NaNet Status

Alessandro Lonardo (INFN)

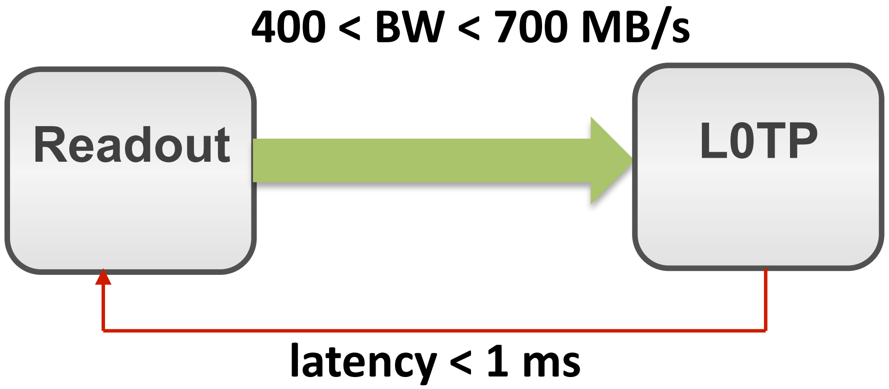
On behalf of the NaNet collaboration:

R. Ammendola[a], A. Biagioni[b], O. Frezza[b], G. Lamanna[c], F. Lo Cicero[b], A. Lonardo[b], F.Pantaleo[c], P.S. Paolucci[b], R. Piandani[c],L. Pontisso[d], D.Rossetti[b], F. Simula[b], L.Tosoratto[b], M. Sozzi[c], P. Vicini[b]

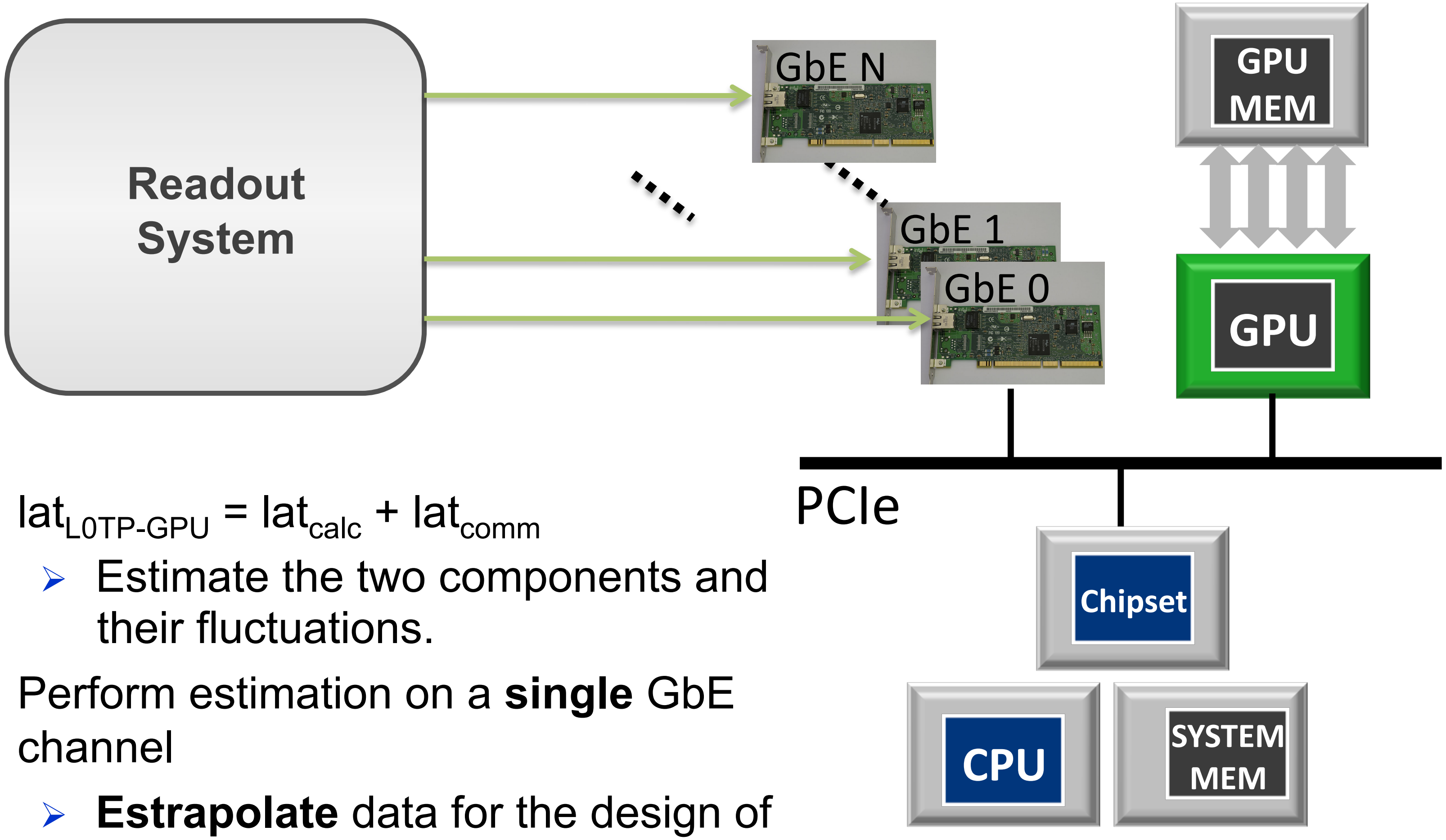
(a) INFN Sezione di Roma Tor Vergata (b) INFN Sezione di Roma (c) INFN Sezione di Pisa
(d) Università di Roma "Sapienza"

NA62 RICH L0 Trigger Processor Requirements

- ❑ Network Protocol: UDP over GbE.
- ❑ System Throughput: 10 MEvents/s
  - Network bandwidth between 400 MB/s and 700 MB/s, depending on data protocol choice (hit finetime data,…).
- ❑ System response latency < 1 ms
  - determined by the size of Readout Board memory buffer storing event data to be passed to higher trigger levels.



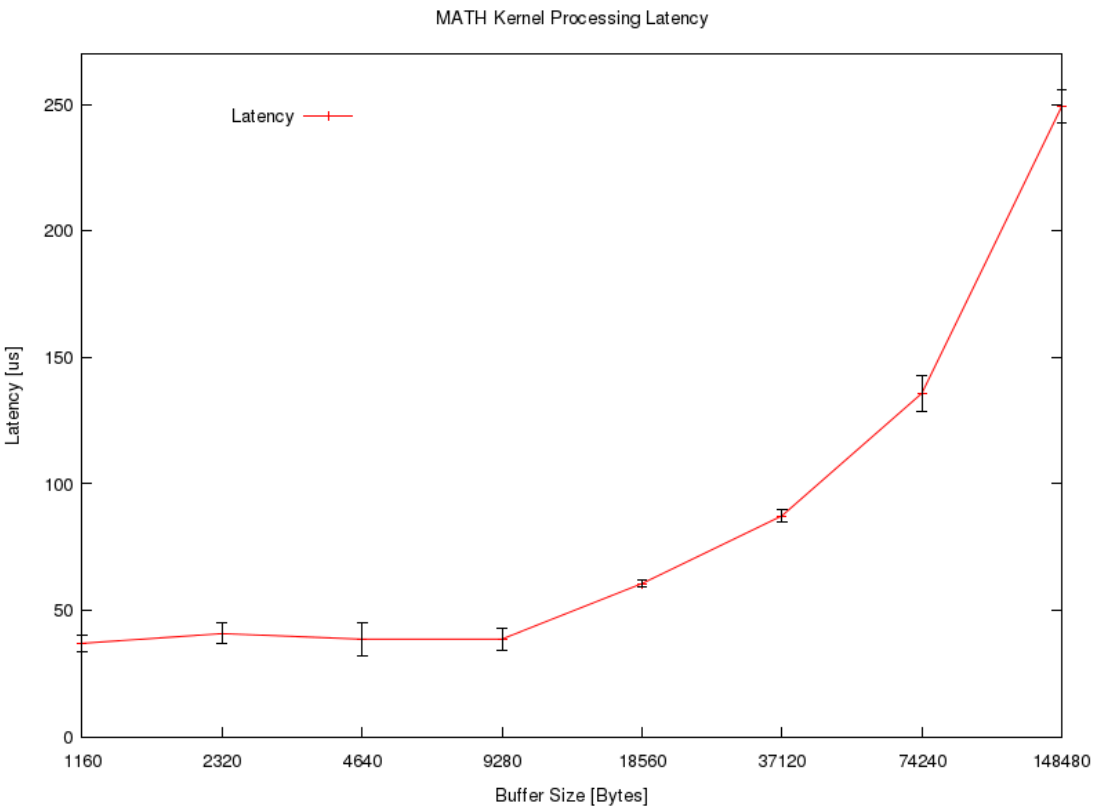
GPU-Based RICH Level 0 TP System Latency Estimation



- $\mathrm{lat}_{\mathrm{L0TP\text{-}GPU}} = \mathrm{lat}_{\mathrm{calc}} + \mathrm{lat}_{\mathrm{comm}}$
  - Estimate the two components and their fluctuations.
- Perform estimation on a **single** GbE channel
  - **Estrapolate** data for the design of the “full bandwidth” system.

System Latency Estimation
Processing Latency

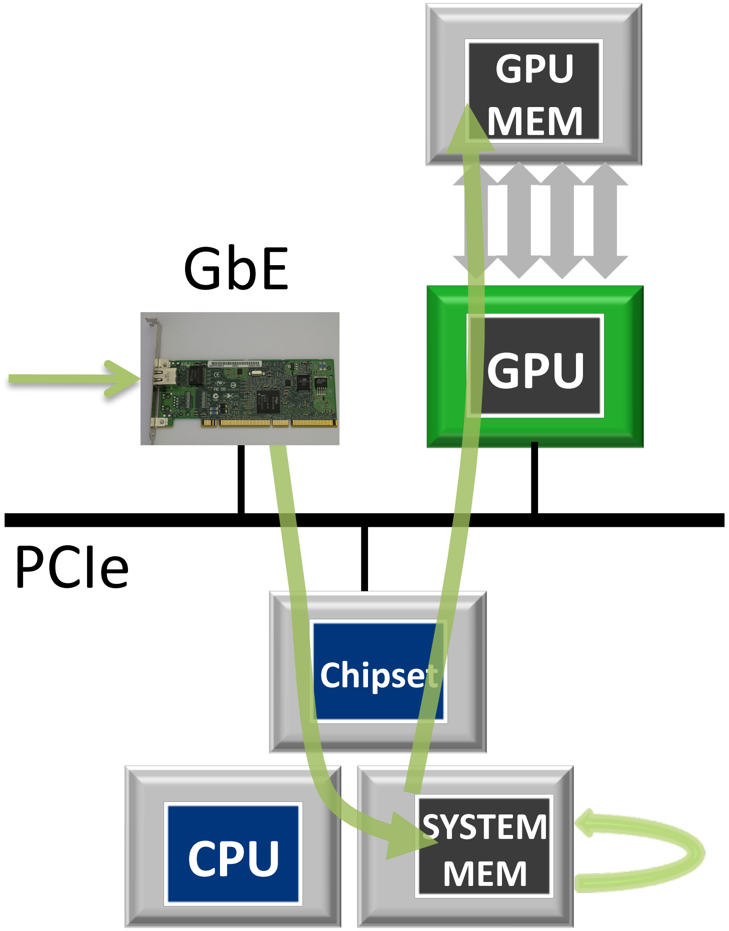
- **lat$_{calc}$** : time needed to perform rings pattern-matching on the GPU with input and output data on device memory using the **MATH** algorithm.
- Test setup: Supermicro X8DTG-DF motherboard (Intel Tylersburg chipset), dual Intel Xeon E5620, Nvidia M2070, Intel 82576 Gigabit Network Connection, kernel 2.6.32-358.6.2.el6.x86_64, CUDA 4.2, Nvidia driver 325.15.



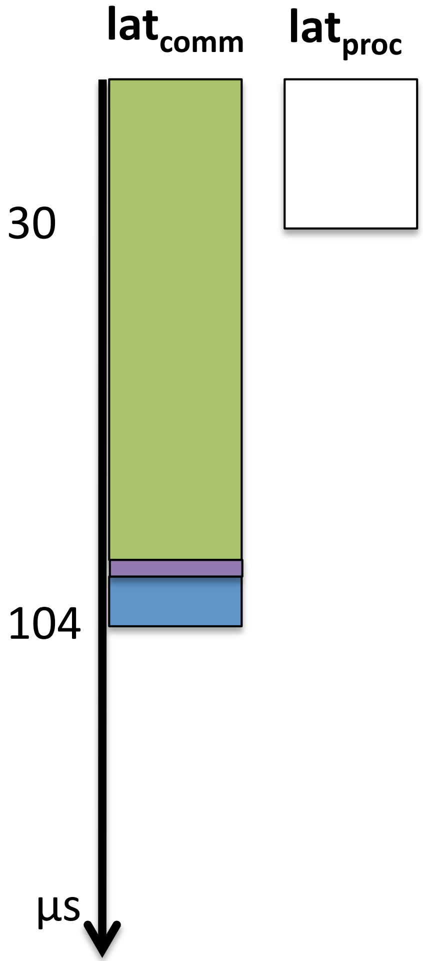
System Latency Estimation
Communication Latency

- **lat**$_{\text{comm}}$: time needed to receive input event data from GbE NIC to GPU memory.

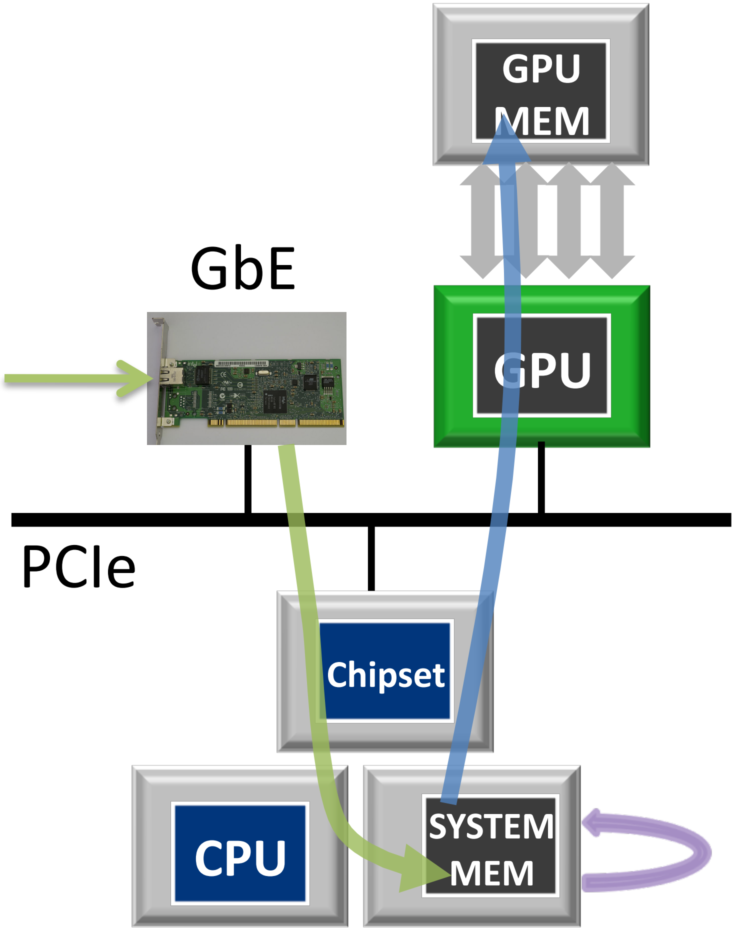
- We considered 3 alternatives for the network subsystem:
  - Standard Network Interface Card (Intel 82576 based)
    - Standard Software Stack (OS Kernel, Drivers, libraries…)
    - Real time OS
  - Custom NIC (NaNet-1)
  - PF_RING low latency drivers, not discussed here.



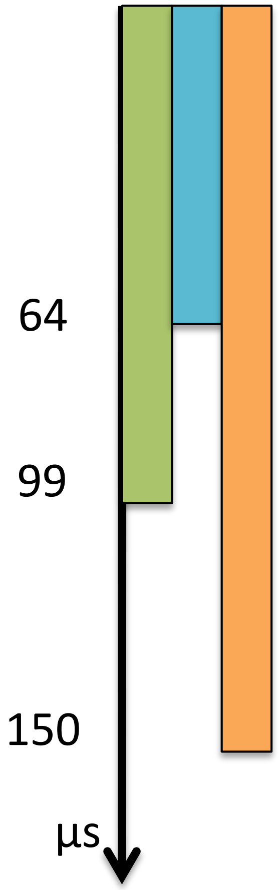
Communication Latency
Standard GbE NIC / SW Stack



- 20 events data (1400 byte)
- $lat_{comm} \cong$ 104 µs
- Total latency is dominated by communication latency: $lat_{comm} \cong 3.5 \times lat_{proc}$ (much higher considering kernel activation latency)



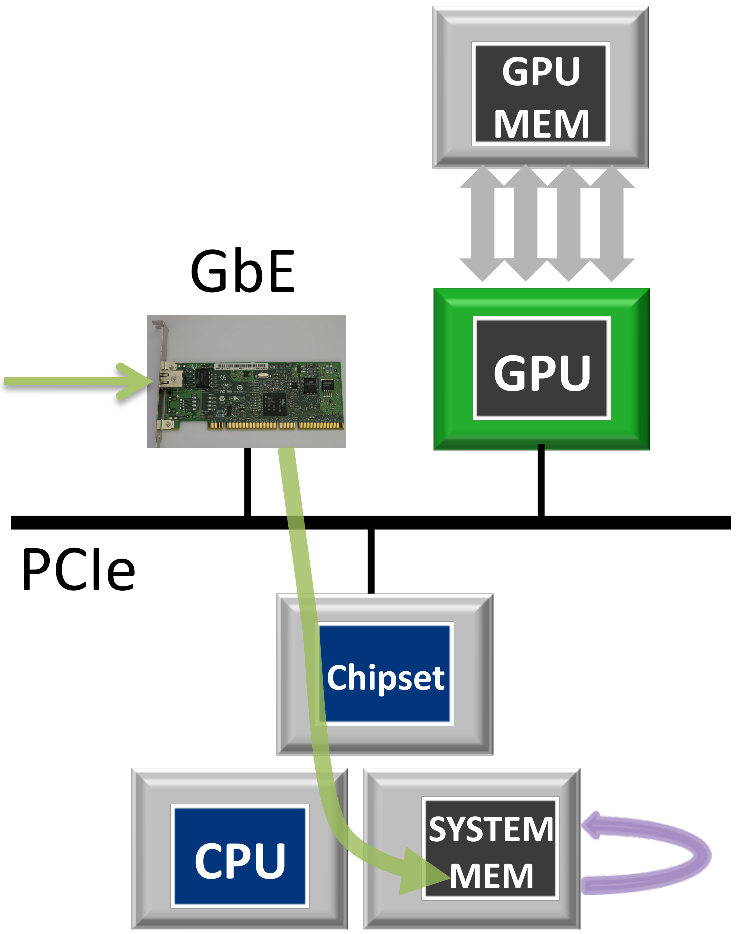
Standard GbE NIC / SW Stack Communication Latency Fluctuations



- Fluctuations on the **GbE component** of lat_{comm} may hinder the real-time requisite, even at low events count buffer sizes.

```
sockperf: Summary: Latency is 99.129 usec
sockperf: Total 100816 observations; each
percentile contains 1008.16 observations

sockperf: ---> <MAX> observation = 657.743
sockperf: ---> percentile 99.99 = 474.758
sockperf: ---> percentile 99.90 = 201.321
sockperf: ---> percentile 99.50 = 163.819
sockperf: ---> percentile 99.00 = 149.694
sockperf: ---> percentile 95.00 = 116.730
sockperf: ---> percentile 90.00 = 105.027
sockperf: ---> percentile 75.00 = 97.578
sockperf: ---> percentile 50.00 = 96.023
sockperf: ---> percentile 25.00 = 95.775
sockperf: ---> <MIN> observation = 64.141
```



Standard GbE NIC / Real-Time Kernel

Might a Real-Time kernel come to our rescue?

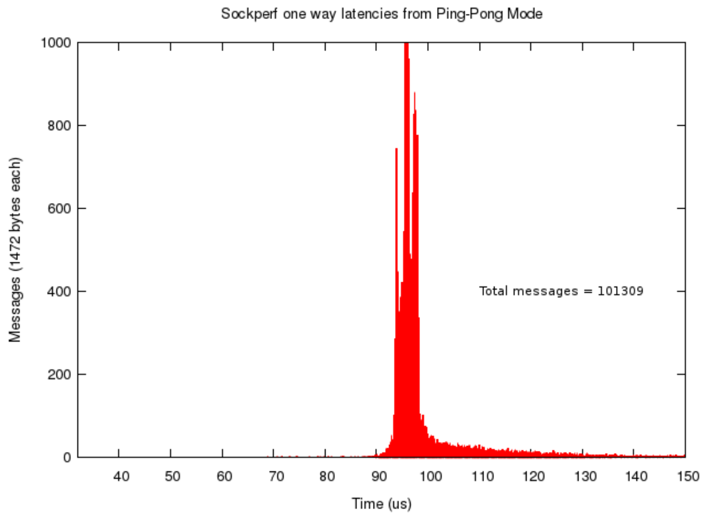
Main features of such kernels:

- predictability in response times
- reduced jitters
- microsecond accuracy
- improved time granularity

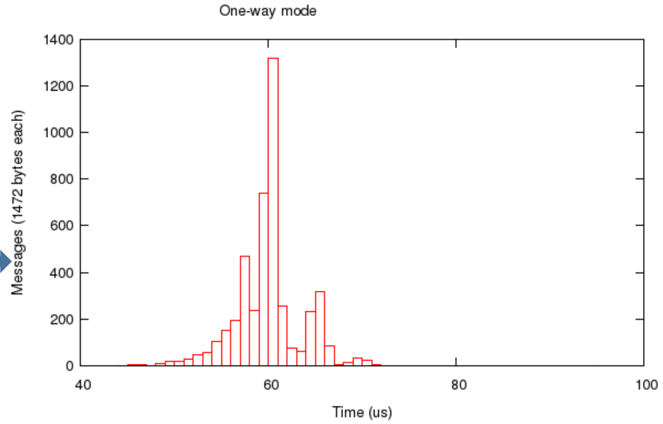
...but not a panacea...

To avoid other possible sources for latency, the CPUSPEED and IRQBALANCE services has been stopped. The INTERRUPT moderation was disabled either.

vanilla kernel: 2.6.33



kernel 2.6.33.9-rt31-EL6RT



NaNet-1: a custom 1GbE NIC with GPUDirect RDMA capability

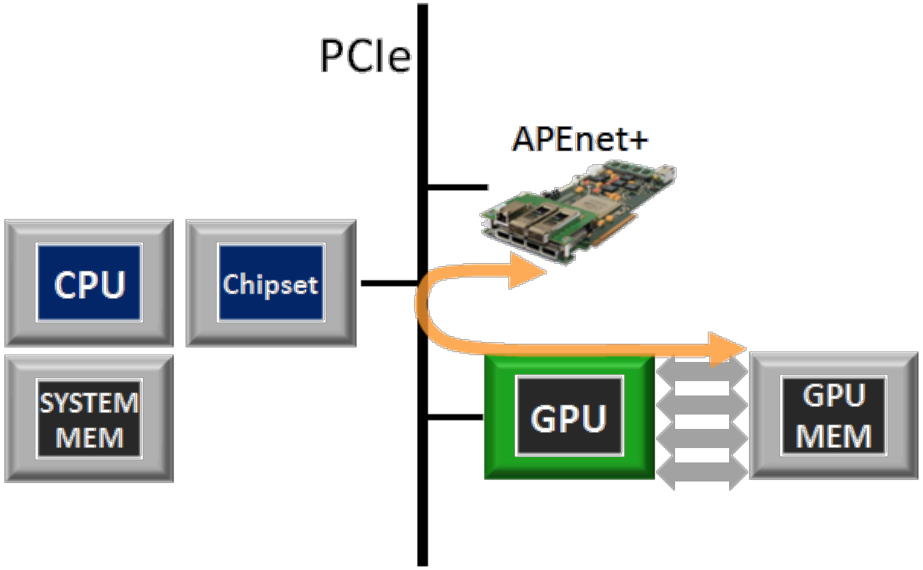
- Problem: lower system communication latency and its fluctuations.

- How?
  1. Injecting directly data from the NIC into the GPU memory with no intermediate buffering.
  2. Offloading the CPU from network stack protocol management, avoiding OS jitter effects.

- NaNet solution:
  - ❑ Re-use the **APEnet+** design, implementing **GPUDirect RDMA**.
  - ❑ Add a network stack protocol management offloading engine to the logic (**UDP Offloading Engine**).

NaNet GPUDirect P2P Support

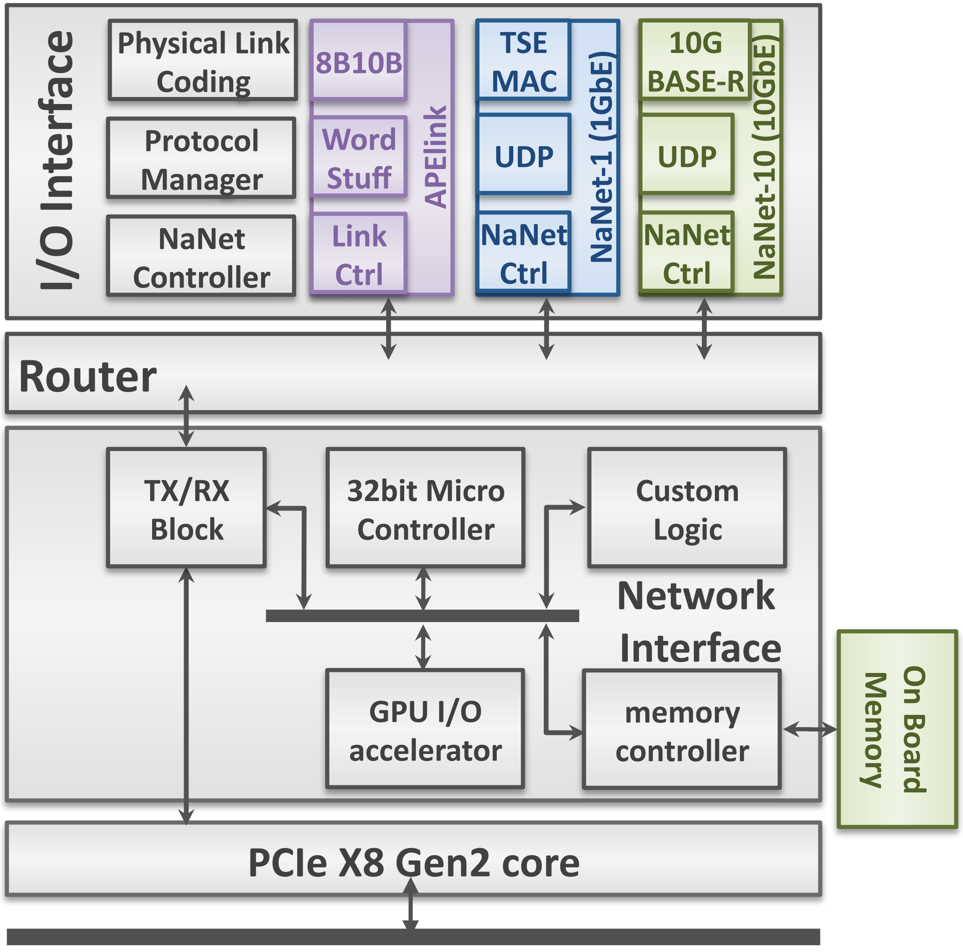
APEnet+ & NaNet Flow



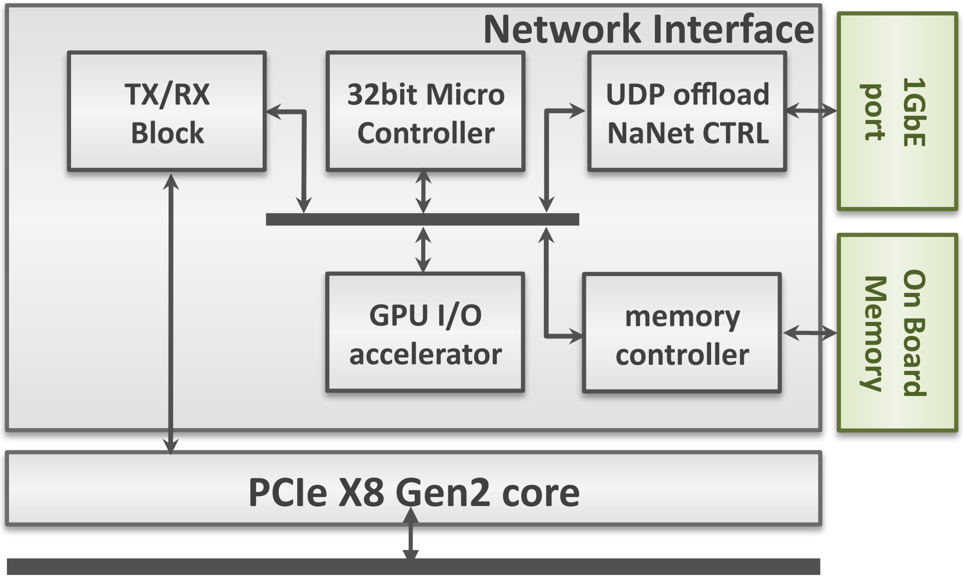
- PCIe P2P protocol between Nvidia Fermi/Kepler devices and APEnet+
  - ❑ First non-Nvidia device supporting it in **2012**.
  - ❑ Joint development with Nvidia.
  - ❑ APEnet+ board acts as a peer.
- No bounce buffers on host. APEnet+ can target GPU memory with no CPU involvement.
- GPUDirect allows direct data exchange on the PCIe bus.
- Real zero copy, inter-node GPU-to-host, host-to-GPU and GPU-to-GPU.
- Latency reduction for small messages.

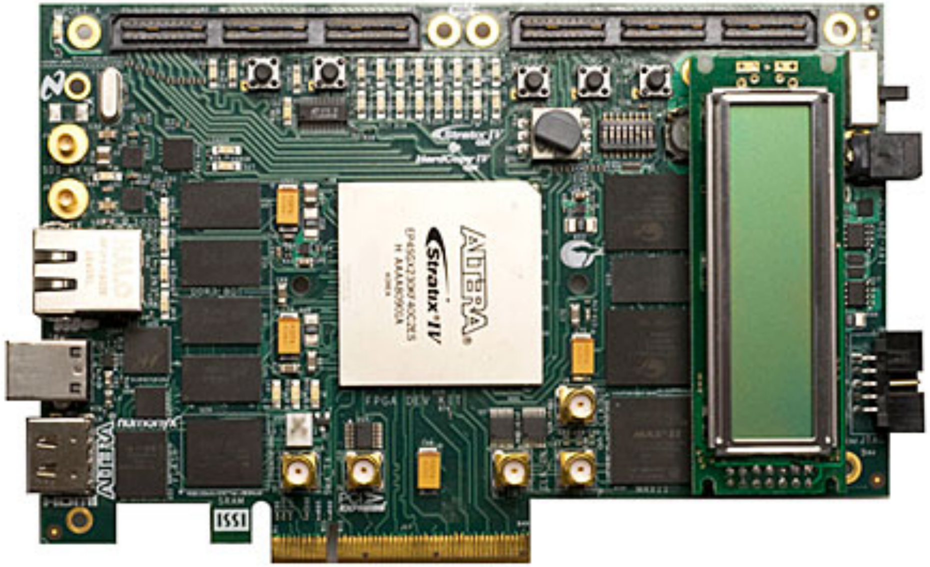
NaNet Architecture

- Multi-link support in I/O IF
  - ❑ Up to 6 APElink channels (34 gbps/ QSFP).
  - ❑ 1 GbE, 10 GbE (work in progress).
- Phyisical Link Coding.
- Data Protocol Manager (e.g. UDP).
- NaNet Controller: encapsulate incoming data payload in APEnet+ packets and forwards them to the RX logic (and viceversa in TX).
- DDR3 memory controller.
- NIOS II microcontroller.
- GPU I/O accelerator implementing GPUDirect P2P protocol.
- Custom Logic (e.g. addressing)
- PCIe Gen2 X8 host interface



NaNet-1 Implementation



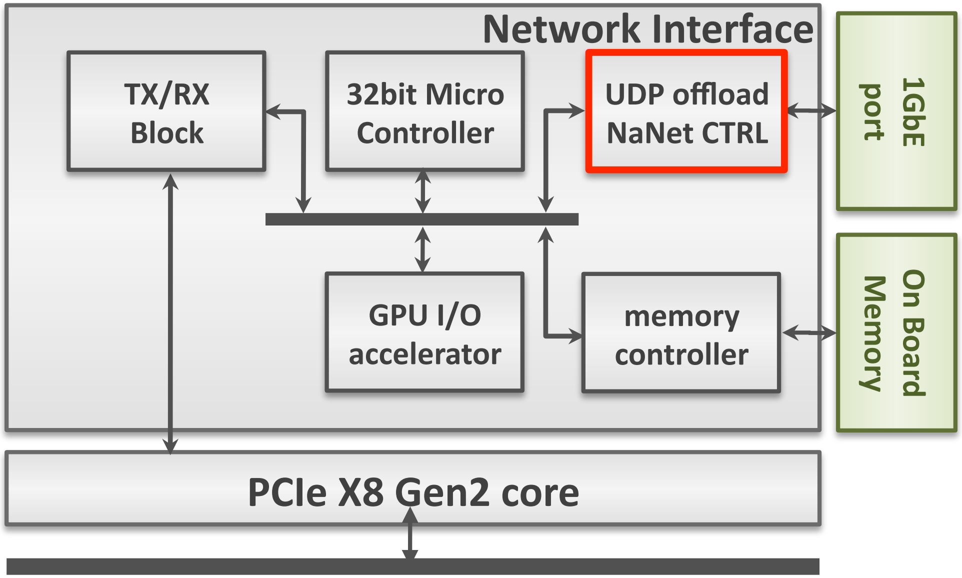


- Implemented on the Altera Stratix IV dev board (EP4SGX230KF40C2)
- PHY Marvell 88E1111
- HAL based Nios II microcontroller firmware (no OS)
  - System Configuration and Initializazion
  - Virtual Memory Management for the GPUDirect RDMA (TLB filling)

NaNet-1 UDP Offload

NiosII UDP Offload :

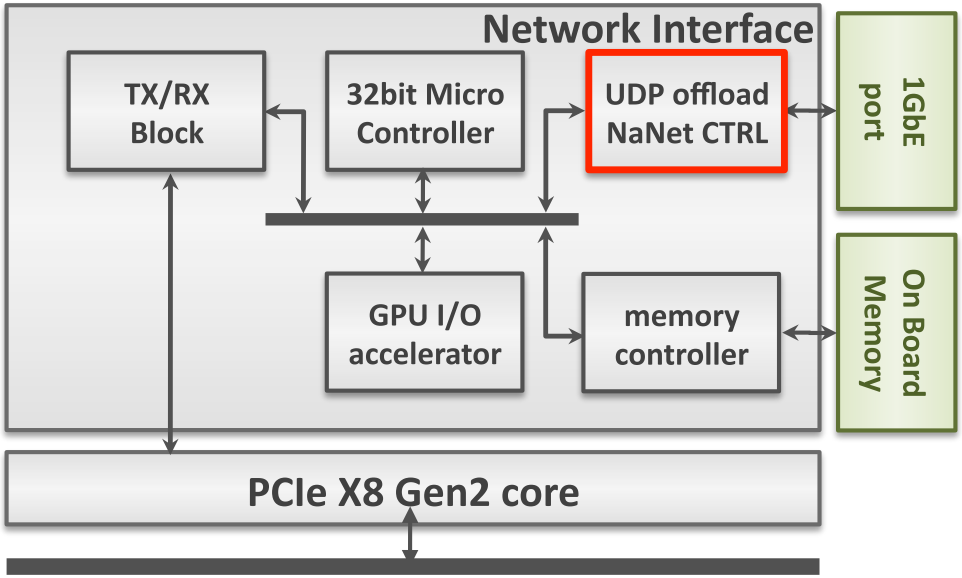
- Open IP: http://www.alterawiki.com/wiki/Nios_II_UDP_Offload_Example
- It implements a method for **offloading the UDP protocol management** from the Nios II microcontroller.
- It collects data coming from the Avalon Streaming Interface of the Altera Triple-Speed Ethernet Megacore (TSE MAC) and redirects UDP packets into an hardware processing data path.
- Current implementation provides a single 32-bit width channel.
- **6.4 gbps** (six times greater than the GbE Requirements)
- The output of the UDP Offload is the PRBS packet (Size + Payload)
- Ported to Altera Stratix IV EP4SGX230 (the project was based on Stratix II 2SGX90), clk@200MHz (instead of 35 MHz).



NaNet Controller

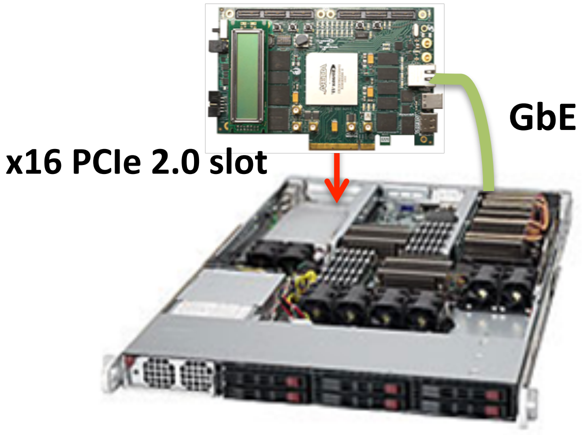
NaNet Controller:

- It manages the GbE input stream by encapsulating packets in the APEnet+ packet protocol (and viceversa for output stream)
- It implements an Avalon Streaming Interface
- It generates the Header for the incoming data, analyzing the PRBS packet and several configuration registers.
- It parallelizes 32-bit data words coming from the Nios II subsystem into 128-bit APEnet+ data words.
- It redirects data-packets towards the corresponding FIFO (one for the Header/ Footer and another for the Payload)



NaNet-1 Test & Benchmark Setup

- Supermicro SuperServer 6016GT-TF
  - X8DTG-DF motherboard (Intel Tylersburg chipset)
  - dual Intel Xeon E5620
  - Intel 82576 Gigabit Network Connection
  - Nvidia Fermi M2070
  - kernel 2.6.32-358.6.2.el6.x86_64, CUDA 4.2, Nvidia driver 325.15.
- NaNet-1 board in x16 PCIe 2.0 slot
- NaNet-1 GbE interface directly connected to one host GbE interface
- Common time reference between sender and receiver (they are on the same host).
- Ease data integrity tests.

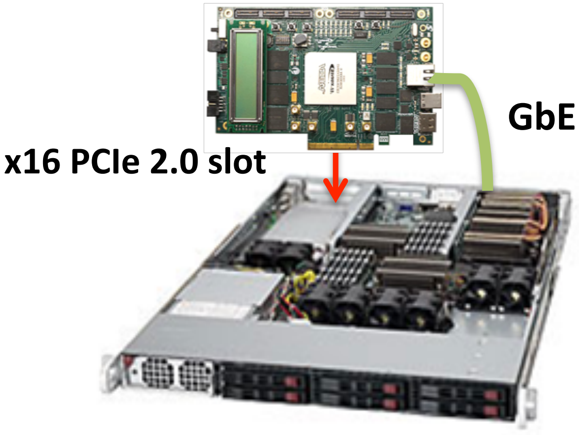


NaNet-1 Latency Benchmark

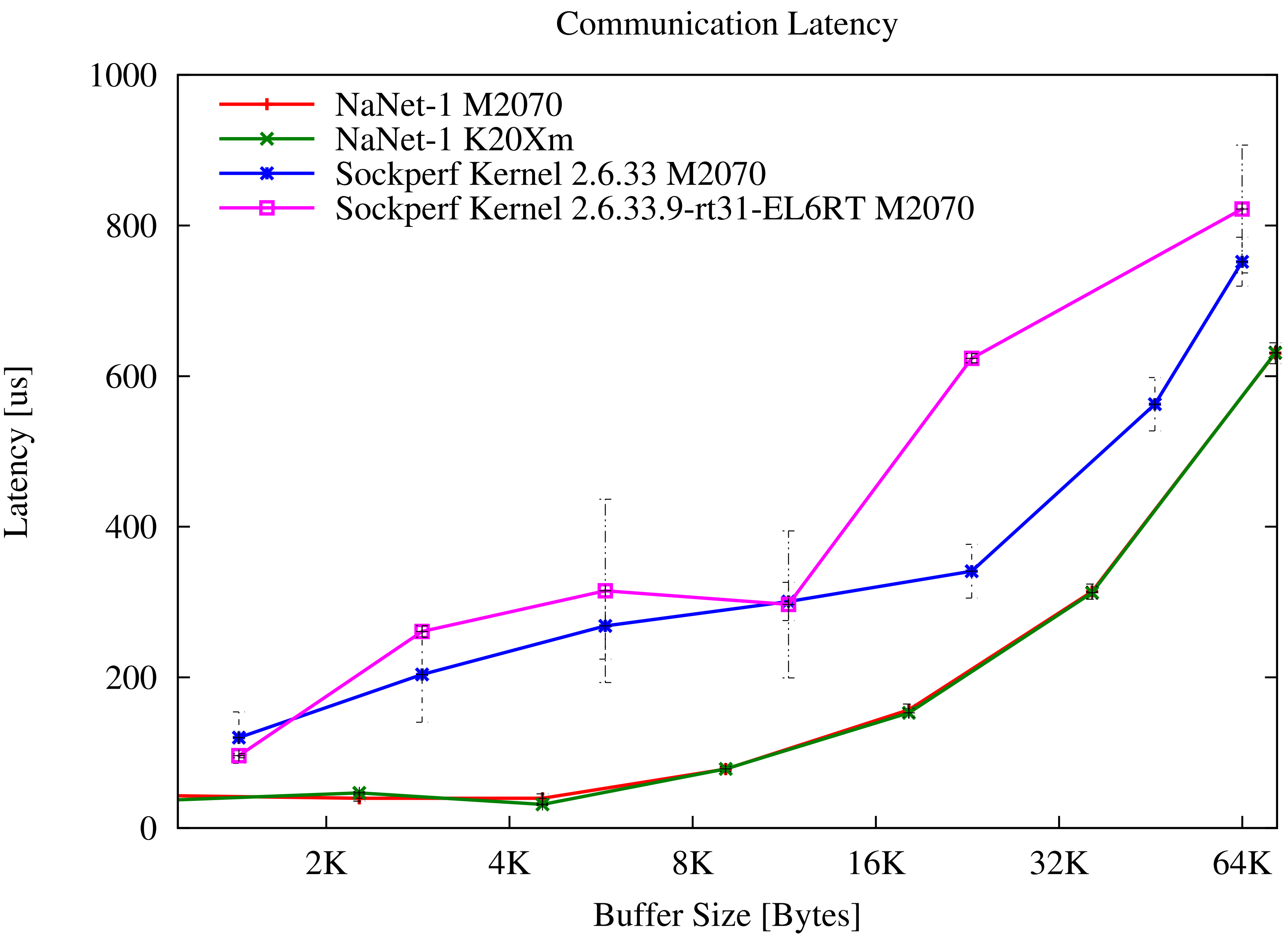
In a single host process:

- Allocate and register (pin) *N* GPU receive buffers of size *P* x1168 byte (*P* x 16 events) in a **circular list**.
- In the main loop:
  - Read TSC Register (*cycles_bs*)
  - Send *P* UDP packet with 1168 byte payload size over the host GbE intf
  - Wait for a received buffer event
  - Read TSC Register (*cycles_ar*)
  - Launch the GPU kernel on next buffer in circular list.
  - Synch Streams
  - Read *(cycles_ak)*
  - Record *latency_cycles_comm* = *cycles_ar* - *cycles_bs*
  - Record *latency_cycles_calc* = *cycles_ak* – *cycles_ar*

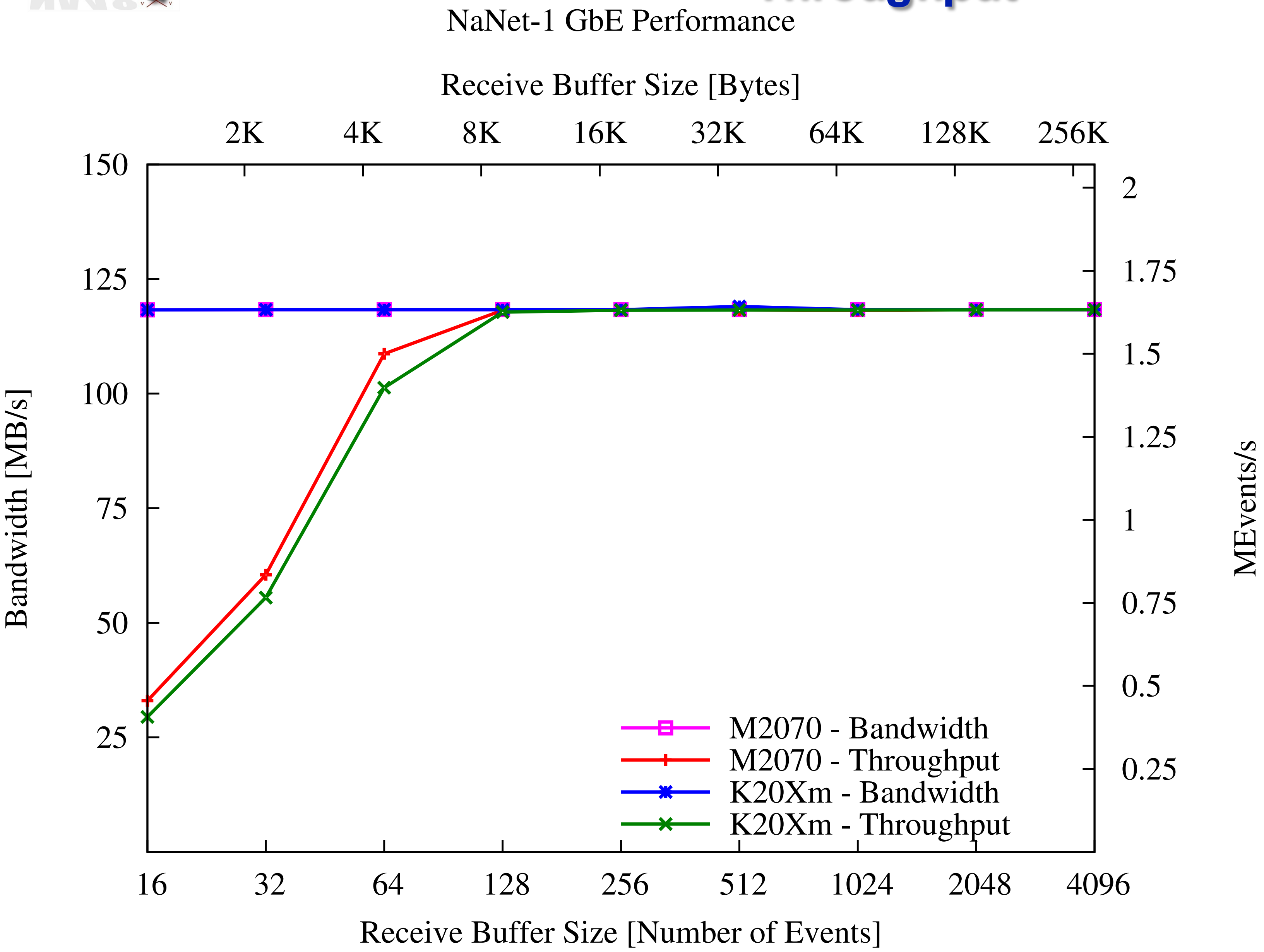
- Results in good agreement with oscilloscope measures (TEL62).



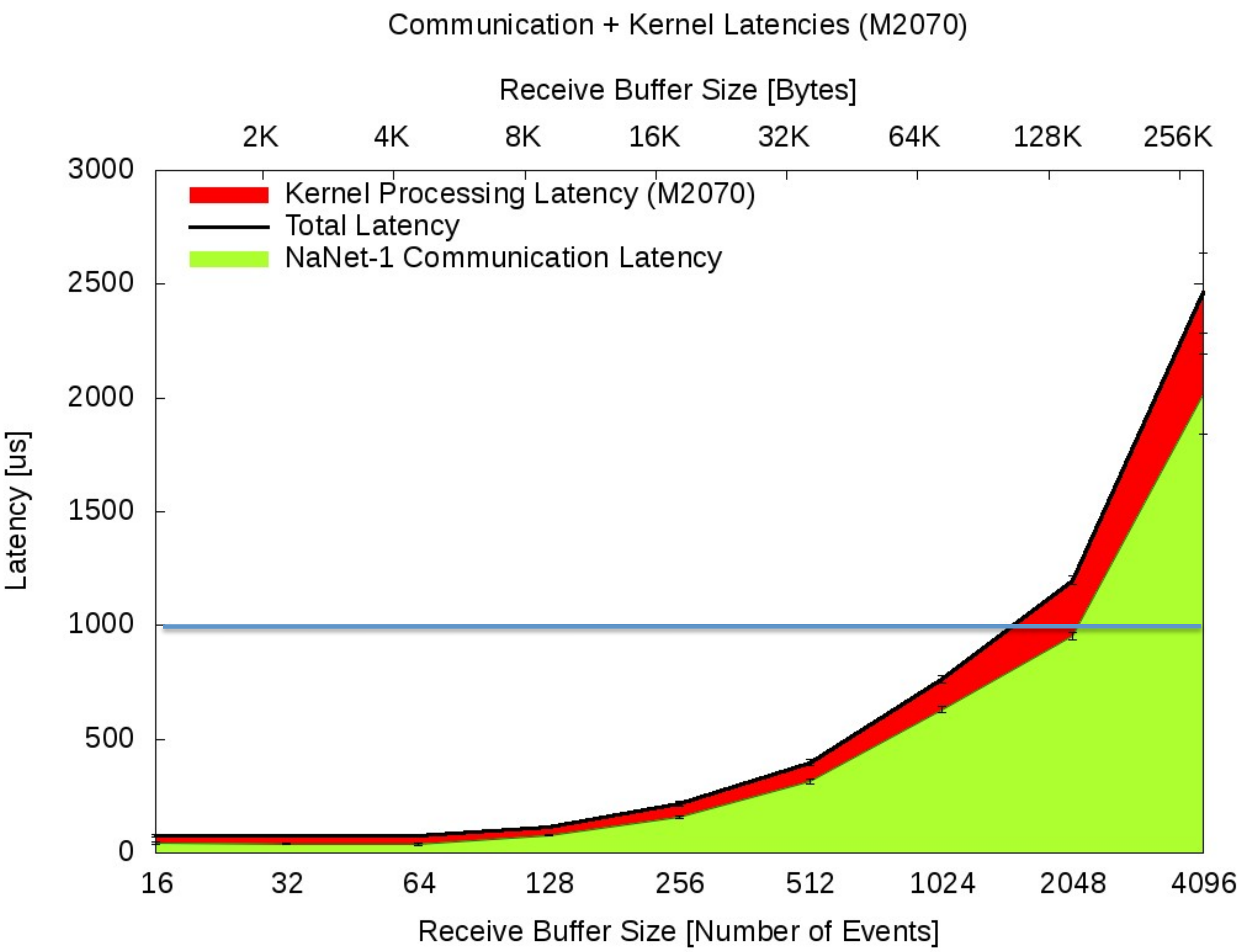
NaNet-1 Communication Latency



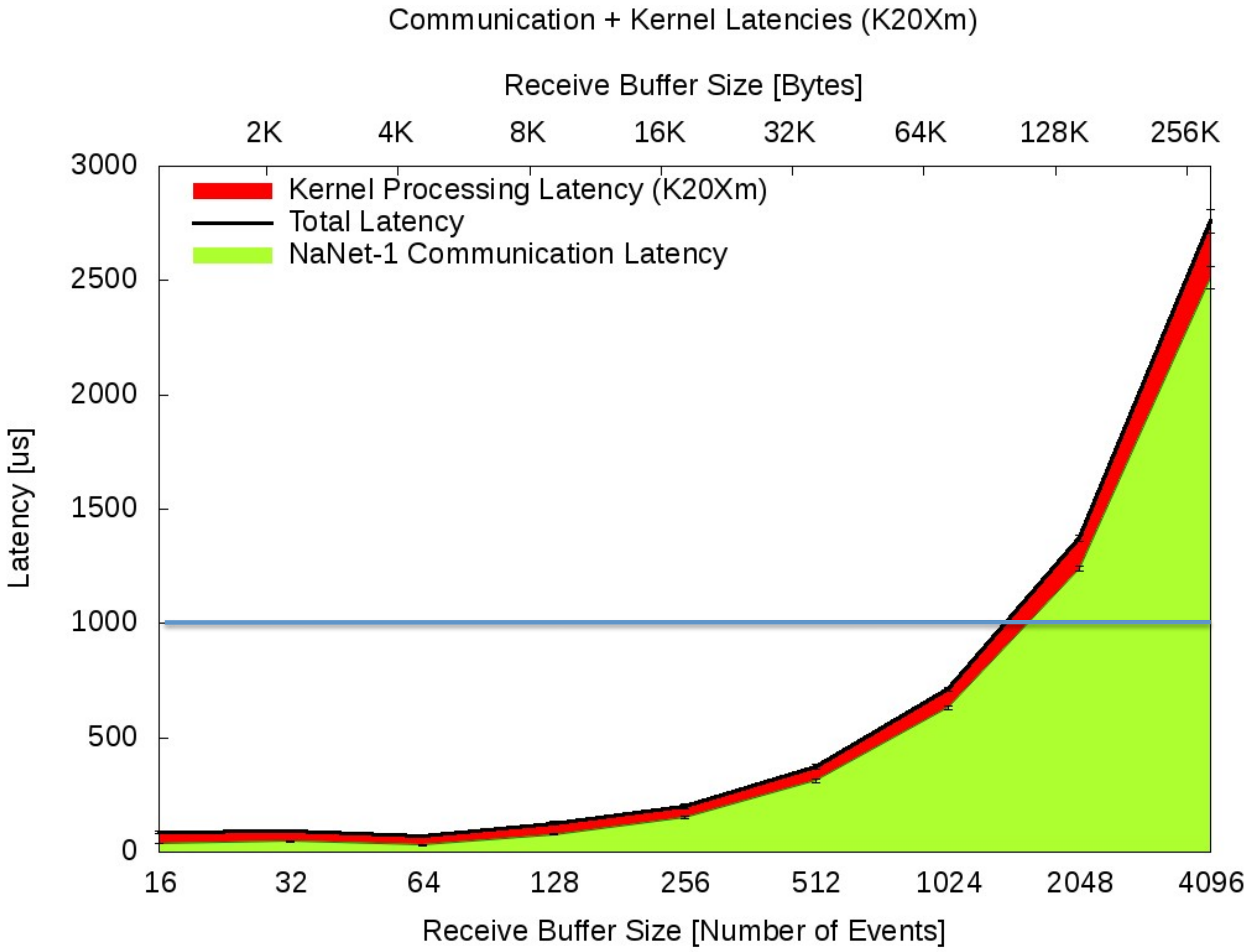
NaNet-1 1 GbE Bandwidth & Throughput



Total latency of the GPU-Based RICH L0-TP using NaNet-1 (Fermi M2070)

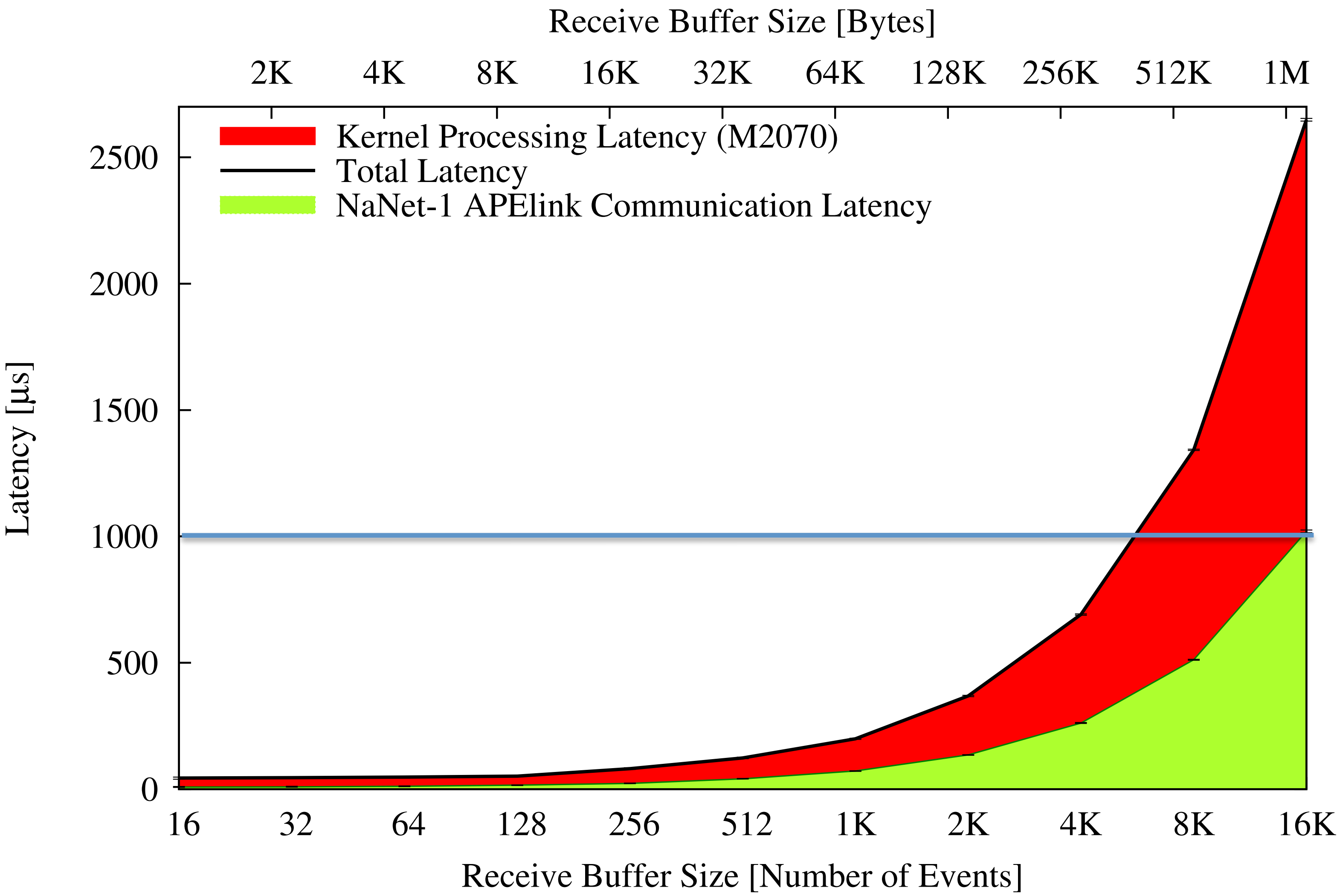


Total latency of the GPU-Based RICH L0-TP using NaNet-1 (Kepler K20X)

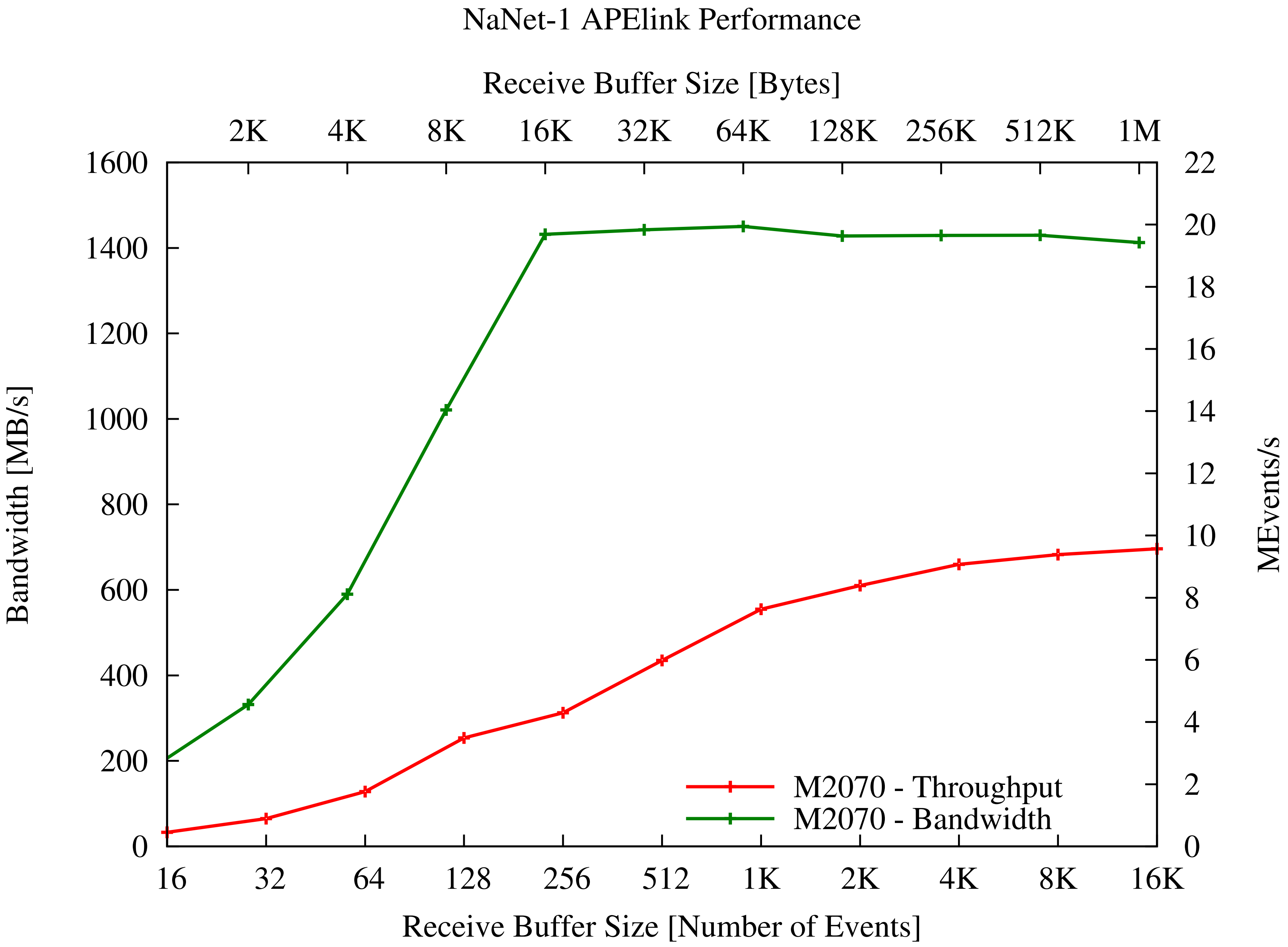


Total latency of the GPU-Based RICH L0-TP using NaNet-1 (APElink)

NaNet-1 APElink Communication + Kernel Latencies (M2070)



NaNet-1 APElink Bandwidth & Throughput



Total latency of the GPU-Based RICH L0-TP using NaNet-1 - Comments

- Worst case: to perform measure communication and processing tasks were serialized, while in normal operation…
- data communication and GPU processing are overlapped!
- Circular list of receive buffers: when one of the buffers is full, the GPU kernel is lauched, data arriving from the network are accumulated in the next buffer in the circular list concurrently with processing.

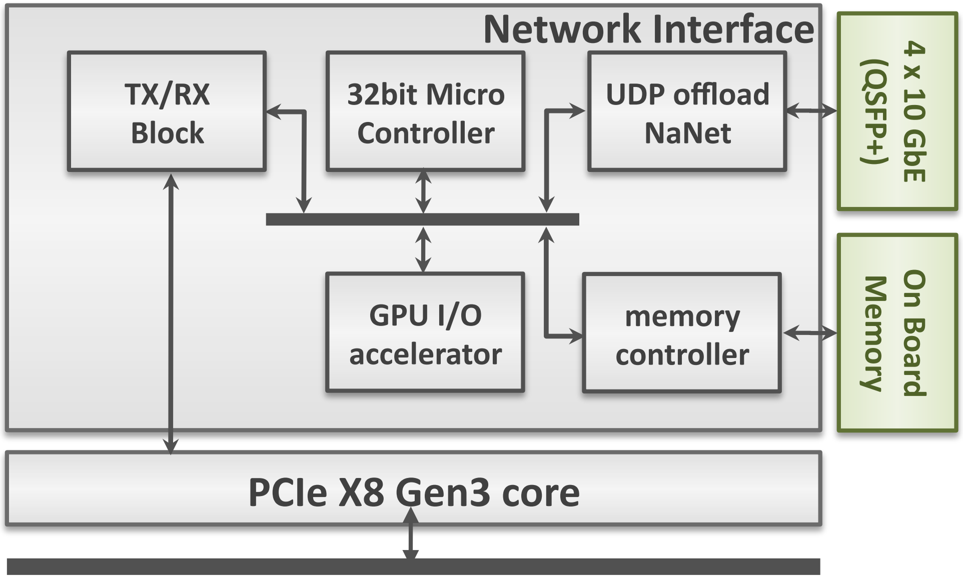
- A RICH detector Level 0 Trigger Processor GPU-based system using NaNet-1 (1 GbE link) performing the simple MATH processing is able to sustain a throughput of **~1.6 Mevents/s** with real-time behaviour.

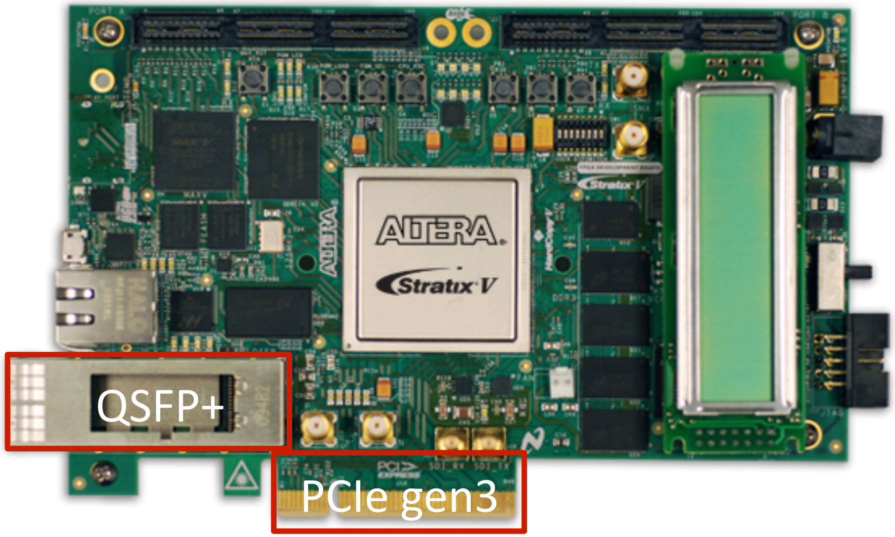
- Using a single APElink channel instead of the GbE one, the system shows a throughput of **~10 Mevents/s: good scaling with link bandwidth.**

Activities

- Implement a dedicated custom logic block to completely offload the microcontroller from address generation (done).
- Speedup virtual to physical address translation using a CAM based TLB (done).
- Test with TEL62 (started, preliminary results).
- Implement 10 GbE link support (in progress).
- Implement PCIe Gen3 support (in progress).
- Real-time caracterization of the 1/10 GbE switch (not started yet).

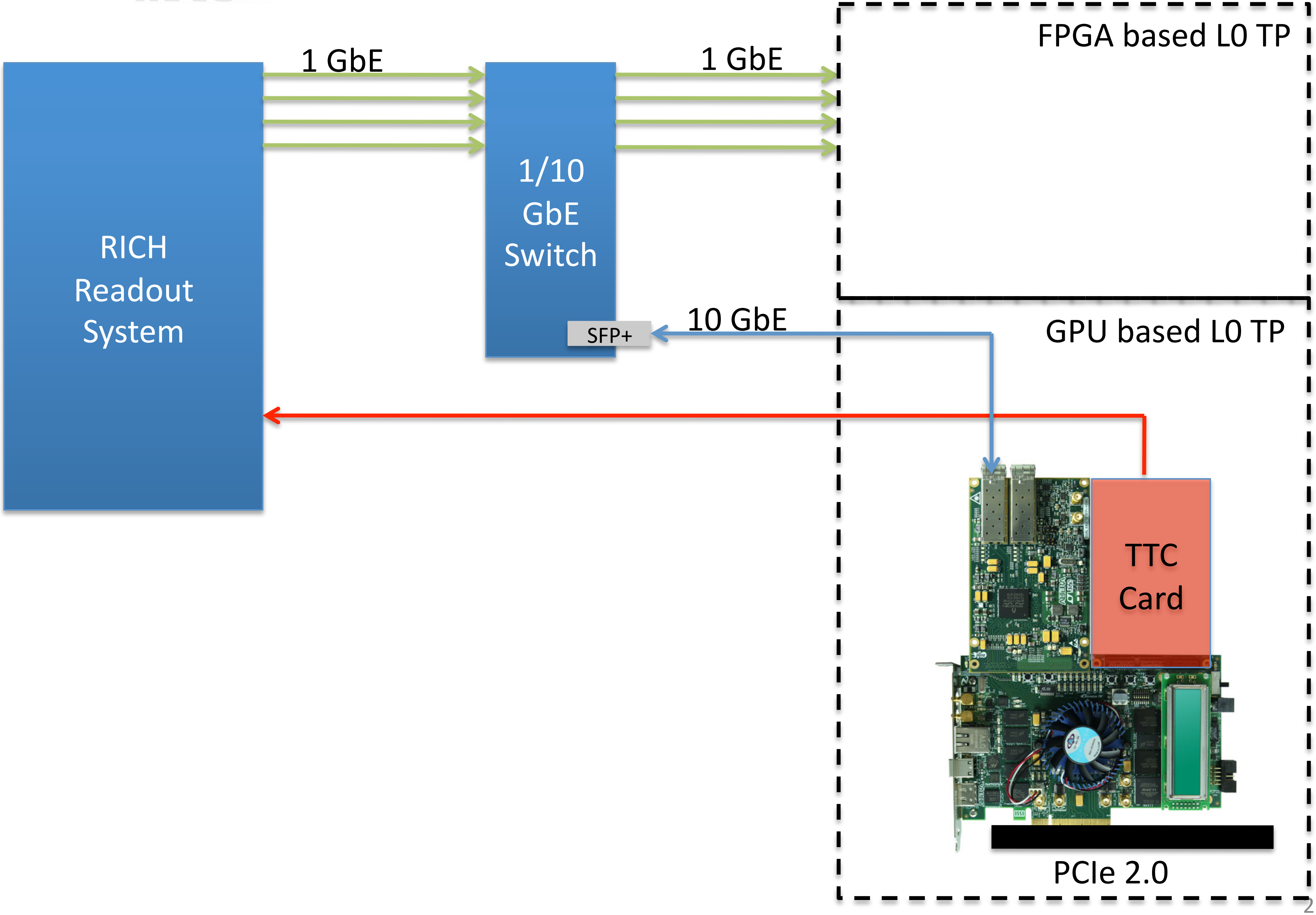
NaNet-10 on Stratix V





- Implemented on the Altera Stratix V dev board
  - PCIe gen3 (8 GB/s)
  - Embedded Altera Transceiver (up to 14.1 Gbps)
  - hardened 10GBASE-R PCS features to support 10 Gbps Ethernet (10GbE)

NaNet-10 for NA62 RICH L0 GPU Trigger Processor



THANK YOU

BACK-UP SLIDES

APEnet+: a 3D NIC for HPC with GPUdirect RDMA capability

3D Torus Network:

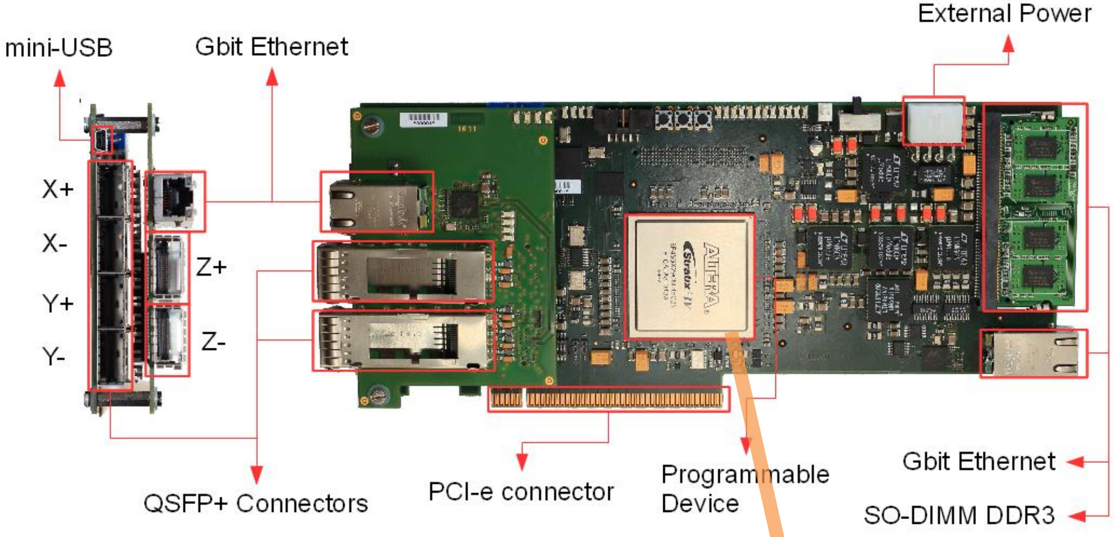
- Scalable (today up to 32K nodes)
- Direct Network: no external switches.

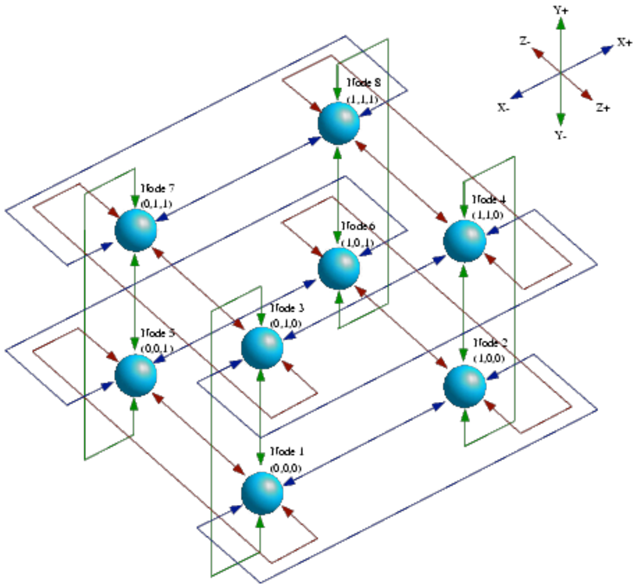
APEnet+ Card:

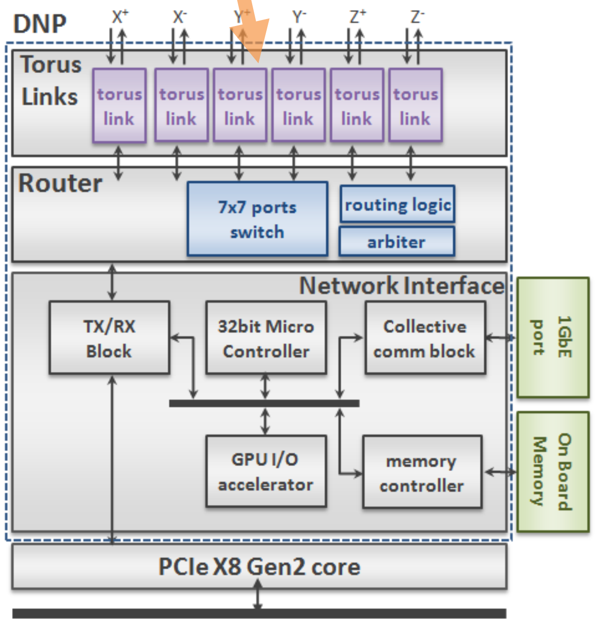
- FPGA based (ALTERA EP4SGX290)
- PCI Express x16 slot, signaling capabilities for up to dual x8 Gen2)
- Fully bidirectional torus links, 34 Gbps

APEnet+ Logic:

- Torus Link
- Router
- Network Interface
  - NIOS II 32 bit microcontroller
  - RDMA engine
  - GPU I/O accelerator.
- PCI x8 Gen2 Core



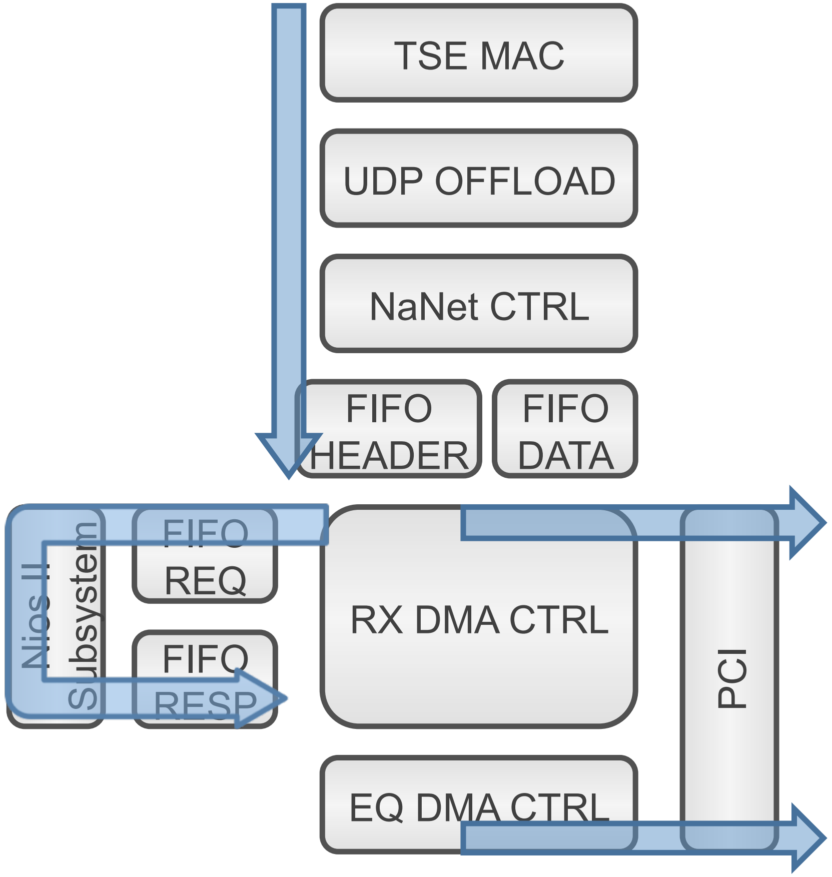




Poster "Architectural improvements and 28nm FPGA implementation of the APEnet+ 3D Torus network for hybrid HPC systems" presented by Roberto Ammendola.

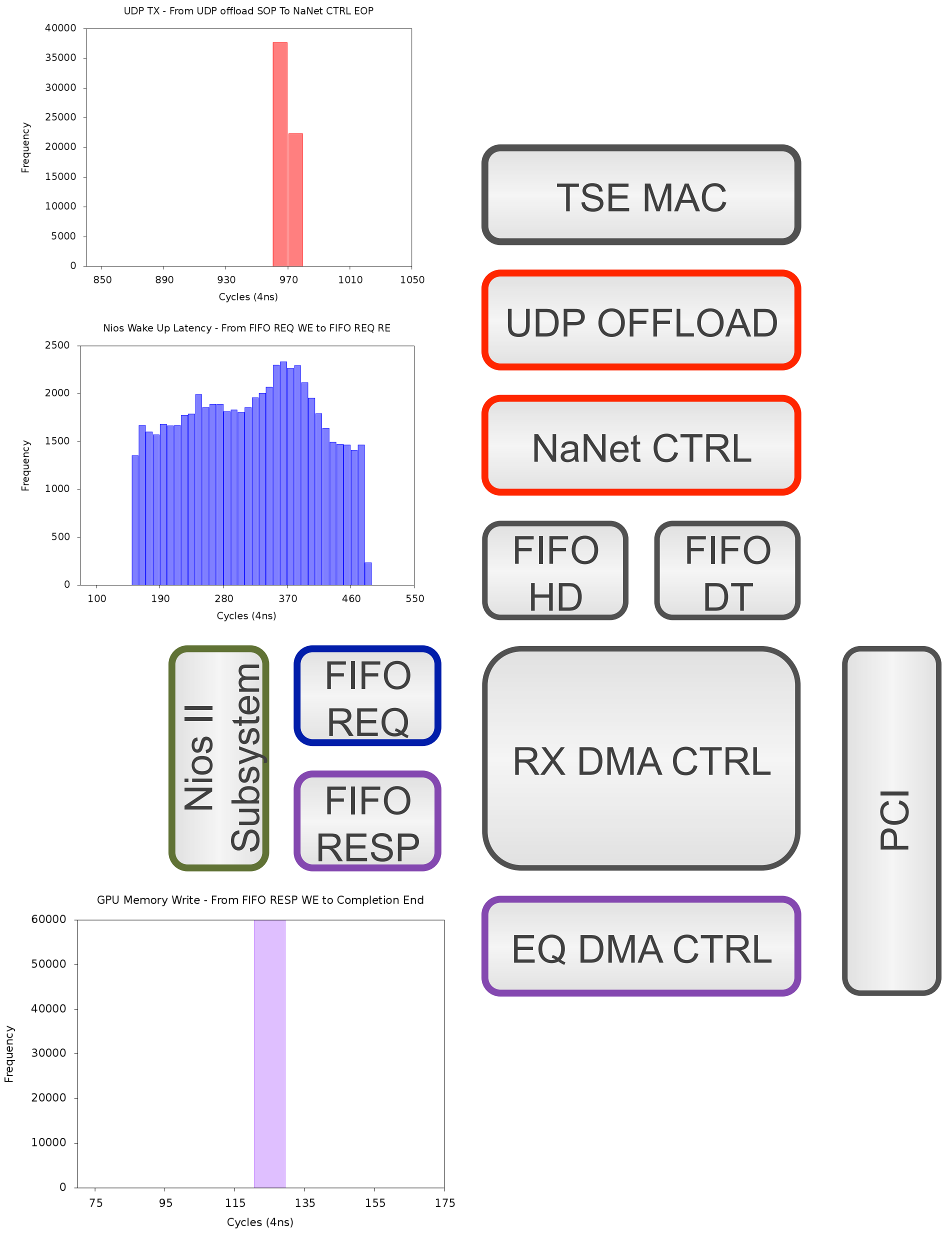
NaNet Data Flow

- TSE MAC, UDP OFFLOAD and NaNet CTRL manage the GbE data flow and encapsulate data in the APEnet+ protocol.
- RX DMA CTRL:
  - ❑ It manages CPU/GPU memory write process, providing hardware support for the Remote Direct Memory Access (RDMA) protocol
  - ❑ It communicates with the μC :
    - It generates the request for the Nios necessary to perform the destination buffer search and the virtual2physics address translation (**FIFO REQ**)
    - It exploits the instruction collected in the **FIFO RESP** to instantiate the memory write process.
- Nios II handles all the details pertaining to buffers registered by the application to implement a zero-copy approach of the RDMA protocol (**OUT of the data stream**).
- EQ DMA CTRL generates a DMA write transfer to communicate the completion of the CPU/GPU memory write process.
- **A Performance Counter is used to analyze the latency of the GbE data flow**



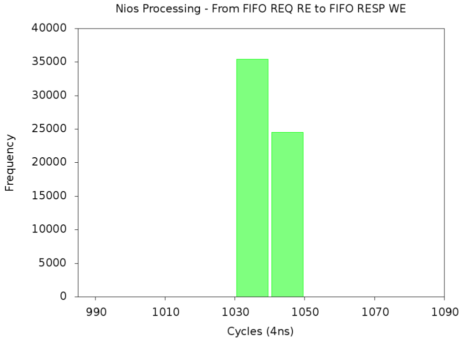
UDP Packet Latency Inside NaNet-1 Detailed Analysis

- Fine grained analysis of the UDP packet hardware path to understand the latency variability
- UDP TX (3.9 μs, stable)
  - From the UDP offload Start of Packet
  - To the NaNet CTRL End Of Packet
- Nios Wake Up (0.6 μs ÷ 2.0 μs, **culprit**)
  - From the FIFO REQ Write Enable
  - From the FIFO REQ Read Enable
- Nios Processing (4.0 μs, stable)
  - From the FIFO REQ Read Enable
  - From the FIFO RESP Write Enable
- GPU Memory Write (0.5 μs, stable)
  - From the FIFO REQ Read Enable
  - From the FIFO RESP Write Enable


UDP TX - From UDP offload SOP To NaNet CTRL EOP

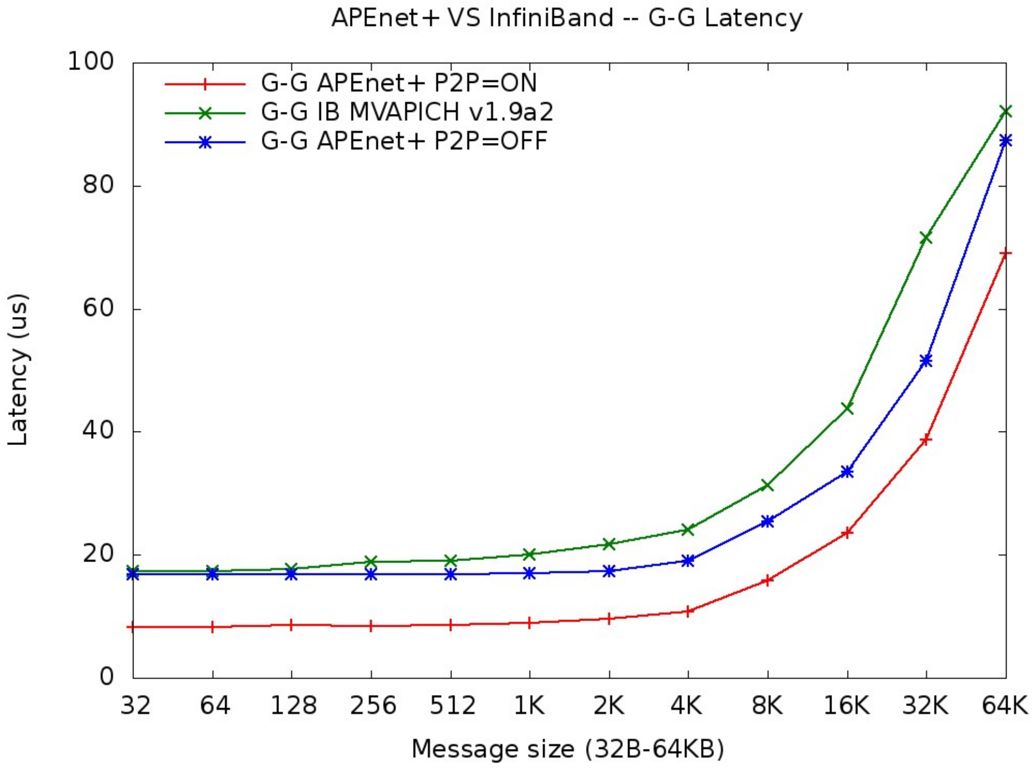
Nios Wake Up Latency - From FIFO REQ WE to FIFO REQ RE

GPU Memory Write - From FIFO RESP WE to Completion End


Nios Processing - From FIFO REQ RE to FIFO RESP WE

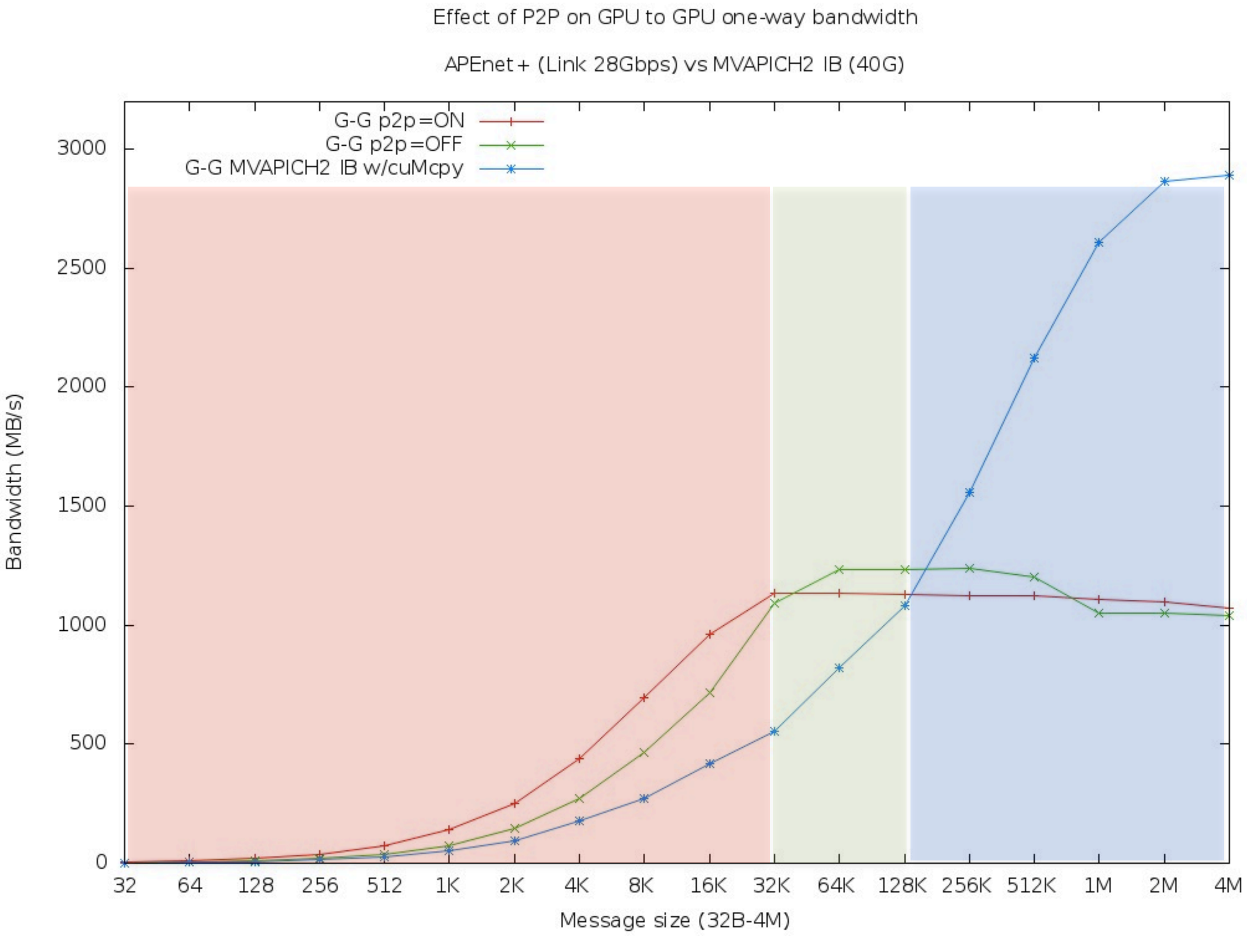
APEnet+ vs InfiniBand – g2g Latency

- GPU P2P behaviour on small buffer size
- APEnet+ leverages on Nvidia P2P implementation
  - APEnet+ P2P latency ~8.2 μs
  - APEnet+ staging latency ~16.8 μs
  - MVAPICH/IB latency ~17.4 μs
- P2P=OFF
  - `cudaMemcpyD2H/H2D()` on host bounce buffers
  - Buffers pinned with `cuMemHostRegister`
  - `cuMemcpy()` ~ 10 μs
- MVAPICH2 tested on same test system

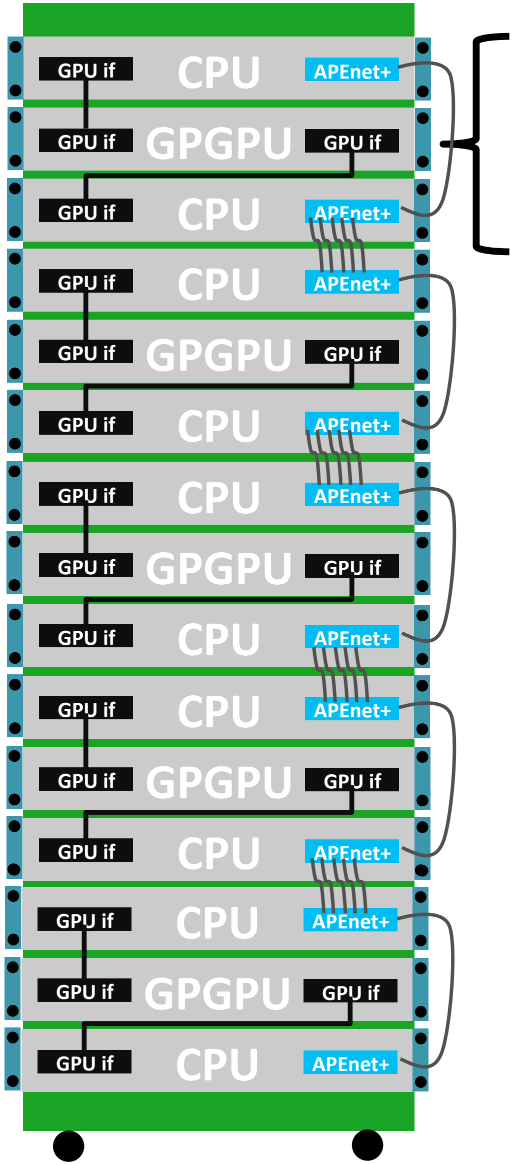


APEnet+ VS rest of the World

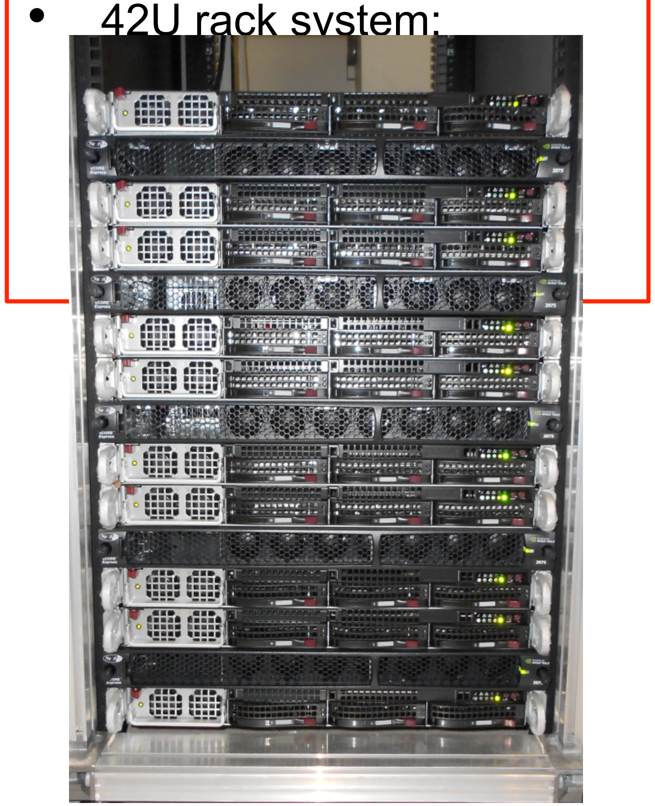
- Below 32 KB P2P wins
- 32 kB – 128 KB P2P shows limits
- over 128 KB Pure bandwidth wins



QUonG status and next future



- QUonG elementary mechanical assembly:
  - multi-core INTEL (packed in 2 1U rackable system)
  - S2090 FERMI GPU system (5 TFlops)
  - 2 APEnet+ board
- 42U rack system:



The EURETILE HW Platform demonstrator at 2012 project review will be a stripped version of QUonG elementary mechanical assembly with

- 2 CPU systems with/without GPUs connected with ApeNet+ boards
- To demonstrate:
  - running prototype of EURETILE HW platform
  - preliminary implementation of “faults awareness” hardware block (sensors registers and link error counter read,…)