IP CACHING FOR TERABIT SPEED ROUTERS

Bryan Talbot, Timothy Sherwood, Bill Lin
University of California San Diego
9500 Gillman Dr., La Jolla, CA 92093
{btalbot,tsherwoo,billlin}@ucsd.edu

Abstract

As network speeds continue to grow, current methods of translating destination IP addresses to output port numbers during routing become inadequately slow. Eventhough this lookup is often performed in hardware, it may still be limited by DRAM access latencies. We present a method of speeding up access to DRAM based lookup mechanisms by more than a factor of 10 using CPU style memory caches. Real IP router traces are used to validate the caching scheme as well as study some IP address properties that may affect caching performance.

1 Introduction

An IP router must perform an IP routing table lookup every time a packet is to be forwarded to the next hop. This lookup table serves to translate the destination IP address into an output port number onto which the packet is to be transmitted.

A possible naive approach is to simply enumerate the entire table in memory, however with IPV4 there are 2^{32} possible IP address, and with the advent of IPV6 this number increases to 2^{128}. Even the smaller of these address spaces is too large to simply enumerate, thus clever methods are needed.

Another point to note is that the IP lookup mechanism lies directly on the critical path for IP routing, and as networks begin to reach terabit speeds the IP lookup time can quickly become a bottleneck. A terabit capable router must be able to make over two-hundred million lookups per second assuming IP packet sizes of 500 bytes. This means that IP lookups must be completed on the order of nanoseconds rather than hundreds of nanoseconds. The use of DRAMs for table lookup in the common case ceases to be feasible at these speeds. This is due to the fact that DRAM access speeds are on the order of 50 ns and decrease only a few nanoseconds per generation. Indeed at terabit speeds even going off chip without the use of highly tuned bus transceivers is unlikely to be realistic. However by changing some basic assumptions we may be able to avoid both of the above problems.

We believe that destination IP address lookup speeds can be increased by taking advantage of temporal locality of the IP addresses. If an address is referenced, it will likely be used again soon. We attempt to show that real packet traces exhibit temporal locality of their destination addresses, and that this locality can be exploited to provide very fast IP lookups in real network routers. To take advantage of the temporal locality, we propose specialized caches to capture commonly referenced addresses for fast lookup. We examine the timing ramifications of our cache design using the cache simulator "Dinero IV" [3] as well as show some interesting characteristics of large packet streams.

The rest of the paper is laid out as follows: section 2 is an overview of related IP routing work and section 3 is a description of our proposed scheme. Section 4 describes our experimental methodology including trace descriptions. Sections 5 and 6 present the results, and section 7 draws final conclusions.

2 Related Work

There has been a great deal of work done towards high speed routers as of late [11][6][5][2], some of which was directed at fast IP lookup. Two of the schemes which IP Caching draws heavily upon are the Stanford Lookup scheme [4] and IP Switching [10].

2.1 Stanford Lookup

The Stanford lookup scheme [4] is built on the idea that when doing a histogram of the actual sizes of bits used

in the routing table, only the least significant 24 bits of the 32 bit IP address are used in 99.3% of the routing table. The authors argue that by simply enumerating those 2^{24} possible entries in DRAM, which would only take approximately 16 million entries, the common case could be satisfied with a single DRAM access and little other overhead. This method works quite well for current networks with 32 bit address and at slow enough speeds, however the Stanford method is quite limited by the fact that one DRAM access per lookup is required.

2.2 IP Switching

Lin and McKeown present work relevant to IP Caching in the form of arguments for IP switching [5]. The basic idea of IP switching is to characterize the stream of IP packets into flows, which can then be assigned, to virtual circuits. To make this work two criterion must hold true: a significant portion of the packet stream must be comprised of logical flows, and the number of flows has to be small enough such that the flows can be stored efficiently on the routers. Lin and McKeown go on to show that this solution may be feasible for several packet traces. However we claim that if the above two conditions hold true then there may be a simpler way to speed IP lookup, simply cache the IP address.

3 IP Caching

We propose that by building a special IP lookup cache we can increase the throughput and decrease the latency of the lookup operation in the Stanford scheme. Lookup requests would be checked in the cache before an access out to memory. One interesting thing to note is that since there is no spatial locality in the packet stream, large cache line sizes give you nothing in return for their increased access time. Because of this we use only one entry per cache line, and a cache line size of 1 byte.

Upon access to the cache, the IP address is split (as a virtual address would be in a microprocessor) into two parts: the indexing bits and the tag bits. The cache works in the traditional way, the indexing bits are used to address a cache line and the tag bits are used to verify that the indexed entry is the one that was being searched for. We investigate tag and index bit choices in the results section.

Site	Description
ANL	Argonne National Laboratory to STARTAP
FLA	Florida universities GigaPOP
FXW	FIX-West facility at NASA-Ames (FDDI)
MRT	Michigan universities (Merit)
NCL	NC Networking Initiative NCNI GigaPOP
NRN	NASA's NREN connection at the AIX
ODU	Old Dominion University
OSU	Ohio State University
SDC	SDSC commodity connection

Table 1: Packet trace source locations.

Site	Packets	Unique IPs	Duration (s)
ANL	1,371,854	2,564	1,649
FLA	3,294,418	2,323	3,601
FXW	43,080,646	162,839	3,599
MRT	58,980,772	101,352	3,600
NCL	6,919,914	6,196	3,602
NRN	33,600,336	1,208	3,602
ODU	4,808,121	10,070	3,600
OSU	32,765,590	57,514	3,598
SDC	36,044,403	73,816	3,600

Table 2: Packet trace sizes.

4 Methodology

In order to evaluate caching performance we obtained packet traces containing real IP numbers from busy Internet routers from around the United States. Real packet traces were necessary because randomly generated IP numbers would not exhibit the same distribution of bit values in the address nor would it exhibit any locality.

The destination IP addresses were then used, rather than traditional memory references, as input to a caching simulator to compare the performance of various cache configurations.

4.1 Packet Traces

Since cache effectiveness is based on the assumption that IP destination addresses seen by a backbone router have specific properties, we needed router traces containing real IP addresses to confirm those properties and the validity of our caching scheme. There were two properties we were most interested in investigating. First we wished to find out which bits of the destination IP address are the most "volatile". That is, which bit positions change the most from address to address. Secondly, the temporal locality of the addresses should be tested. If a high degree of such locality does

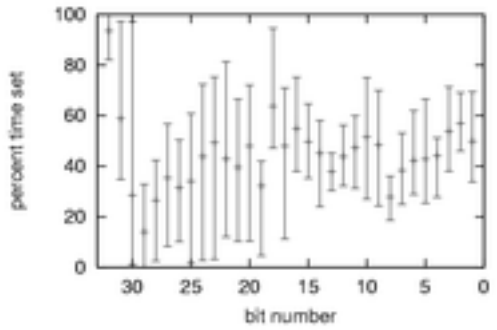

Figure 1: Average bit values.

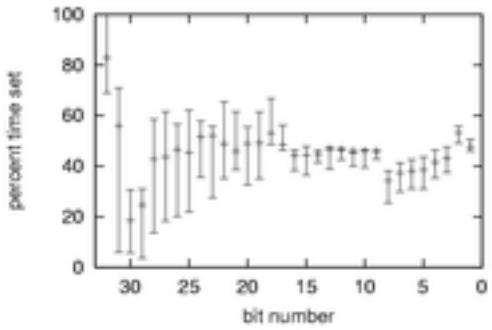

Figure 2: Average bit values for unique addresses.

indeed exist, then caching the destination IP lookup tables should be quite effective.

To this end we contacted the National Laboratory for Applied Network Research [7] and arranged access to such traces. They provided us with long traces ranging in length from nearly $\frac{1}{2}$ hour to over 1 hour collected from nine routers from around the U.S. Tables 1 and 2 summarize the sites and traces used.

4.2 Caching Simulation

We first formatted the original router trace files to conform to an input format usable by Dinero with the IP destination address in place of memory address. We first tested 21 L1 cache configurations ranging in size from 4K to 256K entries each with three different associativity levels including direct-mapped, 2, and 4 way. One and two megabyte L2 caches with direct-mapped, 2-way, 4-way, and full associativity were then simulated with all the above L1 caches in an attempt to obtain optimum hit rates.

5 Results

5.1 Average bit values

The first series of tests performed were designed to measure the average values of the bits distributed in the IP address. This measurement could help determine if some of the bits in the address would be better to index a cache with or to use as tags. The best indexing bits should be those with an average value of $\frac{1}{2}$; meaning that over a large series of addresses, they are set 50% of the time. Indexing with these bits should help to spread references more evenly across the cache minimizing conflict misses. Every address in each trace file was checked. The average bit values were calculated over the entire stream of references and also for only those references which were unique. Figure 1 plots the range of average bit values for the entire reference stream. All nine traces' average bit values were computed separately and then an overall average value covering all the traces was computed. The plot shows the high and low average values; the overall average value is marked on each vertical line.

The average bit values vary quite a bit for most bits and vary quite wildly for several which indicates that the average bit value seen by each router tends to be dominated by the networks it is connected to.

The graph in Figure 2 shows the average values for only the unique addresses found in each trace. The average value of the lower 16 bits of the address vary much less from router to router than the non-unique values do. The high bit values still vary quite a bit, but much less than before. Since most of the lower bits appear to have the most desirable and tightest average value, we used the bits as they appeared in the IP addresses in the cache simulations with the lowest bits indexing the cache.

5.2 Caching simulations

The L1 cache miss rates for four of the traces are plotted in Figure 3. The x-axis plots cache size which varies from 4KB to 256KB and the y-axis is the miss rate. There are three plots per graph with the topmost plot (having the worst hit rate) being for a direct mapped cache. The middle plot is a 2-way set associative cache. The best hit rates are achieved using a 4-way set associative cache.

Several of the traces had such a small number of unique IP address in them that the hit rates were exceptionally high even on the smallest, 4K entry, caches we simulated. Those plots are not shown since they are

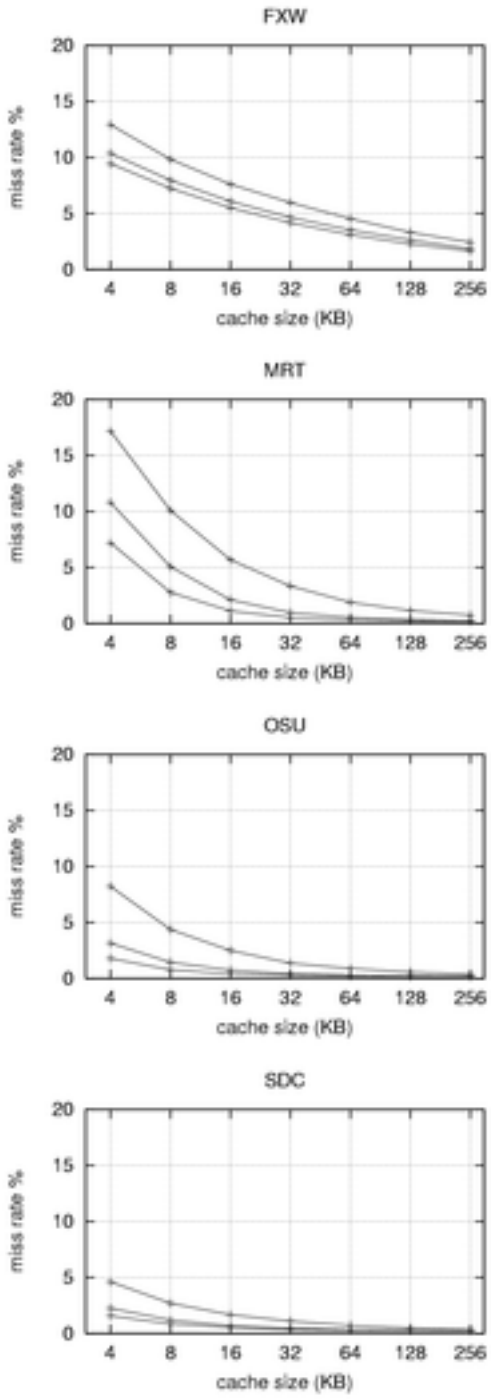

Figure 3: L1 cache miss rates for FXW, MRT, OSU, and SDC using direct mapped, 2-way and 4-way caches

Site	no cache	128K L1	4K L1	L1+L2
ANL	55	2.11	2.32	2.13
FLA	55	2.04	2.16	2.06
FXW	55	3.40	8.85	3.28
MRT	55	2.17	11.11	3.45
NCL	55	2.06	2.82	2.16
NRN	55	2.00	2.03	2.01
ODU	55	2.12	2.65	2.19
OSU	55	2.12	6.36	2.74
SDC	55	2.15	4.45	2.46

Table 3: Average IP lookup times in nanoseconds

so flat. All of these locations display a high degree of temporal locality; at least for the duration of the traces we have.

The miss rates do steadily decrease with increasing cache sizes as expected. This fact suggests that caching IP destination addresses scales well with cache size. A heavily aggregated site with less temporal locality in the IP destinations could improve cache hit rates by using larger cache sizes. Since cache memory densities have historically grown at a rate similar to network traffic growth rates (doubling in size every 18 months) this should be a maintainable improvement into the future.

6 IP Lookup Performance Times

Realistically L1 cache sizes and associativity are limited by cycle time. We found several sources for L1 caches in the 128K, 2-way, 2ns cycle time, 1-cycle latency range. Table 3 shows calculated average IP lookup times, in nanoseconds, for several different lookup configurations. In each case, the main memory DRAM is assumed to be organized using the Stanford scheme from.[4] The "no cache" column gives access times if a DRAM access is required for every lookup. The next two columns show access times if a high-performance 128K or 4K entry, 2-way, 2ns cycle time with a 1-cycle latency L1 cache were used in front of the DRAM stage. The last column attempts to demonstrate the effectiveness of a large fast 1M entry L2 cache for improving access times even when the 4K L1 hit rates are low. Low for this data however is still over 80% for even the smallest 4K caches, so the effect of the L2 cache may not be as dramatic as it could be with different reference streams.

The cache timing we used for the L1 caches are from [8] and L2 cache timings are from the Intel Xeon CSRAM modules as reported in [1] and [9]: 2ns (and less) cycle times, 5 cycle latency, 4-way set associative.

7 Conclusions and Comments

Our experiments demonstrate that caching of destination IP addresses is indeed an effective speedup to IP lookup in high-speed routing. The traffic patterns of the nine sites we were able to obtain test data for all responded well to our caching strategy. Hundreds of millions of packets over several hours of time were analyzed. In each case the lowest bits of the IP address demonstrated the most randomness making them well suited to index a cache. On top of that, the scheme would appear to scale well for use on sites with higher aggregation levels of destination addresses. The sites we tested all showed impressive speedups using small 4K entry caches. This suggests that if all nine of these networks were concentrated at one router at most $9 \times 4k$ cache entries would be necessary to maintain similar performance.

Non-real world worst case traffic streams, such as sequential non-repeating destination addresses, would show no gain using this method; although, the overhead imposed by the presence of the L1 cache in these cases would be a small fraction compared to the DRAM access time. The presence of the cache has shown a 10x or better speed improvement when used with all of the real packet streams we've tested it with.

The best performing method we've seen to date, the Stanford method, required at least one DRAM cycle (55ns) plus off chip delays to perform this task. The caching we propose would supplement that method, or most any other DRAM based method, and shows sizable speed increases. With an average IP lookup time of well under 4ns, a lookup engine could support 250 million lookups per second. Even the occasional worst case performance is barely greater than 1 DRAM access. Assuming average packet sizes of 500 bytes per packet, this lookup scheme could support speeds exceeding 1Tbps.

Acknowledgments

We would like to thank Hans-Werner Braun of NLANR for providing us with packet traces and for providing us with the machine time to perform the data analysis.

References

[1] "Intel Bets On Custom Cache Device For Slot 2 Platform", Anthony Cataldo, EE Times, February 11, 1998

[2] "Small Forwarding Tables for Fast Routing Lookups", Mikael Degermark, Andrej Brofnik, Savante Carlsson, and Stephen Pink., ACM SIGCOMM 1997

[3] Dinero IV Cache Simulator 1997 NEC Research Institute, Inc. and Mark D. Hill.

[4] "Routing Lookups in Hardware at Memory Access Speeds", Pankaj Gupta, Steven Lin, Nick McKeown, Computer Systems Laboratory, Stanford University

[5] "A Simulation Study of IP Switching", Steven Lin and Nick McKeown, From ACM Sigcomm'97.

[6] "Scalable High Speed IP Routing Lookups", Marcel Waldvogel, George Varghese, Jon turner, and Bernard Plattner, From ACM Sigcomm'97

[7] National Science Foundation Cooperative Agreement No. ANI-9807479, and the National Laboratory for Applied Network Research, http://nlanr.net/

[8] "A 2ns Access, 285 Mhz, Two-Port Cache Macro using Double Global Bit-Line Pairs", Ken-ichi Osada, Hisayuki Higuchi, Koichiro Ishibashi, Naotaka Hashimoto, Kenji Shiozawa. From IEEE International Solid-State Circuits Conference (ISSCC) 1997.

[9] "Intels Xeon for Workstations and Servers", Andreas Stiller, c't 14/98, page 162, Translation by Sabine Cianciolo

[10] "Scalable High-Speed IP Routing Lookups.", M. Waldvogel, G. Varghese, J. Turner, B. Plattner, Procedures ACM SIGCOMM 1997, pp. 25-36

[11] "Routing on Longest-Matching Prefixes", Doeringer, Willibald, Karjoth, Gunter, Nassehi, Mehdi, IEEE ACM Transactions On Networking, Vol. 4, No. 1, pp. 86-97, February 1996.