# Towards Effective Network Algorithms on Multi-core Network Processors

Yaxuan Qi[1, 2], Zongwei Zhou[1], Baohua Yang[1], Fei He[1], Yibo Xue[1, 2] and Jun Li[1, 2]
[1]Research Institute of Information Technology, Tsinghua University, Beijing, China
[2]Tsinghua National Lab for Information Science and Technology, Beijing, China
{yaxuan, yiboxue, junl}@tsinghua.edu.cn
{zhou-zw02, ybh03, hefei06}@mails.tsinghua.edu.cn

## ABSTRACT
To build high-performance network devices with holistic security protection, a large number of algorithms have been proposed. However, multi-core implementation of the existing algorithms suffers from three limitations: performance instability, data-structure heterogeneity, and hardware dependency. In this paper, we propose three principles for effective network processing on multi-core network processors. To verify the effectiveness of these principles, algorithms for two typical network processing tasks are redesigned and implemented on the Cavium Octeon3860 network processor. Test results show that our schemes achieve superior performance in comparison with existing best-known algorithms.

## Categories and Subject Descriptors
C.2.0 [**Computer Communication Networks**]: General – *security and protections*

## General Terms
Algorithms, Performance, Experimentation

## Keywords
Packet classification, Pattern matching, Network processor

## 1. INTRODUCTION
To achieve high-performance full inspection for network traffic, numerous network security related algorithms have been proposed in recent years. However, the existing algorithms have the following limitations:

- Performance Instability: The performance stability is one of the most important criteria to evaluate algorithms. However, a lot of existing algorithms can only apply to very specific policy sets, while cannot guarantee worst-case performance in other situations.
- Data-Structure Heterogeneity: The more complex network applications, the more complex algorithms, and the attendant are the more complicated data-structures.
- Hardware Dependency: Many of the existing algorithms require special hardware support. Such hardware dependency severely declines the portability of the algorithms, and increases the power-consumption of the overall system.

In this paper, we solve these problems from the general line of thought. Main contributions are:

- Three principles for network algorithm design: Performance stability, homogeneous data-structure, hardware independency.
- Redesign of two typical algorithms: One for packet classification and the other for pattern matching.
- Performance evaluation on multi-core platform: All algorithms are implemented on Cavium Octeon3860 [1] (Fig. 1).

## 2. Principles for Network Algorithm Design
To achieve high-performance network processing device with integrated service and holistic protection, novel network algorithms must meet the following requirements:

- **Deterministic Performance**: Each algorithm must have explicit worst-case performance bound, so that the worst-case performance of integrated applications can be guaranteed.
- **Homogeneous Data-Structure**: Data-structure of different algorithms must be carefully arranged and well aligned for simple memory management and efficient memory access.
- **Limited Hardware-Dependency**: All algorithms should depend on neither cycle-consuming operations nor application specific co-processors.

## 3. Packet Classification Algorithm Design
We redesigned the AggreCuts algorithm [2] for packet classification based on the three principles:

- To meet performance stability: Our design takes byte as the processing units. Because the total length of packet header is 104 bits for 5-tuple packet classification, the maximum depth of the decision-tree is 104/8 = 13, and hence processing for every packet requires less than 13 memory access.
- To meet the homogeneous data-structure: Each tree-node of the redesigned algorithm (named as **AggreCuts64** due to 64-bit node size) is stored compactly in a 64-bit memory word for effective memory access. In comparison to the original AggreCuts (named as **AggreCuts32** due to 32-bit node size), the bitmap size is expanded from 8 to 32 to achieve better compression rate.
- To meet the hardware requirement: Only AND, ADD, SHIFT and POP_COUNT instructions are used in our design.


*ANCS'08*, November 6–7, 2008, San Jose, CA, USA.

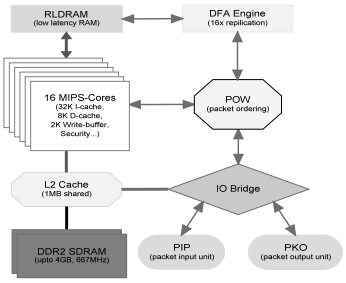


**Figure 1. Architecture of the Cavium Octeon3860 multi-core NP**

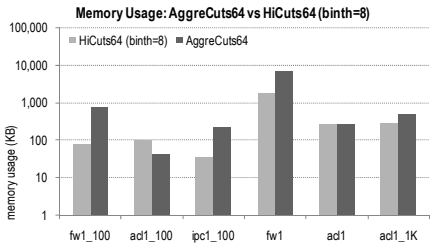


**Figure 2. Memory Usage: AggreCuts64 vs. HiCuts64 (binth=8)**

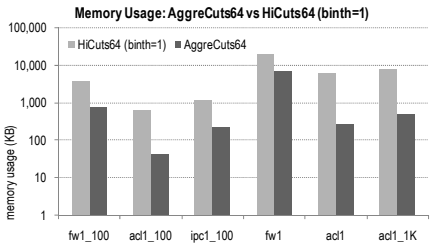


**Figure 3. Memory Usage: AggreCuts64 vs. HiCuts64 (binth=1)**

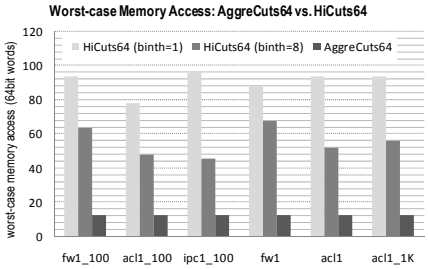


**Figure 4. Memory Access: AggreCuts64 vs. HiCuts64**

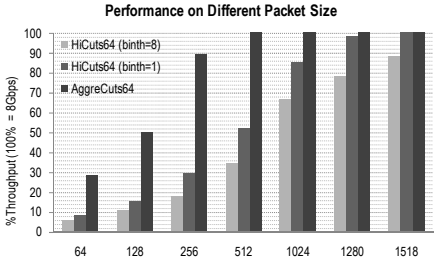


**Figure 5. Performance on Different Packet size: AggreCuts64 vs. HiCuts64**

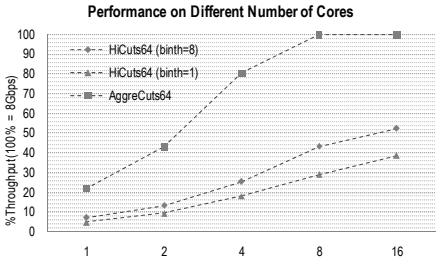


**Figure 6. Speedup with 512B Pkt: AggreCuts64 vs. HiCuts64**

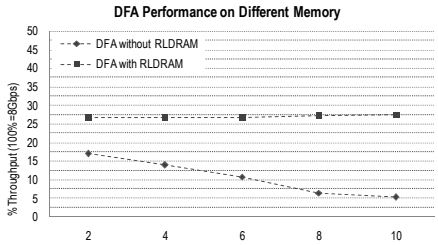


**Figure 7. DFA Performance with Different *min_length* (512B Pkt)**

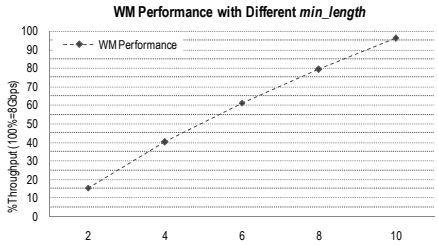


**Figure 8. WM Performance with Different *min_length* (512B Pkt)**

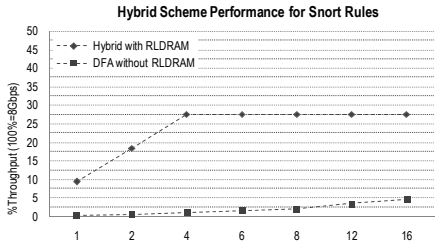


**Figure 9. Hybrid Scheme Performance Speedup (512B Pkt)**

## 4. Pattern Matching Algorithm Design

On the one hand, although DFA-based algorithms [3] meet the basic byte operation requirement of pattern matching, the memory blowup issue limits their applicability. On the other hand, while HASH-based algorithms [4] successfully reduce the memory usage, their performance is instable when the minimum pattern length (*min_length*) is small and/or collision rate is high. Considering all these issues, in this paper we leverage the advantages of these two types of algorithms and propose a hybrid pattern matching algorithm optimized for multi-core network processors. Basic ideas are:

- Divide the pattern set into two sub-sets according to pattern length.
- Apply a DFA-based algorithm to short patterns, so that DFA can be stored in low-latency memory for fast processing.
- Apply a HASH-based algorithm to long patterns, so that the matching can take large "jump" to avoid unnecessary check.

## 5. Evaluation

Performance of packet classification algorithms is shown in Fig. 2-6. Test of pattern matching algorithms is shown in Fig. 7-9.

## 6. Acknowledgment

This work was supported by National High-Tech R&D 863 Program of China under grant No. 2007AA01Z468.

## 7. REFERENCES

[1] http://www.cavium.com/OCTEON_MIPS64.html

[2] Y. X. Qi, B. Xu, F. He, B. H. Yang. J. M. Yu and J. Li, "Towards High-performance Flow-level Packet Processing on Multi-core Network Processors," Proc. of ACM/IEEE Symposium on Architectures for Networking and Communications Systems (ANCS), 2007.

[3] A. V. Aho and M. J. Corasick, "Efficient String Matching: An Aid to Bibliographic Search," Communications of the ACM, 18(6):333–340, 1975.

[4] S. Wu and U. Manber, "A Fast Algorithm for Multi-pattern Searching," Technical Report TR-94-17, Department of Computer Science, University of Arizona, 1994.