Scalable, High Performance Ethernet Forwarding Lookup

Dong Zhou[1†], Bin Fan[1†], Hyeontaek Lim[1†], David G. Andersen[1], Michael Kaminsky[2]

[1]Carnegie Mellon University, [2]Intel Labs, [†]Student Author
{dongz,binfan,hl,dga}@cs.cmu.edu, michael.e.kaminsky@intel.com

This poster will describe a new Ethernet forwarding lookup architecture that enables the simple construction of switches that can contain tens of millions of forwarding entries. The core design is based upon recent work on concurrent multi-reader cuckoo hashing [1] to create fast and memory-efficient lookup tables. This work has two primary benefits: First, it suggests that were a need for such scale to arise—whether for flat addressing or for more extreme uses such as content-based networking—such huge-scale tables could be practically constructed. Second, as a pragmatic byproduct, the technique allows storing several hundred thousand forwarding entries entirely in fast L3 CPU cache, facilitating the construction of high-throughput software-based switches.

Why Big, Fast Tables? Recent technology advances, such as scalable enterprise networks [3] and data center networks [2, 5], have made much larger layer-2 networks a reality. Such ever-larger flat networks require Ethernet switches to store an increasing number of entries in the forwarding tables. At the same time, line speeds are increasing to 10G and 40G.

Current Techniques won't do High-speed memories (e.g. TCAM) are widely used by hardware switches to implement high performance forwarding tables. However, their small size severely limits scalability. For example, the Mellanox SX1016 64-Port 10GbE Switch can only support 48K layer-2 forwarding entries. Some approximate solutions are very memory efficient, e.g. BUFFALO [6], but we seek instead a solution that provides exact matching, thus inducing no path stretch.

MAC addresses can also be mapped to outging port using a hash table. However, prior hashing schemes suffer from either memory inefficiency and/or unacceptable lookup performance to handle collisons, which makes them much less attractive.

Solution Overview To address the aforementioned challenges, we use *optimistic cuckoo hashing* [1] to achieve high memory efficiency (94% table occupancy [4]) and concurrent access to the hash table with read-intensive workloads.

The high performance of our cuckoo hash table comes in part from that each forwarding lookup issues only 2 *parallel* cacheline-sized reads. Each read fetches a bucket consisting of four 64-bit slots, where each slot contains the full 48-bit MAC address along with a corresponding 16-bit outgoing port identifier.

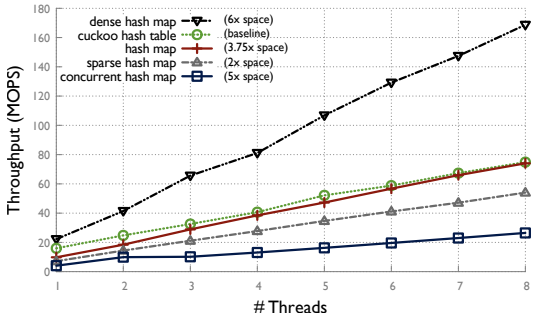

Figure 1: Lookup performance in a table of 1.5M entries

Preliminary Results Figure 1 compares the space consumption (listed next to the key) and throughput of our hash table with four other popular hash tables: three non-thread-safe hash tables (the STL hash map and Google's sparse and dense hash maps) and one thread-safe (Intel TBB concurrent hash map). Our implementation uses substantially less memory than any other scheme, and is faster than all but the non-thread-safe dense_hash_map (used read-only with multiple threads).

References

[1] B. Fan, D. G. Andersen, M. Kaminsky. MemC3: Compat and Concurrent MemCache with Dumber Caching and Smarter Hashing. In *Proceedings of the 10th USENIX conference on Networked Systems Design and Implementation*. 2013.

[2] A. Greenberg, et al. VL2: a scalable and flexible data center network. In *Proceedings of the ACM SIGCOMM 2009 conference on Data communication*. 2009.

[3] C. Kim, M. Caesar, J. Rexford. Floodless in SEATTLE: A Scalable Ethernet Architecture for Large Enterprises. In *Proceedings of the ACM SIGCOMM 2008 conference on Data communication*. 2008.

[4] H. Lim, et al. SILT: A Memory-Efficient, High-Performance Key-Value Store. In *Proceedings of the 23rd ACM Symposium on Operating Systems Principles (SOSP)*. 2011.

[5] R. Niranjan Mysore, et al. PortLand: A Scalable Fault-tolerant Layer 2 Data Center Network Fabric. In *Proceedings of the ACM SIGCOMM 2009 Conference on Data Communication*. 2009.

[6] M. Yu, A. Fabrikant, J. Rexford. BUFFALO: Bloom Filter Forwarding Architecture for Large Organizations. In *Proceedings of the 5th International Conference on Emerging Networking Experiments and Technologies*. 2009.