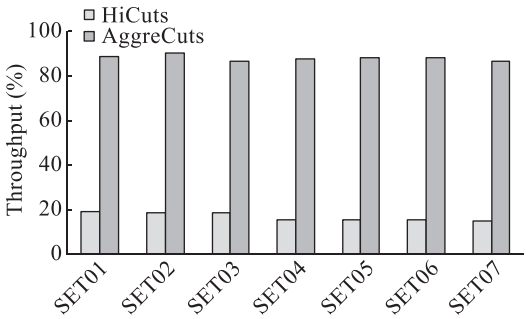

Fig. 10 Throughput on the IXP2850

Figure 11 shows the throughput of AggreCuts and HiCuts on the OCTEON3860 multi-core platform. For the OCTEON platform, we used different packet sizes in our test. From Fig. 11, we can see that the AggreCuts algorithm achieves over 3 times more throughput than HiCuts on 64-byte Ethernet packets. As the packet size increases, the AggreCuts algorithm has near linear speedup and reaches 100% throughput (8 Gpbs) with 512-byte and larger packets. In comparison, the HiCuts

algorithm cannot reach the 100% line rate even with 1518-byte packets.

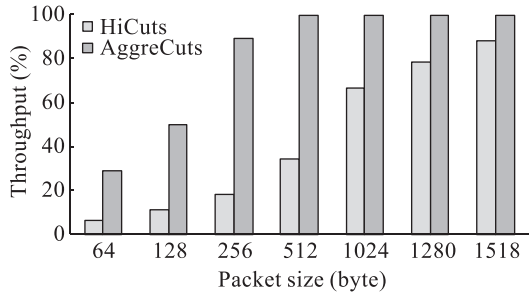

Fig. 11 Throughput on the OCTEON3860

4 Conclusions

Packet classification is crucial to the implementation of several advanced network services that require the capability to distinguish traffic in different flows, such as firewalls, intrusion detection systems, and many QoS implementations. To reach multi-Gbps packet classification speed, this computation-intensive task can utilize new generation network processors to perform at nearly the line rate. In this paper, we have proposed the AggreCuts algorithm which has explicit worst-case search time with modest memory usage. The data structure of AggreCuts is also optimized for multi-core platforms. To evaluate the performance of AggreCuts, we implemented the algorithm on both Intel IXP2850 32-bit and Cavium OCTEON3860 64-bit multi-core platforms. The experimental results show that AggreCuts outperform the best-known existing algorithm in terms of memory usage and classification speed.

References

[1] Naik U, Chandra P. Designing High-Performance Networking Applications. USA: Intel Press, 2004.

[2] Sherwood T, Varghese G, Calder B. A pipelined memory architecture for high throughput network processors. In: Proceedings of the 30th ACM/IEEE International Symposium on Computer Architecture (ISCA). San Diego, USA, 2003.

[3] IXP2850. http://www.intel.com/design/network/products/npfamily/ixp2xxx.htm. 2010.

[4] OCTEON3860. http://www.caviumnetworks.com/OCTEON-Plus_CN38XX_solutions.html. 2010.

[5] XRL732. http://www.netlogicmicro.com/Products/Multi-Core/XLP.asp. 2010.

[6] Qi Yaxuan, Xu Bo, He Fei, et al. Towards optimized packet classification algorithms for multi-core network processors. In: Proceedings of the International Conference on Parallel Processing (ICPP). Xi'an, China, 2007.

[7] Overmars M H, Van der Stappen A. Range searching and point location among fat objects. *Lecture Notes in Computers Science*, 1994, **855/1994**: 240-253.

[8] Gupta P, McKewon N. Algorithms for packet classification. *IEEE on Network*, 2001, **15**(2): 24-32.

[9] Gupta P, McKeown N. Packet classification on multiple fields. In: Proceedings of ACM SIGCOMM. Boston, USA, 1999.

[10] Xu Bo, Jiang Dongyi, Li Jun. HSM: A fast packet classification algorithm. In: Proceedings of the 19th International Conference on Advanced Information Networking and Applications (AINA). Taipei, China, 2005.

[11] Gupta P, McKeown N. Packet classification using hierarchical intelligent cuttings. In: Proceedings of Hot Interconnects. California, USA, 1999.

[12] Singh S, Baboescu F, Varghese G, et al. Packet classification using multidimensional cutting. In: Proceedings of ACM SIGCOMM. Karlsruhe, Germany, 2003.

[13] Qi Yaxuan, Xu Lianghong, Yang Baohua, et al. Packet classification algorithms: From theory to practice. In: Proceedings of the 28th IEEE International Conference on Computer Communications (INFOCOM). Rio de Janeiro, Brazil, 2009: 648-656.

[14] Qi Yaxuan, Xu Bo, He Fei, et al. Towards high-performance flow-level packet processing on multi-core network processors. In: Proceedings of the 3rd ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS). Orlando, USA, 2007.