@qant

15th ANNUAL WORKSHOP 2019

FASTER FABRICS RUNNING AGAINST LIMITS OF THE OPERATING SYSTEM, THE PROCESSOR AND THE I/O BUS

Christoph Lameter, <cl@linux.com> R&D Team Lead

March 12, 2019

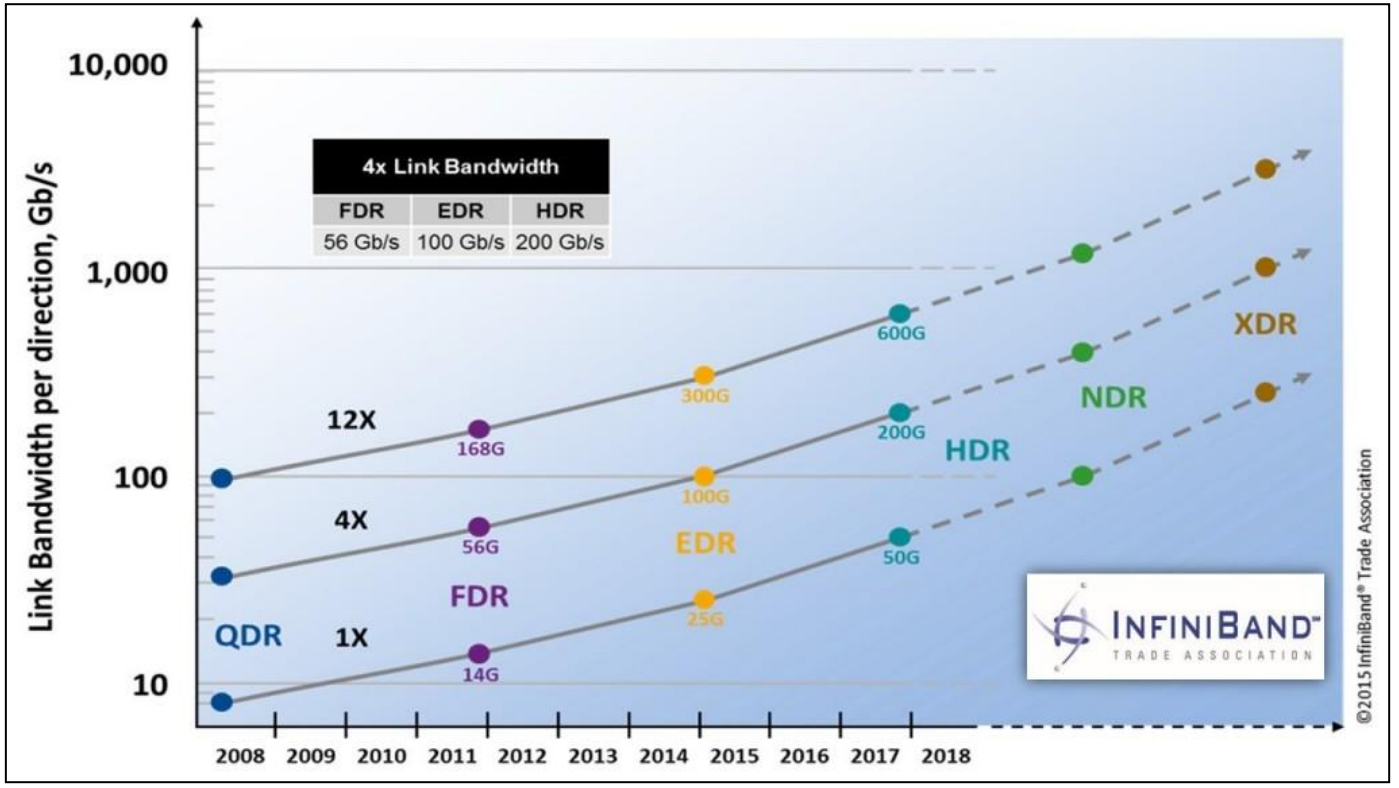
Jump Trading LLC

Interconnects/Fabrics are getting too fast

The Problem

Have we been too successful?

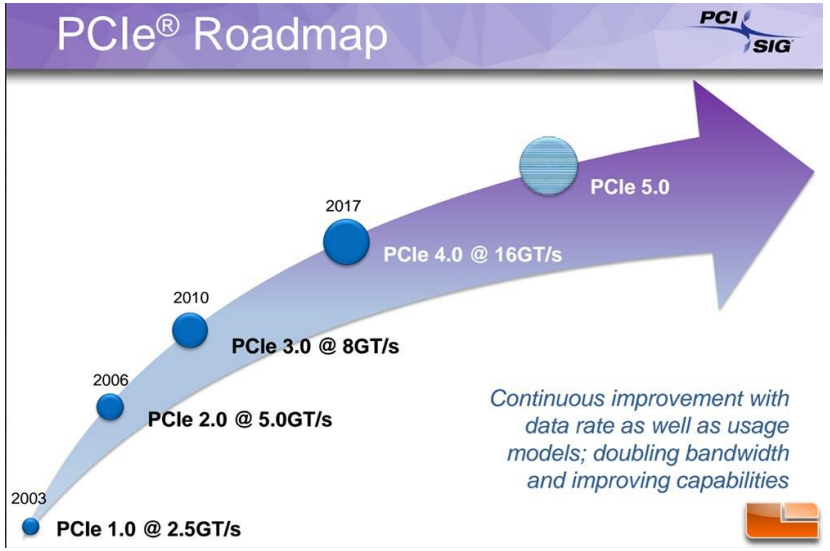
- **We have stopped upgrading Infiniband Fabrics**
- **A mismatch to processor and memory speeds.**
- **The PCI-E 3 barrier**
  - ~1Gbyte/s per lane.
  - 8Gbps vs 25Gps on EDR and 50Gpbs on HDR.
  - Lots of scarce PCI-E lanes required for full bandwith (16/32 of 44 available!!)
  - A decade long use of PCI-E 3 and no end is in sight.
- **The roadmap well is roadkill**



Excess capacity is not useful for general purpose computing. Special offload hardware can only use that performance (GPUs and FPGAs)

PCI E woes

- PCI E 4 doubles the data rate. So about 2 Gbytes/sec
- PCI-E 4 is stalled. Rumors of PCI E 4 in 2019 did not pan out. Maybe AMD has something later this year.
- Reliability issues due to difficulties with high frequency.
- PCI-E latencies are a problem. Latencies with other interconnects are faster.
- The whole Intel general compute paradigm becomes more and more of an issue. It is well supported and mature but not suitable for High Performance Compute.



- OpenCAPI 3.0 is an alternative independent of PCI-E and based on 25GT line rate. About 3x of PCI-E but OpenCapi is a technology only available for IBM Power9 processors with very limited device support.

The fastest interconnect is no longer Infiniband but NVLink running at 300GByte per second. Also connects to Power9 processors.

Memory Bandwidth

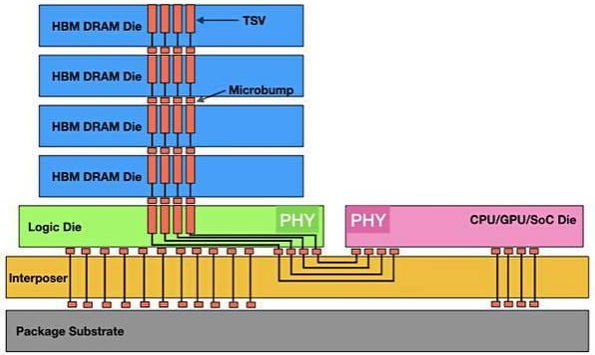
Processors have been adding more memory channels. Intel has 6. AMD 8 channels. That in turn requires more pins and leads to difficulties with routing all these lanes.

~20GBs per channel. Intel's 6 channels result in ~ 120GBs. AMD 8 come to ~150GBs.

Individual thread throughput is ~12-20 Gigabytes per sec to and from memory.
A single thread is challenged to keep up with the data rate on >100G network.

So data must be processed by multiple cores in any case.

GDDR and HBM memory are an interesting to look at because of the higher bandwidth. The use of these new types of memory by NVIDIA GPUs with NVLINK shows that radically higher bandwidths are possible.



The Processor Bottleneck

A recent attempt was made to scale up the Linux network stack by Jesper Brouer.

10 Gbps works. At 10G a 1500 byte packets has a latency budget of 1.2 microseconds for processing. So packet routing in the kernel is possible.

But at 100 Gbps? 120ns per packet.
We cannot even fetch a single cache line.
Large "packets" can be processed but general handling must be in hardware.

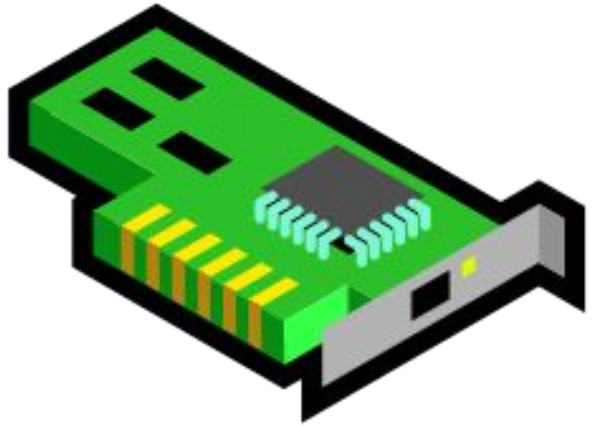
XDP can provide a packet classifier and the routing of packets to processes is delegated to hardware.

So the latency budget not sufficient for packet handling in the classic OS network paradigm.

Existing Solutions

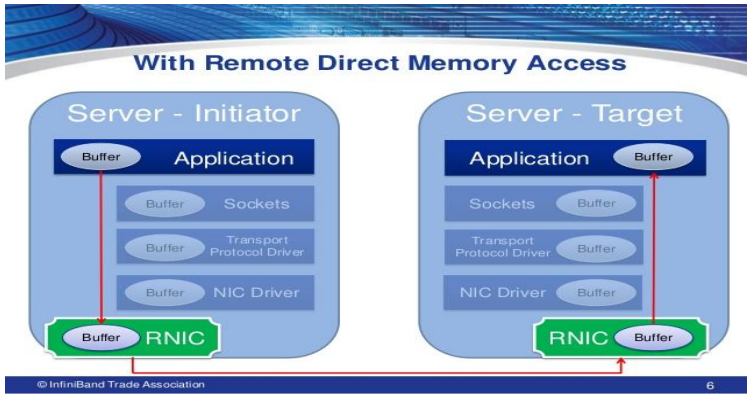
“Normal” (OS Stack) NIC developments

- **Support of an extensive set of offload techniques that has been growing for the last couple of decades.**
  - Checksum processing
  - Package aggregation and segmentation
  - Interrupt aggregation: NAPI
- **XDP related: Hardware flow detection and direction to OS threads.**
- **Full routing in hardware instead of software through programmable redirectors and local queues in the network stack.**
- **Per hardware thread queuing for lockless operation**
- **Multiple virtual NICs that can receive different flows**
- **NIC segmentation for Virtual Guests.**



RDMA NIC developments

- QP endpoints implicitly direct the flow to hardware threads. Flow steering is therefore inherently already provided by the existing RDMA semantics.
- Flow steering to route traffic that does not have a explicit QP endpoint
- Packet classification. Special operations on inbound and outbound packets.
- Virtualization support
- Build in switch functionality.
- Single NIC interfaces with multiple servers
- Device Local memory mitigating PCI E problems
- PeerDirect for connection to other PCI E peers without going through the hosts memory and for transferring into device local memory of another PCI E device (in particular NVME devices to support NVMEoF etc)

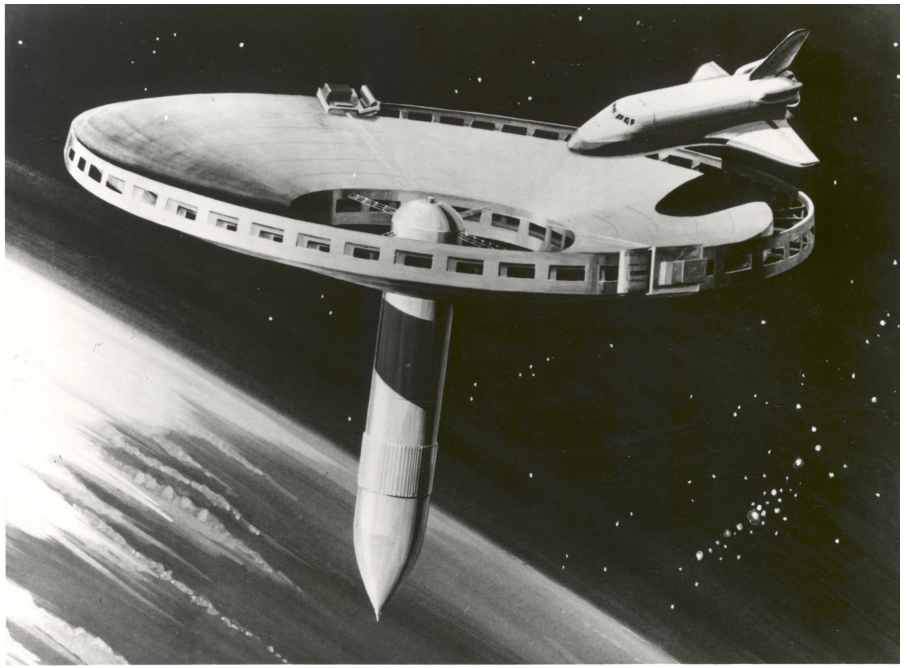


In line Processing using Specialized Chips

- Special hardware network elements that can process data at line rate (currently only for 10G)
- Mostly used for individual ports on Ethernet switches
- Timestamping
- VLAN tagging
- Packet diagnostics
- Maybe also simple packet modifications in the future

The Far or not so Far Future

- **Someone ought to be thinking about the problem of handling a terabit real time video stream as required to transmit 3d images (holograms anyone?)**
- **Exascale Computing challenges.**
- **How to move even more massive amounts of data.**
- **Do we need a new hardware architecture for HPC?**
- **Maybe we see one emerging with**
  - Specialized computing devices like GPUs
  - Super high speed networking in NVLINK
  - New types of RAM (HBM/GDDR)



RDMA stack evolution?

RDMA Stack Challenges

Lots of different APIs are being added to the RDMA stack to accommodate new technology and devices.

The RDMA stack becomes more an almagam or a collection of various features and subprotocols that end up being vendor specific.

The challenge is to come up with a mechanism to incentivize adoption of common APIs instead of vendor specific ones.

Any help to simplify the RDMA stack and API is appreciated. Help wanted!

Given the current trends maybe what I am going to say on the following pages is just a pipe dream that will flounder because of the vendor specific solutions being pushed.

“RDMA” a different take

- **RDMA Data may never reach main memory. Both endpoints of the RDMA operation may be remote.**
- **The RDMA stack may need to become a control API for diverse hardware devices processing flows of data?**
- **A mechanism is needed to describe a “RDMA” data flow through a variety of**
  - NICs
  - PCI-E devices
  - In line processing elements
- **That is more like how a network controller functions through redirection of streams based on load and network processing elements available.**
- **Maybe using programmable connections between devices (like possible with the new breed of “layer 1” switches by Arista and Exablaze would allow a programmable traffic flow through these elements.**

- **Software defined Networking the “RDMA” way?**

@qant

15th ANNUAL WORKSHOP 2019

Quo vadis, RDMA?

Christoph Lameter <cl@linux.com>

Jump Trading LLC