Cache-aware data structures for packet forwarding tables on general purpose CPUs: project description

Maksim Yegorov

2018-02-13 Tue 01:47 PM

context

- network router receives a packet
- inspects destination ip
- looks up by prefix
- forwards to outgoing interface for next hop

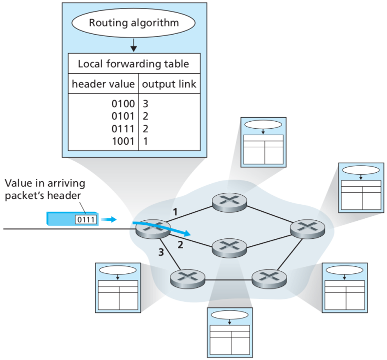


Figure 1: forwarding; from Kurose & Ross, *Computer Networking*

constraints

- time: key-value lookup
- space: table encoding
- hardware cost, power efficiency etc.

estimate throughput demand

ballpark estimate:

- $\approx$ 50 Gbps router $\equiv$ 75 Mpps
- varying prefix length, conservatively estimate x32
- $\rightarrow$ 2.5e9 lookups per second
- @4GHz single CPU $\rightarrow$ 50 CPU cycles per packet
- ignore other overhead (routing table etc.) as performed by dedicated CPU/memory

FIB table size

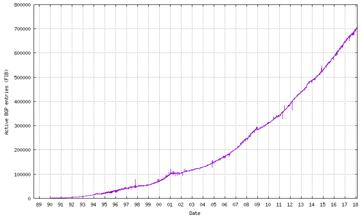


Figure 2: IPv4 table; source: http://bgp.potaroo.net/

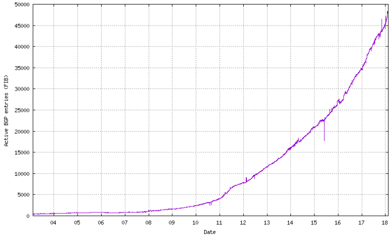


Figure 3: IPv6 table

time and space requirements

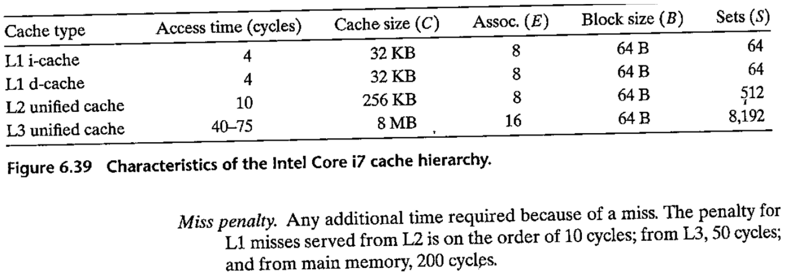
space

@ 100B per key-value pair, 1e6 entries $\rightarrow$ 100MB hash table
(vs. 4MB Bloom filter)

time

- @10 CPU cycles per byte of hash $\rightarrow$ 40 cycles / 32b hash
- fetching from memory: expensive

cache loads and misses



Cache type	Access time (cycles)	Cache size (C)	Assoc. (E)	Block size (B)	Sets (S)
L1 i-cache	4	32 KB	8	64 B	64
L1 d-cache	4	32 KB	8	64 B	64
L2 unified cache	10	256 KB	8	64 B	512
L3 unified cache	40–75	8 MB	16	64 B	8,192

Figure 6.39 Characteristics of the Intel Core i7 cache hierarchy.

Miss penalty. Any additional time required because of a miss. The penalty for L1 misses served from L2 is on the order of 10 cycles; from L3, 50 cycles; and from main memory, 200 cycles.

Figure 4: from Bryant & O'Hallaron, *Computer Systems*

. . . *ouch*!

existing solutions

- well researched area
- traditionally the lookup is performed in highly optimized hardware, using compactly encoded datastructures (trie / Bloom filter and derivatives)
- TCAM: mask and compare incoming packet in parallel for all prefix lengths
- SRAM: a couple orders of magnitude cheaper than TCAM

recent proposals

- state of the art multithreading, algorithms and vector packet processing make forwarding feasible on commodity hardware
- advantages: cost, added flexibility/programmability, lower barrier to entry (not requiring advanced manufacturing capabilities)
- **bottleneck**: memory access
- challenge: how is it even possible?

cache optimization strategies

- design for sequential reads & writes
- minimize misses by prefetching in cache strategically
- helper threads
- datastructures designed for concurrency
- arrange elements with memory alignment in mind
- split tables into long-tail low-fidelity and short-tail high-fidelity:
  - avoid having the more frequent prefix lengths require additional false-positive test
  - design split table size to (partially) fit in L3 cache

plan

- **milestone 1**: review data structures, memory hierarchy & cache optimization literature
- **milestone 2**: implement filters (Bloom, cuckoo) and hash table (low- vs. high-fidelity versions) with and without cache optimization
- **milestone 3**: analyze patterns of cache hits and misses, run tests, and benchmark