Towards 100 Gbit Flow-Based Network Monitoring

Luca Deri <deri@ntop.org>

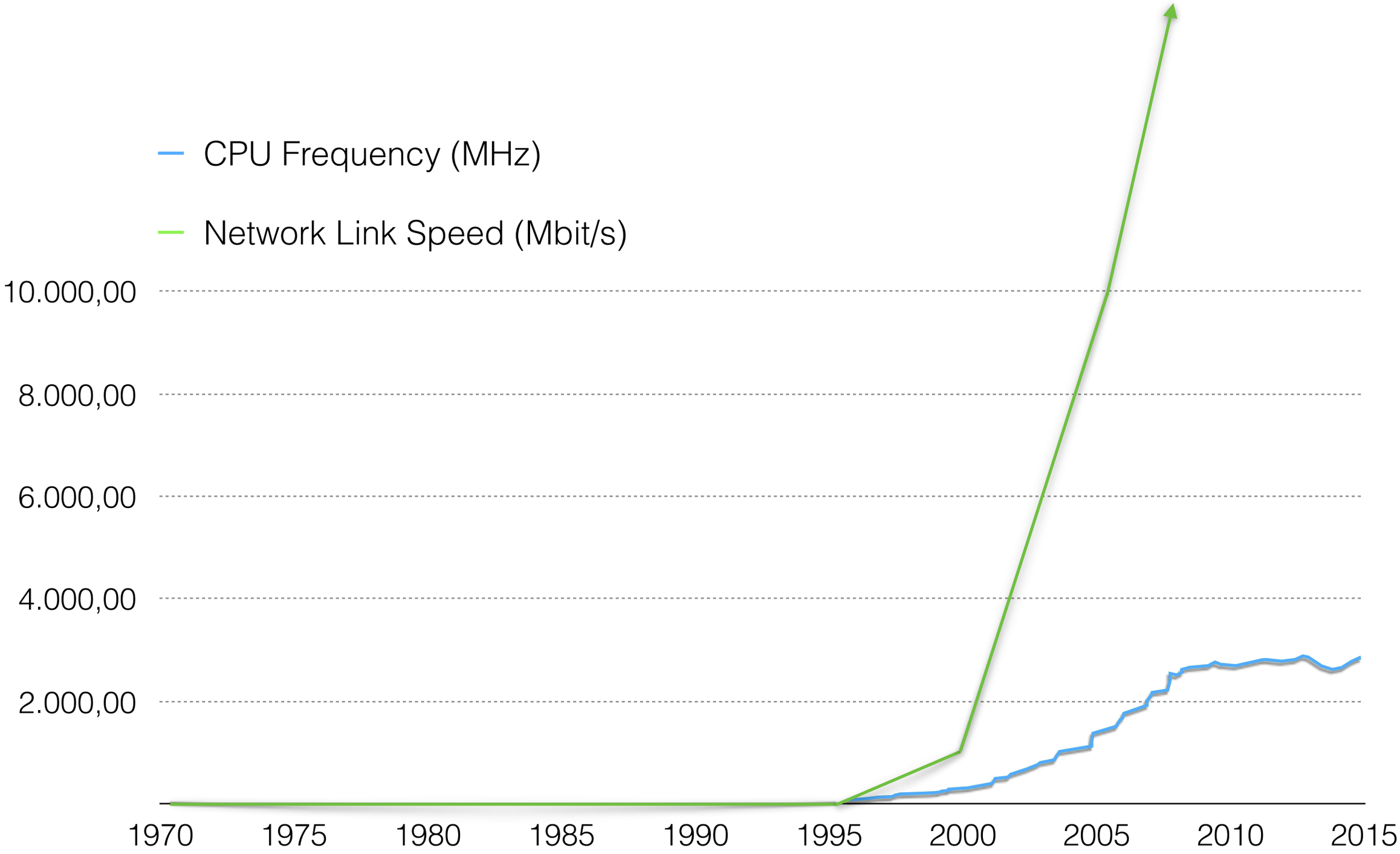
Part 1

100 Gbit: Benefits and Challenges

Towards 100 Gbit Networking

- In the past few years the market has moved from 10 Gbit to 40 Gbit, and recently to 100 Gbit. 100 Gbit is still not very popular but it is becoming more and more popular as per-port/optical adapters price drop.
- As it happened years ago with the transition from 1 to 10 Gbit, 100 Gbit has been initially available only on switches and routers, but there are now 100 Gbit host network adapters on the market.

Trends: Network vs CPU



Ethernet Speed is Increasing [1/2]

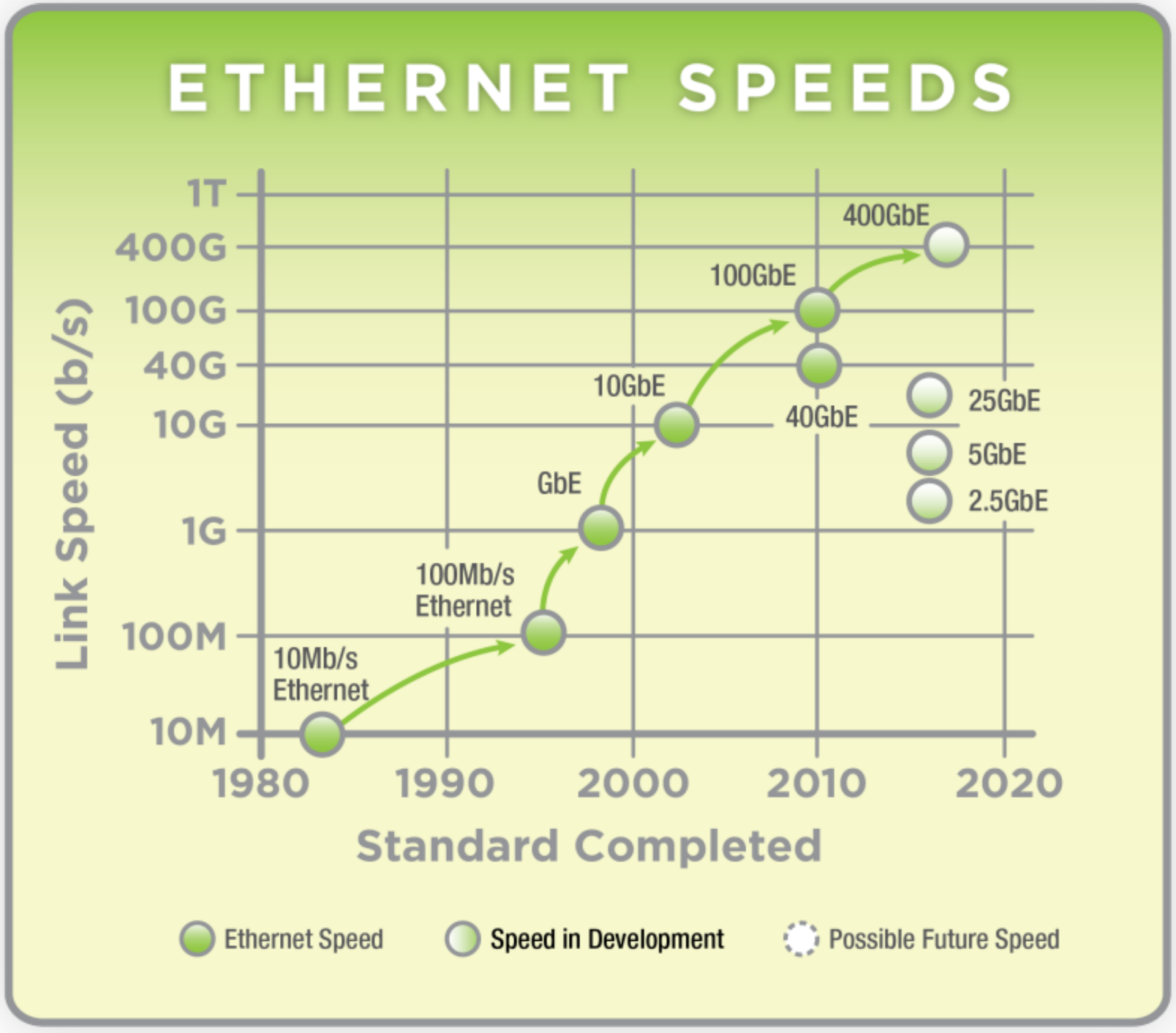
- 6 new speeds in development
  2.5 GbE, 5 GbE, 25 GbE, 50 GbE, 200 GbE, 400 GbE.

- Cloud transition to 10GbE has passed, pushing fast towards 25G, 50G

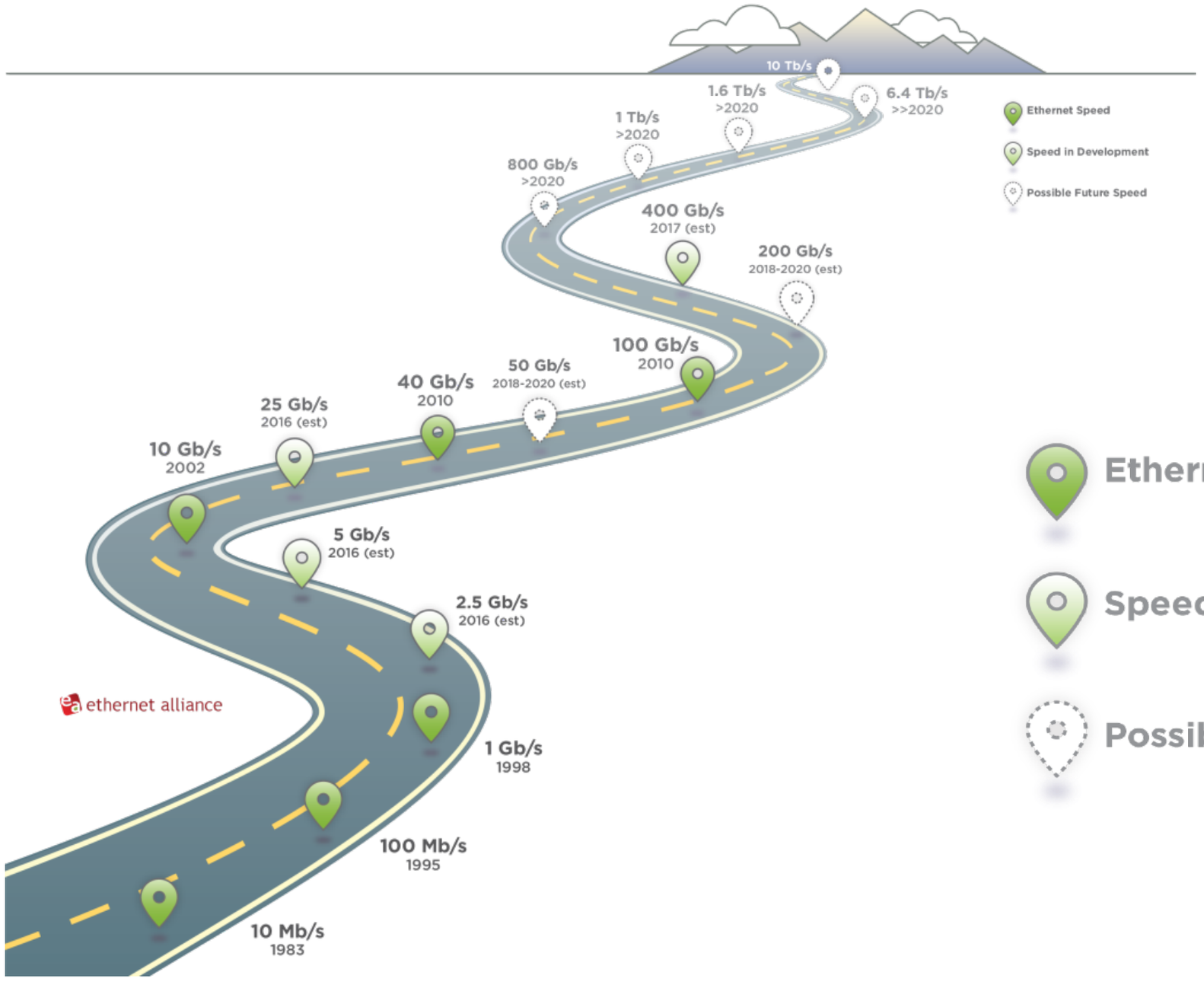
- Enterprise servers are still making the transition to 10GbE

- The current 1 GbE will be replaced by 2.5/5 GbE, 10 GbE by 25/50 GbE, 40 GbE by 100 GbE.

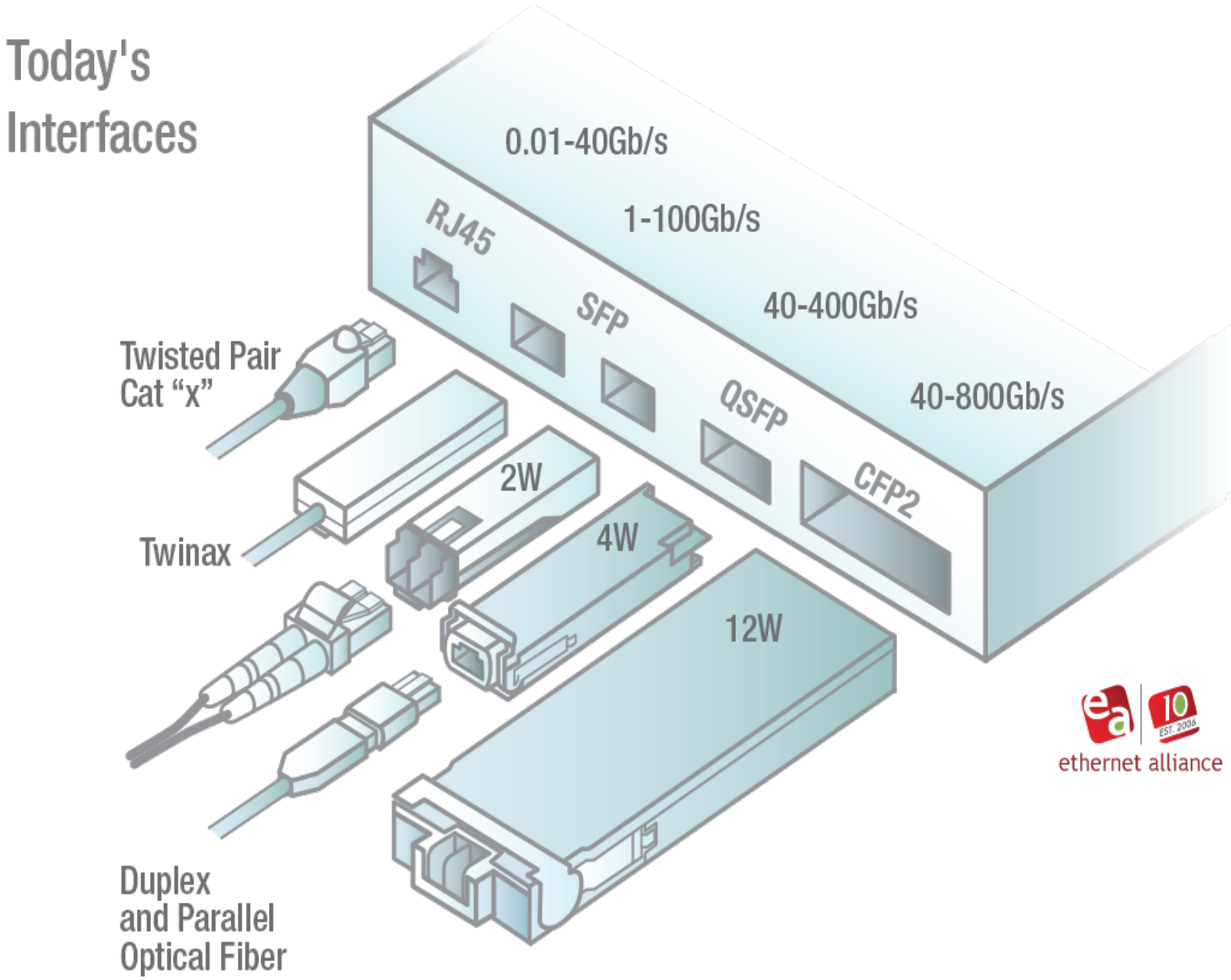
Ethernet Speed is Increasing [2/2]



Ethernet Roadmap



Ethernet Modules



100 Gbit Modules

- CXP (C is the Roman numeral for 100)
  - (12 x 10 Gbit or 3 x 40 Gbit) Fibres bundled on the same connector. Derived from Infiniband.
  - Compatible 40 and 100 Gbit. Available also in copper.

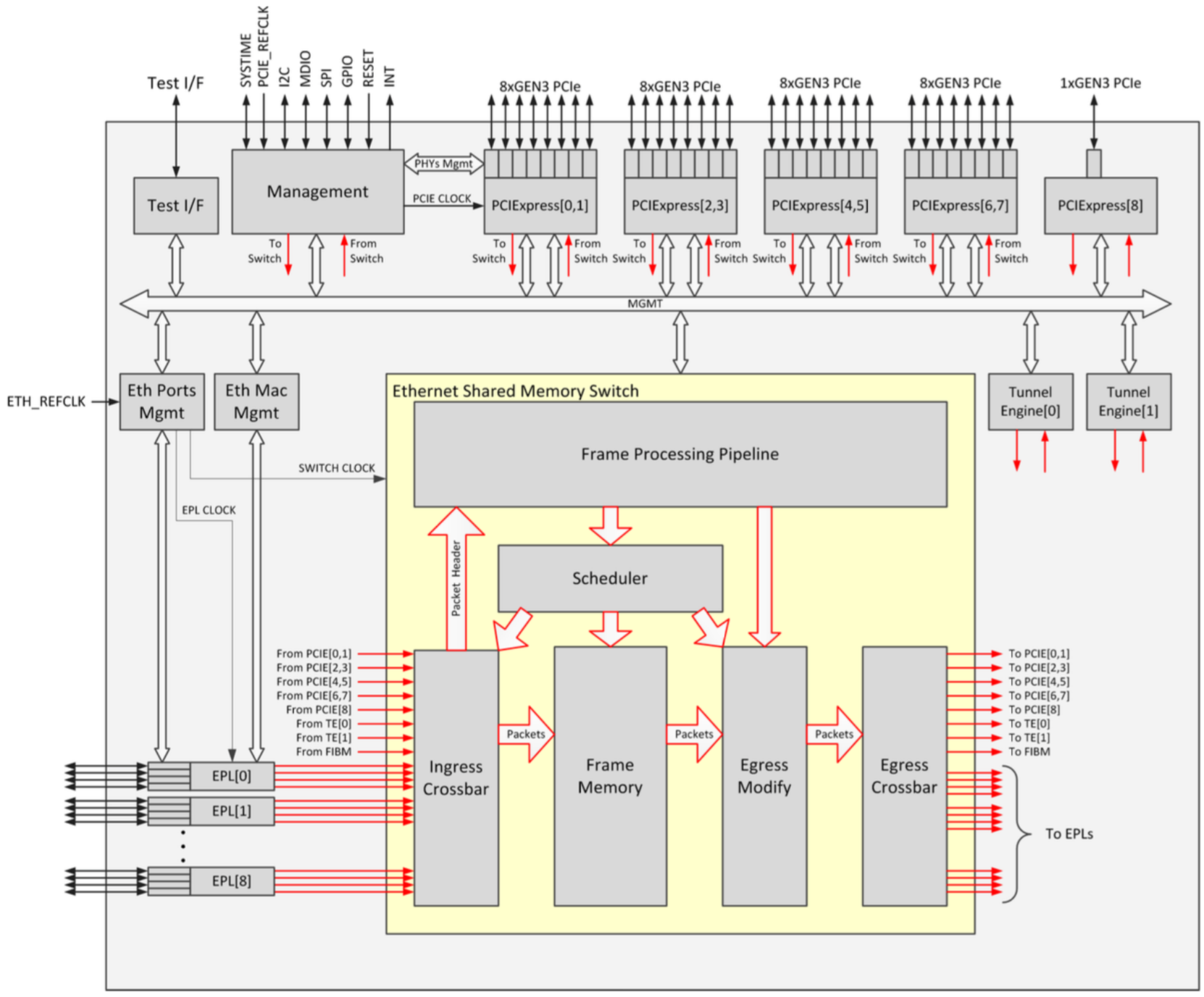
- QSFP28 (Quad Small Form-factor Pluggable)
  - 2/8 Fibres bundled on the same connector (4 x 28 Gbit).
  - Compatible 25, 40 and 100 Gbit.
  - This is the connector of the future as it uses fewer fibres thus increasing port density.

100 Gbit Packet Capture

- FPGA-based NICs (e.g. Accolade and Napatech) have been out for more than a year now, with prices (not including optics) starting in the 10K USD range.

- Recently Intel has introduced the FM10000 10/25/40/100 Gbit Ethernet controller (Red Rock Canyon) that offloads to the controller selected features (e.g. packet switching/distribution/drop). The first products have been announced and are available since 1Q16 for < 1.5k USD (dual 100 Gbit).

Intel Red Rock Canyon [1/2]



Intel Red Rock Canyon [2/2]

- In essence RRC (FM1000) is a family of network switches that integrate 10/25/40/100 Gbit ports.

- It is possible to program the switch to drop/redirect/ filter traffic across ports (as every network switch does).

- Currently there are network adapters available on the marked using RRC but they are limited in performance due the PCI 8x slot performance.

- Good low-end solution, but no line rate yet.

PCIe Bus Speed

- 100 Gbit is not just a lot of data in terms of network traffic but also for the PCIe bus.

- PCIe 3.0 bus speed (8 GT/s)

1x	4x	8x	16x
7,87	31,5	63	126

Gbit/sec

- For 100 Gbit the 16x slot is required, but there is <u>not enough bandwidth available</u> for 2 x 100 Gbit (typical in case of using a network tap with 100G links).

- PCIe 4.0 will appear in 2017 (double GTs, and Gbit/) and it will enable 200 GbE in a single slot.

From 10 Gbit to 100 Gbit [1/3]

- Just as RSS (Resource Side Scaling) allows network adapters to distribute traffic across cores on modern network adapters, it is possible to do the same using network switches.

- Some products (e.g. Dell Z9100-ON) allow traffic to be statically (no RSS-like features) distributed across multiple 10/25/40 Gbit ports for reducing 100 Gbit monitoring to multi 10-Gbit monitoring.

From 10 Gbit to 100 Gbit [2/3]

- OpenFlow-programmable switches, can be very well used to distribute ingress traffic across their interfaces and this implement the “divide and conquer” paradigm.

- You can setup ACLs (same as FM1000) for selectively dropping/filtering/diverting traffic across ports.

- ACLs do not support any sort of RSS-like features, meaning that you cannot dynamically balance traffic across ports, but you need to implement this statically.

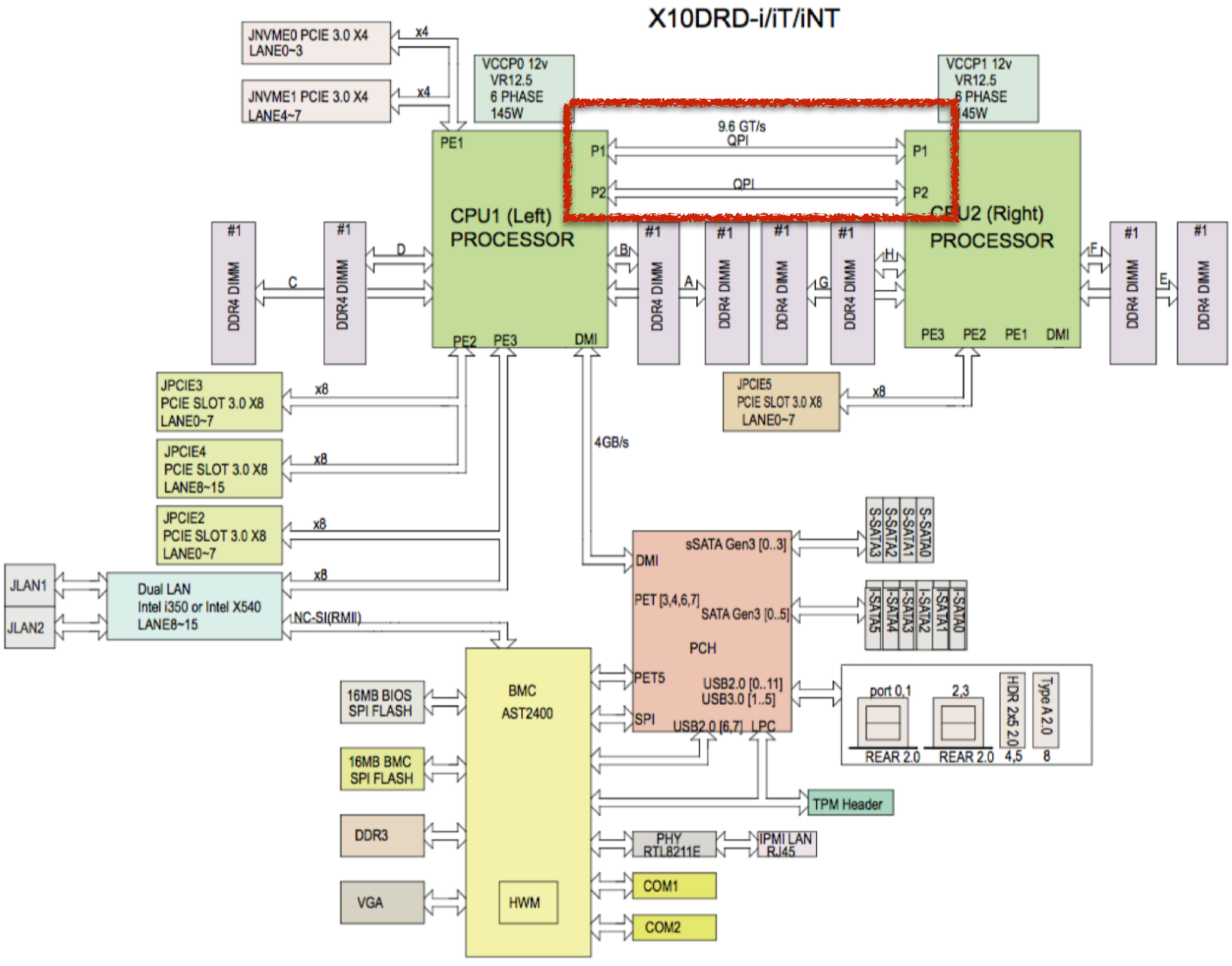
From 10 Gbit to 100 Gbit [3/3]

- In essence both Red Rock Canyon and external OpenFlow-programmable switches can achieve the same goal with a different form-factor.

- ACLs can help dropping/filtering/redirecting traffic but they are not dynamic and flow-aware.

- These solutions are good for selected cases where the traffic to analyse is very specific and predictable in terms of IPs and ports.

What About Computing Architectures [1/2]?



What About Computing Architectures [2/2]?

- As 100 Gbit link can have over 148 Mpps, a single CPU node might not be enough to sustain the ingress rate, and thus we need to spread the load across CPUs.

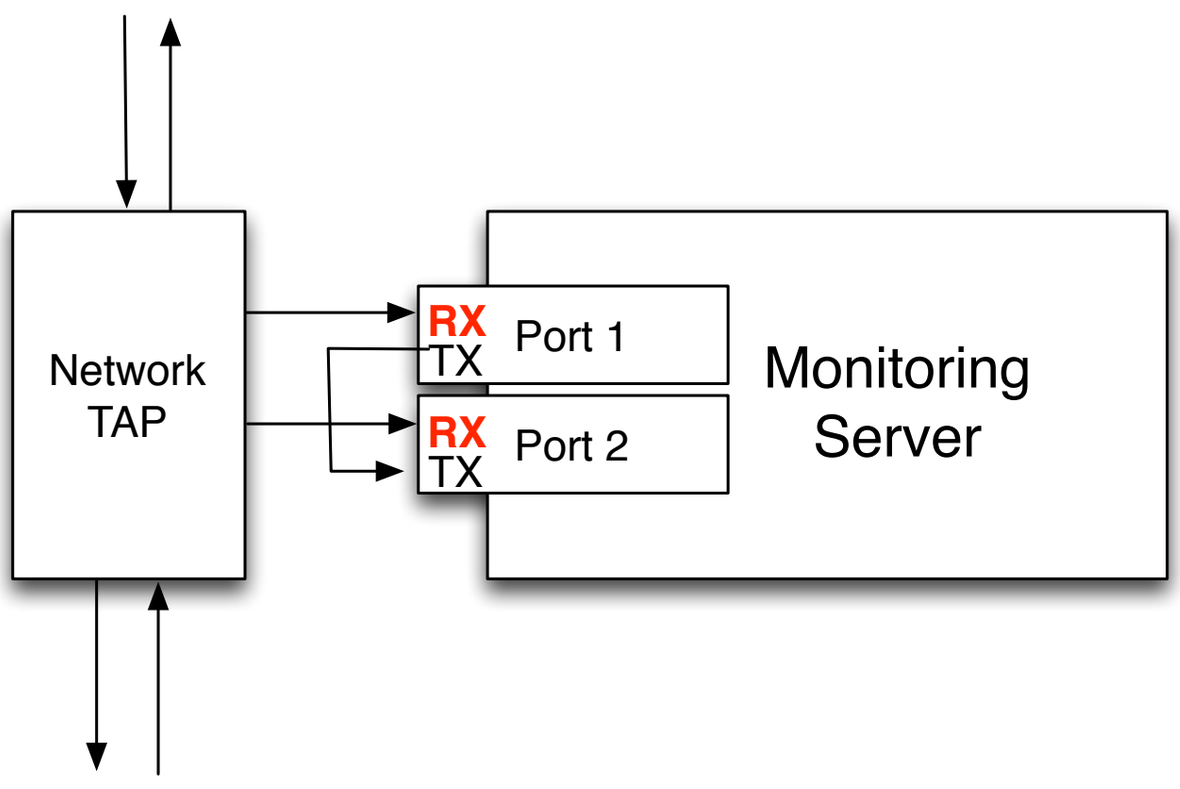
- QPI (Quick Path Interconnect) is the bus that interconnects the CPUs on a NUMA system: packets across CPUs will flow through it.

- …but QPI has a narrow bandwidth (9.8 GTs) and not just packets have to pass through it. In essence we need to avoid using QPI.

Bypassing QPI [1/3]

- Traffic filtering can help reducing the bandwidth required, but this is not always an option.

- In case a networks tap is used, out of a single 100G interface, two physical 100G interfaces are necessary.

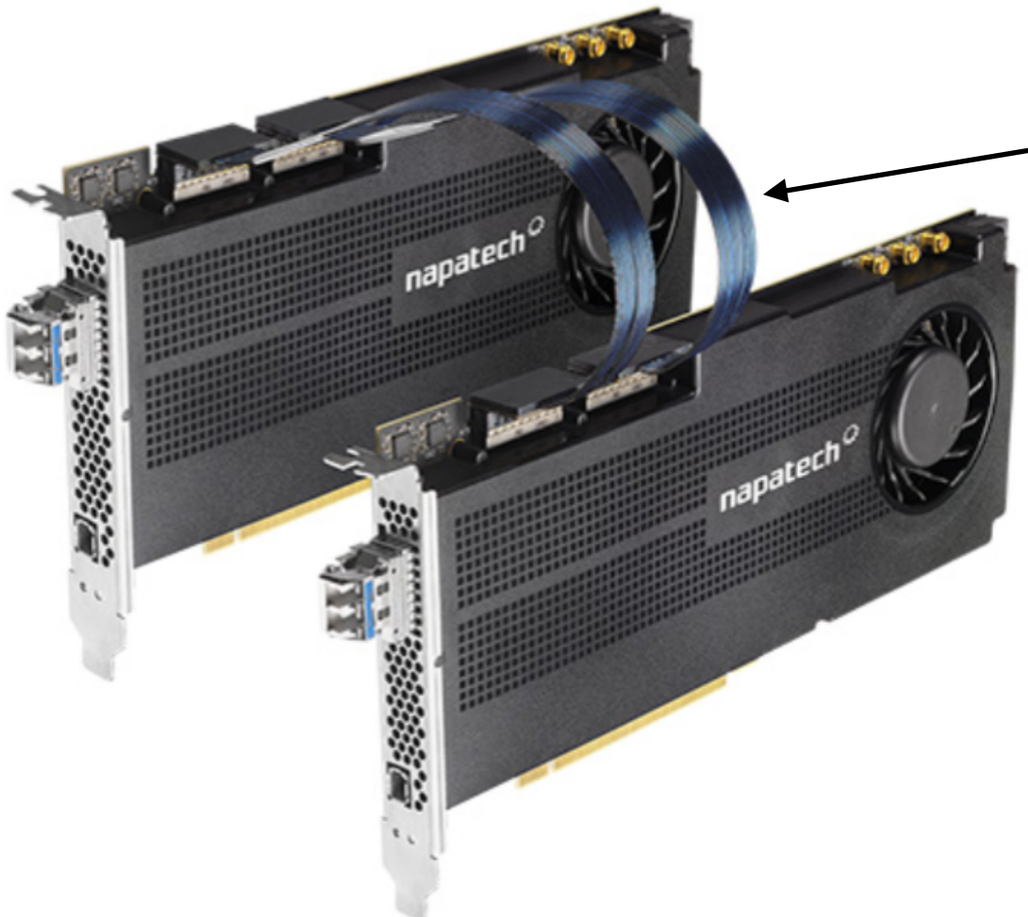


Bypassing QPI [2/3]

- As cores of a single CPU might not be enough, we need to:
  - Distribute the traffic across CPUs, without using the QPI bus that has a narrow bandwidth.
  - Operate efficiently when spreading across CPUs (affinity, caching etc.): in NUMA systems, each CPU has its local memory directly attached.

Bypassing QPI [3/3]

- Therefore vendors overcome this problem using various technologies.

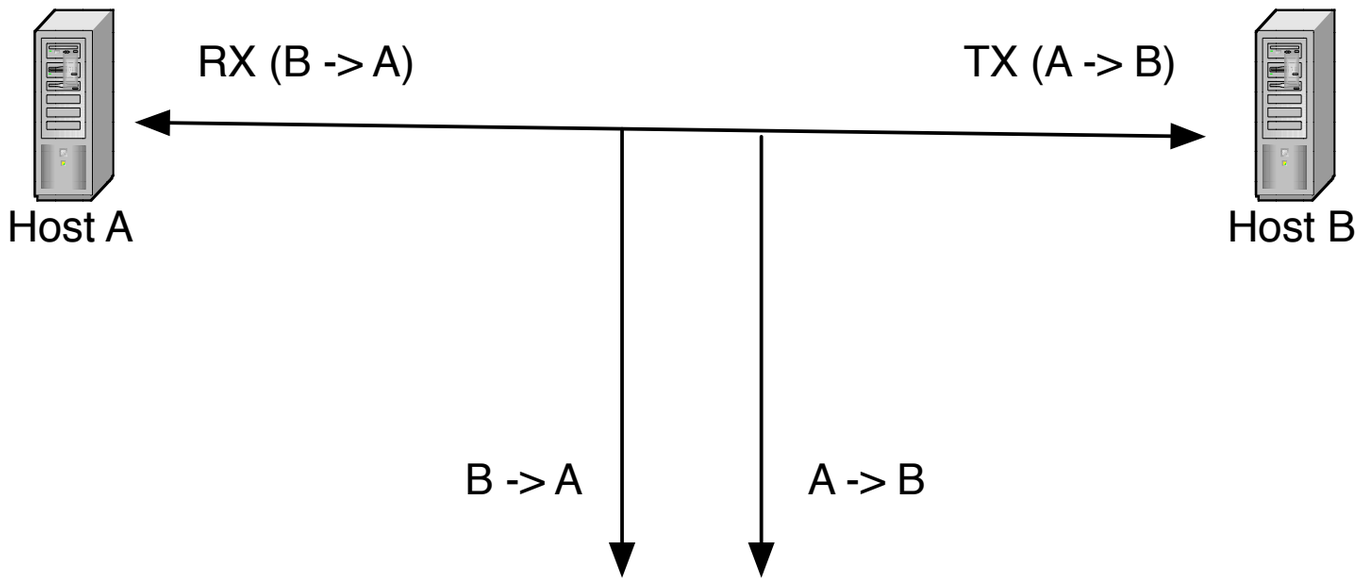


Napatech Custom Interconnection Cable

Traffic Coherency [1/4]

- Direct NIC interconnection can solve the problem of the QPI bypassing but it also solves another problem.

- Monitoring applications usually require “coherent” traffic (i.e. the same app/thread must see a->b and b->a), so it is necessary to preserve this property.



Traffic Coherency [2/4]

- With a network tap, a->b is on one port and b->a on the other port. Thus a monitoring application must monitor both ports at the same time for a total (potential) amount of 200 Gbit.

- The interconnection cable allows this problem to be solved as the NIC sends over the cable flow packets in order to preserve the traffic coherency. Without it, it would be necessary to process both traffic directions over the same CPU.

Traffic Coherency [3/4]

- Traffic is distributed across cores using a mechanism called RSS (Resource Side Scaling).

- In essence the adapter is virtualised into multiple virtual interfaces (often called streams) and traffic is hashed (IP, port, protocol, VLAN): based on the hash value is put onto an interface/stream.

- In case of fragmented IP traffic, it is desirable to use 2-tuple hash (vs 5 -tuple) so fragments and full packets are put on the same interface/stream.

Traffic Coherency [4/4]

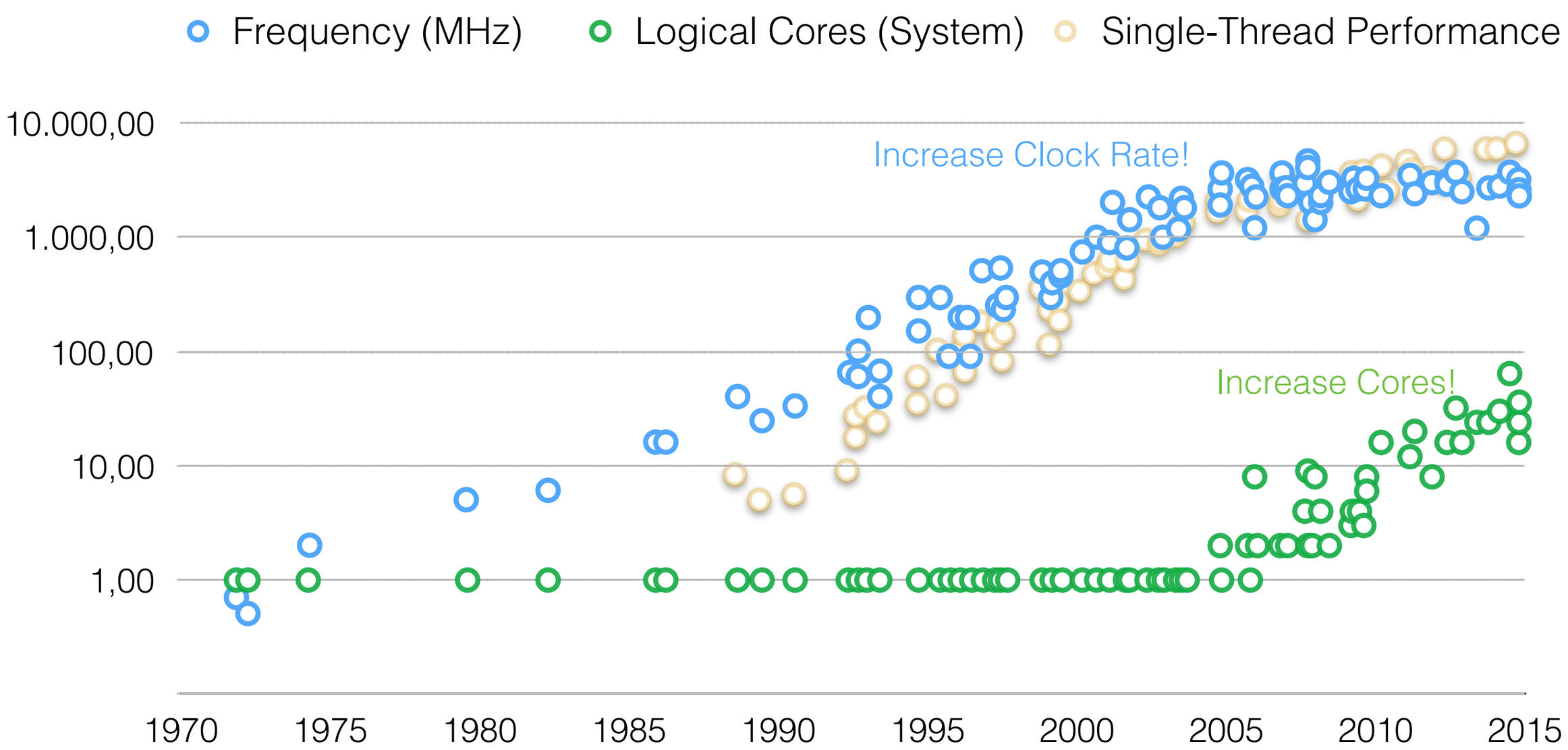
- In essence the QPI bypass is a transparent (to applications) mechanism for spreading traffic across CPUs (and cores) in a coherent fashion.

- However applications must be able to be:

  - Partitioned into several independent processing threads (no intra-thread locks are desirable).

- Data must stay local as cross-CPU migration has impact on QPI (the bypass is used only for packets). This means that all threads must do all tasks (e.g. implementing a defragmenting thread for the whole application is not a good idea).

Trends: CPU Performance



Network vs CPU speed: Network is the winner.

At 100 Gbit Every Detail Matters

- Before going into details about 100 Gbit traffic monitoring, it is important to understood that at 100 Gbit every detail matters.

- As at line rate we need to process ~150 Mpps, a 10% increase obtained optimising a data structure, means that we can process the same amount of packets of a 10 Gbit interface (14.48 Mpps).

- While the move to 10 Gbit has required many design decisions to be changes, at 100 Gbit the change is even more extreme.

Memory Trends [1/2]

- DDR3 memories have been replaced by DDR4 in modern chipsets/memories.

- Column Access Strobe (CAS) latency, is the delay time between the moment a memory controller tells the memory module to access a particular memory column on a RAM module, and the moment the data from the given array location is available on the module's output pins.

- True Latency (ns) = Clock Cycle Time (ns) x Number of Clock Cycles (CL)

Memory Trends [2/2]

	Speed (MT/s)	Clock Cycle Time (ns)	CAS Latency (CL)	True latency (ns)
DDR3	1333	1,5	9	13,5
DDR3	1608	1,25	11	13,75
DDR4	1866	1,07	13	13,93
DDR4	2133	0,94	15	14,06
DDR4	2400	0,83	17	14,17
DDR4	2666	0,75	18	13,5

- In essence only DDR4-2666 has the same true latency of DDR3-1333.
- These are not details at 100 Gbit as applications must be aware of this fact and use data prefetching wisely to avoid performance penalties.

http://pics.crucial.com/wcsstore/CrucialSAS/pdf/en-us-c3-whitepaper-speed-vs-latency-letter.pdf

Hugepages [1/3]

- CPU uses a page-based mechanism to translate virtual addresses into physical addresses.

- Pages are commonly 4KB, CPU can hold a limited number of virtual to physical mappings in the TLB (Translation Look-aside Buffers).

- TLB entries are hundreds on modern CPUs (a few MBs of memory can be addressed). A TLB hit takes ~1 clock cycle, a miss (TLB update) takes ~20 clock cycles.

Hugepages [2/3]

- All modern Intel CPUs allow larger page sizes (2MB, which is 512 times the default 4KB, or even 1GB) to be used.

- Linux provides an interface to allocate pages with large size called hugepages. Applications must be modified to use hugepages.

```
$ echo 1024 > /sys/kernel/mm/hugepages/hugepages-2048kB/nr_hugepages

$ mount -t hugetlbfs nodev /mnt/huge

$ cat /sys/devices/system/node/node*/meminfo | grep Huge
Node 0 HugePages_Total:  1024
Node 0 HugePages_Free:   1024
Node 0 HugePages_Surp:      0
```

Hugepages [3/3]

Advantages:

- Using a large amount of memory with the default 4KB page size leads to processing overhead for managing the TLB entries: using 2MB hugepages increases performance.

- Large amounts of physical memory can be reserved for memory allocation, that otherwise would fail especially when physically contiguous memory is required.

- Sharing data across applications (e.g. zero copy packets between apps/NICs) is more efficient (up to 10% performance increase) when using hugepages.

Data Alignment [1/2]

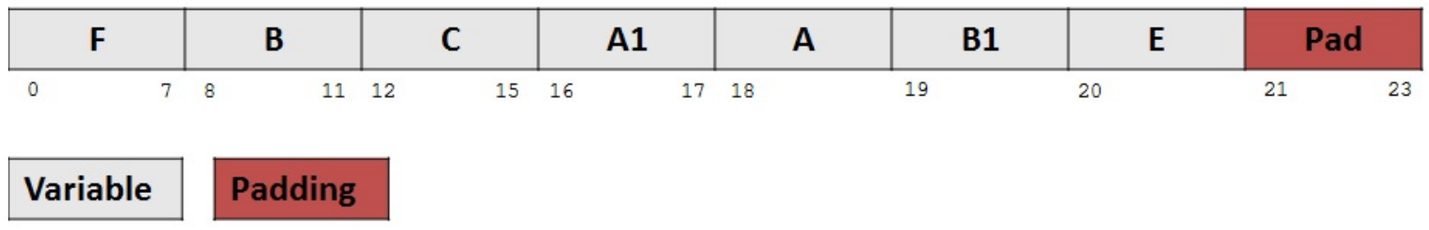
- CPU reads/writes from/to memory in words.
- Data alignment means putting the data at a memory address multiple of the word size.
- Data alignment increases performance due to aligned memory accesses (or if you wish you have a performance penalty when using unaligned data).
- It is a good practice to align data structures, reordering fields and/or using padding.

Data Alignment [2/2]

```
struct s1
{
        u_int8_t a;
        u_int16_t a1;
        u_int8_t b1;
        float b;
        u_int32_t c;
        u_int8_t e;
        double f;
};
```

Size of Struct s1 = **32**

```
struct s2
{
        double f;
        float b;
        u_int32_t c;
        u_int16_t a1;
        u_int8_t a,b1,e;
};
```



Size of Struct s2 = **24**

Data Alignment Tips

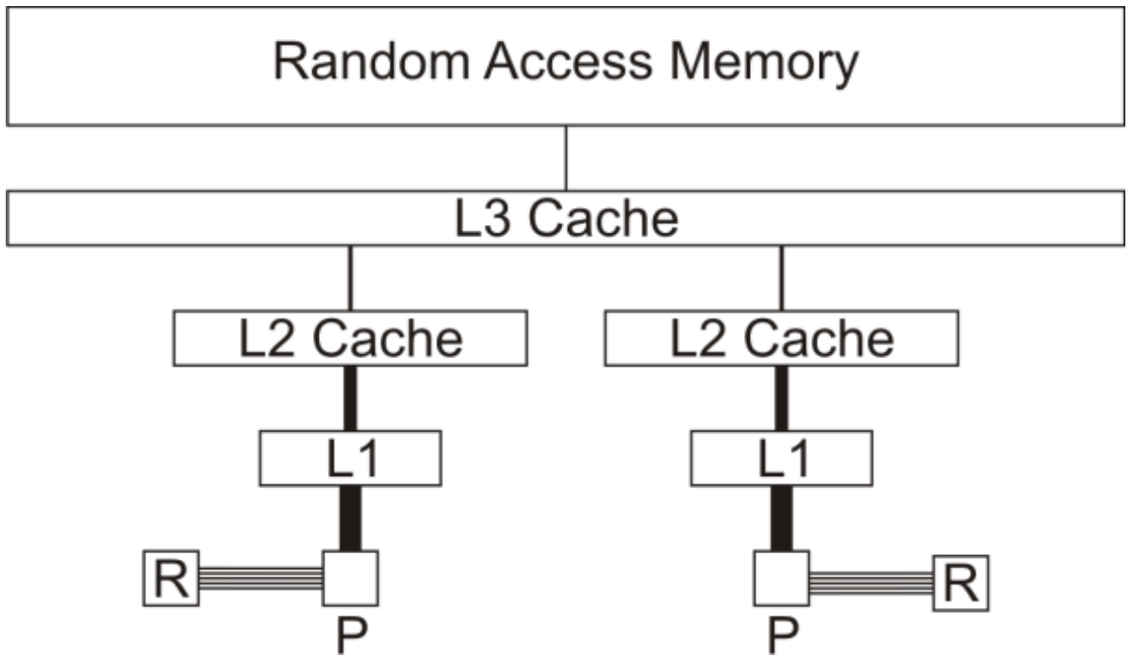
- Padding improves performance at expense of memory: better to reorder whenever possible!

- Load and store operations are generally only atomic when they access aligned memory. Note: some SSE (SIMD) instructions on Intel CPUs require the data to be 128-bit aligned.

- If the data structure is larger than the cache line (usually 64 byte), it is better to cluster together read-only and read-write sections. So in case of change (e.g. flow packet/bytes/ timestamp count update), if read-write variables are all close, the read-only part is not accessed/modified. This optimisation can improve 5-10% CPU performance.

Cache-Aware Structs [1/3]

- A Cache is a small memory which stores blocks of data from frequently-used memory locations. Cache is organised as a hierarchy of more levels (L1, L2, ..).
- A cache miss is an attempt to access a memory address which is not available in the cache, resulting in an access in a memory level with higher latency.
- A typical cache configuration is:
  - 64B Cache line
  - 32KB L1 Data Cache size
  - 256KB L2 Cache size
  - 2-55MB L3 Cache size (shared)



Cache-Aware Structs [2/3]

- Allocating data structures properly aligned to cache lines reduces cache misses.

- Rearrange data structures for optimising cache locality by grouping small items that are frequently used together into a struct, and forcing the struct to be allocated at the beginning of a cache line. This:
  - Guarantees that the structure is loaded into the cache as soon as one of the items is accessed.
  - Avoids that more cache lines than what is the minimum number needed are loaded.

Cache-Aware Structs [3/3]

- Rearrange data structures by grouping items based on threads, and use padding to isolate each group. Keep this in mind when using hashes that instead spread information across all the available memory.
- If a data structure is accessed from more than one thread, and each thread is accessing a portion of the data structure, having those sub-structures unaligned to cache lines leads to performance degradation.

Part 2

100 Gbit Packet Processing

From Theory To Practice

- So far we have covered 100 Gbit in theory.
- It is now time to see what we can do in practice trying to develop a 100 Gbit probe using x86 hardware.
- This to learn what kind of performance can be obtained, and what are the issues that have been encountered.

Trends: User Requirements [1/2]

- For years the industry and community focused mainly on packet capture/filtering.

- Applications were relatively simple and self-contained: network security, traffic monitoring, high-frequency trading, packet-to-disk….

- Cost of data center space, and desire to deploy fewer appliances to run the monitoring applications required, are driving towards a single monitoring box.

Trends: User Requirements [2/2]

- 100 Gbit (and partially 40 Gbit) are raising the bar once more, and (FPGA-based) NICs are solving “just” the packet capture problem.

- Unfortunately “packet capture acceleration” is no longer enough as we often need to combine it with multi-app traffic distribution, balancing, and pipelining in order to collapse functionalities currently done on multiple boxes into one high-speed converged box.

Trends: Application Performance

- IDS/IPS Applications (e.g. Snort, Bro and Suricata) are CPU bound. Suricata on a E5-2690v2 3GHz (10 cores+HT all in use) and a FPGA NIC can do ~14.1 Gbps with real traffic (~512 byte packets or more in average).
- ntop's NetFlow probe (nProbe) can process ~3.5 Mpps (per core, so line rate, with real traffic) on a Intel E3-1230v3.

Problem Statement

- Is it possible to implement line-rate 100 Gbit (or 10x10 Gbit) flow based monitoring on a x86 box ?

- Can we instrument a flow-monitoring tool to egress/drop selected traffic to other applications that can further process the traffic ?

- Can we combine flow visibility with other functionality such as IDS or packet to disk on the same box ?

- Can we build multi 1/10 Gbit flow sensors using low-cost hardware?

What is Still Missing?

- Modern packet processing frameworks (e.g. PF_RING ZC or Intel DPDK) allows to be efficiently received/transmitted (line rate, 64 bytes packet) from/to ethernet devices or in-host/VM queues.

- Existing NetFlow probes (e.g. nProbe) have been designed in the 1/10 Gbit days, so fast but not 100 Gbit-ready (too in depth packet analysis).

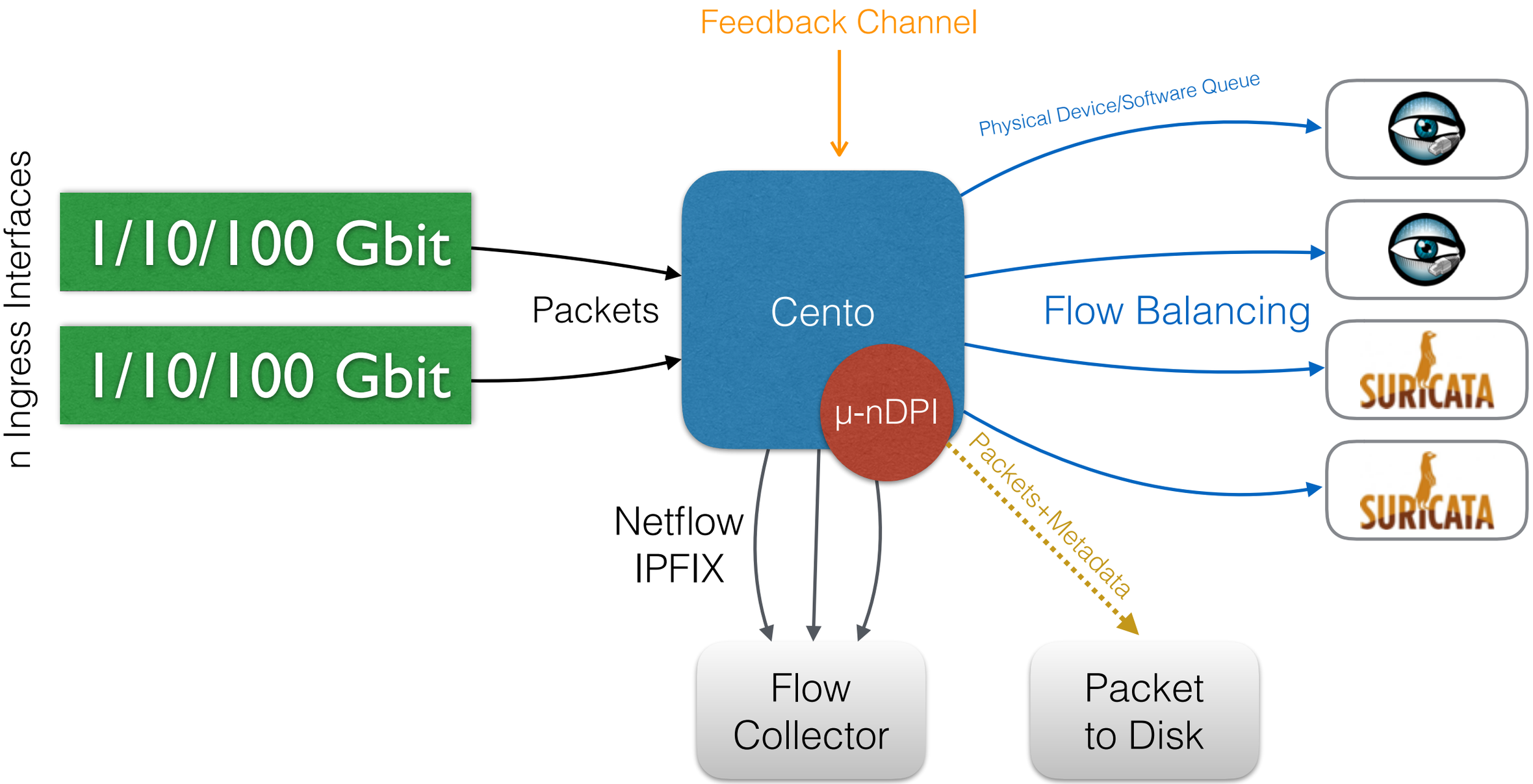
- Can we design (from scratch) a new 100 Gbit probe able to do both flow processing and other tasks such as distribute traffic to other applications in order to fully exploit modern multi-core Intel boxes ?

Welcome to nProbe Cento

- Leveraging more than 15 years of high-speed packet processing and NetFlow monitoring, we decided to code a new flow-based sensor from scratch.

- Cento ("one hundred" in Italian) has been designed as the first component of a monitoring system, the one that captures ingress packets, classifies them via DPI (Deep Packet Inspection), and performs optional actions on selected packets.

nProbe Cento Architecture



Cento Design Principles [1/3]

- Export in NetFlow v5/v9/IPFIX, JSON, Text (soon also Kentik Flow and Kafka).

- Coded in “simple” C++ to avoid runtime slowdowns.

- Ingress traffic is split across multiple interfaces/memory areas by RSS-like techniques: one thread per interface.

- Avoid locks when possible: one flow exporter/cache per sensor thread.

- Short locks when unavoidable: in case of synchronisation (e.g. during export) locks enclose very short code and are clustered (e.g. one lock every 500k flow export).

Cento Design Principles [2/3]

- All data structures are cache-line aligned (typically 64 bytes).

- Large hash tables (used to host flows) put a lot of pressure on memory and so prefetching is widely used to increase performance.

- As ZC prefetches packets as well, prefetching must be carefully used to avoid slowing down the application instead of accelerating it (big problem with “smart” FPGA-based NICs).

Cento Design Principles [3/3]

- Ingress packet processing has maximum priority over other tasks (e.g. flow export), and thus multiple low-priority actives can be collapsed on the same core, while dedicating a complete core to packet processor threads.
- If cento has been configured to egress packets. This is done (by PF_RING ZC) by putting the packet reference onto the egress queue/device (no memory copy unless FPGA-adapters are used as they allocate memory on a custom/proprietary fashion).

Comparing Memory Allocation Techniques

- We compared the application performance using a few flow cache allocation mechanisms.

E3-1230v3 (4C+HT) - 4 x dummy Interface, 500k flows/interface

Huge Pages (HP)	66.7 Mpps
No huge pages, single posix_memliagn() with mlock()	**64.6 Mpps**
No huge pages, single posix_memliagn() without mlock()	66.7 Mpps
new/delete	**67.4 Mpps**

Hardware Hashing and Prefetching

- Modern network adapters are able to provide both packet payload and metadata. Inside the packet metadata many adapters (e.g. Intel) can store a packet hash (e.g. used for RSS) in addition to hardware timestamp.

- We planned to use this hash to both:
  - avoid hashing in software
  - prefetch hash buckets in order to improve the performance.

- Unfortunately when using the 5-tuple hash (e.g. Toeplitz Hash on Intel) the number of hash collisions was higher than the software hash we implemented, and thus the performance was worse. Not a good idea, thus we are staying with the software hash.

100 Gbit vs. 3x40 vs. 5x25 vs.10x10

- A 100 Gbit interface cannot be handled with one CPU core, so the traffic must be necessarily split.

- Hardware packet hashing (RSS) allows ingress traffic to be divided into multiple virtual interfaces.

- As 100 Gbit interfaces are still relatively expensive, splitting a 100 Gbit link into various 40/10 Gbit interfaces (that can also use RSS) is also an option.

- Switches or appliances can be used to ease migration towards 100 Gbit by splitting ingress traffic onto various 10/25/40 Gbit egress interfaces.

DPI Impact on Performance

- ntop develop and maintains an open source DPI toolkit named *nDPI* that supports over 200 protocols (e.g. Skype, SSL, BitTorrent…).

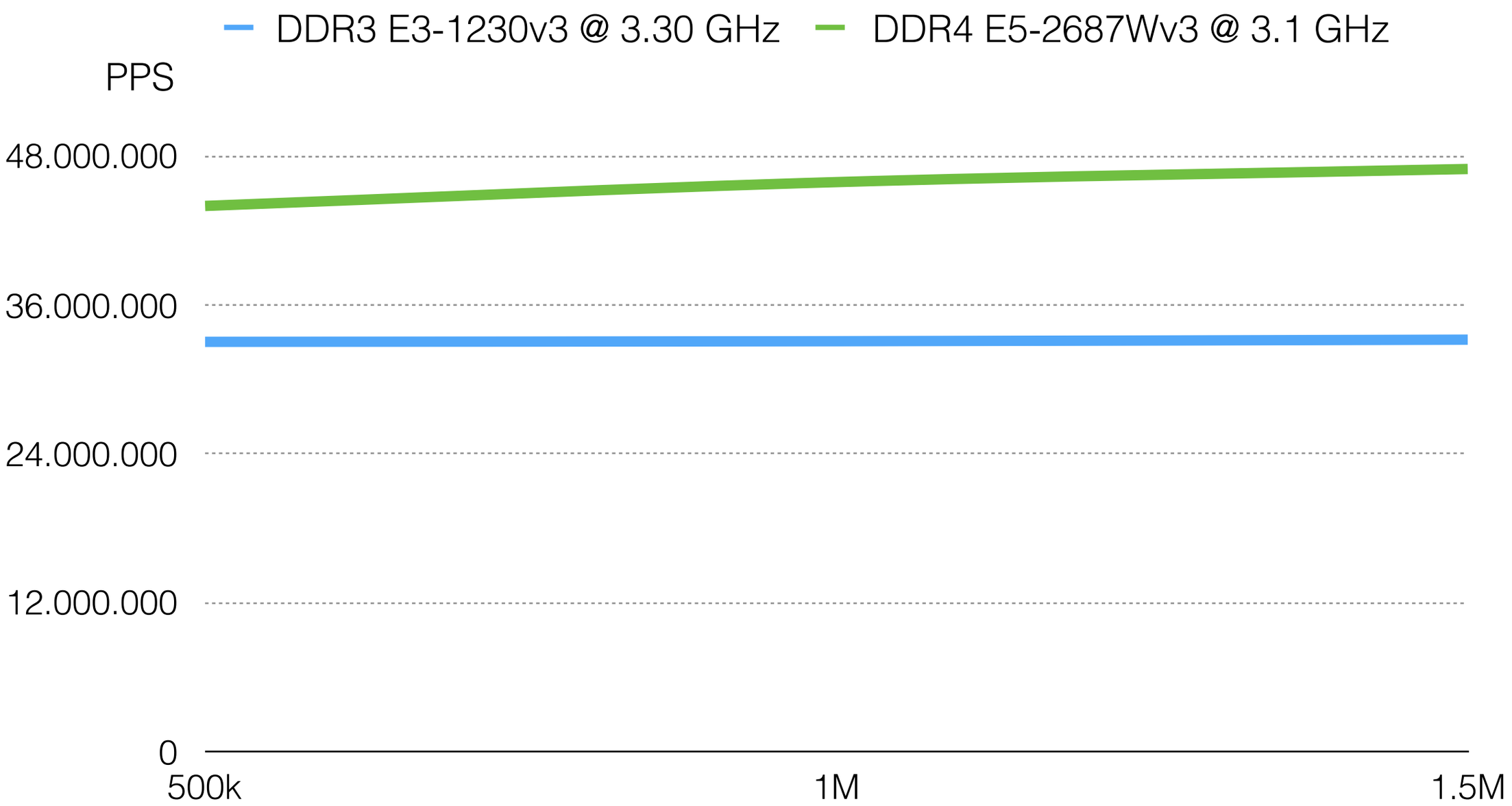
- Cento can optionally use nDPI to identify application protocols. Detection happens at flow start using less than 10 packets.
- Enabling nDPI with all protocols on Cento the performance is reduced by 20/40% (depends on flow duration).
- We have developed a μ-nDPI that contains just a few protocols such as HTTP/SSL/DNS and enabling it, the performance degradation it is almost unnoticeable.

Performance Evaluation

In order to evaluate the performance we have tested cento on:

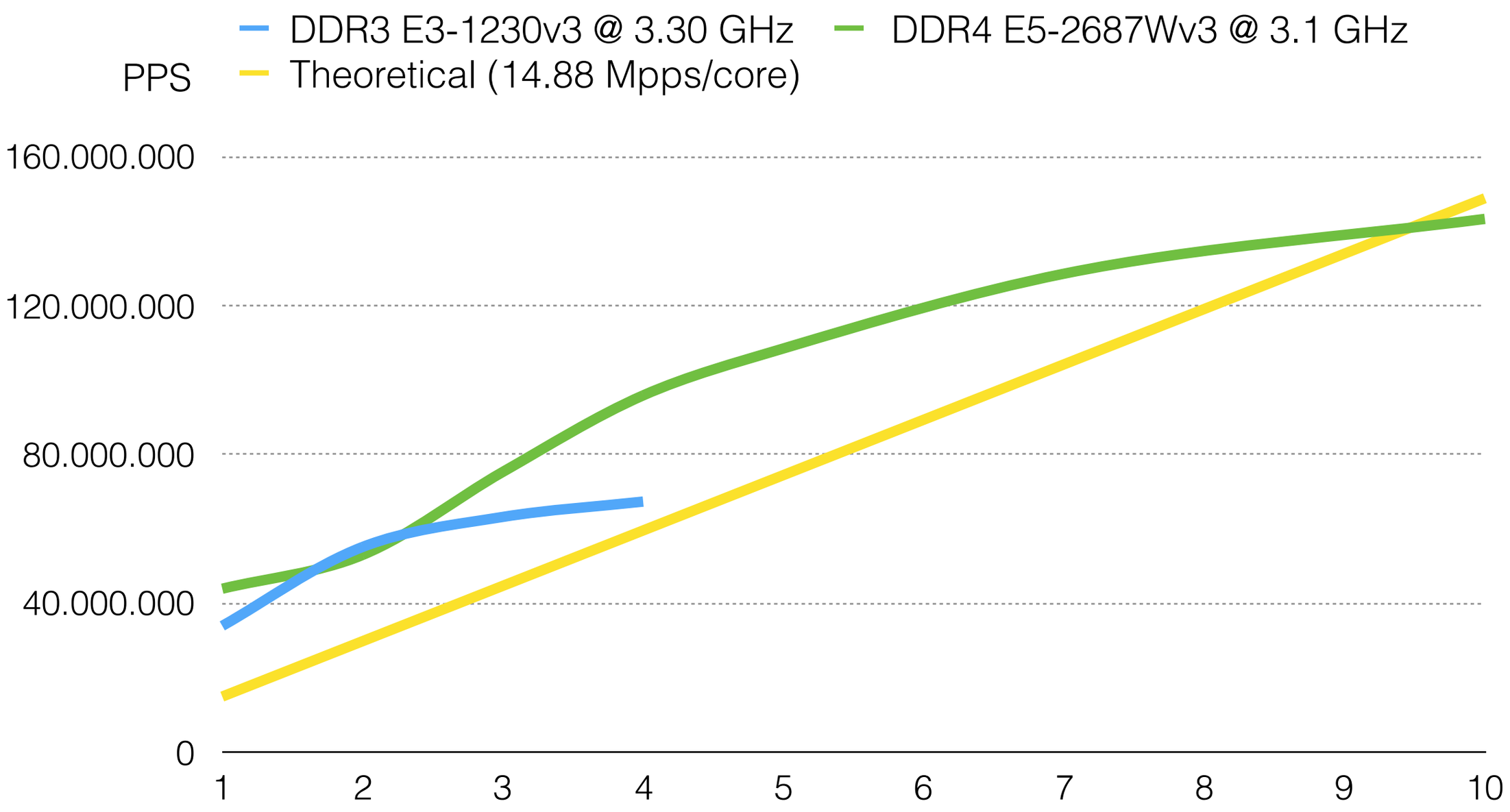
- “Dummy” interface (32k packets/interface in memory to simulate impact on CPU cache).
- PF_RING ZC on Intel 82599 and X710.
- 100 Gbit interfaces (Napatech).

Processing Speed vs Flow Cache Size



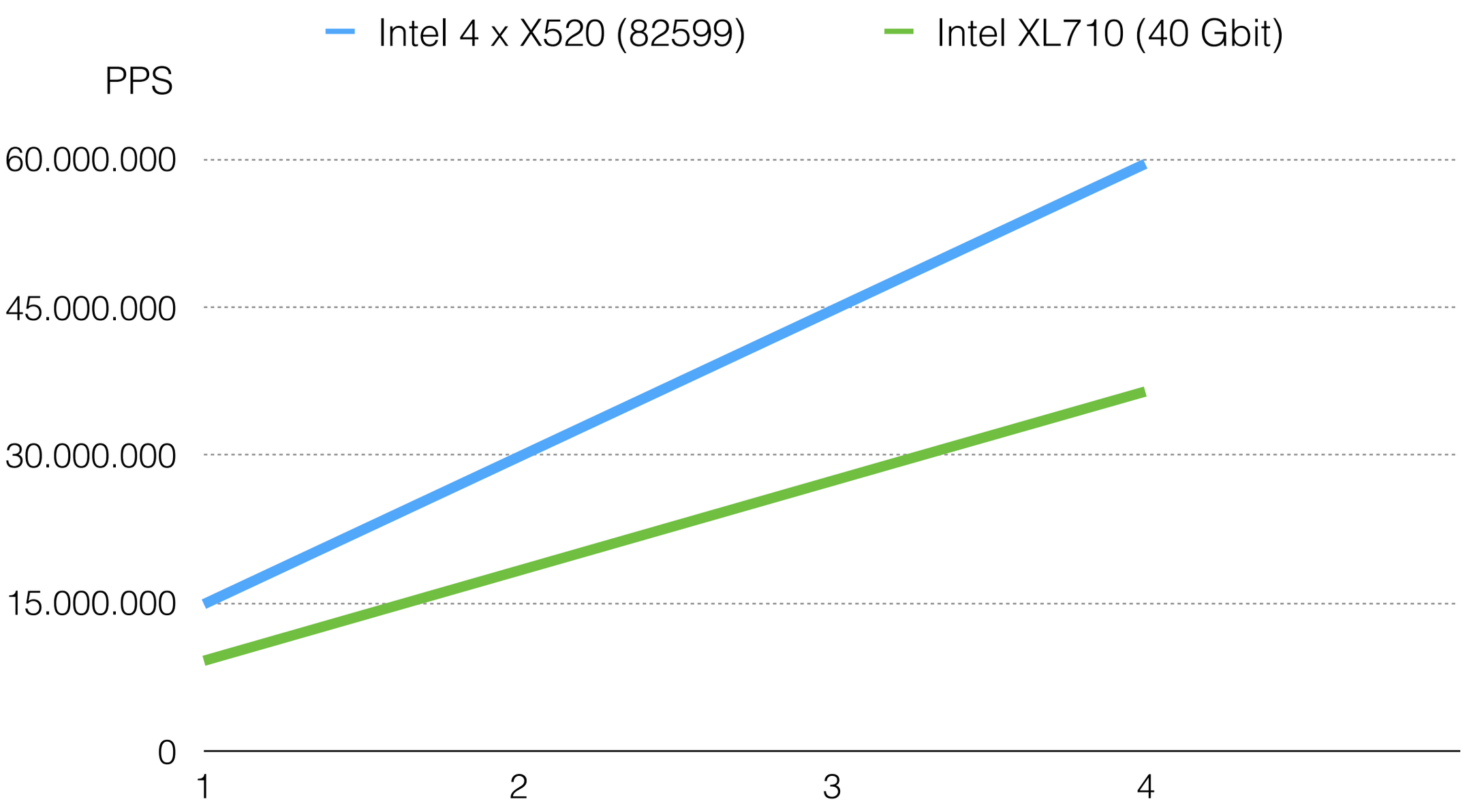
Single core Processing Performance vs Flow Cache Size (Dummy Interface)

Scalability: Dummy Interface



Per-core Processing Performance (500k Flows/Interface, Dummy Interface)

Scalability: Intel multi-10 Gbit



Per-core Processing Performance (500k Flows/Interface, Intel E3-1230v3)

Scalability: Real 100 Gbit Traffic
More Streams (70 bytes)

26 Streams

```
01/Dec/2015 16:00:12 [nt:stream0] [5'077'577 pps/3.49 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:13 [nt:stream1] [5'076'595 pps/3.49 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:13 [nt:stream2] [5'078'730 pps/3.49 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:14 [nt:stream3] [5'079'094 pps/3.49 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:14 [nt:stream4] [5'079'561 pps/3.49 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:15 [nt:stream5] [5'079'930 pps/3.49 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:15 [nt:stream6] [5'079'448 pps/3.49 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:16 [nt:stream7] [5'080'528 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:16 [nt:stream8] [5'080'557 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:17 [nt:stream9] [5'081'058 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:17 [nt:stream10] [5'080'978 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:18 [nt:stream11] [5'081'977 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:18 [nt:stream12] [5'082'670 pps/3.50 Gbps][100/200/0 act/exp/drop flows][0/0 RX/TX put drops]
01/Dec/2015 16:00:19 [nt:stream13] [5'082'088 pps/3.50 Gbps][100/200/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:19 [nt:stream14] [5'082'256 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:20 [nt:stream15] [5'082'632 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:20 [nt:stream16] [5'083'798 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:21 [nt:stream17] [5'084'046 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:21 [nt:stream18] [5'083'934 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:22 [nt:stream19] [5'084'371 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:22 [nt:stream20] [5'084'487 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:23 [nt:stream21] [5'084'625 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:23 [nt:stream22] [5'084'374 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:24 [nt:stream23] [5'083'037 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:24 [nt:stream24] [5'083'832 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:25 [nt:stream25] [5'083'563 pps/3.50 Gbps][100/100/0 act/exp/drop flows][0/0 RX/TX pkt drops]
01/Dec/2015 16:00:25 [cento.cpp:513] Actual stats: 132'125'746 pps/0 drops
```

napatech ~100% Line Rate

Final Remarks

- 100 Gbit traffic monitoring is challenging in terms of number of packets to process and hardware configuration complexity.

- NUMA systems and RSS can be combined to balance the load across cores and thus make 100 Gbit monitoring possible.

- We have demonstrated that 100 Gbit traffic monitoring is feasible using an x86 server and a 100 Gbit interface as well multiple 10/40 Gbit interfaces, without custom FPGA acceleration for flow processing or generation.

- It is not uncommon to have 10/14 physical CPU cores, and so it is possible to take advantage of them, so you can effectively combine 100 Gbit flow-monitoring with other activities.

- In essence writing optimised code on modern harder it is possible at 10 Gbit line rate to generate flows with as few as 2 cores (1 processing + 1 export) while leaving the remaining cores for Bro, Suricata, n2disk, Wireshark, Snort…