English

Research Express@NCKU Volume 31 Issue 4 - March 3, 2017
[http://research.ncku.edu.tw/re/articles/e/20170303/8.html]

A multi-index hybrid trie for IP lookup and updates

Chia-Hung Li[1], Chia-Yin Hsu[1], Sun-Yuan Hsieh[1,2,*]

[1] Department of Computer Science and Information Engineering, National Cheng Kung University
[2] Institute of Manufacturing Information Systems, National Cheng Kung University

hsiehsy@mail.ncku.edu.tw

IEEE Transactions on on Parallel and Distributed Systems, vol. 25, no. 10, pp. 2486-2498, October 2014.

High-performance routers require high-speed IP address lookup to achieve wire-speed packet forwarding. This study proposes a new data structure, the Multi-Index Hybrid Trie (MIHT), for dynamic router table designs. This data structure was constructed by combining the useful characteristics of the B^+ tree and priority trie. Figure 1 is an example.

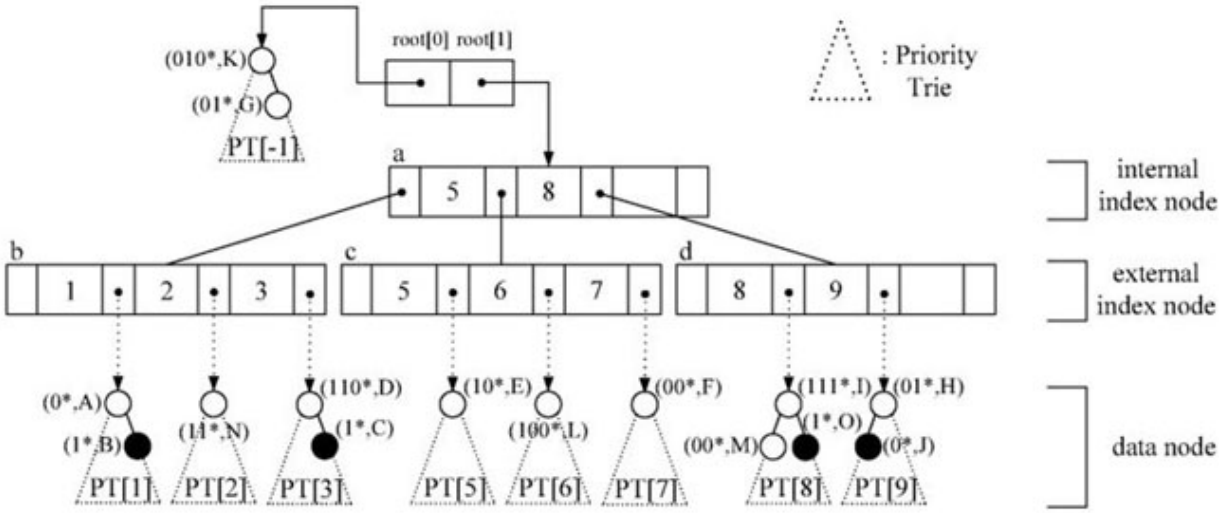

Figure 1. (4,4)-MIHT

IP lookup operations can be performed efficiently by associating each prefix with a key value in the MIHT. Furthermore, because the required tree height and number of prefixes were reduced, dynamic router table operations were performed efficiently using the MIHT. To reduce the memory requirement, each prefix stored its corresponding suffix in a node of the MIHT, rather than storing a full prefix. Experiments using IPv4 and IPv6 routing databases indicated that the proposed data structure has efficient memory usage and performs well for lookup, insertion, deletion operations. This study reports the results of the experiments performed to compare the proposed data structure with other structures using the benchmark IPv4 and IPv6 prefix databases AS1221, AS4637, AS6447, AS65000, AS1221*, and AS6447* with 407,067, 219,581, 417,995, 406,973, 12,155, and 12,278 prefixes, respectively, where AS1221* and AS6447* are IPv6 BGP routing tables. Tables 1-3 are the performance comparison for the various data structures.

Table 1. Average Tree Heights and Storage Comparisons of the Various Data Structures

Data Structure	Avg. Tree Height				Storage (KB)			
	AS1221	AS4637	AS6447	AS65000	AS1221	AS4637	AS6447	AS65000
Binary Trie	32	32	32	32	11,519	6,731	11,870	11,513
LC-Trie	23	21	25	23	13,772	8,266	14,011	13,762
Prefix Tree	29	29	32	29	6,360	3,430	6,531	6,358
Priority Trie	28	27	32	29	6,360	3,430	6,531	6,358
DTBM	10	10	10	10	25,291	14,506	25,997	25,285
4-MPT	11	10	10	10	21,484	11,391	22,049	21,501
4-PCMST	10.08	9.72	10.13	10.04	22,172	12,463	23,051	22,043
(16,16)-MIHT	8.04	7.66	8.07	8.04	6,883	3,778	7,064	6,881
(16,16)-PMIHT	4.11	3.47	4.16	4.11	6,875	3,776	7,056	6,873

Table 2. Performance Comparisons for Lookup for the Various Data Structures

Data Structure	Avg. Lookup Time (# of Clock Cycles)				Avg. # of Memory Accesses			
	AS1221	AS4637	AS6447	AS65000	AS1221	AS4637	AS6447	AS65000
Binary Trie	2,990	2,684	3,018	3,000	23.56	23.40	23.59	23.56
LC-Trie	3,006	2,857	3,020	3,015	9.25	9.75	9.26	9.24
Prefix Tree	3,206	2,793	3,215	3,206	21.71	21.31	21.72	21.71
Priority Trie	2,864	2,481	2,872	2,862	20.09	19.65	20.09	20.09
DTBM	2,094	1,912	2,091	2,114	8.34	8.29	8.35	8.34
4-MPT	2,878	2,509	2,850	2,866	15.69	14.71	15.61	15.70
4-PCMST	2,012	1,805	2,112	2,151	10.16	9.52	9.57	9.16
(16,16)-MIHT	2,222	1,920	2,227	2,221	9.51	9.17	9.51	9.51
(16,16)-PMIHT	1,954	1,657	1,935	1,953	8.31	8.05	8.30	8.30

Table 3. Performance Comparison for Updating for the Various Data Structures

Data Structure	Avg. Update Time (# of Clock Cycles)				Avg. # of Memory Accesses			
	AS1221	AS4637	AS6447	AS65000	AS1221	AS4637	AS6447	AS65000
Binary Trie	4,138	4,021	4,216	4,181	26.55	26.70	26.59	26.54
Prefix Tree	2,445	2,395	2,426	2,438	20.87	20.33	20.77	20.85
Priority Trie	3,834	3,830	3,831	3,849	21.42	21.00	21.43	21.42
DTBM	2,755	2,705	2,749	2,754	8.90	8.91	8.91	8.90
4-MPT	2,144	2,419	2,178	2,277	6.24	6.26	6.21	6.24
4-PCMST	1,345	1,517	1,079	1,153	4.04	4.05	3.02	3.24
(16,16)-MIHT	1,907	1,780	1,853	1,900	8.90	8.70	8.90	8.90
(16,16)-PMIHT	1,679	1,693	1,622	1,679	7.56	7.46	7.56	7.57