Computer Communications 70 (2015) 86–94

Contents lists available at ScienceDirect

Computer Communications

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

# A compression approach to reducing power consumption of TCAMs in regular expression matching

Jinmin Yang[a,*], Jie Yang[a], Kun Huang[b], Huigui Rong[a], Kin Fun Li[c]

[a] College of Computer Science and Electronic Engineering, Hunan University, China
[b] Institute of Computing Technology Chinese Academy of Sciences, China
[c] Department of Electrical and Computer Engineering, University of Victoria, Canada

A R T I C L E I N F O

*Article history:*
Received 13 January 2015
Revised 11 June 2015
Accepted 2 August 2015
Available online 8 August 2015

*Keywords:*
Regular expression matching
Ternary content addressable memory
Power consumption
Compressibility identification
Source state differentiation

A B S T R A C T

Ternary content addressable memory (TCAM) is a popular device for fast regular expression (Regex) matching in networking and security applications. The rapid growth of Regexes necessitates large TCAM memory consumption, which in turn impacts power consumption. Compressing the transition table can cut down TCAM memory consumption, thereby reducing its power consumption. This work identifies the compressibility of transition entries and then proposes a compression scheme. In our scheme, the compression ratio is improved by skillfully assigning identifiers to states in a deterministic finite automaton (DFA). Furthermore, our scheme utilizes the wildcard function and the priority matching mechanism provided in TCAM to exploit the minimum differentiation among a set of source states. A complete implementation of the identifier assignment and transition table compression is presented. Experimental results on real-world Regex sets show that our scheme is significantly more effective, reducing power consumption by 87.4% and memory space by 93.2%, and improving throughput up to 114.7% on average compared to prior work.

© 2015 Elsevier B.V. All rights reserved.

## 1. Introduction

Deep packet inspection (DPI) [1, 2] is a core component of networking and security devices such as network intrusion detection/prevention systems (NIDS/NIPS) [3,4], firewalls, and routers, etc. Its function is to detect malicious intrusions and identify specific applications. This is achieved by examining both the header and the payload of network data packets against a set of predefined string signatures. This process is called ‘string matching’. However, it is difficult to use such a string matching approach to represent and match complex string signatures. Therefore, regular expression (Regex) matching [5–6] is adopted to displace string matching in many DPI systems such as bro [7], snort [8], and Cisco [9], as Regex is more flexible and expressive in describing complex string signatures. In practice, a given Regex can be precisely realized by equivalent deterministic finite automaton (DFA). DFA is a finite state machine in which a source state accepts an input character and then migrates to a unique destination state. In DPI, DFA built from Regex is typically implemented in hardware such as ternary content addressable memory (TCAM) [10].

TCAM is a type of memory that parallel search can be performed at a high speed. A TCAM consists of a set of memory entries. Each entry is a vector of cells, where every cell can store one bit. Therefore, a TCAM memory entry can be used to store a string. TCAM is expensive and its content space is limited, so TCAM is typically implemented with a low-cost memory device, like static random access memory (SRAM) [11]. As to a search, TCAM works as follows: given an input string, it compares this string against all entries in its memory in parallel, and returns the matching data in SRAM. When TCAM is used to perform DFA, every state transition in a DFA is encoded as one TCAM memory entry. Every entry consists of two parts: the TCAM part storing one source state of a DFA and one input character, and the associative SRAM storing the corresponding destination state. All TCAM memory entries built from transitions in a DFA constitute a global transition table.

TCAMs are off-the-shelf chips capable fast parallel lookups, and have been widely deployed in modern network devices. Though, the high power consumption of TCAM has become a critical issue with the proliferation of data centers which have thousands of TCAM-based network devices. Nowadays, more and more companies are concerned with power consumption in their devices. For example, a Cisco system provides an additional 600 Watt redundant power module to support its four 150 Watt external network devices [12]. In these devices, a typical 18 Mb TCAM, such as Cypress’s NSE10K and Netlogic’s NL3280, can consume up to 15 Watts of power when all TCAM entries are activated for a search [13]. The 15 Watts of each TCAM is significant power consumption in data centers with

* Corresponding author. Tel.: +86 13975896967.
*E-mail address:* rj_jmyang@hnu.edu.cn (J. Yang).

http://dx.doi.org/10.1016/j.comcom.2015.08.003
0140-3664/© 2015 Elsevier B.V. All rights reserved.

thousands of TCAM-based network devices. This is especially critical in power-constrained situations where most high performance devices, such as routers and switches, can easily consume tens and hundreds of Watts if these searches are not optimized.

The power consumption of TCAM is directly proportional to the number of its memory entries. A huge DFA with a large number of Regex signatures means the need of many memory entries in the TCAM. Consequently, the high power consumption of TCAM occurs in lockstep with the rapid growth of the size of Regex signatures. Furthermore, DFA suffers from the state explosion problem, where the number of DFA states grows exponentially with the size of Regex signatures. With the growing size of Regex signatures, how to reduce TCAM power consumption has become a hot spot in large data centers with thousands of TCAM-based network devices.

When the memory entries of TCAM are chunked into a set of fixed-size blocks, it is feasible to reduce the power consumption of TCAM. The feasibility comes from the heuristics that a search is involved in only a small part of memory entries of the TCAM. When a search is executed, only the involved blocks need to be activated, while non-involved blocks can stay in the dormant state. Thus, the TCAM power consumption is decreased. The key issue of the strategy is how to identify in advance which blocks are involved in a search. The character-indexed scheme [14] was proposed to address this issue by separating input characters from the global transition table in a TCAM. This scheme constructs an extra front-end character index table for the blocks involved in a specific search. Nevertheless, such a scheme still leads to high power consumption, because every transition table has many redundant transitions. If every transition table could be compressed, the TCAM power consumption can be further reduced.

This work aims to reduce TCAM power consumption by transition table compression. We call our approach power-efficient DFA (PEDFA). The main idea is to compress all transition tables so as to minimize TCAM space usage, thereby achieving lower TCAM power consumption. PEDFA exploits two features of TCAM. First, TCAM has a ternary state in each cell. Each cell has three states: 0, 1 and * (wildcard). A "*" state matches both "0" and "1", hinting the possibility of memory compression. Second, TCAM returns the first matching entry when two or more entries match the input key, which is called TCAM priority matching. This mechanism can be used to further reduce the number of TCAM entries. It implies that those TCAM memory entries with the same prefix codes and destination state can be displaced by a single TCAM memory entry consisting of the prefix codes plus wildcard state "*". Suppose that there are two TCAM memory entries "000 → 010" and "001 → 010". They have the same prefix code "00" and destination state "010". Consequently, they can be compressed into one TCAM memory entry "00* → 010".

The proposed PEDFA is evaluated using the real Regex sets *snort*, *bro*, and *cisco*. A series of experiments are conducted to compare PEDFA to state-of-the-art character-indexed (CIDFA). The preliminary results demonstrate that PEDFA reduces 93.2% memory space as well as 87.4% power consumption, and improves throughput up to 114.7%, on average against CIDFA.

The structure of this paper is as follows. In Section 2, we discuss related work. Section 3 gives a detailed description of the proposed PEDFA. Section 4 presents our evaluation methodology and the experimental results on real Regex sets. Finally, conclusion is presented in Section 5.

## 2. Related work

For TCAM-based implementations of Regex matching, memory efficiency is a key issue. It has a direct influence on matching speed and throughput, and especially on power consumption. Many approaches have been proposed to improve memory efficiency.

Sharing common data [15] is an effective approach to memory efficiency. For transition entries stored in TCAM, there exists a lot of redundancy in terms of input character, source state, and destination state. The redundancy can be reduced or eliminated by extracting common parts and then sharing them. One way is to transform the storage structure from a table into a tree [15]. In this approach, many schemes were proposed, such as transition sharing, default transition, and failure transition in [15], and character indexing in [14]. Transition sharing merges multiple transitions of DFA into one TCAM entry by exploiting both character redundancy and state redundancy, while a table consolidation scheme [15] extracts the common part of multiple transition tables to form a shared table.

Memory efficiency can be also achieved by transformation [16–20]. For example, delay input DFA ($D^2$FA) [16] constructs a tree structure for every DFA according to its inherent logic relationship, respectively, and then analyzes local similarity in a tree or among trees, similar to default transition path [17] and variable striding [16]. After that, every similar part is extracted and transformed into a simple transition with a bit of additional information and/or treatment. For example, $CD^2$FA [18] uses a prefix and a hash function to determine the specific outlet of a default path in matching. The DFA deflation scheme [19] implements one simple TCAM lookup to match an input character by exploiting the structural connection between DFA and non-deterministic finite automaton (NFA). NFA is space-efficient, as the memory needed by NFA grows linearly with the size of Regex signatures. However, NFA is time-inefficient, as it maintains possibly multiple active states at the same time. CFA [20] focuses on the transformation of the complex terms of Regex patterns like the wildcard closure ".*" and the associated character "{}". Those complex terms have an important influence on DFA state size, so CFA can effectively reduce TCAM space consumption. However, CFA requires devices to have certain logic handling capability.

Compression [21,22] is another approach to memory efficiency. Although multiple transition entries have different values in one or more attributes, they can be compressed into one entry by exploiting TCAM characteristics. The wildcard provided by TCAM contributes to this target. In TCAM, a value with one or more wildcards can cover and express multiple values. In this compression approach, higher compression ratio is usually the primary research effort. The global scheme in [21] directly compresses the rows of the transition table, while the local scheme in [22] conducts compression after transforming the two-dimensional table into a hierarchical tree. Compact DFA [22] achieves a higher compression ratio by adding a bit of information about state properties in the state code.

Although memory efficiency contributes to lower power consumption of TCAM, special emphasis should be put on power efficiency. This is critical for large-scale deployment of devices in network environment such as data center networks. String matching probability was exploited to reduce power consumption [22]. Based on the characteristic that the mismatch has a very high probability, prefixes are extracted from strings to form two-level matching in separate modules. Thus the scale of parallel matching becomes smaller, leading to lower power consumption. Another low-power TCAM solution was called character-indexed DFA (CIDFA) [14]. CIDFA reduces the power consumption by indexing characters and structuring TCAM units. In CIDFA, TCAM memory consists of two kinds of blocks: character-index blocks and transition blocks. In the character-index blocks, all input characters are associated with the corresponding transition block IDs. In transition blocks, all transitions are clustered based on the input characters. In a Regex matching, the first thing is to scan character-index blocks to determine which transition blocks need to be activated.

CIDFA can avoid activating irrelevant transition blocks, resulting in lower power consumption. However, its memory consumption is still high because it does not compress its TCAM space. High memory consumption implies high power consumption in TCAMs.

Download English Version:

https://daneshyari.com/en/article/449970

Download Persian Version:

https://daneshyari.com/article/449970

Daneshyari.com