Computer Standards & Interfaces 36 (2014) 880–888

Contents lists available at ScienceDirect

Computer Standards & Interfaces

journal homepage: www.elsevier.com/locate/csi

# A regular expression matching engine with hybrid memories

Shuhui Chen [a,*], Rongxing Lu [b,*]

[a] College of Computer Science, National University of Defense Technology, Changsha 410073, China
[b] School of Electrical and Electronics Engineering, Nanyang Technological University, 50 Nanyang Avenue, 639798, Singapore

ARTICLE INFO

*Article history:*
Received 22 February 2013
Received in revised form 2 December 2013
Accepted 11 December 2013
Available online 31 December 2013

*Keywords:*
Deep Packet Inspection
Regular expression
Matching engine
Markov chain

ABSTRACT

A key technique of network security inspection is by using the regular expression matching to locate the specific fingerprints of networking applications or attacks in the packet flows, and accordingly identify the underlying applications or attacks. However, due to the surge of various networking applications and attacks in recent years, even more fingerprints need to be investigated in this process, which leads to a high demand on a large memory space for regular expression matching. In addition, with the frequent upgrading of the network links nowadays, the network flow rate also increases dramatically. As a result, it demands the fast operation of regular expression matching accordingly with the enhanced throughput for network inspection. However, due to the limited space of the fast memory, the requirements on fast operations and large memory space are conflicting. On addressing this challenge, in this paper, we propose to use hybrid memory for regular expression matching. In specific, by investigating on the transition table state access probability through the Markov theory, it can be observed that there exist a number of states which are much more frequently accessed than others. Therefore, we devise a matching engine which is suitable for FPGA implementation with two-level memories, where the first-level memory uses the on-chip memory of FPGA to cache the frequently accessed state transitions, and the second-level memory, composed of slow and cheap DRAM, stores the whole state transitions. Furthermore, the L7-filter's regular expression patterns have been applied to obtain the state access probability, and different quantities of memory assignment approaches have also been investigated to evaluate the throughput.

## 1. Introduction

Deep Packet Inspection (DPI) is a key technique for Network Intrusion Detection System (NIDS) and Network Forensic System (NFS) [1,2]. In these network security devices, the payload of packets is matched against certain pre-defined patterns, e.g., fingerprints, to identify specific classes of applications, viruses, attacks and criminal evidences, etc. One approach in DPI is by using the string matching. However, it is not quite flexible and powerful to describe complicated fingerprints. As a remedy to that, regular expressions become a main approach rapidly in place of explicit string patterns as the pattern matching language in packet scanning applications. Due to its expressive power and flexibility for describing useful patterns, regular expression has been widely used. For example, in the Linux Application Protocol Classifier (L7-filter) [3], all protocol identifiers are expressed as regular expressions. Similarly, the Snort [4] intrusion detection system has evolved from no regular expressions in its rule set in April 2003 to several thousand rules using regular expressions currently. Another intrusion detection system, Bro [5], also adopts regular expressions as its pattern language.

To conduct the match, regular expressions are compiled into Finite State Machine (FSM), and then the packet payload is scanned with the FSM. In the case of a light-weight network, with low flow rate and a small number of patterns, pattern separated approach (in which every regular expression is compiled into several FSMs, and the packet payload is matched those FSMs one by one sequently) can satisfy the throughput requirement. However, with the dramatic increase of bandwidth, multi-Gbps links are nowadays widely applied in campus networks, and the scale of the patterns of the typical DPI system is over one hundred, as thus the traditional pattern separated approach no longer meets the critical performance requirement. If a pattern integrated FSM is utilized, the inflation of the states makes it impossible to be filled into high speed memory like Static Random Access Memory (SRAM) or on-chip memory of Field Programmable Gate Array (FPGA).

In regular expression matching, the storage capacity and the memory speed become an irreconcilable conflict. In order to decrease the storage requirement, previous researches (a detailed survey of the previously reported literatures will be provided later) were exerted to reduce the transition table memory. In this paper, we do not consider how to deduce the memory of the state transition table, instead, we will take hold of how to conjointly use high speed memory with small space and slow speed memory with large space to make matching engine at almost the throughput of the high speed memory.

* Corresponding authors.
*E-mail addresses:* shchen@nudt.edu.cn (S. Chen), rxlu@ntu.edu.sg, rxlu.cn@gmail.com (R. Lu).

0920-5489/$ – see front matter © 2013 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.csi.2013.12.001

Every byte of packet payload needs at least one memory access in DPI systems; as a result, the throughput constraint of the network security devices is memory access time. Generally, the immanent fluidness of network packets results in a very low hit ratio of the cache. Since the number of the memory access is pre-defined and cannot be changed subsequently, the potential to enhance system performance is to improve the efficiency of each memory access.

Traditional network security devices like NIDS and NFS adopt high performance CPU platforms such as ×86 and MIPS, to improve the system performance. These typical CPUs are developed for the improvements on the calculation performance, so they are made as perfect or effective as possible on cache coherence, branch prediction, out-of-order execution, multi-core parallelism, etc. The improvements on memory access concentrate on how to increase the hit rate of the cache, but the fluidness of network packets cannot exploit these advantages; therefore, customizing hardware based approach such as FPGA is introduced as its better pipelining and parallelism. In addition, FPGA has specially designed on-chip memory for high bandwidth linkage with logic units, which could accomplish high speed communication between matching engine and memory. Nevertheless, the space requirement of the state transition table compiled from real-world patterns is far more than the capacity of the FPGA's on-chip memory.

To address the contradiction between the performance of the matching engine and the capacity of the memory, in this paper, we present a new hybrid memory matching engine, which enables the high throughput and a large scale of state transition table storage by utilizing two levels of memories to accommodate the state transition table. Specifically, the contributions of this paper are three-fold.

- First, based on the transition table, a state transition possibility matrix has been constructed, and the state accessing probability using Markov chain with stable state vector is investigated. To the best of our knowledge, it is the first work researching on formed states of regular expressions.
- Second, a two-level-memory hierarchy storage mechanism has been introduced, in which the Markov theory is used to obtain the frequently accessed states and their transition table entries are stored in the first-level memory while the second-level memory is used to hold the whole transition table entries.
- Finally, real-world regular expression patterns are used to produce the state transition table and state probability table, and a simulation analysis is employed to show that our hybrid-memory architecture can obtain the throughput almost as the first-level memory while the memory cost is nearly the same as the second memory.

The remainder of this paper is organized as follows. The related work is provided in Section 2, and the motivations on introduction of Hybrid Regular Matching Engine are presented in Section 3. Then, Section 4 depicts the system architecture and framework. The process of obtaining the stable vector is introduced in Section 5. The experiments and result analysis are in Section 6. Finally, we conclude our work in Section 7.

## 2. Background and related work

Finite State Automata (FSM) is a natural formalism for regular expressions. Although Deterministic Finite Automaton (DFA) and Non-deterministic Finite Automaton (NFA) are two kinds of FSMs that can be used to conduct DPI, the more preferred approach is DFA as it supports non-backtracking search. In DFA-based systems, a number of researches based on DFA systems compile $m$ regular expressions into a composite DFA, which provide guaranteed performance benefit over running $m$ individual DFAs. Specifically, a composite DFA reduces processing cost from $O(m)$ ($O(1)$ for each automaton) to $O(1)$, i.e., a single lookup obtains the next state for any given character. However, the number of states in the composite automaton grows to $O(\sum^{nm})$ in the theoretical worst case.

Even though the approach compiling $m$ regular expressions into a single FSM is relatively fast, its $O(1)$ scanning complexity still cannot meet the current links' bandwidth requirement, as the aggregated Internet traffic has been experiencing an annual bandwidth growth of 40%–50% in recent years [6]. To break the performance bottleneck of regular expression matching engine, a number of researches have been studied to improve the overall throughput by achieving efficient content-matching. Previously reported researches mainly focus on improving the throughput of the rule matching algorithms, and/or employing FPGA [7–9], GPU [10–13] or TCAM [14] for efficient content-matching.

Another kind of researches exerted to reduce the memory requirements can be clustered into two classes:

(1) The first FPGA-based solutions implement NFA [15–17]. Although NFAs are always smaller than DFAs, they need more memory bandwidth as an NFA may be in several states simultaneously whereas a DFA is always only in one state. Thus each byte that is processed might need to access the transition table for $|Q|$ times. Prior works have looked for different ways to find good NFA representations of transition table that limits the number of states that need to be processed simultaneously. These approaches combine the matching engine with the state transition table, which may lead to inconvenient pattern update.

(2) The second class of researches exerted to reduce the memory is based on DFA. Although the approach with $m$ regular expressions compiled together into a DFA can gain $O(1)$ computing complexity, its storage cost is extremely higher than that of the separated DFA. The number of states in a DFA scales poorly with the size and number of wildcards in the regular expressions. For a naive DFA with $k$ states, its memory requirement is $|\sum| \times k \times \lceil \ln k \rceil$, which is unsustainable as the $k$ increases so fast with the increasing of the regular expression number $m$ and $n$ (average length of the expressions). In 2007, Becchi and Cadambi [18] proposed a redundancy eliminating method. Its basic idea is merging "non-equivalent" states in a DFA by introducing labels on their input and output transitions. Its evaluation shows that it can drastically reduce the DFA memory requirement, but its performance is influenced as it needs to access several sub-tables.

In [19], the authors analyzed the size of DFAs for typical payload scanning patterns, and developed a grouping scheme which can strategically compile a set of regular expressions into several engines, resulting in a remarkable improvement on regular expression matching speed without much increase in memory usage.

Default transition and $D^2FA$ (Delayed Input DFA) were introduced and employed in [20,21]. $D^2FA$ is a special FSM based on DFA but with default transitions where each state can have at most one default transition to one other state. The directed graph named as deferment forest consisting of only default transitions in a $D^2FA$ must be acyclic. $D^2FA$ representation reduces transitions by more than 95% but with a speed penalty for long default transition paths in the $D^2FA$ matching.

In [14], the authors introduced a hardware-based RE matching approach that uses Ternary Content Addressable Memory (TCAM) and its associated Static Random Addressable Memory (SRAM) to hold state transition table. They proposed three novel techniques: transition sharing, table consolidation, and variable striding, to reduce TCAM space and improve regular expression matching speed. The experiments based on 8 real-world regular expression sets show that small TCAMs can be used to store large DFAs and achieve potentially high regular expression matching throughput.

We find the above-mentioned DFA based space compressing methods except [14] sacrifice performance to some extent to occupy less memory.

Our research is orthogonal to these researches as the second-level memory is not so sensitive to the access performance, which signifies that using these research achievements can reduce the second-level memory space in our proposed approach.