Layered Interval Codes for TCAM-based Classification

Anat Bremler-Barr Boris Farber
The Interdisciplinary Center, Israel
{bremler, farber.boris}@idc.ac.il

David Hay Danny Hendler
Ben Gurion University, Israel
{davidhay, hendlerd}@cs.bgu.ac.il

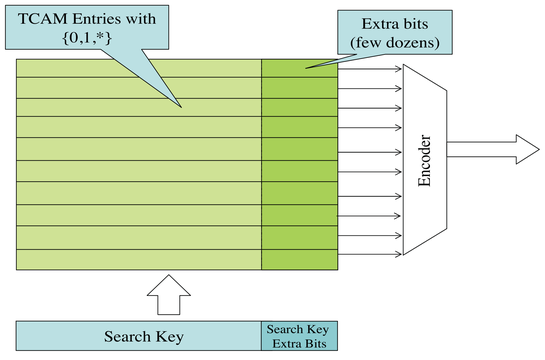
Ternary content-addressable memories (TCAMs), which compare in parallel packet headers against all rules in a classification database, are increasingly used for high-speed packet classification.

TCAMs are not well-suited for representing rules that contain range fields. Such rules are represented by multiple TCAM entries and the range expansion they introduce can dramatically reduce TCAM utilization. This redundancy can be mitigated by making use of extra bits, available in each TCAM entry.

We present a novel scheme for constructing efficient representations of range rules, based on the simple observation that sets of disjoint ranges may be encoded much more efficiently than sets of overlapping ranges.

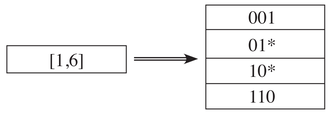
Our technique splits the ranges between multiple layers each of which consists of mutually disjoint ranges. Each layer is then coded independently and assigned its own set of extra bits. An extensive comparative analysis on real-life classification databases establishes that our algorithms reduce the number of redundant TCAM entries caused by range rules by more than 60% as compared with best range-encoding prior art.

The TCAM Device



Each TCAM entry consists of ternary digits: 0, 1, or '' (don't-care). When a key matches multiple TCAM entries, the TCAM returns the index of the first matching entry. This index is then used to locate the information specifying which actions to apply to the packet.*

Range fields, such as port fields, are hard to represent in TCAMs. The traditional techniques for range representation are by a set of prefixes, such that each is stored in a single TCAM entry:



The range [1 ,6] has a range expansion of 4 under the prefix expansion technique

Range expansion was found to cause an increase of more than 16% in TCAM space requirements for real-word databases.

Our Contributions

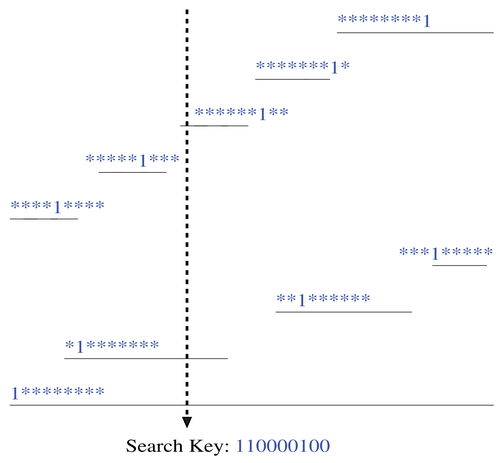
Novel scheme for constructing efficient representations of range rules, by making efficient use of TCAM extra bits.

- Algorithms for partitioning the ranges to sets of disjoint ranges (layers)
- Coding each layer independently with its own set of extra bits
- Supports hot updates for database changes
- Supports multiple range fields

Our empirical results show that all our algorithms reduce the average number of redundant TCAM entries required to represent a range rule decreases by more than 60% as compared to best range-encoding prior art.

Why layering improves the performance?

Encoding and search key generation without layering



Observation: While encoding n arbitrary ranges may require n bits, only log(n + 1) bits are required to encode n disjoint ranges.

Scheme Outline

Layering Stage	How to partition the ranges to sets of disjoint ranges?	Max Size Independent Sets Max Size Colorable Sets Max Weight Independent Sets Max Weight Colorable Sets
↓		
Bit Allocation Stage	How many bits to give to each layer?	Bit Auction
↓		
Encoding Stage	How to encode the rules and search keys?	Encode

All ranges that are not encoded using the extra bits are encoded with multiple entries in a fall back scheme. The objective is to minimize the total number of such redundant entries.

Layering Stage

The layering is equivalent to coloring an interval graph. Thus we need to solve the following problems:

> Let G be an interval graph. The *minimum space layered interval-code* (MLIC) problem is to find a legal coloring $\mathcal{C}$ of G that minimizes $\sum_{i=1}^{|\mathcal{C}|} \log_2(n_i + 1)$, where n_i is the number of the nodes of G assigned color i by $\mathcal{C}$.

We consider also a budgeted version (BMLIC), which aims at finding the best (partial) layering, given the number of available extra bits.

We show that both problems are NP-Hard and provide 2-approximation algorithm for MLIC using maximum size colorable sets. We then propose four different heuristics to solve this problem in practice.

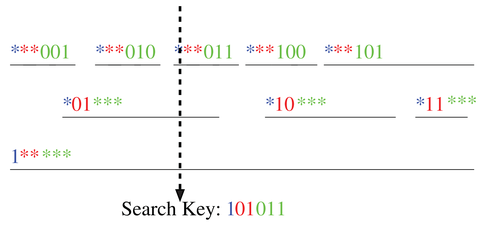
Bit Allocation Stage

The iterative algorithm **Bit Auction**. Each iteration is an auction, in which layers compete for the next available bit: If a layer L_i has already been assigned x_i bits, then assigning it additional k bits allows encoding L_i's next $2^{x_i+k}-2^{x_i}$ intervals. According to this we compute the per bit decrease in redundancy gained by allocating the next k bits to each layer and assign the next bit to a layer for which this quantity is maximal.

Encoding Stage

- **Determining the code of each range:** The ranges are encoded in decreasing weight order within each layer, according to the number of bits allocated to the layer. The extra bits corresponding to other layers are set to '*'. If a range fall out of the bit budget of the layer, all extra bits are set to '*' and a fall-back scheme is used.
- **Encoding rules with ranges:** If the rule's range is encoded – set '*' in its original field and use the extra bits determined in previous step. Otherwise, use fall-back scheme and set '*' in all extra bits.
- **Encoding Search Keys:** The value of the extra bits is determined by concatenating all the codes of the ranges the entry intersect and filling with 0-bits the bits corresponding to layers in which the entry does not intersect any range. The rest of the search key is unchanged.

Encoding and search key generation with a layered interval code



Experimental Results and Comparative Analysis

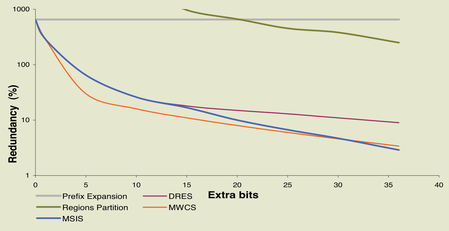
- Settings: Real-life database, which is a collection of 120 separate rule files originating from various applications; approximately 223,000 rules that contain 280 unique ranges. 28% of the rules contain range fields and about half of these include the range [1024, 216-1].
- Results: Either one of the four layering algorithms we analyze reduces the redundancy factor by 50%-70% (depending on the number of available extra bits) as compared with DRES, which is the best prior art range encoding algorithms. In the typical IPv4 TCAM-based classification databases, when 36 extra bits are available our scheme reduces the redundancy factor by either 62.2% or 67.8%, depending on whether the fall-back scheme used is prefix expansion or SRGE coding, respectively.

Algorithm	Expansion	Redundancy
Prefix Expansion	2.68	6.53
Region Partitioning	1.64	2.51
hybrid DIRPE	1.2	-
hybrid SRGE	1.03	1.2
DRES	1.025	0.09
LIC/MSIS and Prefix Expansion	1.0076	0.029
LIC/MSIS and SRGE	1.0061	0.024
LIC/MSCS and Prefix Expansion	1.0088	0.034
LIC/MSCS and SRGE	1.0075	0.029
LIC/MWIS and Prefix Expansion	1.0089	0.034
LIC/MWIS and SRGE	1.0074	0.029
LIC/MWCS and Prefix Expansion	1.0088	0.034
LIC/MWCS and SRGE	1.0074	0.029

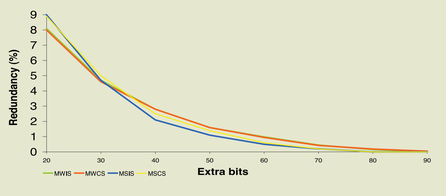
Expansion and redundancy factor, using 36 extra bits, for different range encoding algorithms

Layering Algorithm	Number of Required Bits
LIU	235
LIC/MWCS	96
LIC/MWIS	94
LIC/MSCS	86
LIC/MSIS	85
Optimal MLIC Solution	85

The number of bits required to encode all ranges that occur in our database as a function of the layering algorithm employed



Redundancy factor as a function of the number of extra bits for different encoding schemes.



Redundancy factor as a function of the number of extra bits for different layering algorithms.