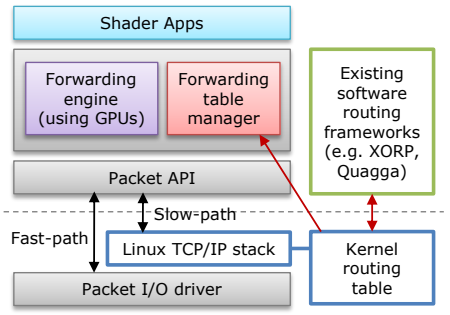
Dynamic Forwarding Table Management for High-speed GPU-based Software Routers

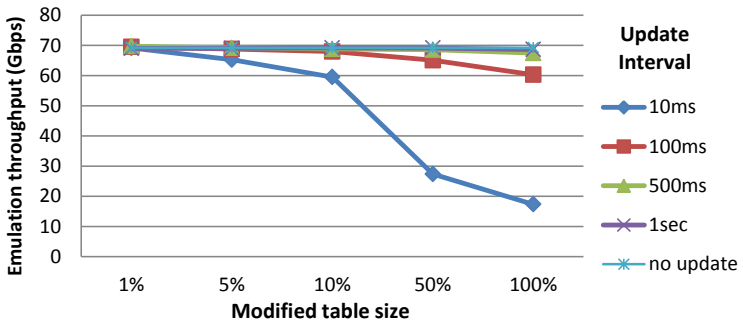
Joongi Kim[*†], Keon Jang[*†], Sangjin Han[*], KyoungSoo Park[**], and Sue Moon[*]

[*]Department of Computer Science, KAIST, {joongi, keonjang, sangjin}@an.kaist.ac.kr, sbmoon@kaist.edu
[**]Department of Electrical Engineering, KAIST, kyoungsoo@ee.kaist.ac.kr
[†]Students

Software routers are gaining momentum for its extensibility and programmability in network packet processing. PacketShader has pushed the performance limit of software routers to 40Gbps on a single machine, utilizing the massive parallel computation power of modern GPUs. The next step is to integrate the control-plane into it. The most crucial part to implement the control-plane is to minimize performance degradation to the data-plane. Today's routers typically maintain large routing tables (over 320,000 entries) and often they have to sustain the updating frequency of 50-150 times per second. Forwarding information bases (FIBs), as known as forwarding tables, are generated from routing tables for faster next-hop lookups, and they usually reside in the linecards in traditional hardware routers. The delay from a link failure or a BGP update to an FIB update can be sometimes seconds or up to minutes and affects a network's performance greatly. The same FIB updating problem exists in GPU-based software routers. The forwarding table resides in the GPU's device memory for the GPU kernels to perform the IP address lookups. Figure 1(a) is the software architecture of PacketShader. For the control-plane integration, we add a forwarding table manager on the GPU side and it pulls most updated routing information from the kernel routing table. Figure 1(b) is the updating costs of the forwarding table in the GPU's device memory. It shows that the forwarding table updates may cause performance fluctuation or degradation in bursty situations. The technical challenge here is to update the forwarding table in GPUs, without compromising the data-plane performance under dynamic and bursty traffic.



(a) Software architecture of PacketShader with the control-plane integrated



(b) Updating cost of the forwarding table in PacketShader

Figure 1

Our basic idea for FIB update is to use double-buffering. A forwarding table is typically a few hundreds MB large, and modern GPUs have enough memory (e.g., 1.5GB for GTX480) for multiple copies of it. While a routing daemon affinitized for a single core updates a copy, the GPU kernels can use the other copy. We expect little or no performance degradation because the current bottleneck of IPv4/IPv6 lookup is not in the memory bandwidth or in the CPU cycles, but in the I/O buses. Another optimization point is use of incremental updates. Since we get the routing table changes from the kernel as in Figure 1(a) and those notifications include only the updated entries, it is better not to duplicate the entire routing table for less overheads. It reduces the memory bandwidth required for updating. We will evaluate the implementation with micro-benchmarks as well as BGP traffic traces.