Modelling of High Bandwidth Systems

Getting DPDK to receive and transmit packets at 100 Gbps is just a start. This whitepaper provides a theoretical model for understanding performance of high-bandwidth packet processing systems with FPGAs and CPUs. Limitations of raw processing power are evaluated as a demonstration of a need for hardware acceleration of packet processing tasks.

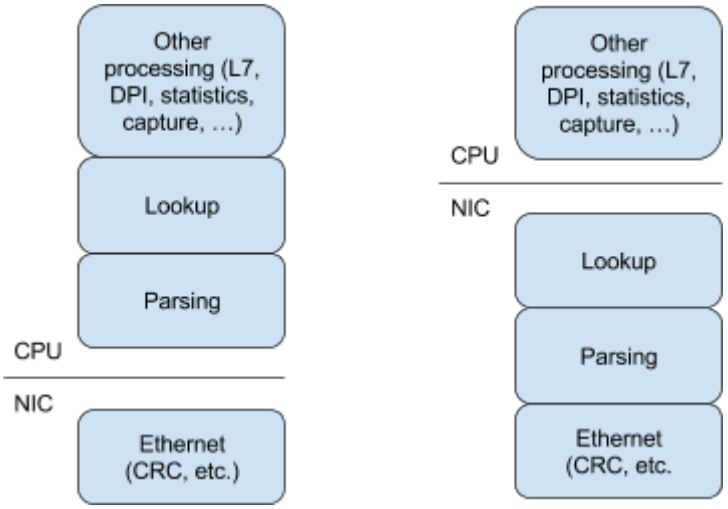
Introduction

With the 100 Gigabit Ethernet being mature and available technology, as well as 400 Gigabit Ethernet standard just before finalization, it seems that CPUs have a hard time trying to catch up. Netcope Technologies have already demonstrated that receiving and transmitting full 100GE line rate in a single-CPU server with FPGA-based NIC is entirely possible, but that doesn't tell much about the usability of such setup from the application perspective.

The goal of this paper is to understand exactly what are performance limitations of CPUs, how can we measure, describe and compare them, and what can we do to overcome the limitations.

Model of Packet Processing Application

A software application that processes ingress packets typically starts with packet header parsing, followed by table lookup (both after receiving the packet, of course). Since these operations are often among the first in the packet processing chain, it often makes most sense to offload them to an accelerator NIC. Conceptually, it's about where you draw the line, as illustrated in the following picture:



That's why our effort is to model the tasks of packet header parsing and field lookup, and to understand how they affect the overall application performance. To do this, we have created a test DPDK application that does exactly that: receives packets, simulates packet parsing and table lookup. Then it modifies the packet by inserting a VLAN header and transmits the packet back to the network, effectively avoiding possible compiler optimizations due to unused results. Packet header parsing, that is often used in DPI and other systems, is simulated by performing several conditional jumps, based on packet data. Table lookup is simulated by computing a hash

function from several bytes of a packet. We set the number of conditional jumps and hash calculations as a parameter *h*. When we establish *p* to be packet length in bytes, we can assume that packet processing time *T* can be expressed as

$T = C + P*p + H*h$

where *C* is a constant per-packet receive+transmit overhead, *P* is a per-byte receive+transmit overhead, and *H* is a single parse+hash step operation overhead. All that this model of packet processing time needs is to find values of *C, P* and *H*. After that, performance of a packet processing system (mainly NIC+CPU+RAM) can be evaluated and compared. Also application complexity can be assessed.

How can we get *C, P* and *H*? We can set *p* and *h* in our experiments arbitrarily by setting packet length on a packet generator, and by adjusting the processing complexity in the test application, respectively. We can also measure *T* by looking at the application throughput. Finally we are getting to some math, since we have a single linear equation with six variables, three of which are unknown, and for the other three we have a lot of experimental data. That's a clear case for linear regression.

Results

We have performed measurements using NFB-100G2Q card and Intel Xeon E5-2687W v4 CPU, then performed linear regression, and then compared the model outputs back with the measured data. It seems that our assumption about the nature of CPU load (the $T = C + P*p + H*h$ formula) is correct, since the vast majority of measured data points differ from the model by less than 0.5 % and only three measured points differ from the model by more than 3 % (these are for short packets and low *h*, where the system is stretched to the extreme).

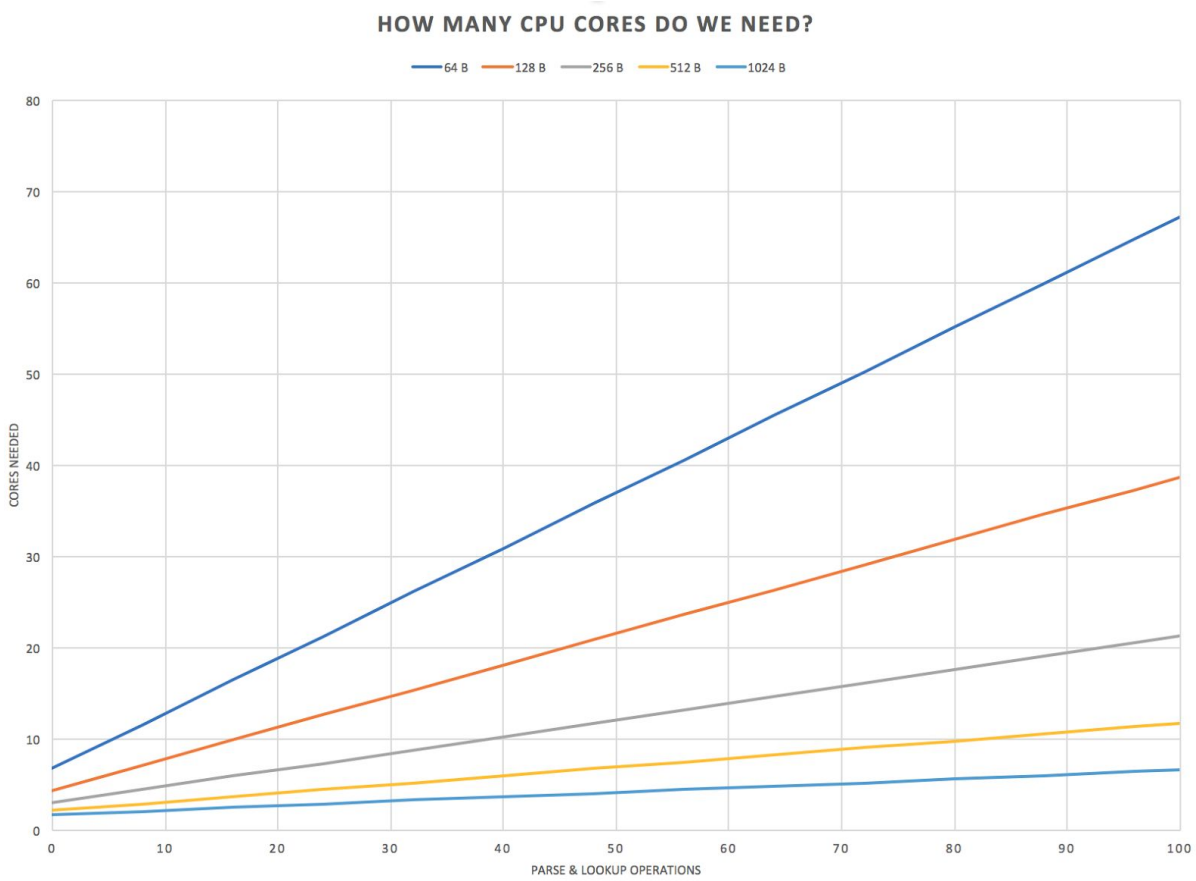
Our measurements show the following results:

C = 38.6 ns/core

H = 4.07 ns/core

P = 0.101 ns/core

Compare to the maximum 100 Gigabit Ethernet rate of 6.7 ns/packet. Suddenly, 5.7 cores are needed just to receive and transmit hypothetical zero-Byte packets (*C*/6.7). That number rises to 6.7 cores for the shortest real 64 Byte packets ((*C*+64**P*)/6.7). Since the Ethernet packet rate is smaller for longer packets, we can get as low as about 1.5 cores for 1518 Byte packets receive&transmit. But these are numbers for *h* = 0, which means that no real work is done with the packets. Let's see how CPU cores scale when increasing the processing complexity:



One word: Poorly. While one might think that 100 parse&lookup operations is too much, imagine one of the most popular packet processing application: a virtual OpenFlow switch. In such software switch, a packet is parsed, then it traverses a chain of lookup tables, before it is sent to the corresponding virtual machine, where the destination protocol stack parses the packet again and performs more lookups. Also let's remind that our test app uses very simple operations, hand coded in pure C. Real protocol parsing and table lookup has more complex code, possibly with branches, so it accounts for several of our steps.

Other applications, such as deep packet inspection systems, use even more complex string- and pattern-matching algorithms to scan through the whole packet payload. On top of that, let us remind that any application working above L4 must perform TCP stream reassembling, which requires not just more lookups, but also maintaining a (possibly large) per-flow state record in RAM, stressing the system even more.

Suddenly 100 parse&lookup operations doesn't seem to be that much. What is way too much is the scale of our vertical axis. Tens of CPU cores consumed just to receive, parse and steer the packets? There is no time left for the revenue-generating application logic.

Acceleration is Here to Help

Our results show two things:

- Once again we confirm that achieving full 100 Gbps DPDK forwarding throughput for all packet lengths is entirely possible with single-CPU server and Netcope FPGA Boards using single PCI-E gen3 x16.
- We show that for virtually any meaningful packet processing, current CPUs are just not powerful enough to support 100 Gbps line rate.

Fortunately, Netcope FPGA Boards are much more than just Network Interface Cards. Their acceleration functions have proven very effective in offloading various tasks off the CPU:

- Netcope Packet Capture (NPC) is a packet capture solution that comes with a hardware filter that supports up to eight thousand filtering rules. L3 and L4 header fields can be used to express conditions. Corresponding action can transfer a packet to the host system, send it to an output network interface, or drop it. Intelligent transfer to the host system can be used to distribute the traffic over CPU cores based on hashing in flow-aware fashion.
- Netcope Session Filter (NSF) is a session-oriented packet capture solution that accelerates per-packet processing and flow-based stateful filtering, which leaves more CPU performance for complex processing of network traffic, like DPI. The cooperation of hardware and software makes it possible to build a powerful solution even for 100G Ethernet networks based on commodity multi-core servers. Key feature of NSF is stateful filtering-based packet manipulation, which offers significant advantage over per-packet stateless processing. NSF perceives network traffic as a set of network flows and it is able to track hundreds of thousands of network flows directly in hardware. Software applications leverage hardware preprocessing of network flows to identify flows of interest for further processing and instructs hardware through API on how to deal with each flow. In other words, it allows you to zoom in interesting traffic, drop the traffic of no interest and gather statistical information about the rest.
- Last but not least, if your application does not map to any of the above, Netcope Development Kit is the ultimate option. NDK is a toolset for rapid development of custom hardware-accelerated network applications based on Netcope FPGA Boards. It is based on a sophisticated build system and a collection of IP cores and software. It offers comprehensive environment that enables prototyping of an application in the shortest time possible, which is an invaluable feature for solution vendors, integrators and R&D teams. And yes, our record-breaking 100+ Gbps DPDK modules are included in NDK. In addition to that, NDK users no longer have to use VHDL or Verilog to implement their networking accelerators. Netcope P4 to FPGA Compiler is able to generate the application firmware from the high-level description in the P4 language.