1. INTRODUCTION

Rather than using special-purpose hardware routers, software routers enable routing on commodity platforms. However, even with faster processors and multi-core platforms, the performance of software routers on commodity platforms today does not scale with high speed.

We identify the limitations of commodity platforms by comparing them to high-end routers. In high end routers, each line card has its own memory for packet queues and the forwarding table (FIB), and its own processors to perform packet processing (e.g., forwarding table lookup). Packet processing is performed locally on line cards, and packets are forwarded to the outgoing interface through a dedicated high-speed switch. However, with a multi-core commodity platform, packets must be stored in a shared main memory and accessed by general-purpose processors. PCIe bus is used to transfer packets between main memory and line cards.

As a result, we must leverage the hierarchical memory architecture and the processing resources in multi-core commodity platforms. First, since packets are forwarded through a detour in the shared memory rather than directly between line cards, memory access becomes the main bottleneck, especially for small-size packets. (The PCIe bus becomes the bottleneck for large-size packets, which can be addressed with improved hardware techniques.) Therefore, software routers should leverage the small fast memory (cache) to improve their performance. Second, to best leverage computing resources with multiple cores, routing functions (e.g., packet forwarding, control plane) should be mapped to the appropriate place in the commodity hardware (e.g., different cores, NICs).

Besides the limitations of commodity hardware, another reason for the poor performance of software routers is the different performance requirements between conventional applications and routing. In addition to throughput, routing is also sensitive to delay and delay jitter. Moreover, routing requires more predictable behavior under a range of workloads than conventional applications, especially under worst-case workloads (e.g., a burst of packets with a wide range of destinations, or routing changes due to network events).

Based on the limitations of commodity hardware and the specific performance requirements of routing, we propose a fast packet forwarding mechanism leveraging today’s multi-core commodity platforms, which works well under worst-case workloads. We also propose enhancements for future multi-core commodity hardware to further improve the performance of software routers.

2. FAST PACKET FORWARDING DESIGN

To leverage the current multi-core platforms, we separate the control and data plane on different cores to avoid cache conflicts and context switching overhead. We also use multiple cores for data plane to give more processing resources for packet forwarding. We divide the FIB into address ranges, where each core is responsible for packet forwarding for one part of the address space. We leverage VMDq (Virtual Machine Device Queues) techniques in modern NICs to demultiplex the packets to the appropriate cores. Each core maintains a Bloom filter for each next hop, to store the set of prefixes associated with that next hop. Each Bloom filter is sized to minimize the overall false-positive rate, subject to fitting in the cache associated with each core. In the next-level cache shared amongst the cores, we store counting Bloom filters to handle routing changes and the entire FIB for forwarding packets that experience false positives. Our mechanism works well across a wide range of workloads.

To further improve packet forwarding performance (especially to reduce the memory access time of FIB and packets), we propose several enhancements to the current multi-core commodity platform. For example, NICs could have better packet demultiplexing support (such as consistent hashing) and even simple packet processing ability (such as Bloom filter lookup). Future commodity hardware could have dedicated packet lookup and forwarding cores placed near NICs, which have a larger cache per core (e.g., 1 MB) and provide APIs for routing software to specify important data structures (e.g., forwarding tables) that should be pinned in the cache.