High Performance Modular Packet Processing with Click and GPU

Weibin Sun
Robert Ricci
{wbsun, ricci}@cs.utah.edu
University of Utah, School of Computing

Software packet processing, like Click modular router, provides much more flexibility than the hardware-based one. Recent advances (Netmap, psio, PF_RING) in software packet I/O have enabled $10Gb/s$ throughput on $10Gb$ NICs with the help of Receive Side Scaling (RSS) on multi-queue NICs and multi-core CPUs, and also zero-copy buffer sharing between kernel and user modes. With such advance, it is possible to build software packet processing systems running at line rate. Those systems include intrusion detection, firewall, VPN, Openflow, IP forwarding, deep packet inspection for ISPs to detect pirated content downloading, and many others.

However, the bottleneck that prevents those packet processing systems listed above from reaching line rate throughput is becoming the computation now. For instance, the pattern matching algorithm for intrusion detection can only run at $2.15Gb/s$ on an Intel Xeon X5680 core according to the Kargus research. It needs five CPU cores to saturate only one $10Gb$ port. This would require significant number of dedicated CPU cores for packet processing, which may contend with packet I/O cores and hence lower the entire system performance. Besides that, the hardware cost of those multiple CPUs is also way more than another feasible commodity hardware choice, which is the parallel GPU computing device. Many research projects have shown huge performance improvements of various computations on parallel GPUs compared with CPUs. As of the pattern matching algorithm, recent work on Kargus shows a $39.1Gb/s$ throughput on a $400$ GTX580 GPU, which is $3x$ faster than the $1639$ Xeon X5680 six-core CPU above. As a result, to build a flexible high performance packet processing system that could run at $10Gb/s$ line rate, we take the modularized Click router and integrate GPU computing into it for computation intensive processing.

To use GPU computing library and other userspace libraries, we are using usermode Click. Obviously, Click was not originally designed for usermode zero-copy packet I/O and parallel GPU computing. We adopt the following technologies to deal with obstacles in Click, multi-queue NICs and multi-core CPUs: a) **GPU Management**: A new GPUT-runtime (specifically, CUDA runtime) Click information element to manage GPU computing memory, kernel and states. There are CUDA mapped, page-locked memory and non-mapped, page-locked memory used for GPU computing. They have different characteristics that worth investigating during our evaluation. b) **Batching**: We use batching to get rid of single packet processing style in Click. A Batcher element batches packets and also prepares GPU memory. It also support sliced copy to only copy specified range of packet data. c) **Wider Interface**: To perform push and pull with packet batch, we defined BEElement and BPort to provide packet batch support. d) **Hybrid CPU and GPU Processing**: A CGBalancer element that load-balances packets between GPU and CPU according to specified policy and system states. This provide flexible GPU offloading mechanism. e) **Zero-copy**: We use Netmap for zero-copy packet capture and transmission, the GPU-enabled Click has been modified to use CUDA memory that GPU driver used for DMA in Netmap. Hence zero-copy buffer sharing among GPU driver, NIC driver and Click. f) **Multi-queue Support**: Netmap provide multi-queue packet capture and transmission. To be NUMA-aware for buffers used on different CPU cores, we use CUDA’s cudaHostRegister() to pin NUMA-aware allocated memory, so that it can be used for GPU DMA and as zero-copy packet buffers.

We have implemented the Click level improvement including GPU-related elements above, and the Netmap/CUDA/Click integration for zero-copy memory sharing. We are developing several computational packet processing elements for evaluation, such as simple firewall, IPv6 routing, IPSec, VPN and Openflow. Besides these functional applications, we also want to investigate the following problems to fully study and analyze our system: a) comparing the costs of CPU-only system and GPU-enabled system, build their theoretical cost model when scaling up. b) comparing different GPU runtime memory types under different workloads and other conditions. c) study the effect of load balancer policy when running different workloads. d) try to answer this question: Is multi-queue really needed? It is derived from these two facts: Netmap research found a single GPU core can handle line rate forwarding, and now we have GPUs dedicated for computing. e) also study and measure the scalability of our GPU-enabled Click with regard to number of NICs.
**The Problem**

- **CPU** limits functionality in line rate packet processing:
  - 10Gb/s line rate packet I/O available [Netmap, psio].
- Compute intensive packet processing:
  - Intrusion detection
  - Firewall
  - Deep packet inspection
  - VPN, Openflow, etc.
- **Flexible** platform needed.

**Our Solution**

- **Flexibility**: Click modular router!
- **Compute intensive processing**: Faster/More CPU cores?
  - **NO!** We have faster and cheaper GPUs.
  - How faster can we run with GPU?

**GPU and Click, How?**

- Manage GPU resource in Click, allow GPU click Elements.
- Batching needed for GPU
- Flexible CPU/GPU load balancing
- Efficient MM for GPU, NIC and Click.
- Utilize what we have now for CPU-based packet processing.

**Heterogeneous GPU Computing In Click**

- **GPURuntime** (CUDA) element for Click to manage GPU resources.
- **Batcher** element for `batch`, `slice`, `copy`, and naturally no REORDER problem.
- **CGBalancer** element, to load-balancing between GPU and CPU.
- **GPUDirect**, to zero-copy among GPU driver, NIC driver, and Click.

**Problems To Investigate (and To Discuss)**

- **How many 10Gb NICs a single CPU core can handle?**
  - To know the ratio of #CPU core / #GPU, and cost saving, and hence the overall system cost comparison.
  - Study mapped GPU memory and non-mapped one, under different workloads, batches, slicing, scattered packet buffers.
  - Effects of workload-specific balancer policy on hybrid CPU+GPU packet processing.
  - Is multi-queue really needed?
  - According to Netmap work, a single CPU core can handle both RX and TX at line rate for forwarding.
  - Using GPU as main computing resource, CPU can just do I/O, interrupt handling.
  - To what extent(#NICs) can this GPU-enabled Click scale up?

**Current Progress and Todo**

- **Done**: infrastructure level, including Click GPU-related elements, Netmap/CUDA integration.
- **Todo**: Computational Click packet processing elements on GPU for evaluation:
  - Simple firewall: online packet inspection.
  - IP routing, IP/Sec, VPN
  - Openflow, ...