Building a Single-Box 100 Gbps Software Router

Sangjin Han†, Keon Jang†, KyoungSoo Park, and Sue Moon†
KAIST

***Abstract*—Commodity-hardware technology has advanced in great leaps in terms of CPU, memory, and I/O bus speeds. Benefiting from the hardware innovation, recent software routers on commodity PC now report about 10 Gbps in packet routing. In this paper we map out expected hurdles and projected speed-ups to reach 100 Gbps in packet routing on a single commodity PC. With careful measurements, we identify two notable bottlenecks for our goal: CPU cycles and I/O bandwidth. For the former, we propose reducing per-packet processing overhead with software-level optimizations and buying extra computing power with GPUs. To improve the I/O bandwidth, we suggest scaling the performance of I/O hubs that limits packet routing speed to well before 50 Gbps.**

I. Introduction

Software routers are attractive platforms for flexible packet processing. While the early routers were built on general-purpose computers, they could not compete with carrier-grade routers with tens of Gbps or higher speed and gave way to specialized hardware in late 90's. With the recent advancements in PC hardware, such as multi-core CPUs, high-bandwidth network cards, fast CPU-to-memory interconnects and system buses, software routers are coming back with competitive cost performance ratio. For example, RouteBricks, a experimental software router platform, reports 8.33 Gbps, or 6.35 Gbps excluding Ethernet overheads, for IPv4 routing of 64B packets on a single PC [3]. In this paper we raise the following question: How far can we push the performance of a single-box software router with technologies from today and in the predictable future? We map out expected hurdles and project speed-ups to reach 100 Gbps on a single x86 machine.

II. Opportunities and Challenges

Recent architectural improvements from Intel and AMD have opened up new possibilities for software routers: (i) Multi-core processors extend the processing power in a scalable manner; (ii) Memory controllers integrated in CPUs provide large memory bandwidth even for many CPU cores; (iii) PCI Express (PCIe) connects high-speed peripherals, such as 10 Gbps network interface cards (NICs); and (iv) Multiple CPU sockets connected to each other via point-to-point interconnects, such as Intel QuickPath Interconnect (QPI) or AMD HyperTransport, expand the computing capacity of a single machine. Effective utilization of these resources is the key to building high-performance software routers.

Figure 1 shows one example of currently available hardware configuration for software routers. It adopts Non-Uniform Memory Access (NUMA) architecture, and has two NUMA nodes. Each node has its own I/O hub (IOH), which bridges peripherals to the CPU. QPI links interconnect CPU sockets and IOHs. Dual-port 10GbE NICs are connected to the IOH via PCIe x8 links. Six dual-port 10GbE NICs would offer 120 Gbps of maximum aggregate throughput. With this configuration, we identify and address several performance bottlenecks for scalable software routers in Sections II-A through II-C.

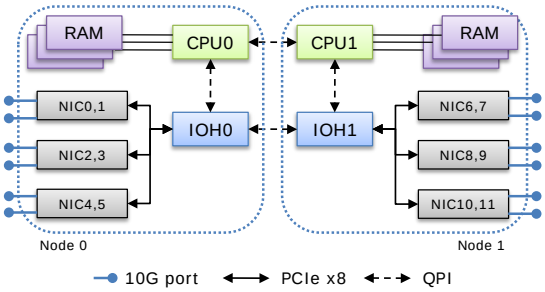

Fig. 1. Block diagram of example system configuration

Note that we only consider systems based on Intel CPUs in this work, but our discussion can easily expand to AMD-based systems.

A. CPU Cycles

Modern NICs support multiple packet queues dedicated to individual CPU cores, and thus packet processing scales well with the number of CPU cores without CPU cache pollution. However, small packets dominate the packet forwarding performance in software routers. Regardless of the packet size, a fixed number of CPU cycles is needed for the forwarding table lookup to find the destination output port. At the line rate of tens of Gbps, the per-packet processing cost poses a serious challenge even with multi-core CPUs.

RouteBricks points at the CPU as the performance bottleneck in building a 10 Gbps router. They report that 1,229 CPU cycles are needed to forward a packet from one NIC port to another NIC port. If we assume minimum-sized (or 64B packets) arriving at 100 Gbps, which translates to 149 million packets per second (Mpps), then we need 277 GHz of CPU cycles. Even with the latest Intel X7560 CPUs (eight 2.26 GHz cores in a chip) configured on four CPU sockets, we only get 72.3 GHz in total and still need four times more CPU cycles to reach our goal. RouteBricks delivers 8.33 Gbps for IPv4 routing per machine, and their choice for over 10 Gbps speed is to stack four servers with Valiant Load Balancing and achieve 15.77 Gbps aggregate speed. Even with multiple servers, the aggregate speed of 100 Gbps seems a distant reality.

†Sangjin Han, Keon Jang, and Sue Moon were supported by NAP of Korea Research Council of Fundamental Science & Technology.

Can we improve per-packet processing overhead? We find the solutions in packet processing software optimizations. RouteBricks uses NIC-driven batching for performance improvement. We propose the following for further improvements: (*i*) remove dynamic per-packet buffer allocation and use static buffers instead; (*ii*) perform prefetch over descriptors and packet data to mitigate compulsory cache misses; (*iii*) minimize cache bouncing and eliminate false sharing [6] between CPU cores. By incorporating these improvements, we achieve about a factor of six reduction in per-packet processing overhead and reduce the required number of CPU cycles to under 200 CPU cycles per packet [4]. Then the total number of CPU cycles required for the 100 Gbps forwarding speed comes down to 30 GHz, which is achievable with today's CPUs.

While packet forwarding is the core functionality of routers, it is one of many tasks that a typical router performs. On top of packet I/O, a router must handle IPv4 and IPv6 routing, IPsec, and myriads of other tasks. Even with today's fastest CPUs only a very limited number of spare CPU cycles is left for other tasks. In order to build a full-fledged software router with the 100 Gbps speed, we should consider other sources of computing power, such as Field-Programmable Gate Arrays (FPGAs) [5] and Graphics Processing Units (GPUs) [1].

B. I/O Capacity

Packet I/O involves the complex interplay among CPUs, NICs, and memory. Packets received from NICs go through PCIe links, IOHs, QPI links, and finally memory buses. Then CPUs process packets with memory access, and the reverse process occurs for packet transmission. Here we examine possible bottlenecks in the packet data path between NICs and CPUs.

PCI Express links: Today's 10GbE NICs have one or two ports, using PCIe x8 as a host interface. PCIe 2.0 interface operates at 2.5 GHz or 5.0 GHz per lane, which translates to bidirectional 250MB/s or 500MB/s, respectively. Intel 82598-based NICs, used in [3], operates at 2.5 GHz and has bidirectional 20 Gbps bandwidth over eight lanes. However, the effective bandwidth is not enough for dual 10 Gbps line-rate links due to encoding and protocol overhead of PCIe and bookkeeping operations for packets, such as packet descriptor write-back. RouteBricks reports 12.3 Gbps for maximum effective bandwidth for each NIC. Newer Intel NICs with 82599 chipsets operate at 5.0 GHz and thus eliminate this bottleneck.

To build a 100 Gbps software router, we need at least five PCIe 2.0 x8 slots. However, a single Intel 5520 or 7500 IOH can only support up to four x8 slots. Moreover, we need spare slots to use other devices, such as graphics cards or management NICs. Thus we need two IOHs in the mainboard as depicted in Figure 1. We use Super Micro Computer's X8DAH+-F that has four PCIe 2.0 x8 slots and two PCIe 2.0 x16 slots, and can have up to six 10GbE dual port NICs in total.

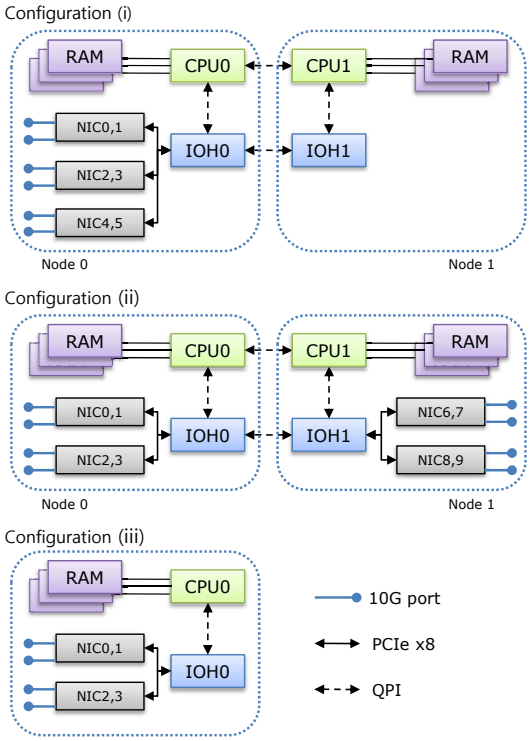

Fig. 2. System configurations for experiments

QuickPath Interconnect: In our system, QPI links play three roles; (*i*) a CPU socket-to-socket link for remote memory access; (*ii*) an IOH-to-IOH link for proxying I/O traffic heading to the other NUMA node; (*iii*) CPU-to-IOH links for interconnection between CPUs and peripherals. Each QPI link has bidirectional bandwidth of 12.8 GB/s or 102.4 Gbps. Let us consider the worst case scenario that every packet received through NICs in one IOH is forwarded to NICs connected to the other IOH. The required bandwidth in the IOH-to-IOH and CPU-to-IOH links should be at least 50 Gbps for each direction, which is only half of available bandwidth. The bandwidth of CPU socket-to-socket QPI link is not a problem as long as packets are processed in the same CPU that receives packets and the NICs move the packets into the memory that belongs to the same socket as the NICs do.

I/O Capacity Measurement: We measure I/O capacity to see whether the system achieves the theoretical limits. At the time of this experiment we have access to only eight NICs, half of which are used as packet generators. We limit our experiment to four dual-port NICs. We use two systems for evaluation. One is a server machine with dual CPU sockets and dual IOHs, and the other is a desktop machine with one CPU socket and a single IOH. The desktop machine has three PCIe slots: two are occupied by NICs and one by graphics card.

To gauge I/O capacity and identify its bottleneck, we consider three configurations in Figure 2: (*i*) three NICs are connected to one IOH in the server system and one NUMA node is used for packet processing, (*ii*) two NICs are connected to each IOH in the server system (four NICs in total), and

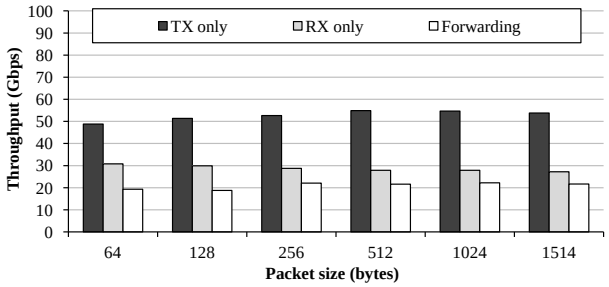

Fig. 3. Packet I/O throughput from Configuration (i) in Figure 2

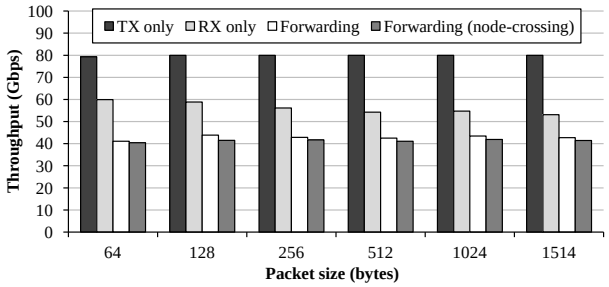

Fig. 4. Packet I/O throughput from Configuration (ii) in Figure 2

(*iii*) two NICs in the desktop system. For each configuration, we measure packet reception (RX), transmission (TX), and forwarding (RX + TX) capacity separately. For all experiments we generate traffic enough to saturate the capacity of all NICs.

Figure 3 depicts the results of experiments from configuration (i). TX throughput is capped at around 50 Gbps with total 60 Gbps NICs. RX throughput is around 30 Gbps, far less than transmission throughput. Forwarding performance is about 20 Gbps.

We plot the throughputs from Configuration (ii) of dual IOHs with four NICs in Figure 4. TX throughput is 80 Gbps, reaching the theoretical maximum. RX and forwarding throughputs are 60 and 40 Gbps, exactly double the single IOH case. The results imply that the actual performance of the dual socket server system cannot reach 100 Gbps of forwarding or receiving performance. In all of our experiments, CPUs are not the bottleneck, and putting more cores or CPUs will not help. The slight degradation of throughput with larger packet sizes is because batching in large packets leads to longer processing time per batch and results in delayed packet reading from NICs. We can reduce the batch size for larger packets to eliminate the performance gap between packet sizes. In Figure 4, we also show the results when packets cross the NUMA nodes, but we see little performance degradation compared to non-crossing cases. It implies that the QPI link is not the limiting factor at 40 Gbps but the question remains as to where the bottleneck is.

To find the cause of throughput difference between RX and TX side, we conduct the same experiment with the desktop system. Figure 5 shows the results. Interestingly RX and TX reach the full throughput of 40 Gbps with two NICs. This leads us to the conclusion that the RX performance degradation in the server system is due to dual IOHs rather than dual CPU sockets. By Googling, we find that the receive I/O throughput degradation with dual IOHs is also known to the GPGPU programming community and that single IOH with dual sockets did not have the problem [2]. Forwarding performance is around 30 Gbps, and is lower than RX and TX throughput. Since QPI and PCIe bus are full-duplex links, I/O should not be the problem. We find that the forwarding performance in the desktop scenario is limited by the memory bottleneck. We explain further details in the following section.

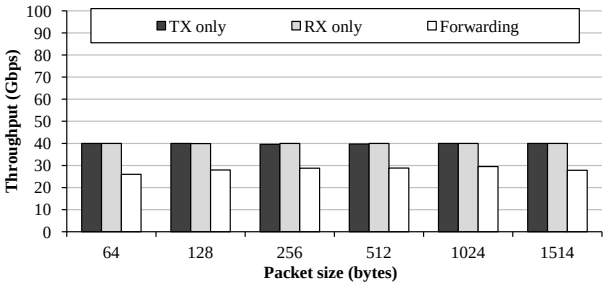

Fig. 5. Packet I/O throughput from Configuration(iii) in Figure 2

C. Memory Bandwidth

Forwarding a packet involves several memory access. To forward 100 Gbps traffic, the minimum memory bandwidth for packet data is 400 Gbps (100 Gbps for transfer between NICs and memory, another 100 Gbps for transfer between memory and CPUs, and doubled for each direction of RX and TX). Bookkeeping operations with packet descriptors add 16 bytes memory read/write access for each packet, giving more pressure on memory buses depending on packet sizes.

In Figure 5, we see that the forwarding throughput is lower than that of RX and TX due to insufficient memory bandwidth. We find that (*i*) CPU usage for forwarding is 100% regardless of packet sizes and load/store memory stall wastes most CPU cycles and (*ii*) with memory overclocking to have more memory bandwidth, we improve the forwarding performance close to 40 Gbps.

For both our server and desktop configurations, we use triple-channeled DDR3 $1,333$MHz, giving theoretical peak bandwidth of 32.0 GB/s for each CPU and 17.9 GB/s empirical bandwidth according to our experiments. Unfortunately, assuming two nodes in the system, we need effective memory bandwidth of 25 GB/s for each node to forward 100 Gbps traffic.

One way to boost the memory bandwidth in NUMA systems is to have more nodes and to distribute the load to multiple physical memory in different nodes. In this case, NUMA-aware data placement becomes particularly important. This is because remote memory access is expensive in NUMA systems in terms of latency and may overload interconnects between nodes. High-performance software routers on NUMA systems should consider careful node partitioning so that communication between node be minimized.

III. Discussion and Future Work

In this paper we have reviewed the feasibility of a 100 Gbps router with today's technology. We find two major bottlenecks in the current PC architecture: CPU cycles and I/O bandwidth. For the former, we propose reducing per-packet processing overhead with optimization techniques and amplifying the computing cycles with FPGAs or GPUs. For the latter, we believe the improvement in IOH chipsets and multi-IOH configuration, and more memory bandwidth with four or more CPU sockets could alleviate the bottleneck. A 100 Gbps software router will open up great opportunities for researchers to experiment with new ideas and we believe it will be a reality in the near future.

References

[1] "General Purpose computation on Graphics Processing Units," http://www.gpgpu.org.

[2] "Nvidia forum," http://forums.nvidia.com/index.php?showtopic=104243.

[3] M. Dobrescu, N. Egi, K. Argyraki, B.-G. Chun, K. Fall, G. Iannaccone, A. Knies, M. Manesh, and S. Ratnasamy, "RouteBricks: exploiting parallelism to scale software routers," in *Proceedings of ACM Symposium on Operating Systems Principles (SOSP)*, 2009.

[4] S. Han, K. Jang, K. Park, and S. Moon, "PacketShader: a GPU-Accelerated Software Router," *submitted for publication*.

[5] J. Naous, G. Gibb, S. Bolouki, and N. McKeown, "NetFPGA: reusable router architecture for experimental research," in *PRESTO '08: Proceedings of the ACM workshop on Programmable routers for extensible services of tomorrow*. New York, NY, USA: ACM, 2008, pp. 1–7.

[6] J. Torrellas, H. S. Lam, and J. L. Hennessy, "False Sharing and Spatial Locality in Multiprocessor Caches," *IEEE Transactions on Computers*, vol. 43, no. 6, pp. 651–663, 1994.