A Memory Efficient and Modular Approach for String Matching

Hoang Le, Edward Yang, Viktor Prasanna
Ming Hsieh Department of Electrical Engineering - University of Southern California
{hoangle,yeyang,prasanna}@usc.edu

USC Viterbi School of Engineering

Background

- Multi-pattern, large-scale string matching
- Return **all** occurrences, if any, of **every** pattern in the given input string at **any** position.
- Memory efficiency (in bytes/char)
- Throughput (in Gbps)

Contributions

A **novel architecture** for large-scale string matching:

- Simple architecture
- High memory efficiency (~ 1 B/char)
- High throughput (3.2 Gbps)
- Modularity

Our Approach – Binary Search Tree

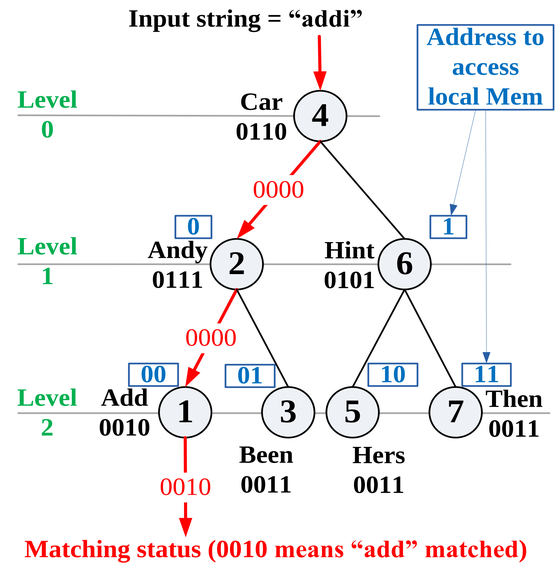
- Height of a complete BST bounded by $\lceil \log N \rceil$ (N - number of nodes)
- Amount of memory **doubled** in the next level
- **No need** to explicitly store the addresses of child nodes
- Last few stages moved onto **external SRAMs**

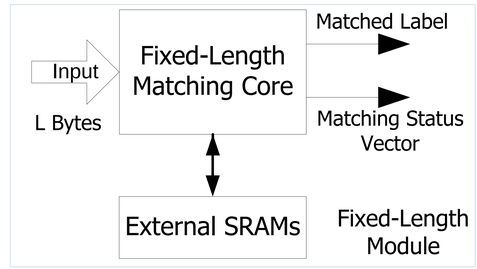
Pattern
An
And
Andy
Add
Bee
Been
The
Then
Her
Hers
Hi
Hint
Car
Ca

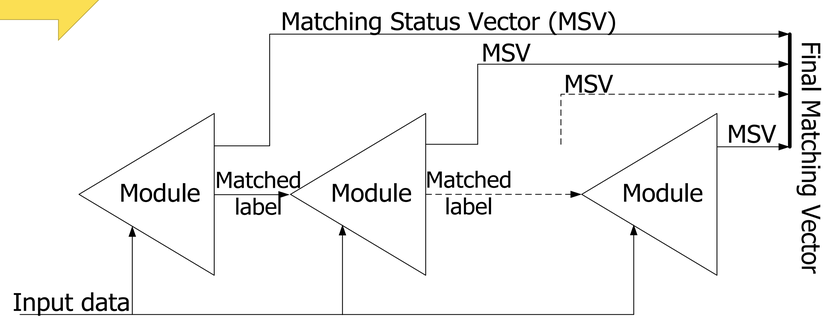
Merging

Parrent Pattern	Bitmap	Node
Andy	0111	2
Add	0010	1
Been	0011	3
Then	0011	7
Hers	0011	5
Hint	0101	6
Car	0110	4

Mapping







Data Structure

- **Overlapping resolved by merging** prefix-patterns with parent patterns ("an", "and" merged into "andy")
- **Bitmap vector** keeps prefix-patterns , a "1" at position *i* shows a prefix of length *i* (ex. "andy_0111")
- **Each node** consists of a parent-prefix and its bitmap vector
- **Standard** BST traversal:
  - Start from root
  - Go left if less than or equal, otherwise right

Modularity

- **Module used as building block** to process long patterns
  - Long patterns are cut into segments of length L
  - Matched label and Matching Status Vector carried to the next module to ensure valid long patterns
- **Variable** module width (L)
- **Duplicated** to improve throughput

Experimental Results

Architecture	Memory Efficiency (B/char)	Throughput (Gbps)
Our work / Rogets	1.07	3.2
Our work / Snort	1.05	3.2
Field-Merged / Rogets	6.33	1.14
Field-Merged / Snort	2.16	1.14
Bit-Split / Rogets	34.1	1.76
Bit-Split / Snort	28.5	1.76
Variable-Stride / Snort	2.4	N/A

Rogets dictionary 178K, maximum length = 24
Snort dictionary 146K, maximum length = 232