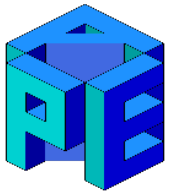


NaNet: a low-latency NIC enabling GPU-based, real-time low level trigger systems.

Alessandro Lonardo (INFN)

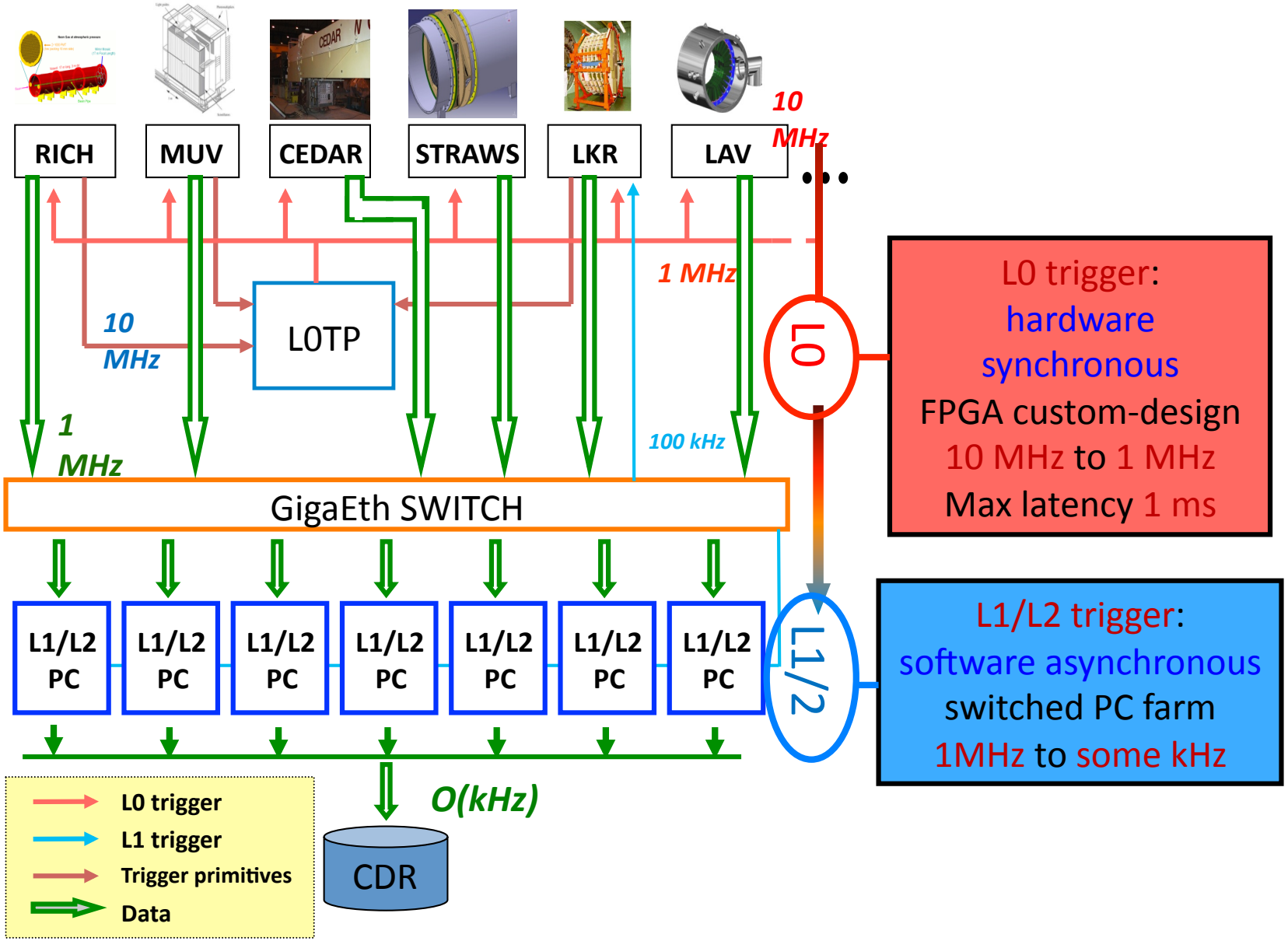
On behalf of the NaNet collaboration

20th International Conference on Computing in High Energy and Nuclear Physics (CHEP2013)

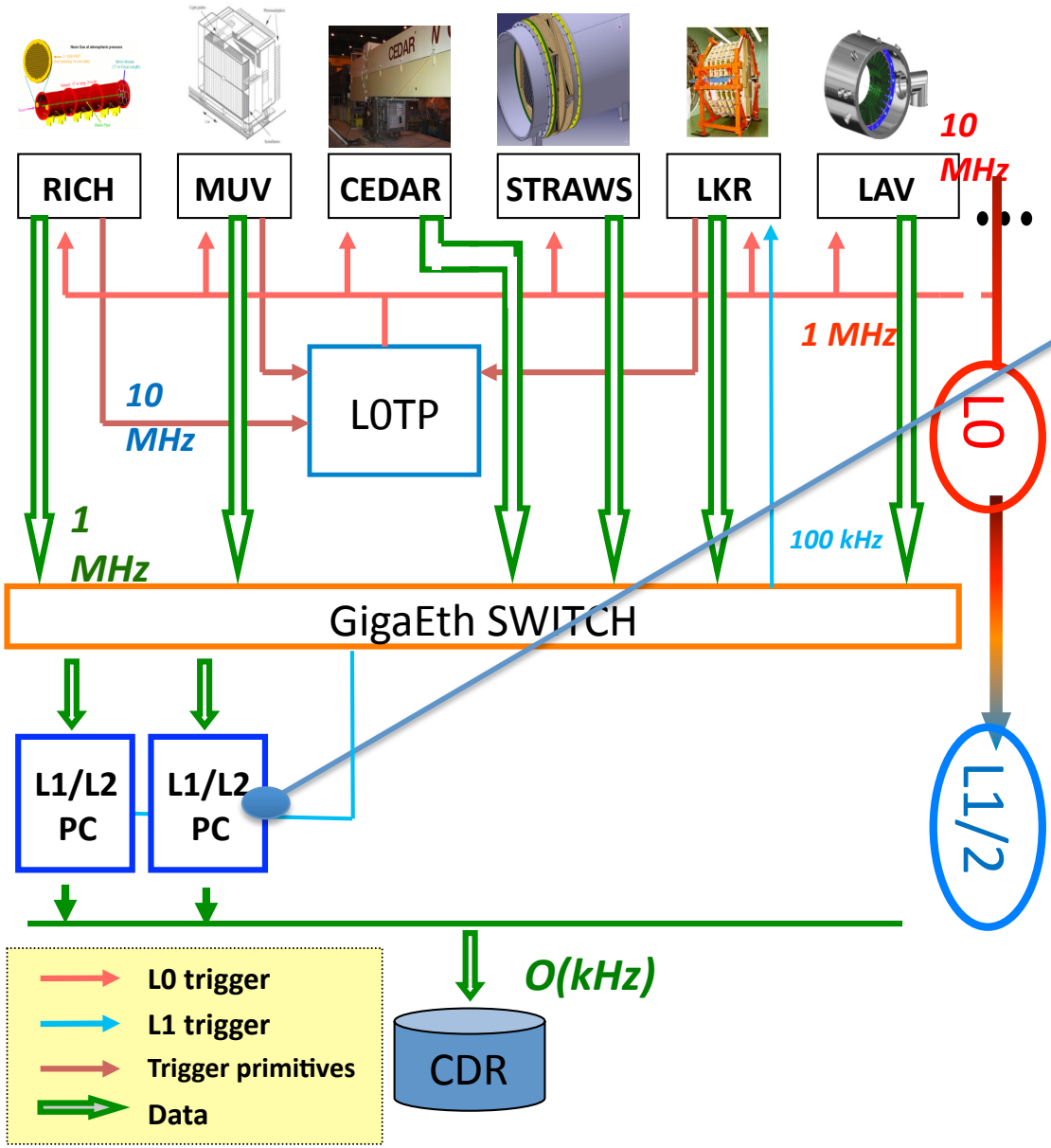
The NA62 Experiment at CERN

- The goal is the precision measurement of the ultra-rare decay process $K^+ \to \pi^+ \nu \bar{\nu}$
  - Extremely sensitive to any unknown new particle.
- Many (uninteresting) events: 10^7 decays/s.
- Ultra-rare: 1 target event on 10^{10} particle decays
  - Expect to collect ~100 events in 2/3 years of operation
  - Need to select the interesting events, filtering efficiently and quickly the data stream coming from the detectors.

The NA62 Multi-Level Trigger and Data Acquisition System



Using GPUs in the NA62 L1/L2 Trigger

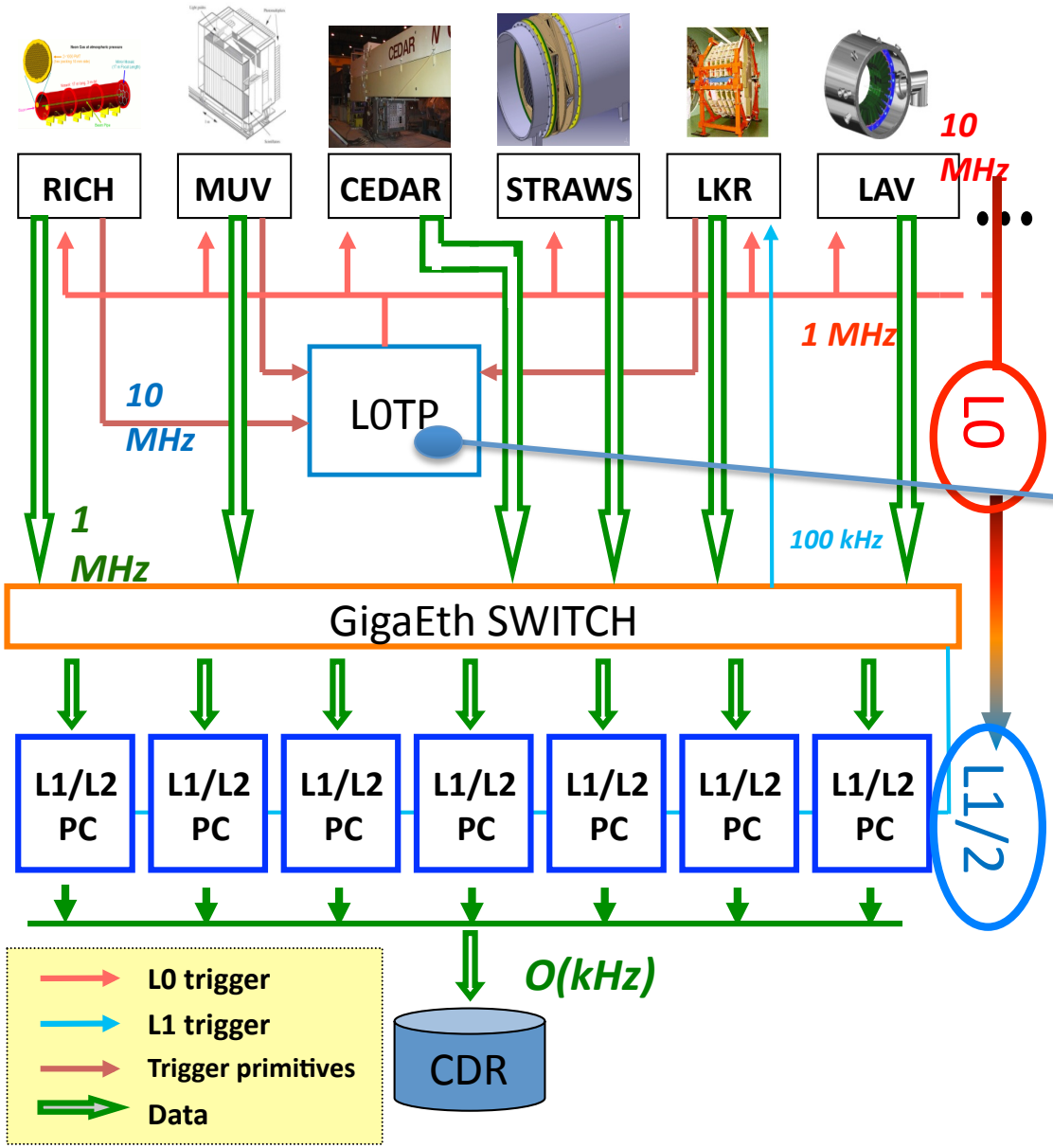


Exploit GPU computing power to **scale down the number of nodes** in the farm:

- event analysis paralelization on many-core architectures.

Not the topic of this talk.

Using GPUs in the NA62 L0 Trigger

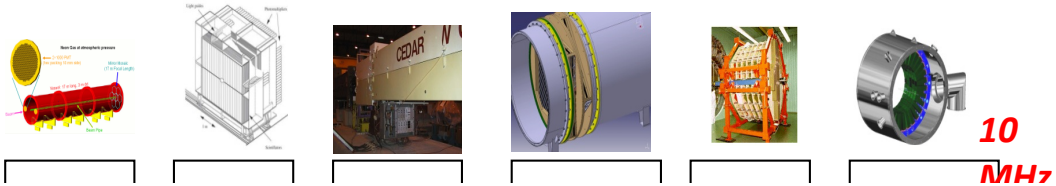


Replace custom hardware with a GPU-based system performing the same task but:

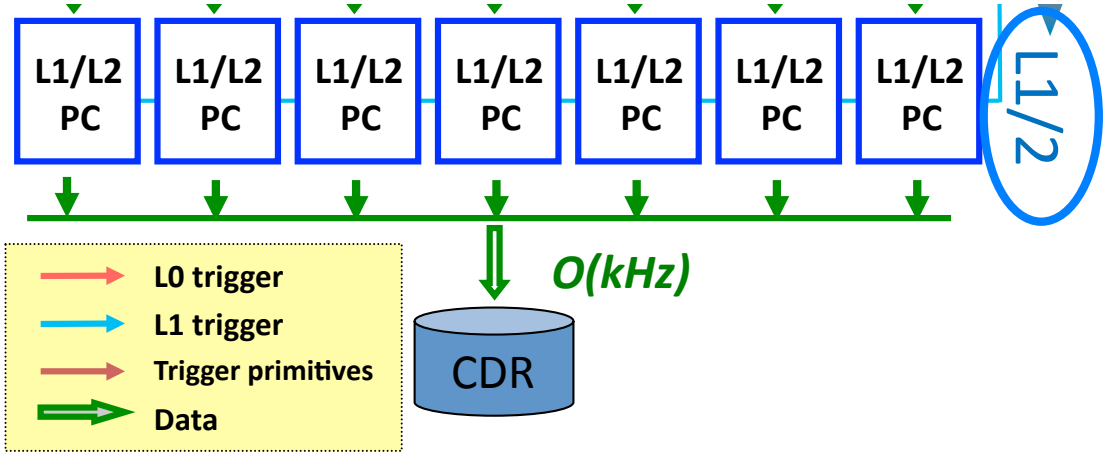
- Programmable
- Upgradable
- Scalable
- Cost effective
- Exploit GPUs computing power to implement more selective trigger algorithms.

The topic of this talk.

Using GPUs in the NA62 L0 Trigger



	TESLA K10[a]	TESLA K20	TESLA K20X
Peak double precision floating point performance (board)	0.19 teraflops	1.17 teraflops	1.31 teraflops
Peak single precision floating point performance (board)	4.58 teraflops	3.52 teraflops	3.95 teraflops
Number of GPUs	2 x GK104s	1 x GK110	
Number of CUDA cores	2 x 1536	2496	2688
Memory size per board (GDDR5)	8 GB	5 GB	6 GB
Memory bandwidth for board (ECC off)[b]	320 GBytes/sec	208 GBytes/sec	250 GBytes/sec
GPU computing applications	Seismic, image, signal processing, video analytics	CFD, CAE, financial computing, computational chemistry and physics, data analytics, satellite imaging, weather modeling	
Architecture features	SMX	SMX, Dynamic Parallelism, Hyper-Q	
System	Servers only	Servers and Workstations	Servers only

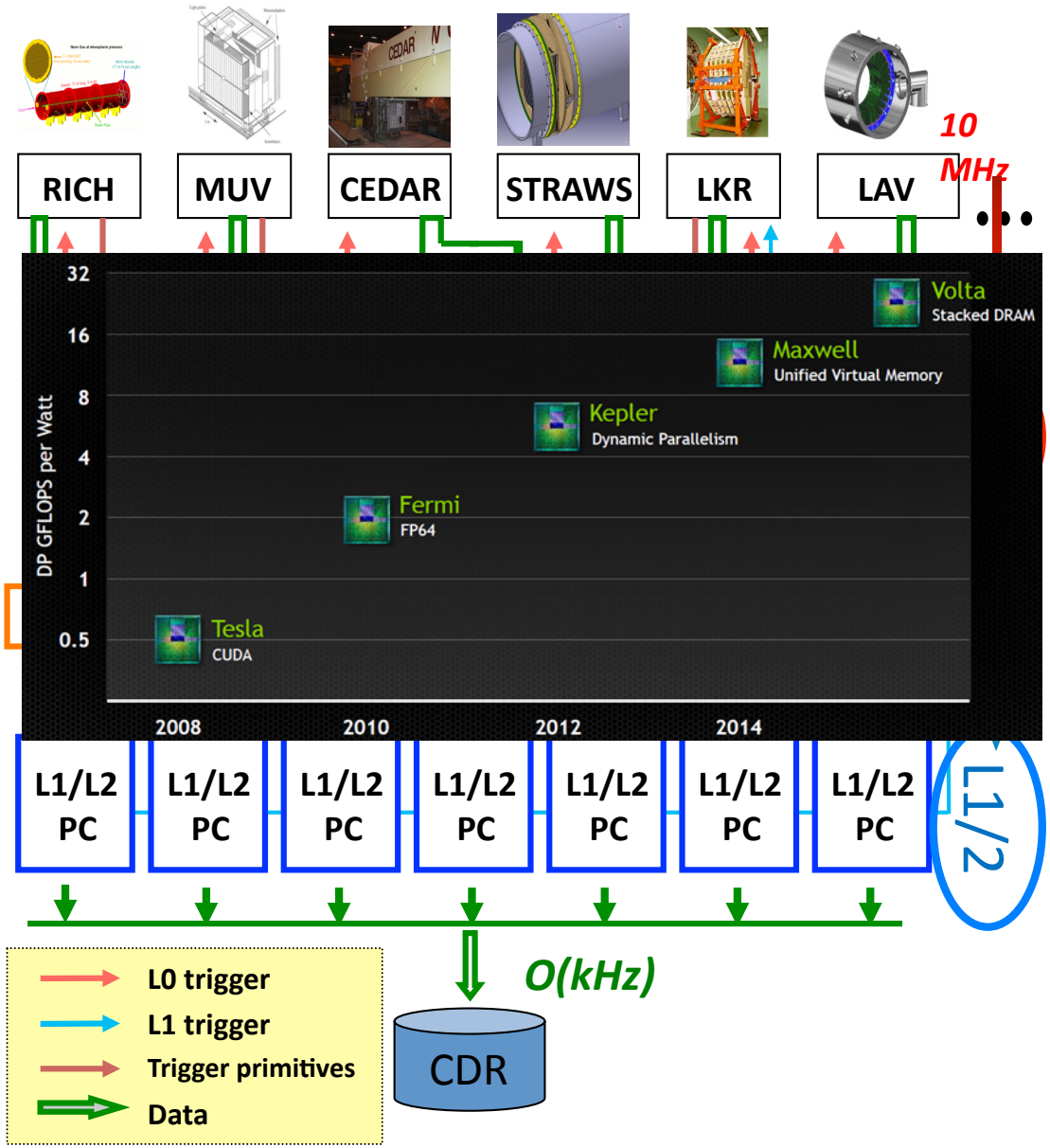


Replace custom hardware with a GPU-based system performing the same task but:

- Programmable
- Upgradable
- Scalable
- Cost effective
- Exploit GPUs computing power to implement more selective trigger algorithms.

The topic of this talk.

Using GPUs in the NA62 L0 Trigger

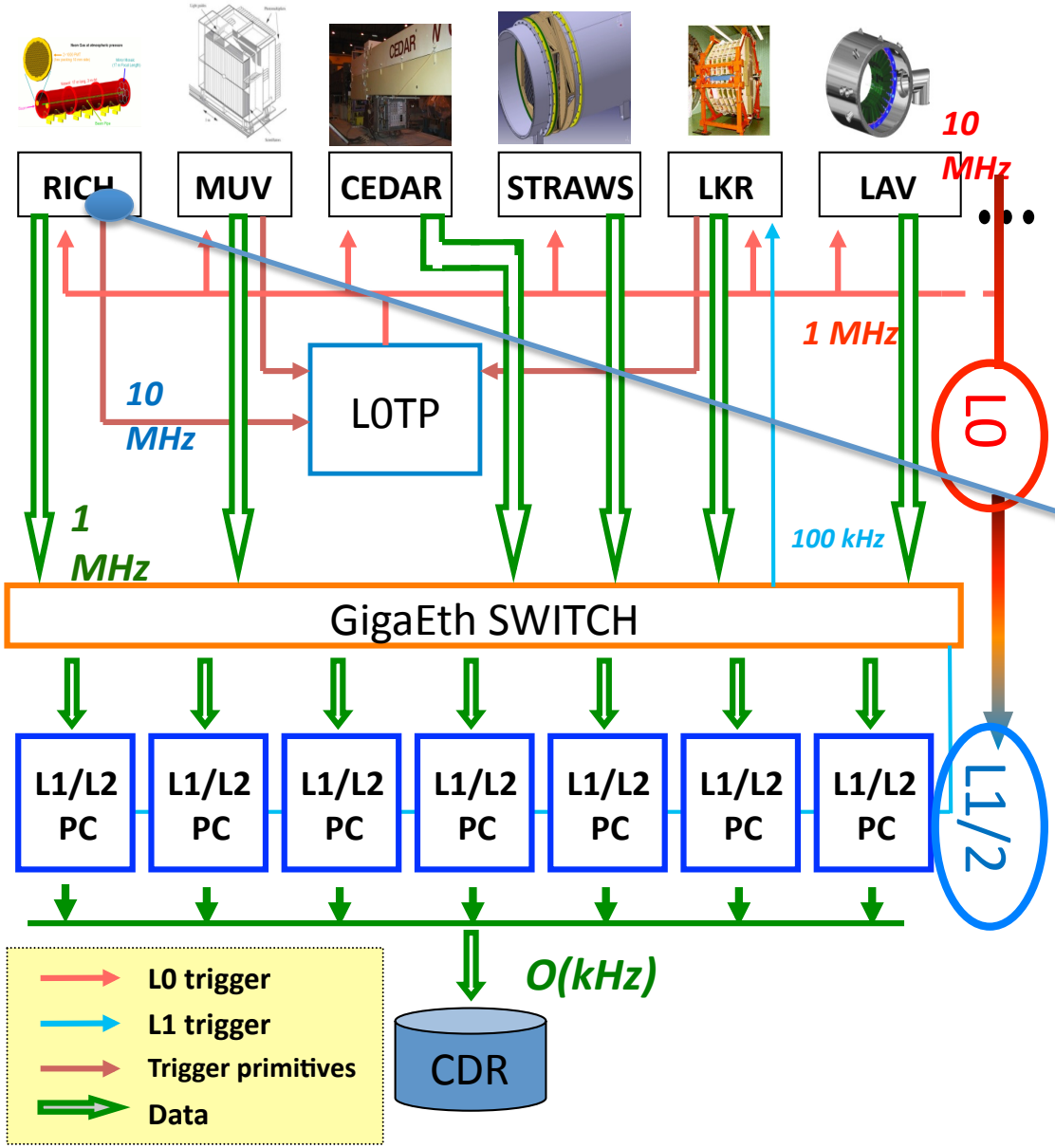


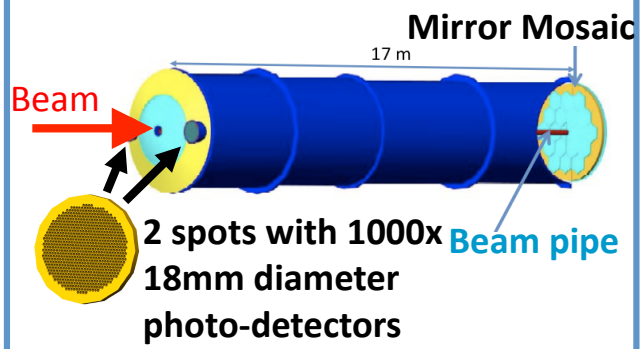
Replace custom hardware with a GPU-based system performing the same task but:

- Programmable
- Upgradable
- Scalable
- Cost effective
- Exploit GPUs computing power to implement more selective trigger algorithms.

The topic of this talk.

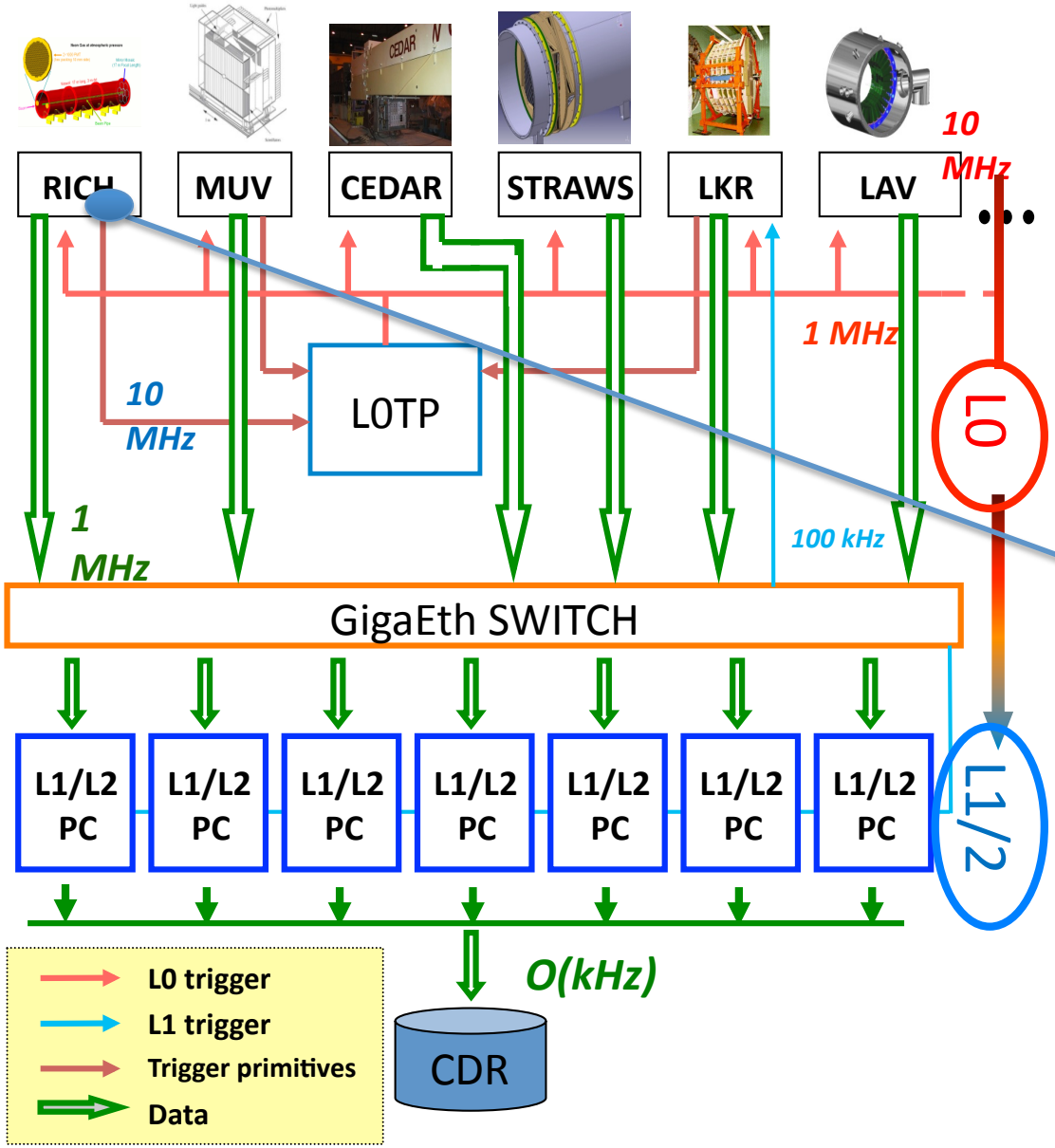
Using GPUs in the NA62 L0 Trigger: the RICH Detector Case Study

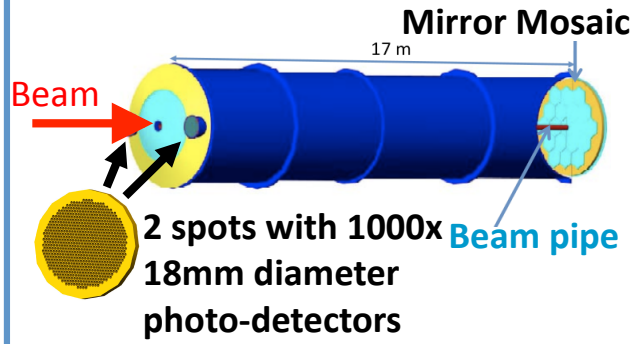




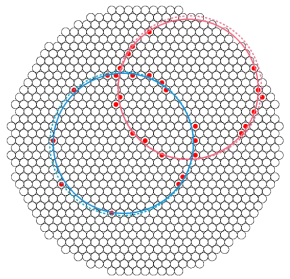
- Light focused by two mirrors on the two spots.
- **100 ps** time resolution.
- **10 MHz** event rate.
- **~18 hits** per ring on average.
- **70 byte** per event (max 32 hits per event).
- **Time reference** for trigger

Using GPUs in the NA62 L0 Trigger: the RICH Detector Case Study





- Rough detection of particle speed (radius) and direction (centre).

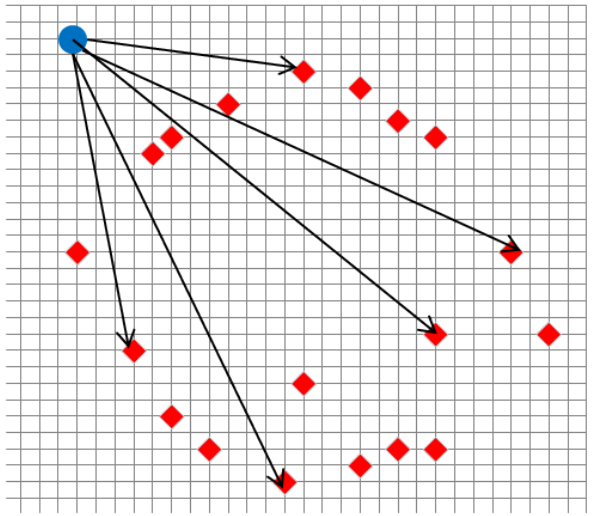


- Matching of partial circular hit patterns can be efficiently performed on GPUs.

GPU Ring Search Algorithms

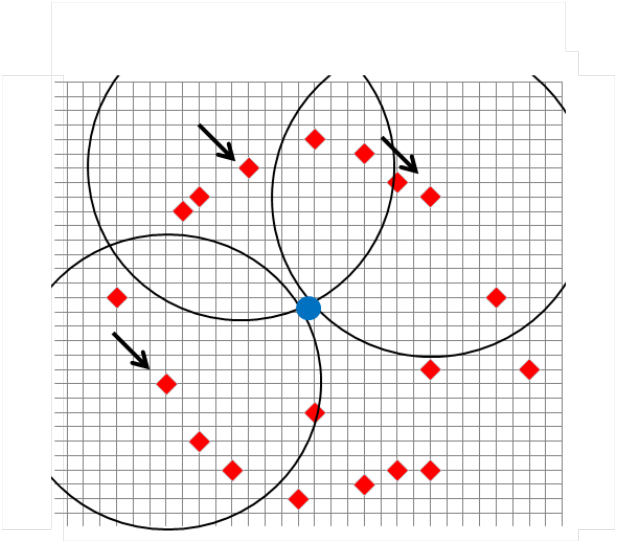
- Events can produce multiple ring hits patterns on PMs.
- 95% of generated hits patterns produce a single ring.
- Single ring search algorithms are used by multi-ring search algorithms.
- Let's focus on single ring pattern search algorithms.

- DOMH/POMH: Each PM (1000) is considered as the center of a circle. For each center an histogram is constructed with the distances btw center and hits.

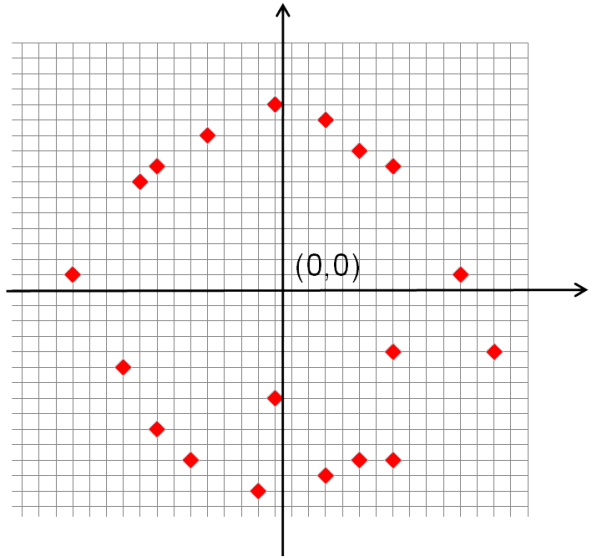


GPU Ring Search Algorithms (2)

- **HOUGH:** Each hit is the center of a test circle with a given radius. The ring center is the best matching point of the test circles. Voting procedure in a 3D parameters space.

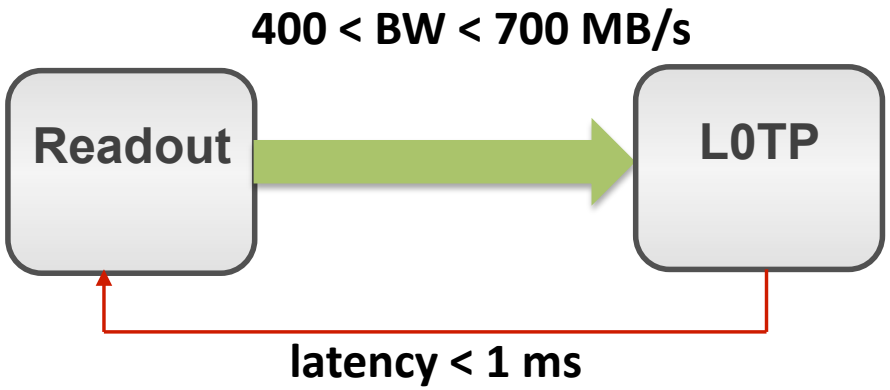


- **MATH:** Translation of the ring to centroid. In this system a least square method can be used. The circle condition can be reduced to a linear system , analitically solvable, without any iterative procedure.

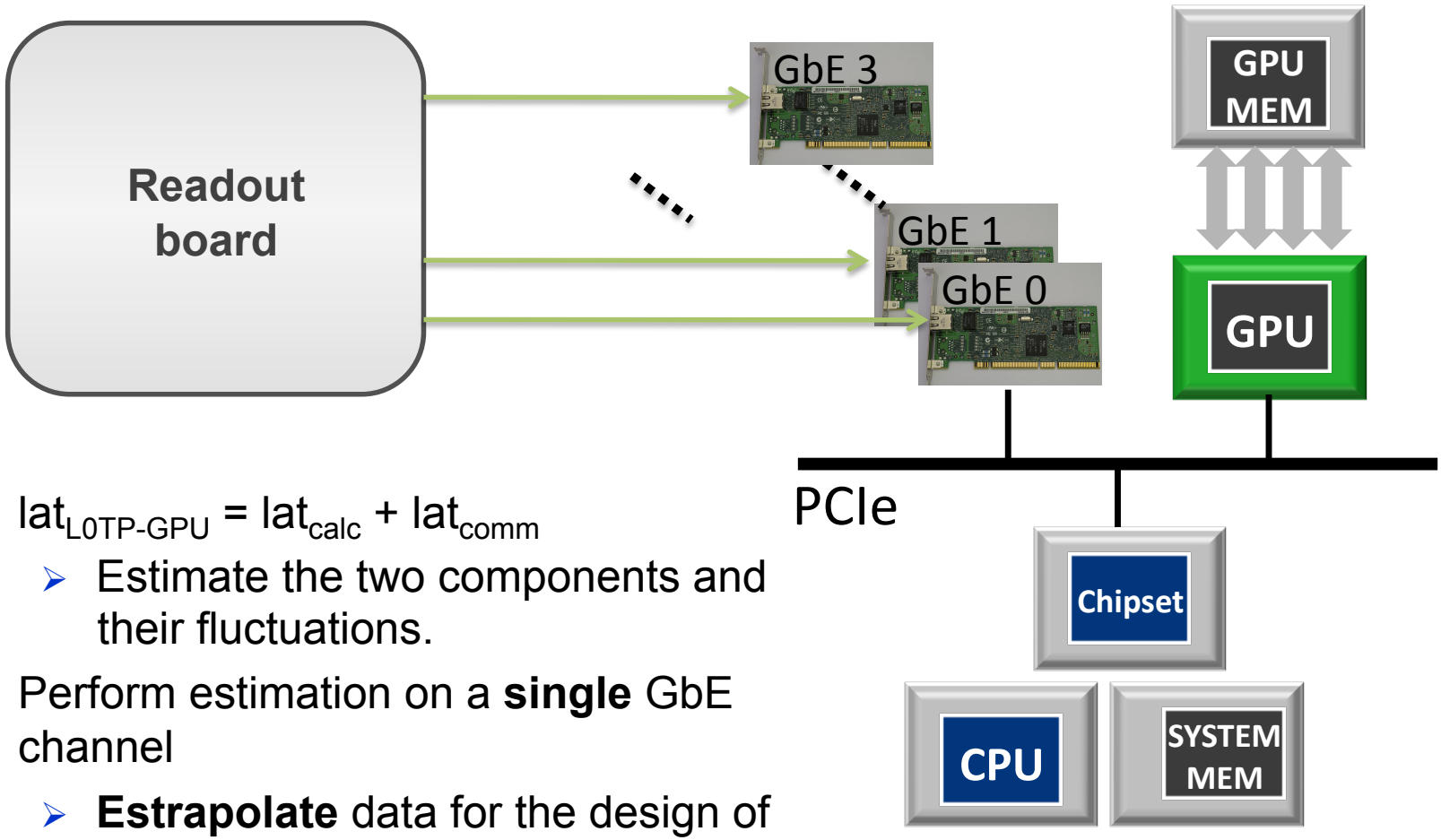


NA62 RICH L0 Trigger Processor Requirements

- Network Protocol: UDP on GbE channel.
- 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.



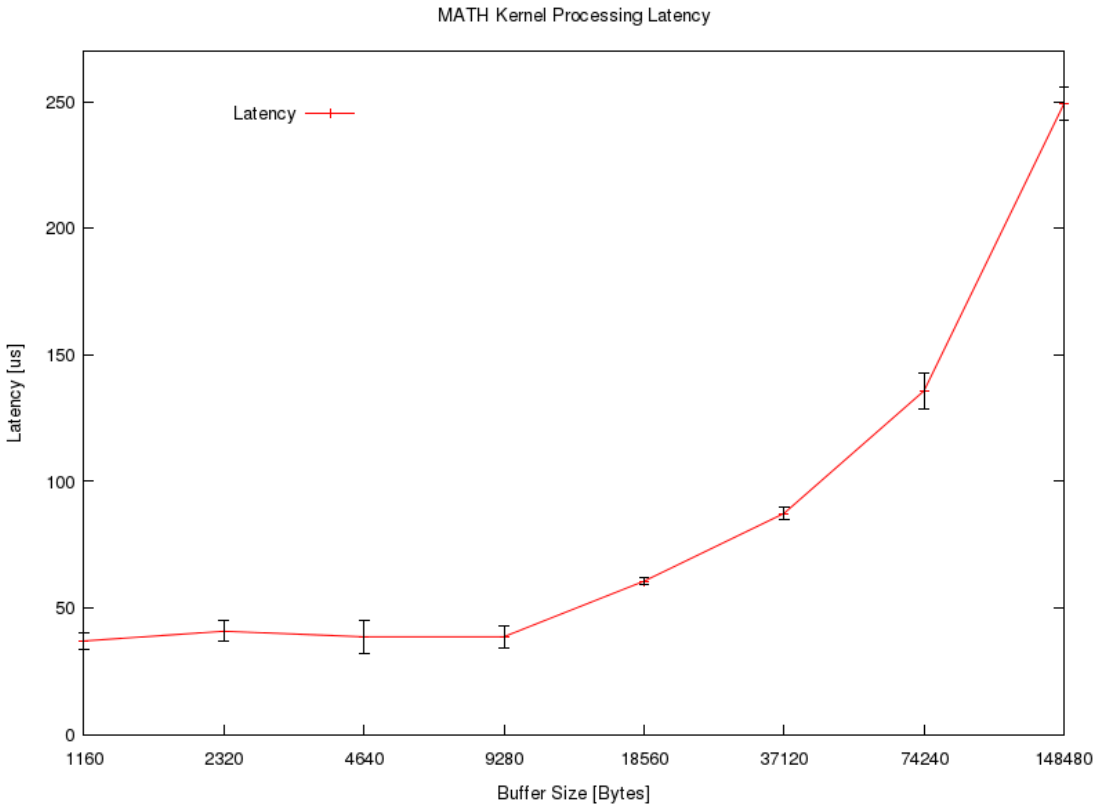
GPU-Based RICH Level 0 TP System Latency Estimation



- $lat_{L0TP\text{-}GPU} = lat_{calc} + lat_{comm}$
  - 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.

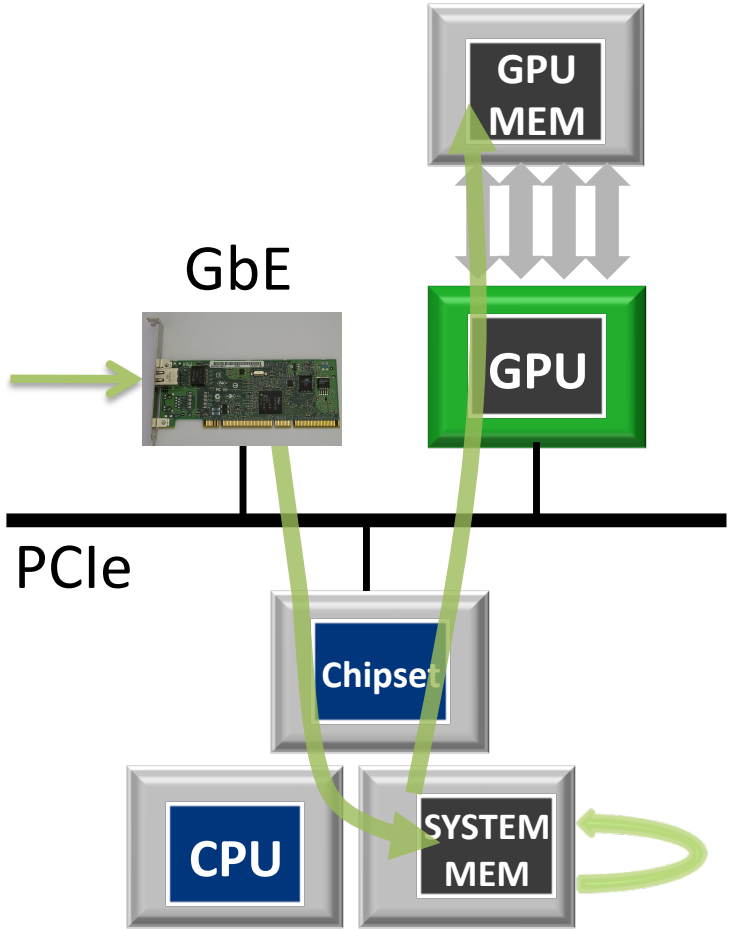


System Latency Estimation
Communication Latency

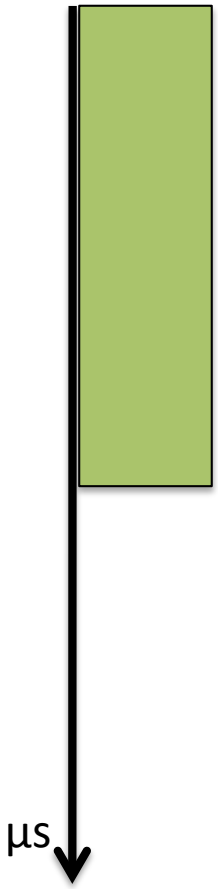
- **lat_{comm}** : time needed to receive input event data from GbE NIC to GPU memory.

- 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)
  - Other possible alternatives: low latency drivers (e.g. PF_RING), systems with dedicated HW for realtime, …

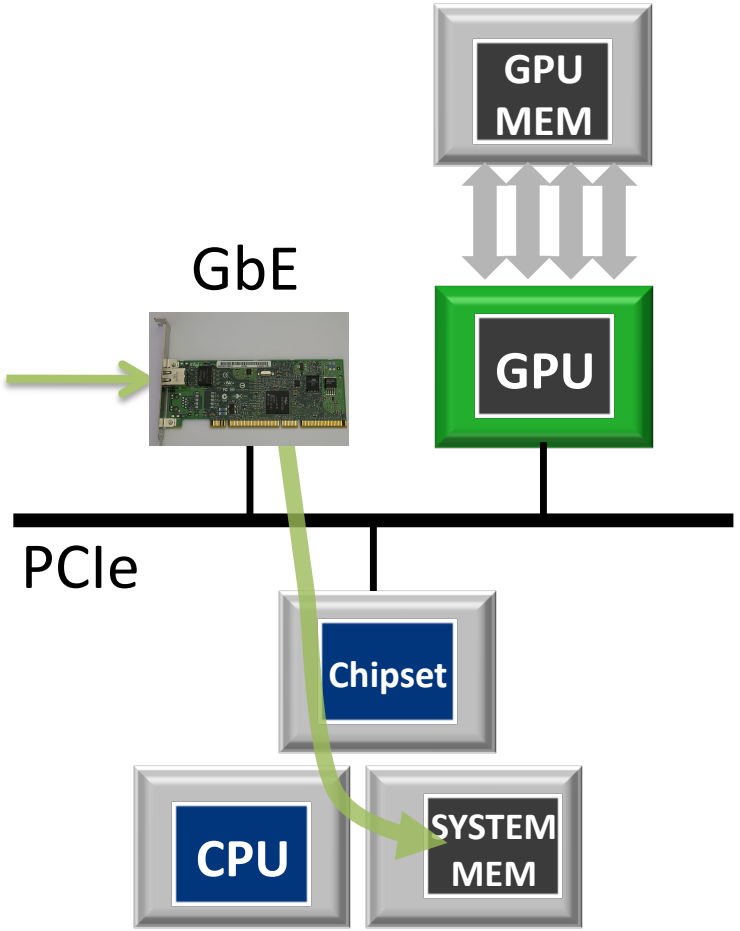
    not discussed here.



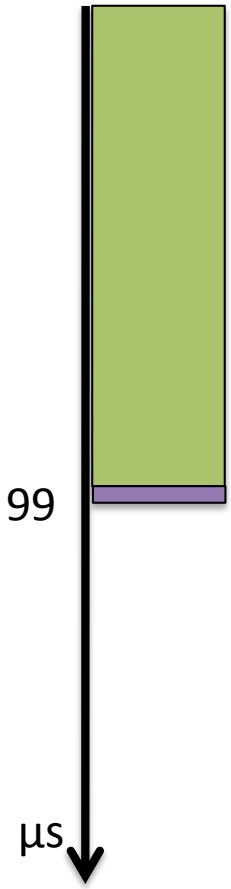
Communication Latency
Standard GbE NIC / SW Stack



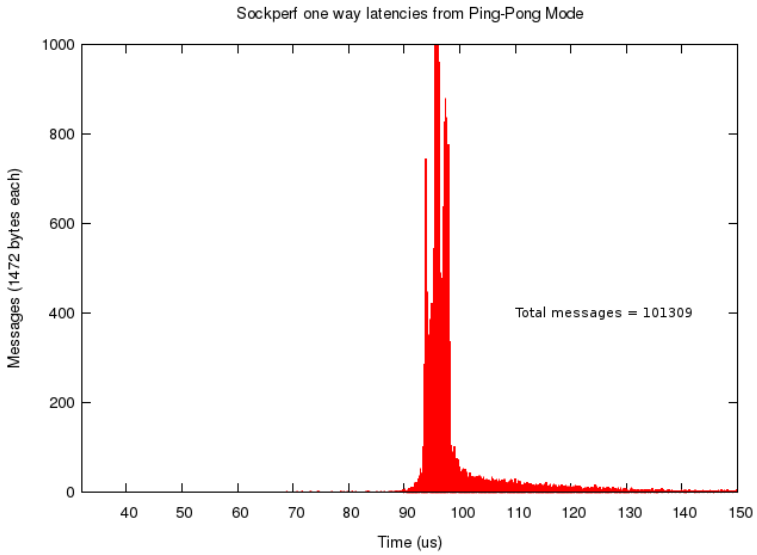
- 20 events data (1404 byte) sent from Readout board to the GbE NIC are stored in a receiving host kernel buffer.
- T=0 start of send operation on the Readout Board.

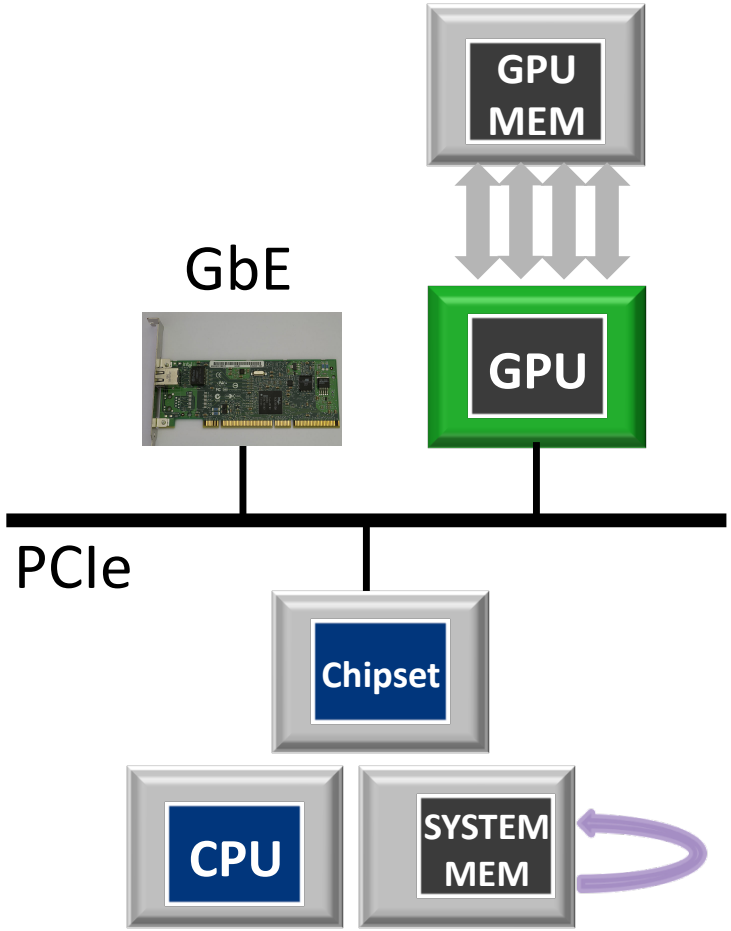


Communication Latency
Standard GbE NIC / SW Stack

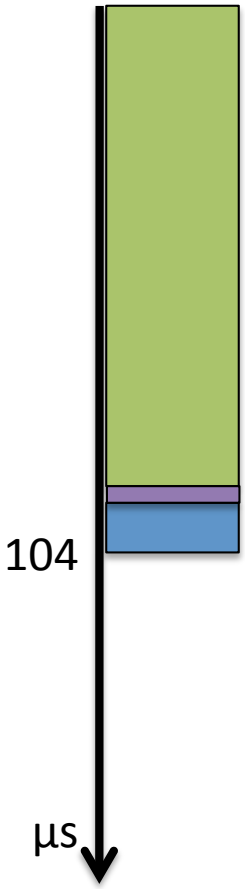


Data are copied from kernel buffer to destination buffer in user space.

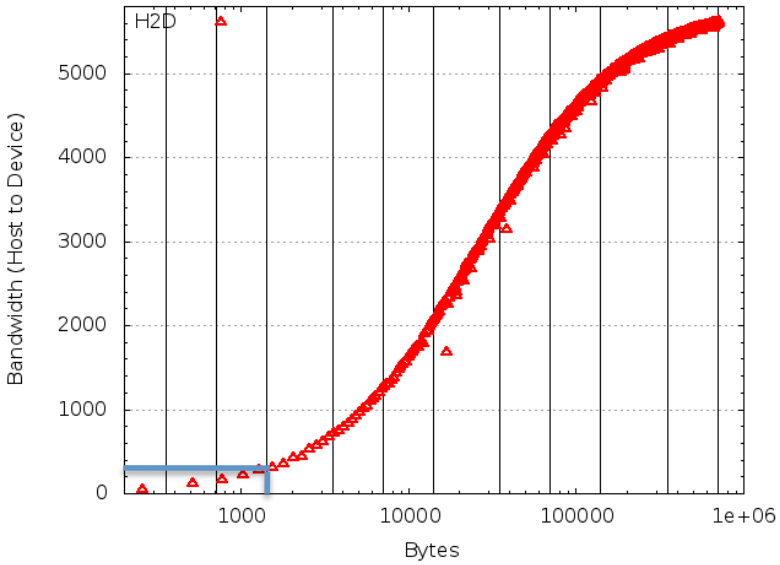


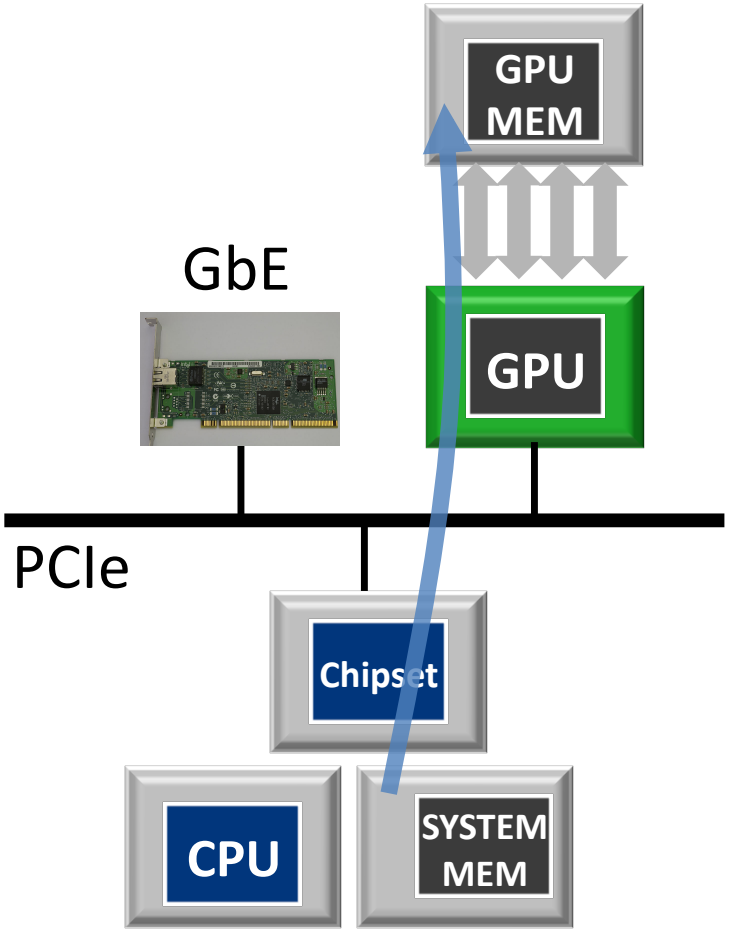


Communication Latency
Standard GbE NIC / SW Stack

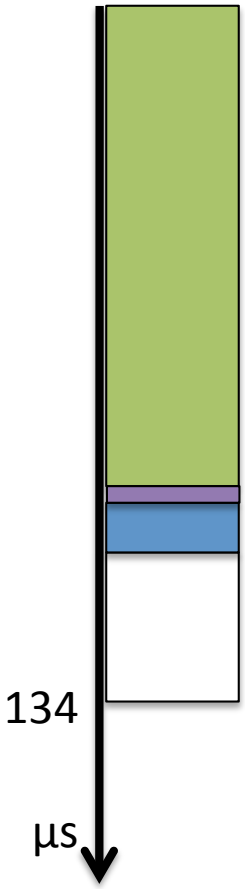


Data are copied from CPU memory to GPU memory

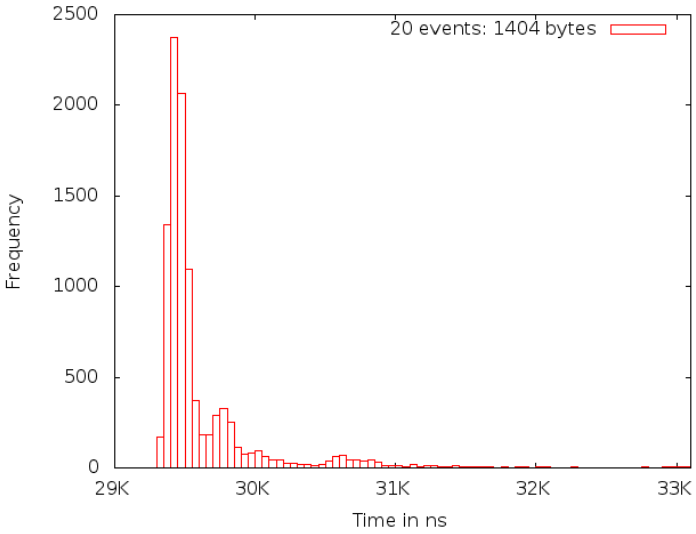


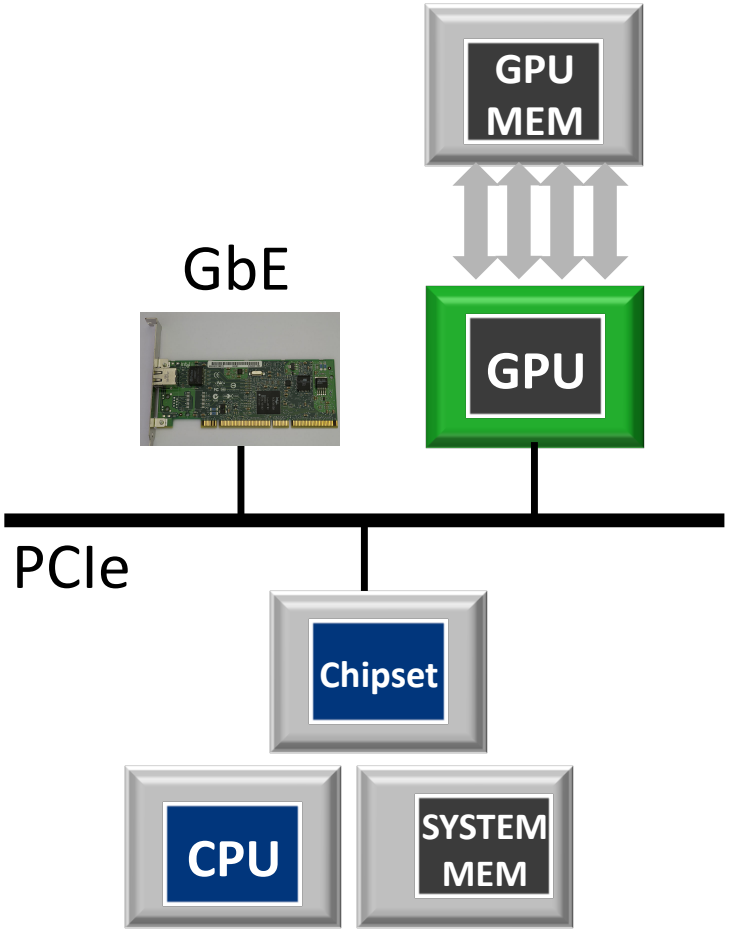


Communication Latency
Standard GbE NIC / SW Stack

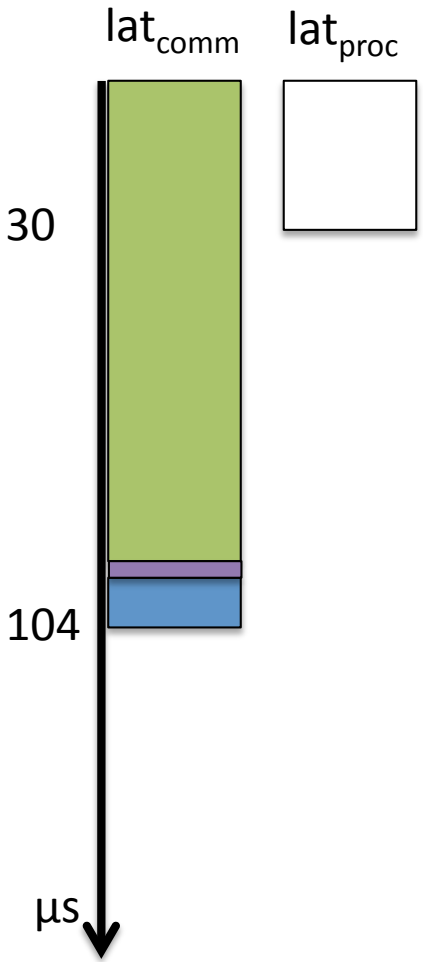


Ring pattern-matching GPU Kernel is executed, results are stored in device memory.

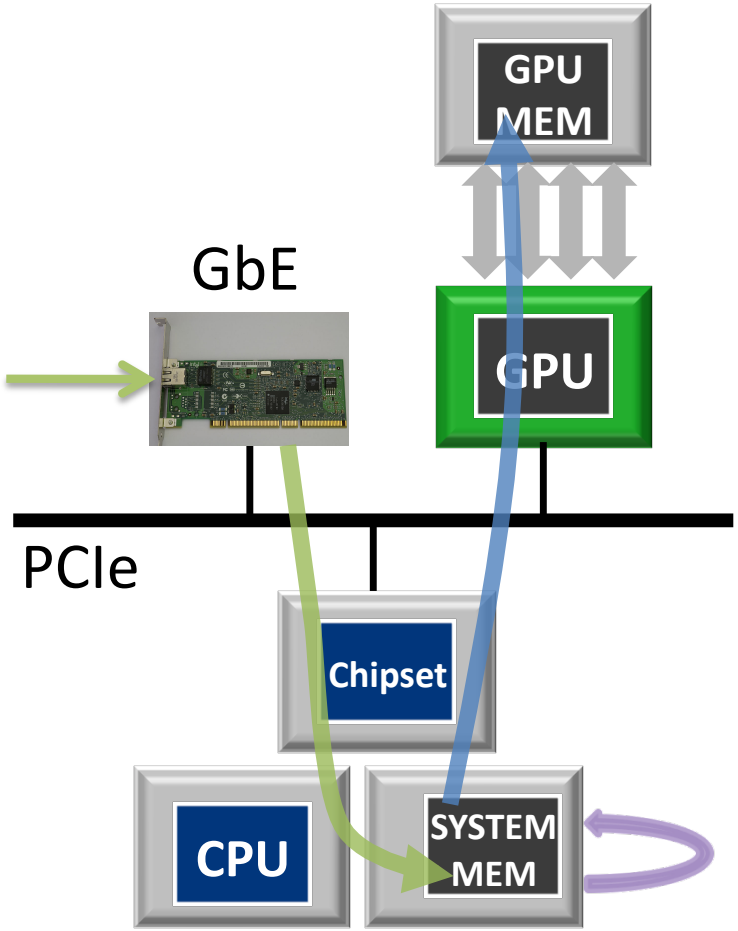




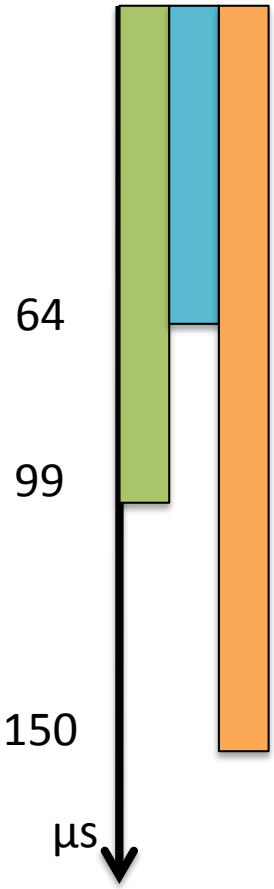
Communication Latency
Standard GbE NIC / SW Stack



- $lat_{comm} \cong 104\ \mu s$
- Total latency is dominated by communication latency: $lat_{comm} \cong 3.5 \times lat_{proc}$



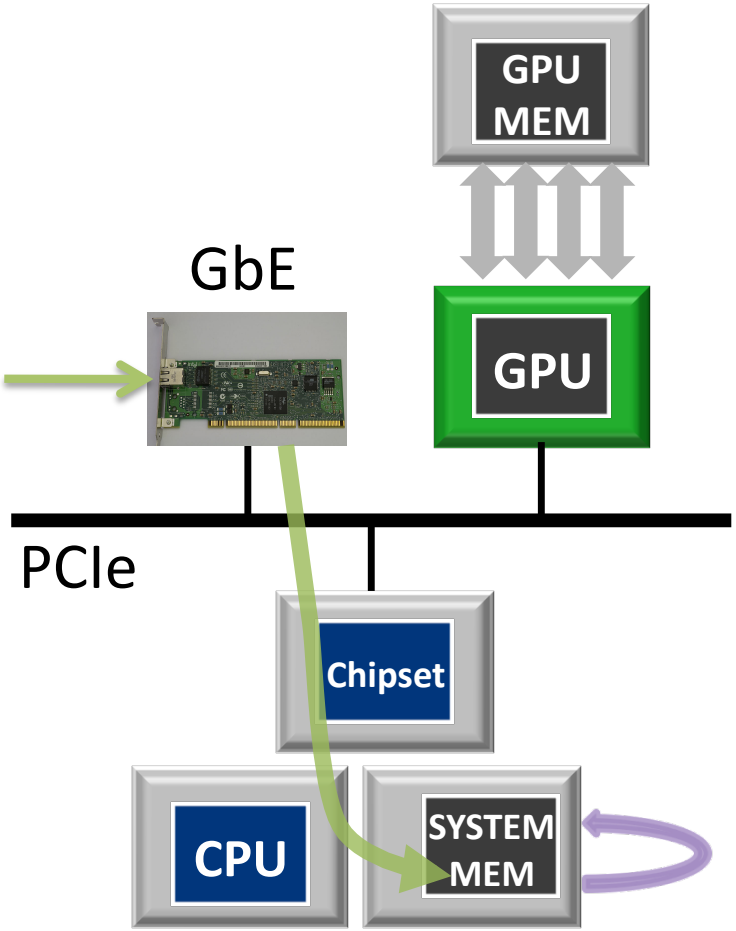
Standard GbE NIC / SW Stack Communication Latency Fluctuations



➢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

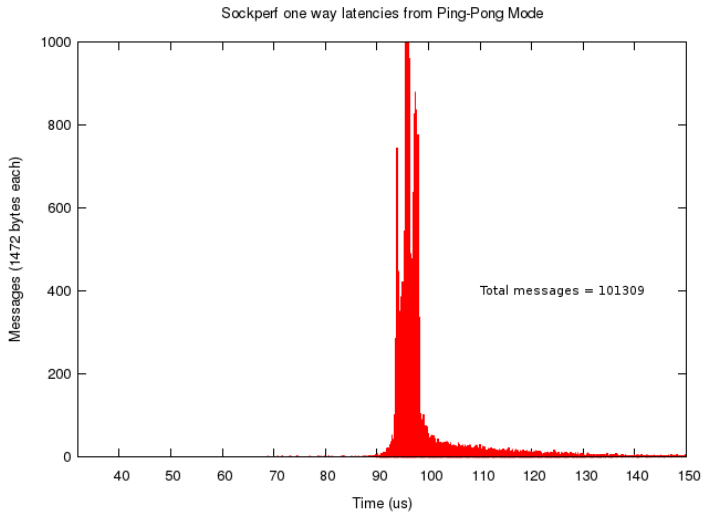
Might a Real-Time kernel come to our rescue?

Main features of such kernels:

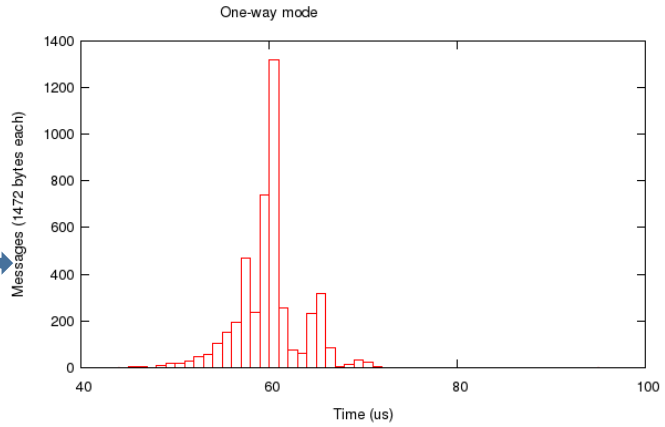
- predictability in response times
- reduced jitters
- microsecond accuracy
- improved time granularity

...but not a panacea...

vanilla kernel: 2.6.33



kernel 2.6.33.9-rt31-EL6RT



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

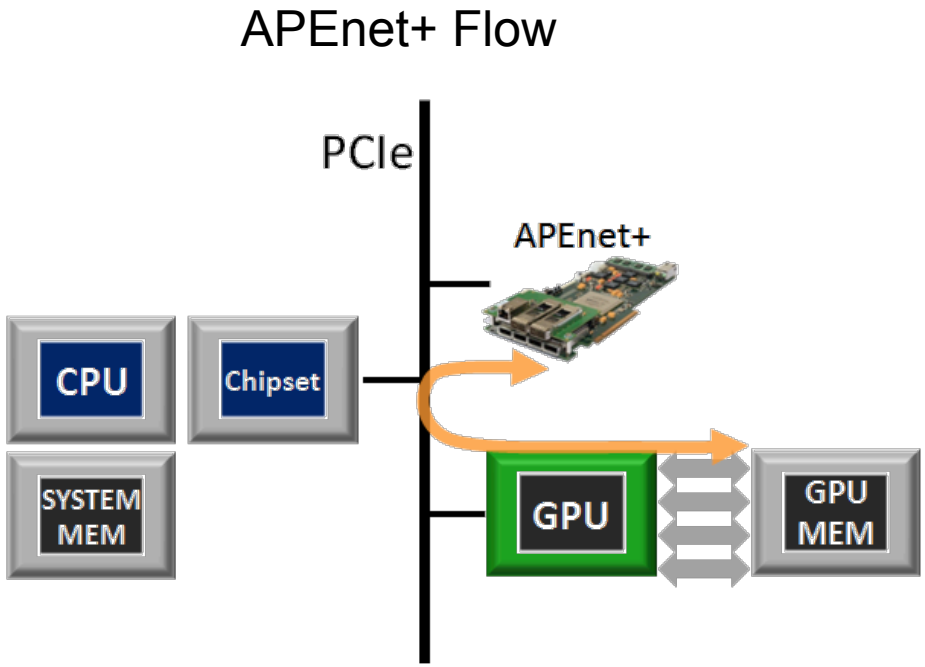
NaNet-1: a custom 1GbE NIC with GPUDirect RDMA capability

- 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**).

APEnet+ GPUDirect P2P Support



- 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.

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

3D Torus Network:

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

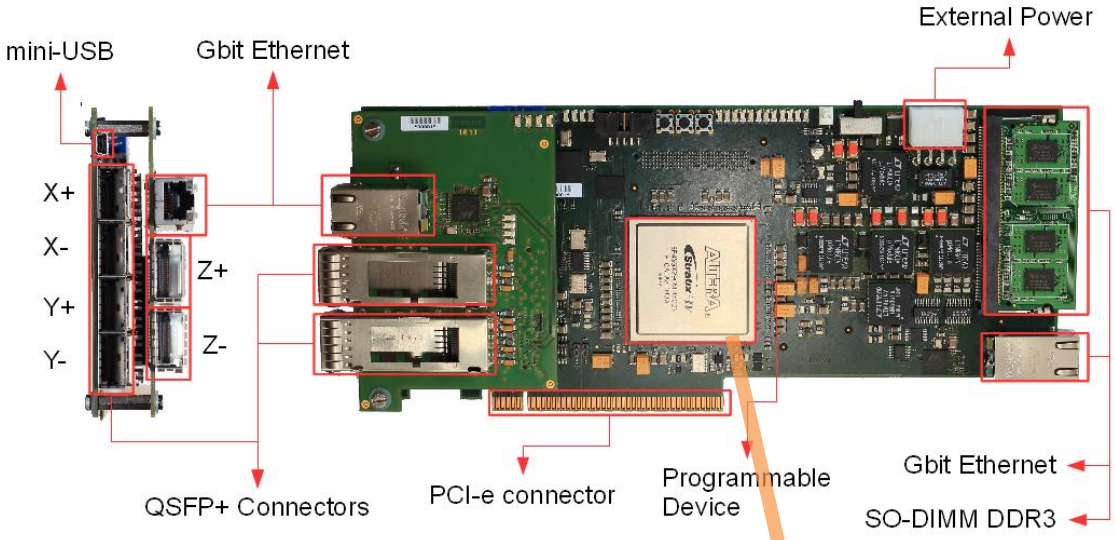
APEnet+ Card:

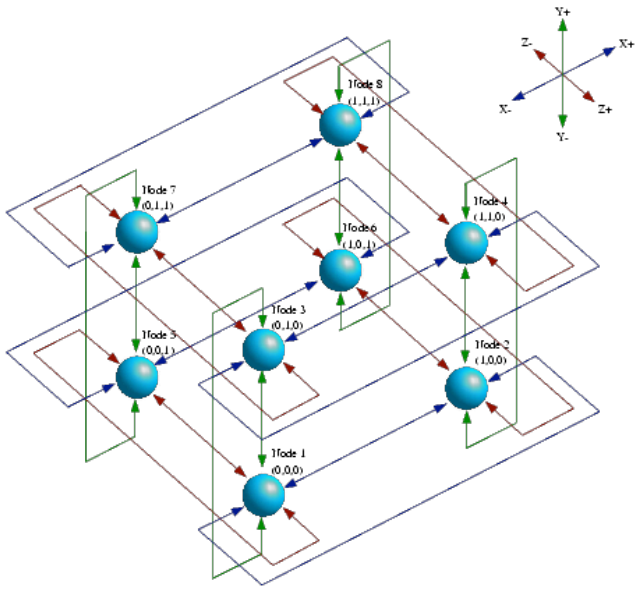
- FPGA based (ALTERA EP4SGX290)
- PCI Express x16 slot, signaling capabilities for up to dual x8 Gen2)
- Fully bidirectional torus links, 34 Gbps

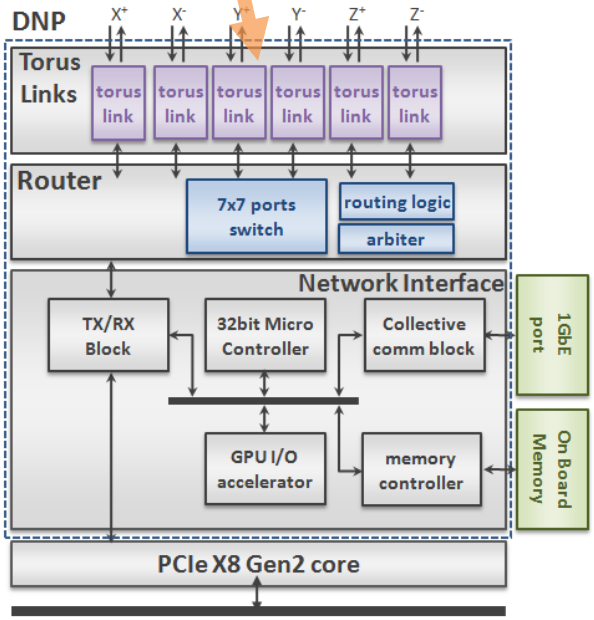
APEnet+ Logic:

- 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.

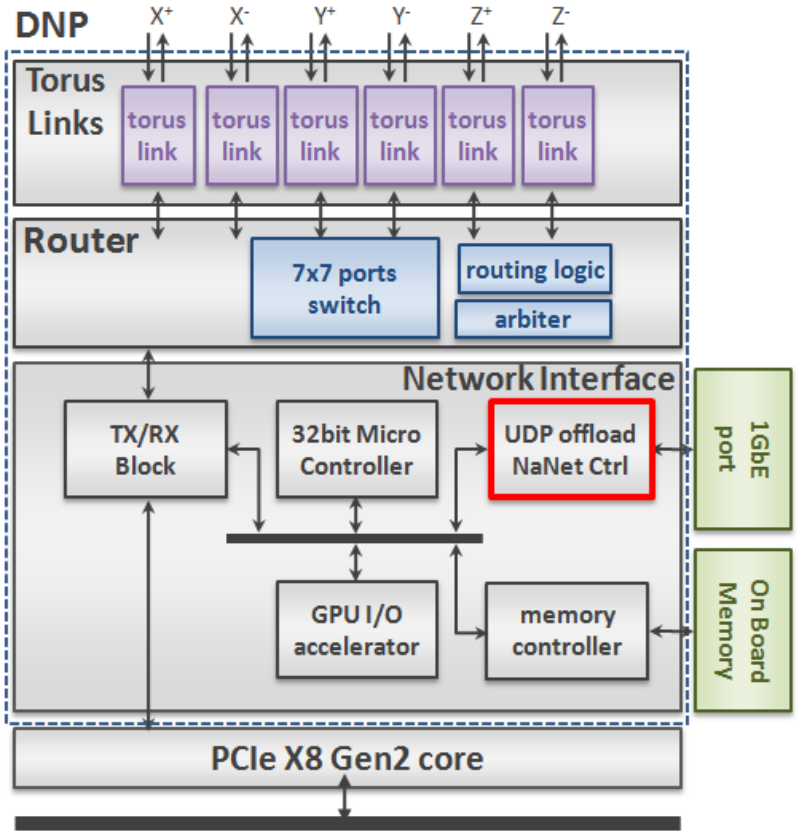




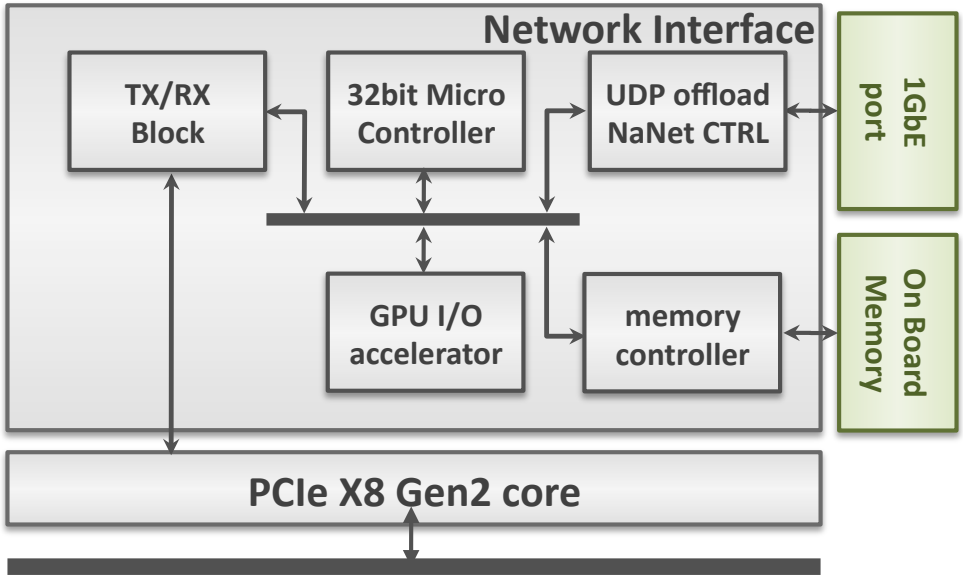


NaNet-1 Architecture

- NaNet-1 is based on APEnet+:
  - PCIe 2.0 P2P protocol for GPUDirect RDMA capability.
  - 6 high speed links (34 Gbps) are in the logic and can be used.
- New features:
  - UDP offload: extract the payload from UDP packets
  - NaNet Controller: encapsulate the UDP payload in a newly forged APEnet+ packet and send it to the RX NI logic



NaNet-1 Implementation

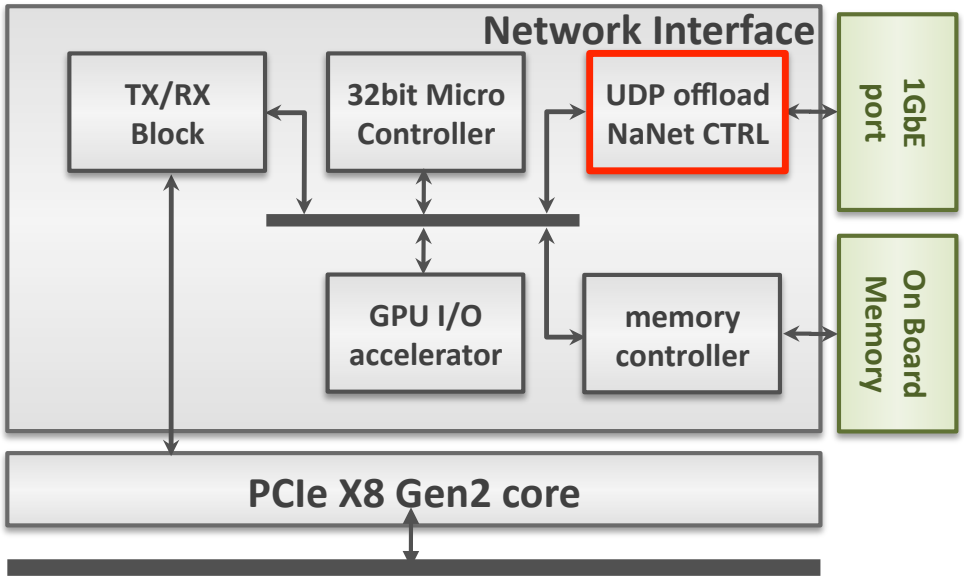


- Implemented on the Altera Stratix IV development system and in APEnet+ card.
- PHY Marvell 88E1111
- HAL based Nios II microcontroller firmware (no OS)
  - System Configuration and Initializazion
  - Virtual Memory Management for the GPUDirect RDMA

NaNet UDP Offload

NiosII UDP Offload :

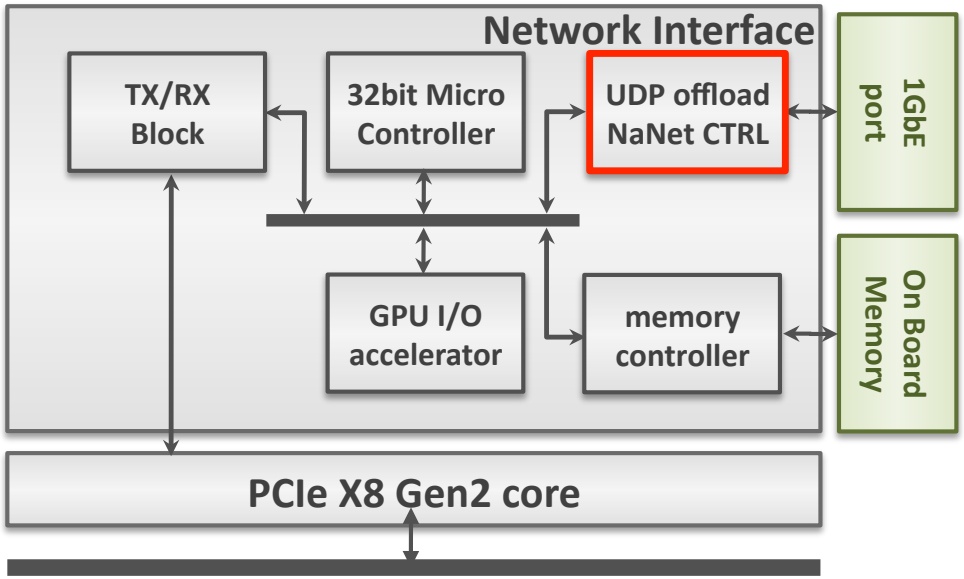
- Open IP: http://www.alterawiki.com/wiki/Nios_II_UDP_Offload_Example
- It implements a method for **offloading the UDP protocol management** from the Nios II microcontroller.
- It collects data coming from the Avalon Streaming Interface of the Altera Triple-Speed Ethernet Megacore (TSE MAC) and redirects 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

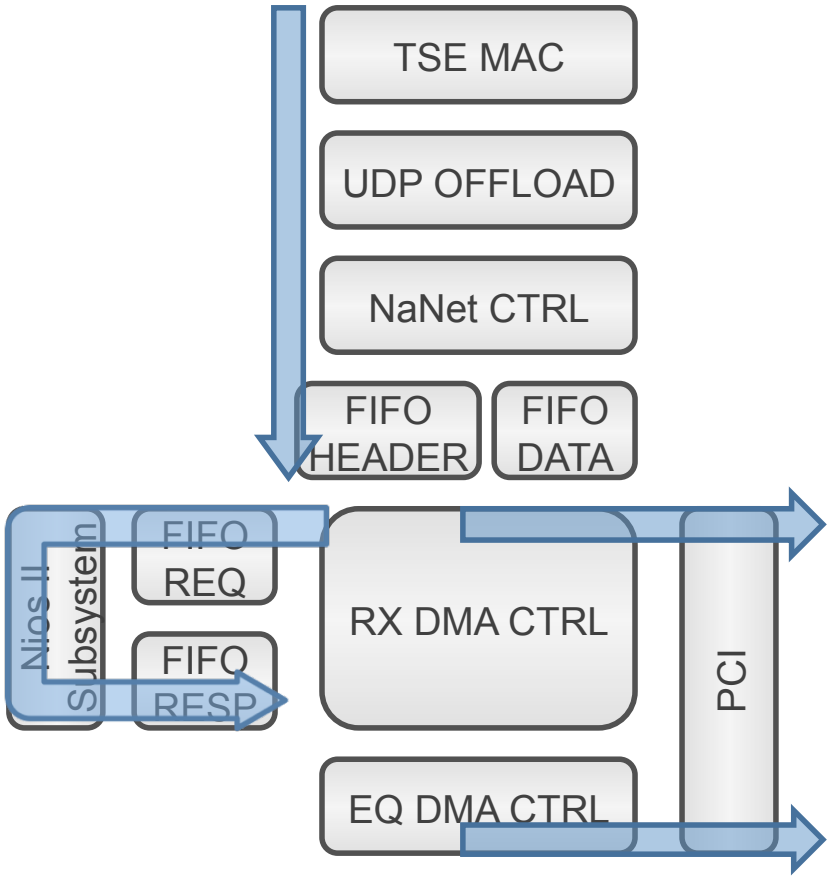
NaNet Controller:

- It manages the GbE flow by encapsulating packets in the APEnet+ packet protocol (Header, Payload, Footer)
- 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)



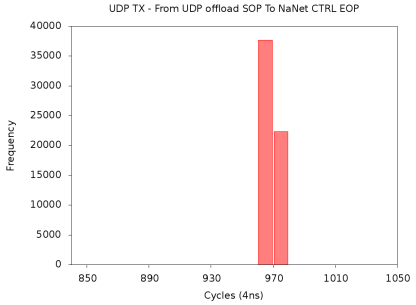
NaNet Data Flow

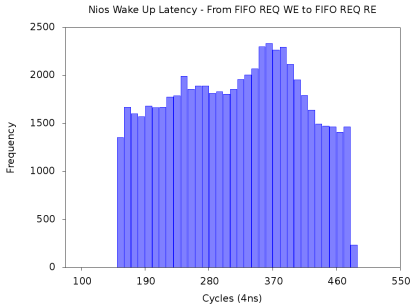
- TSE MAC, UDP OFFLOAD and NaNet CTRL manage the GbE data flow and encapsulate data in the APEnet+ protocol.
- RX DMA CTRL:
  - It manages CPU/GPU memory write process, providing hardware support for the Remote Direct Memory Access (RDMA) protocol
  - It communicates with the μC :
    - It generates 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**

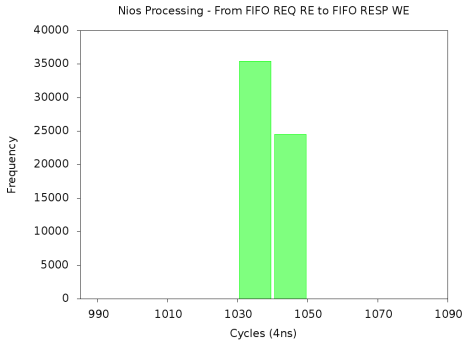


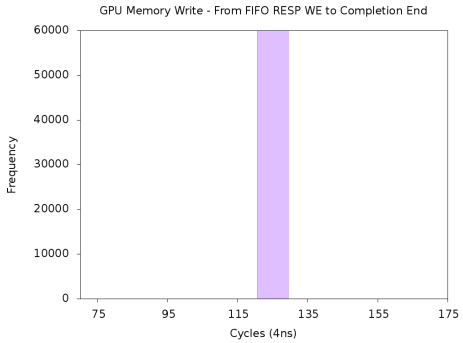
UDP Packet Latency Inside NaNet-1 Detailed Analysis

- 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



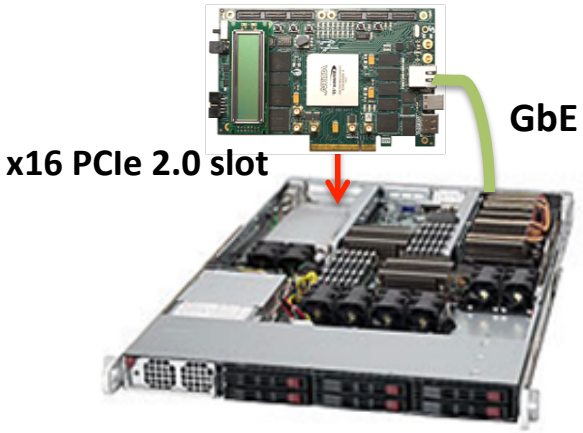






NaNet-1 Test & Benchmark Setup

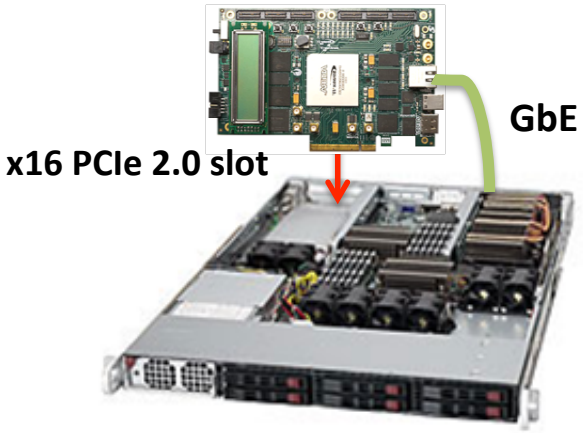
- Supermicro SuperServer 6016GT-TF
  - X8DTG-DF motherboard (Intel Tylersburg chipset)
  - dual Intel Xeon E5620
  - Intel 82576 Gigabit Network Connection
  - Nvidia Fermi M2070
  - kernel 2.6.32-358.6.2.el6.x86_64, CUDA 4.2, Nvidia driver 325.15.
- NaNet-1 board in x16 PCIe 2.0 slot
- NaNet-1 GbE interface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



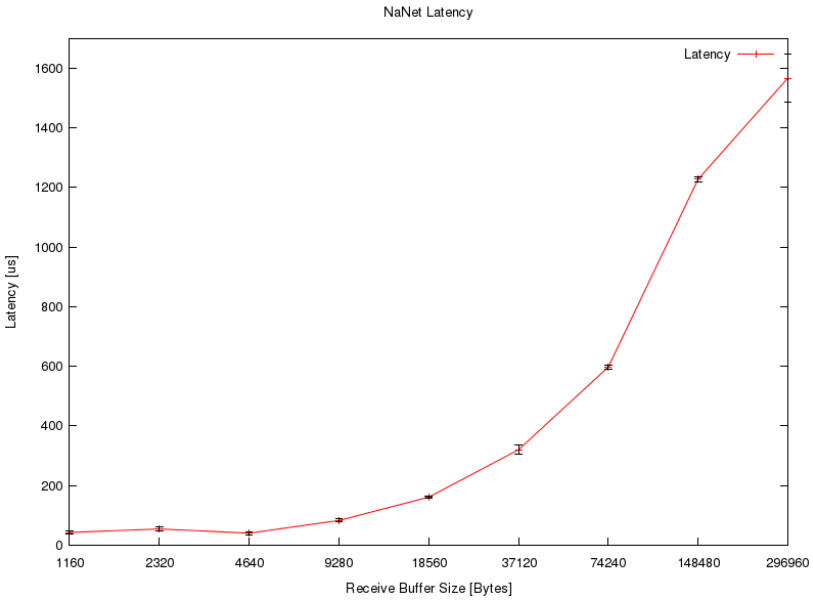
NaNet-1 Latency Benchmark

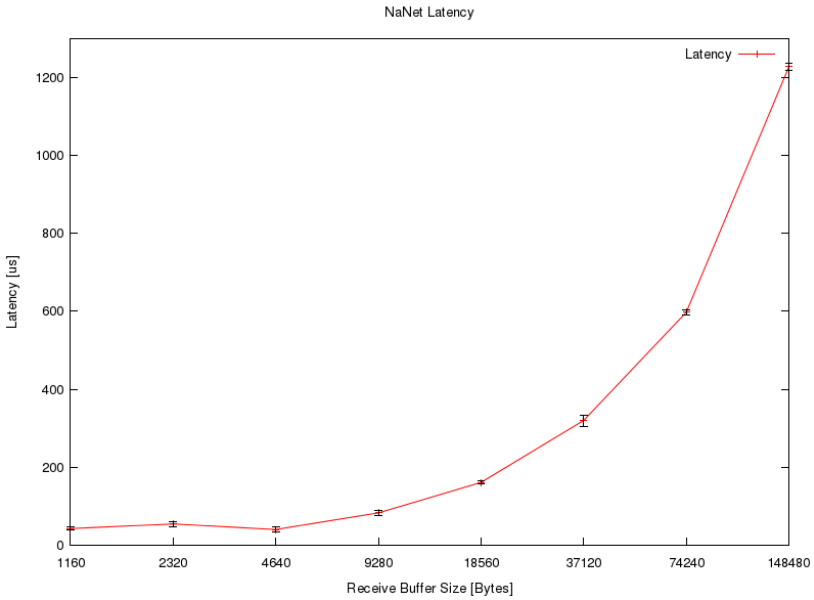
In a single host process:

- Allocate and register (pin) *N* GPU receive buffers of size *P* x1160 byte (*P* x 16 events) in a **circular list**.
- In the main loop:
  - Read TSC Register (*cycles_bs*)
  - Send *P* UDP packet with 1160 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 coherent with oscilloscope data.



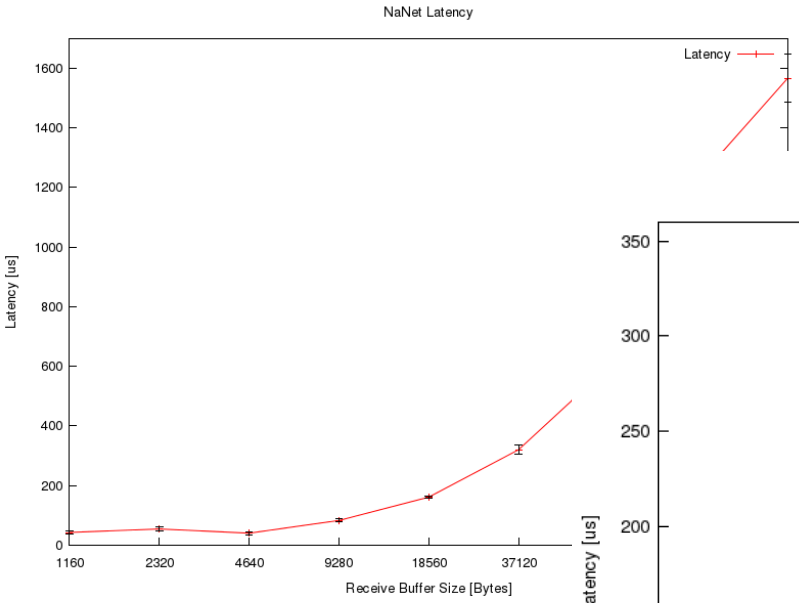
NaNet-1 Communication Latency

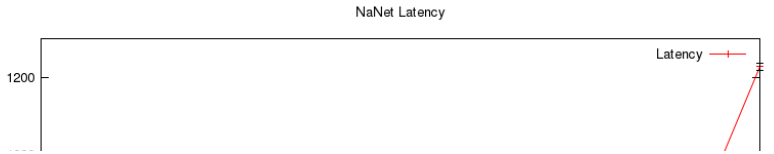


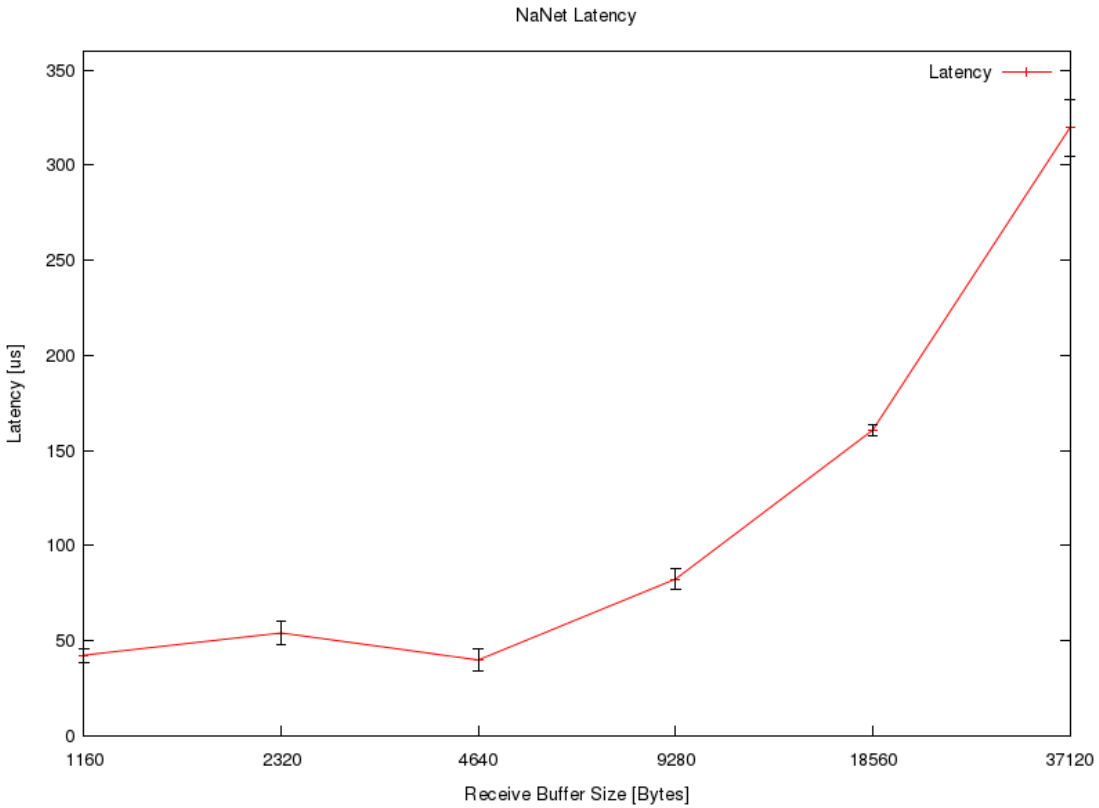


- Receive buffer in GPU global memory
- Lower and more stable results

NaNet-1 Communication Latency

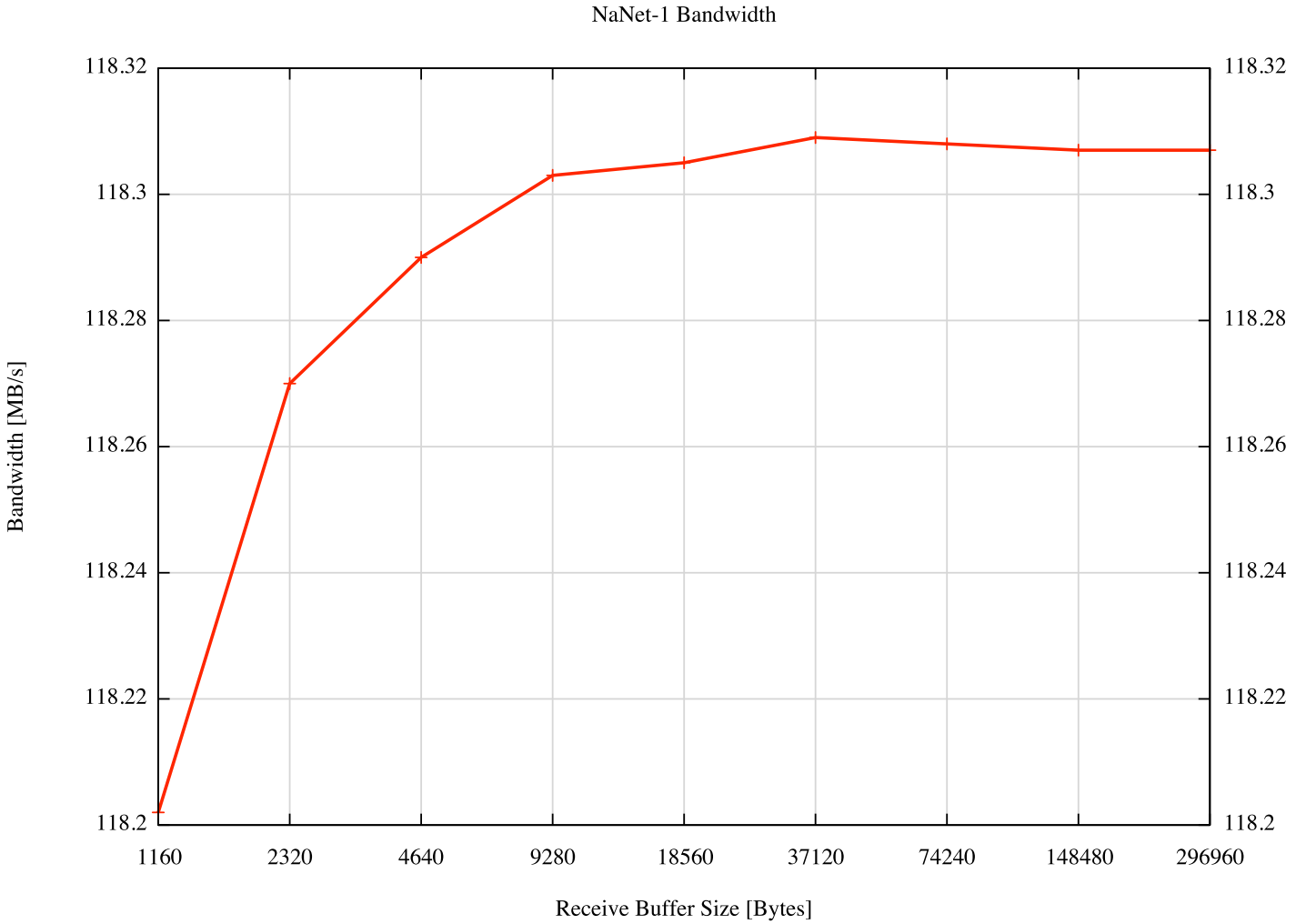




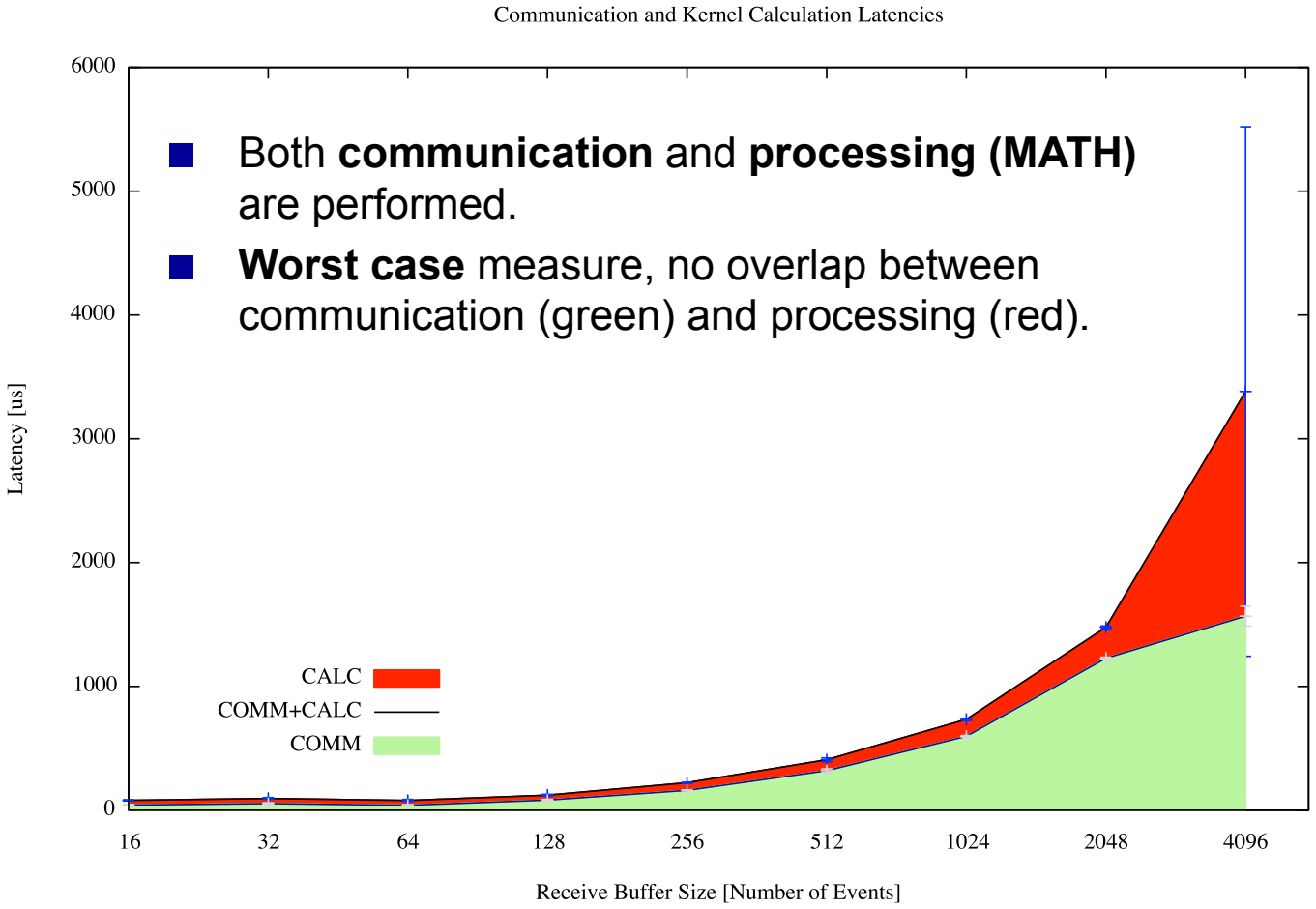


Even for small buffer sizes

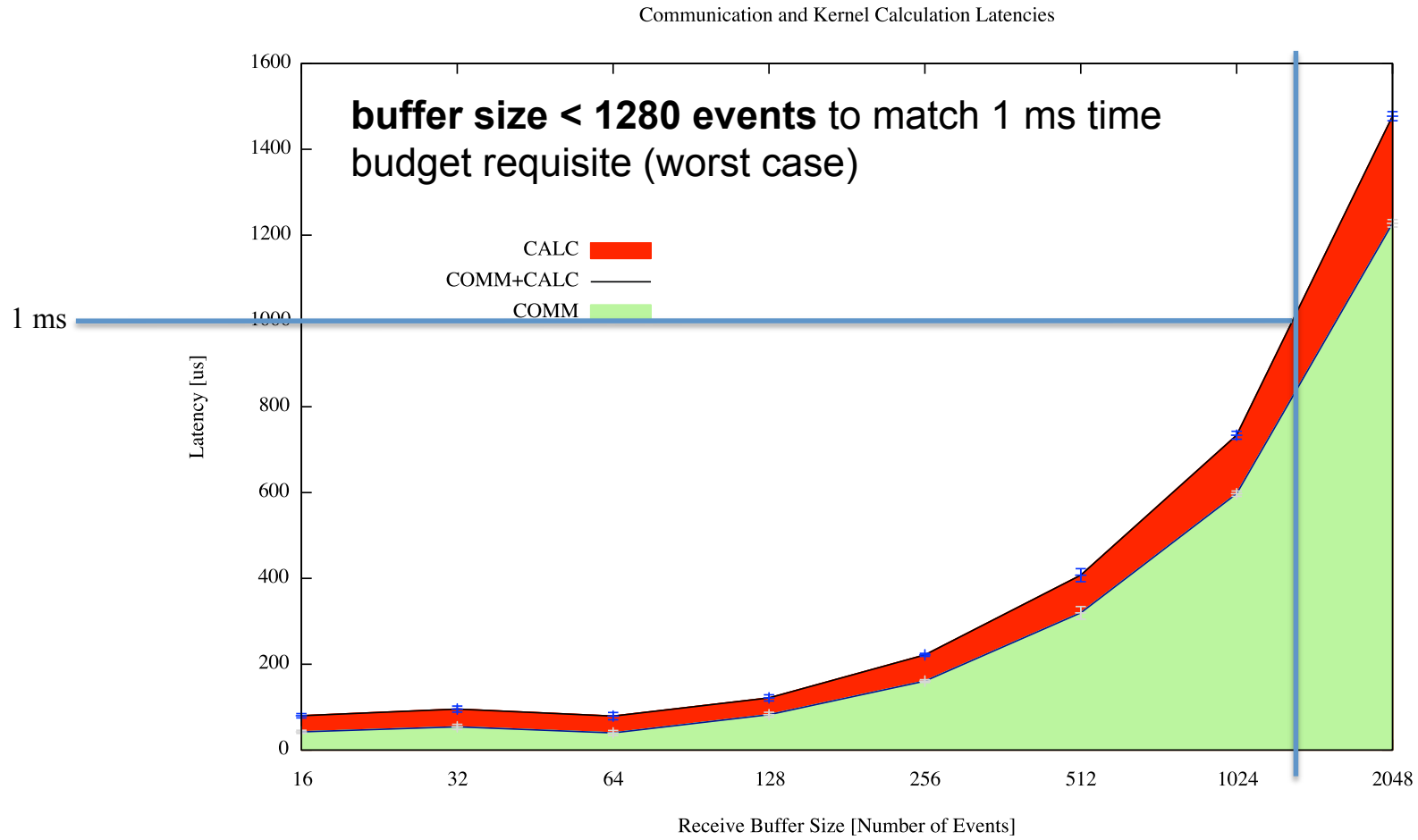
NaNet-1 Bandwidth



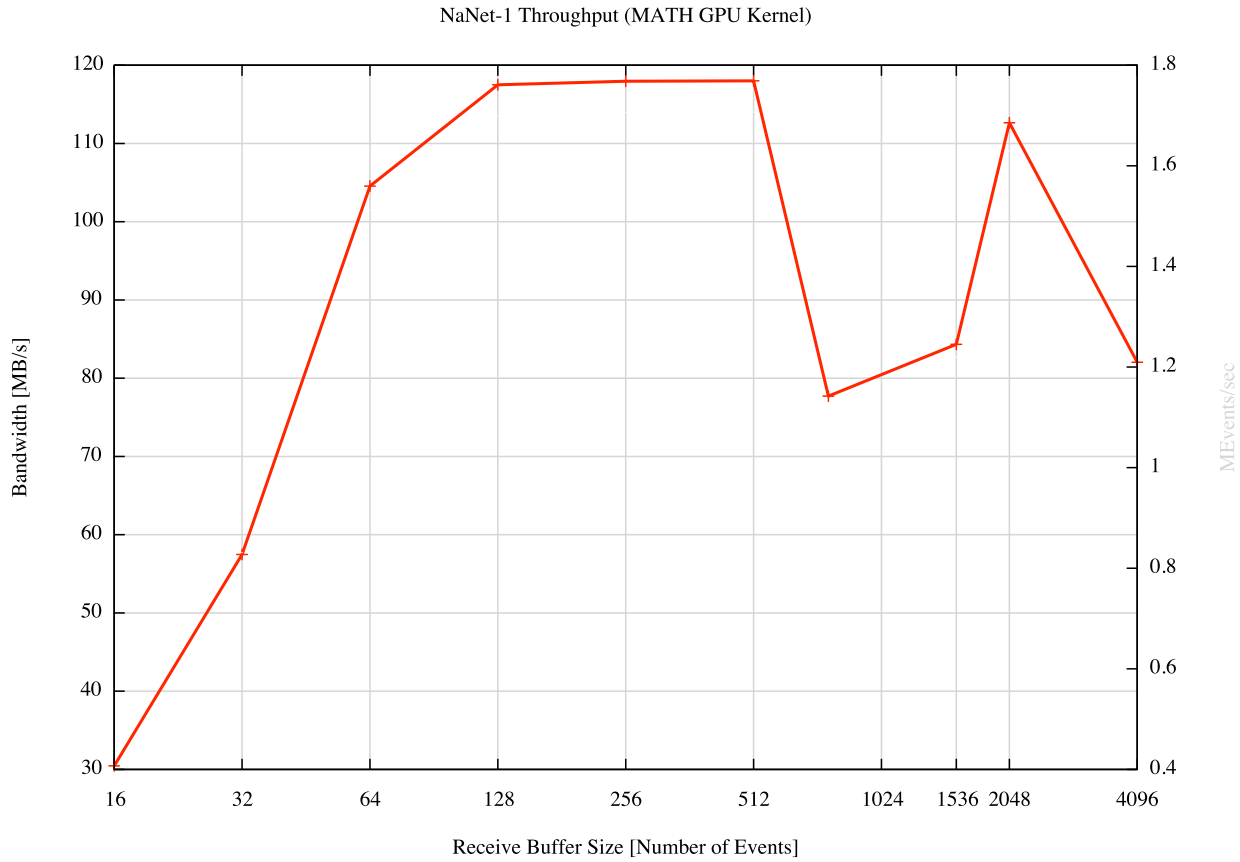
Total latency of the GPU-Based RICH L0-TP using NaNet-1 (1)



Total latency of the GPU-Based RICH L0-TP using NaNet-1 (2)



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



- 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.

Conclusions

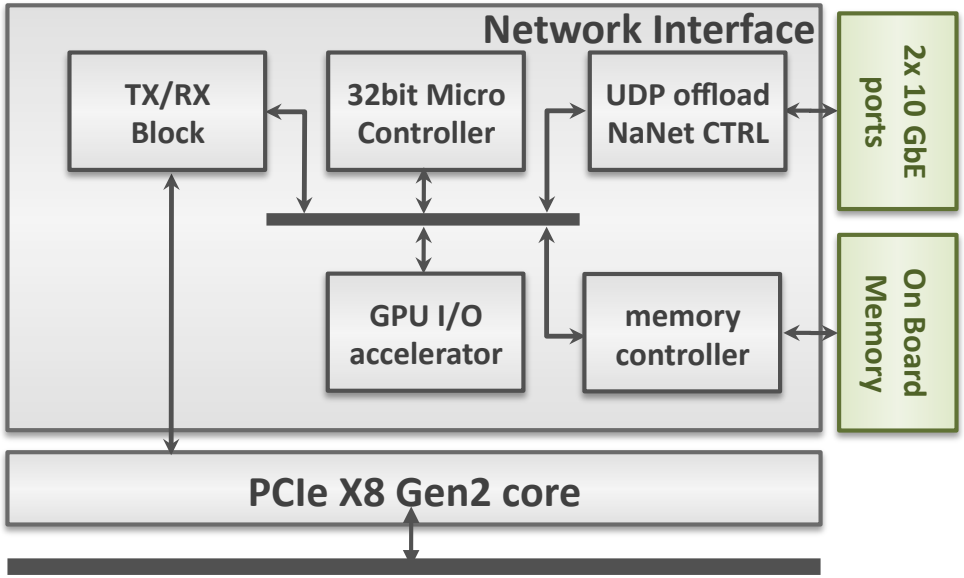
- In NaNet-1 perfoms significantly better in terms of latency compared with other considered approaches.

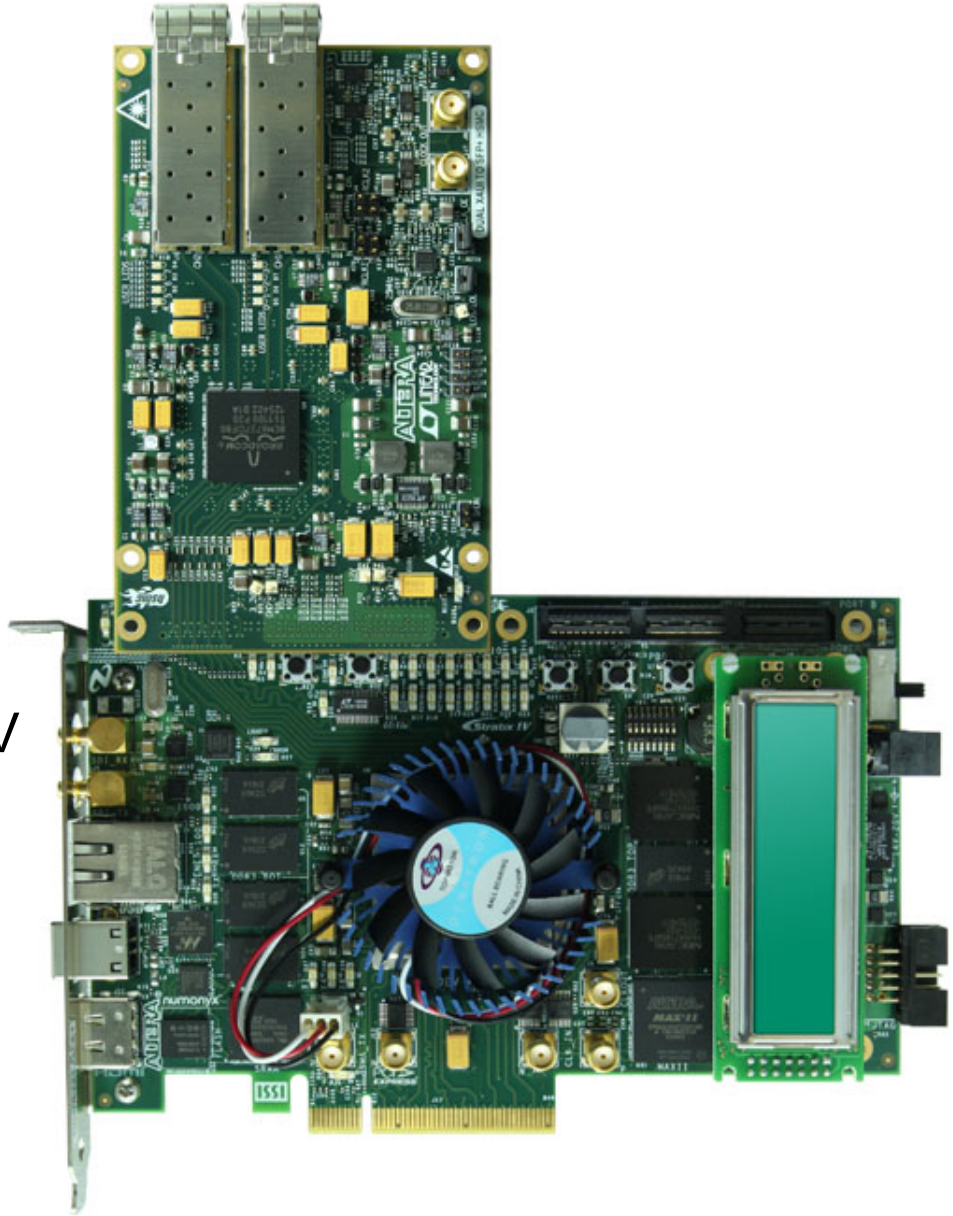
- A RICH detector Level 0 Trigger Processor GPU-based system using NaNet-1 (1 GbE link) is able to sustain a throughput of **~1.8 Mevents/s** being able to match the real-time requisite.

Current & Future Work

- Implement a dedicated custom logic block to completely offload the microcontroller from address generation (started)
- Scale up NaNet-1 implementation to support 4 GbE links
  - Terasic DE4 development Board
- Support 10 GbE links (started).
- Re-use NaNet design concepts in other experiments (KM^3,…) (started)

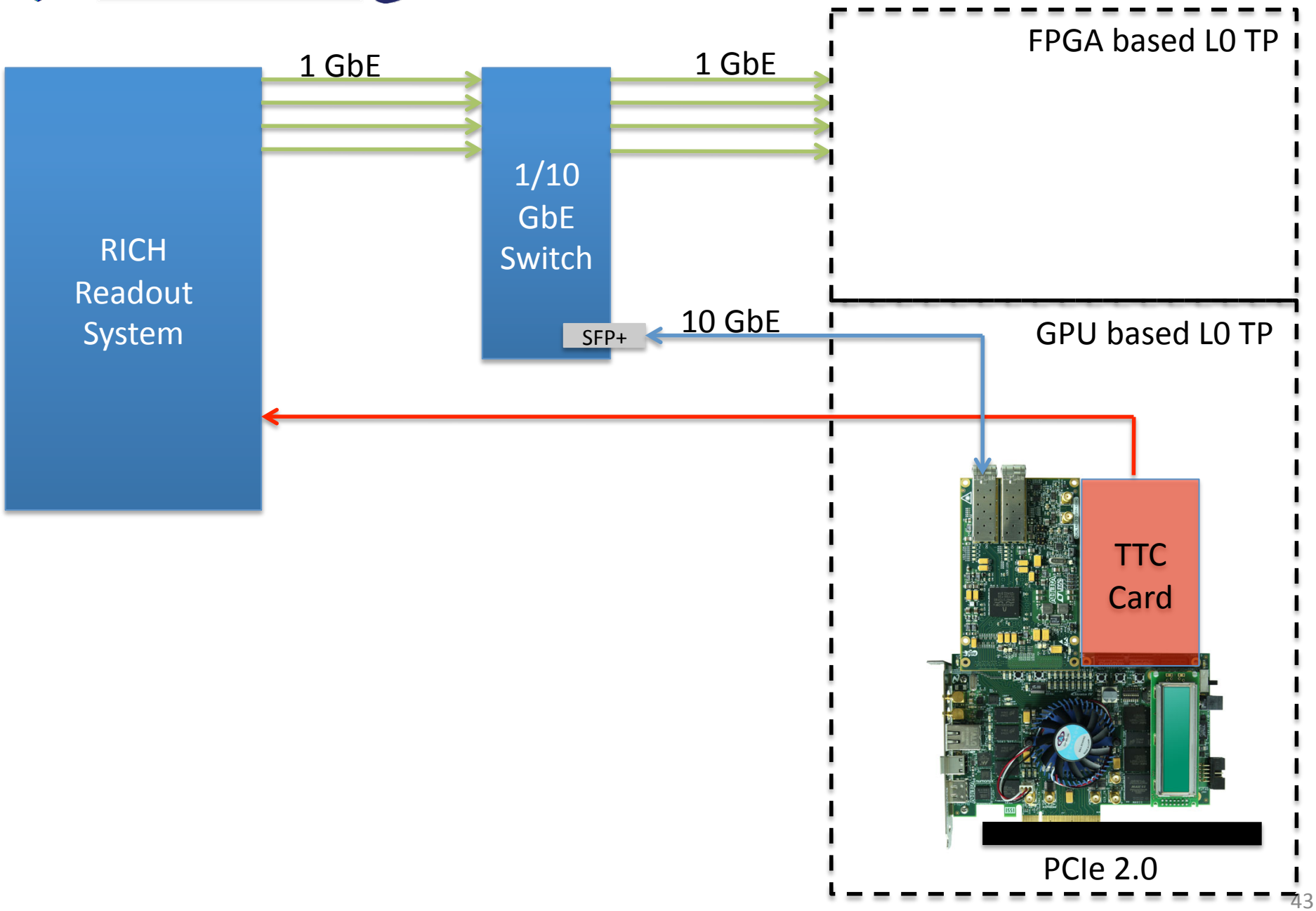
NaNet-10 (dual 10 GbE) work in progress





- Implemented on the Altera Stratix IV dev board + Terasic HSMC Dual XAUI to SFP+ daughtercard
- BROADCOM *BCM8727* a dual-channel 10-GbE SFI-to-XAUI transceiver

NaNet-10 for NA62 RICH L0 GPU Trigger Processor



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)]

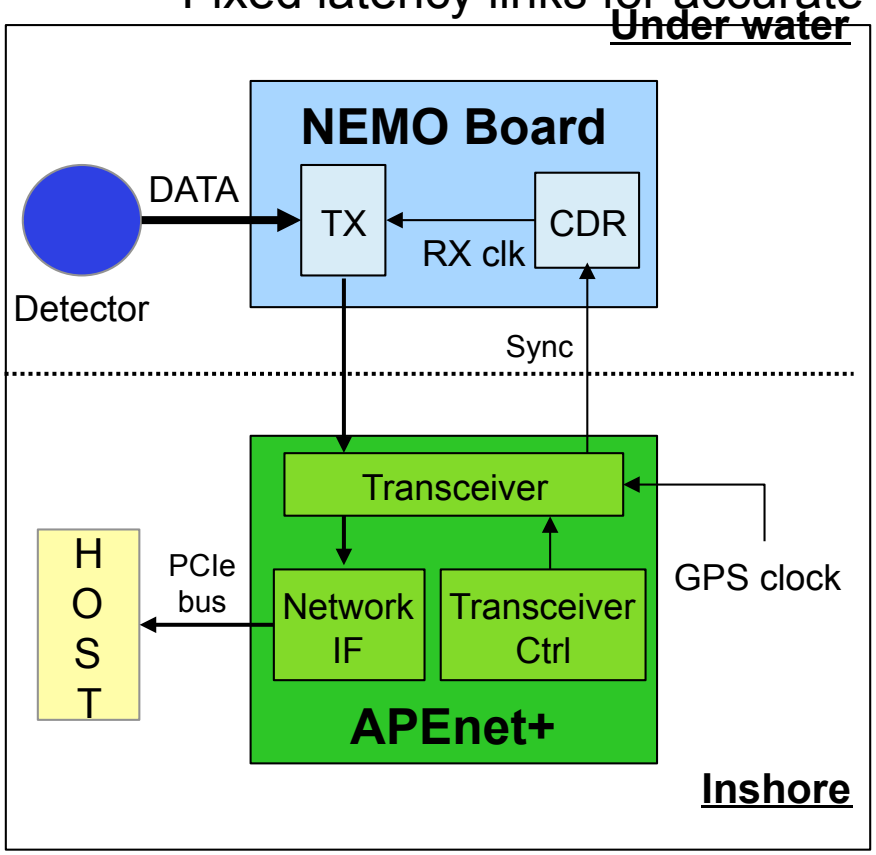
(a) INFN Sezione di Roma Tor Vergata (b) INFN Sezione di Roma (c) INFN Sezione di Pisa (d) Università di Roma "Sapienza"

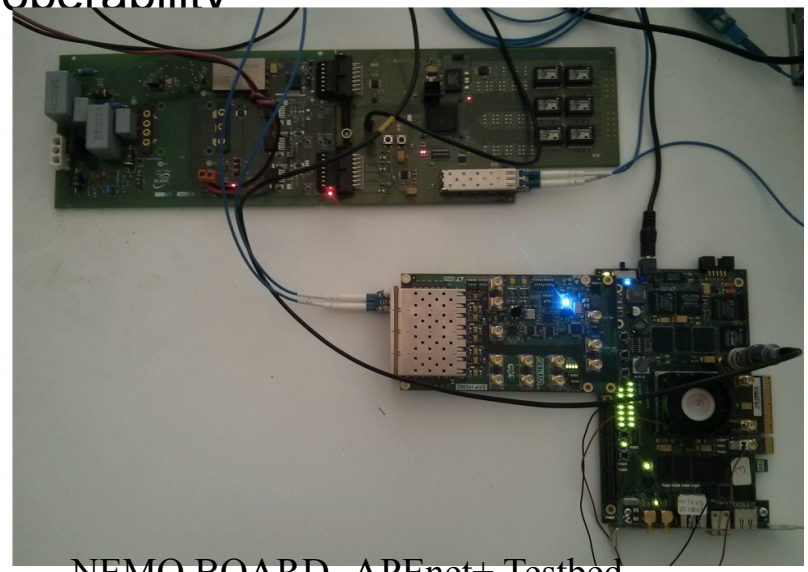

THANK YOU

BACK-UP SLIDES

APEnet+ / KM3NeT : A custom NIC for low and deterministic latency

- **KM3NeT** aims to develop and to deploy an European deep-sea research infrastructure, hosting a neutrino telescope with a volume of cubic kilometers at the bottom of the Mediterranean Sea.
- A customized version of APEnet+ wil be employed as a low latency, high performance on-shore readout system.
- Two main challenges:
  - Xilinx and Altera embedded links interoperability
  - Fixed latency links for accurate timing





NEMO BOARD -APEnet+ Testbed

System features:

- KM3Net Board on Xilinx FPGA
- APEnet+ transceiver: Altera deterministic latency HIP @80 MHz
- Channel: 800 Mbit/s


P2P advantages

P2P means:

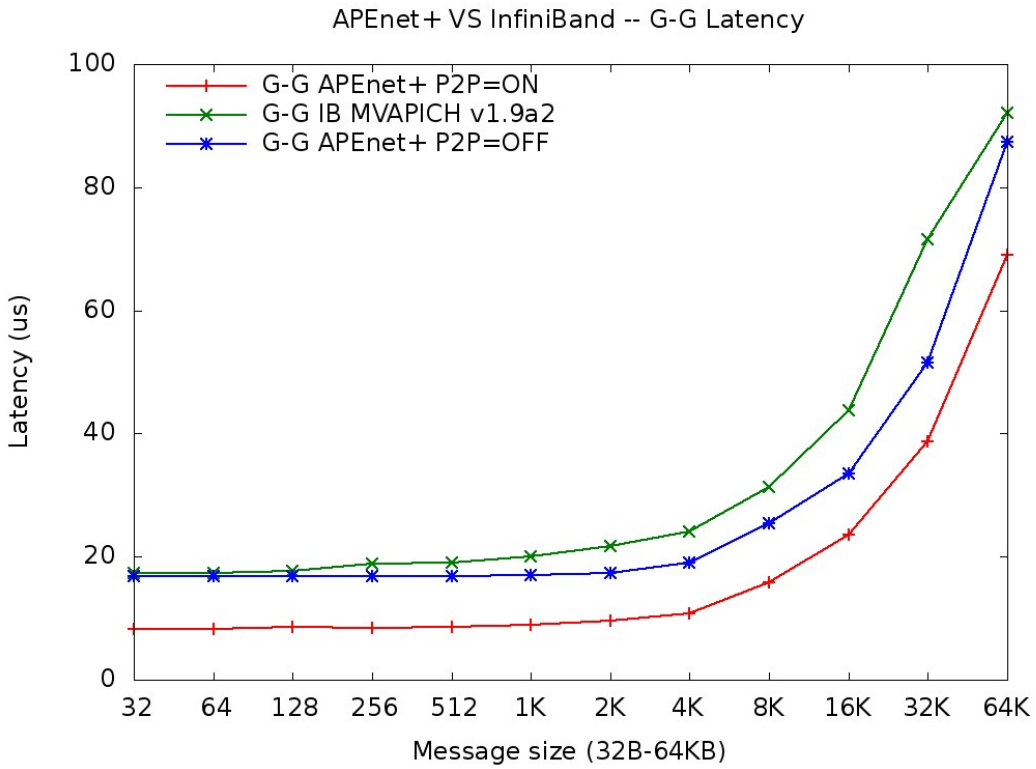
- Data exchange on the PCIe bus
- No bounce buffers on host

So:

- Latency reduction for small msg
- Avoid host cache pollution for large msg
- Free GPU resources, e.g. for GPU-to-GPU memcpy
- More room for comp/comm overlap

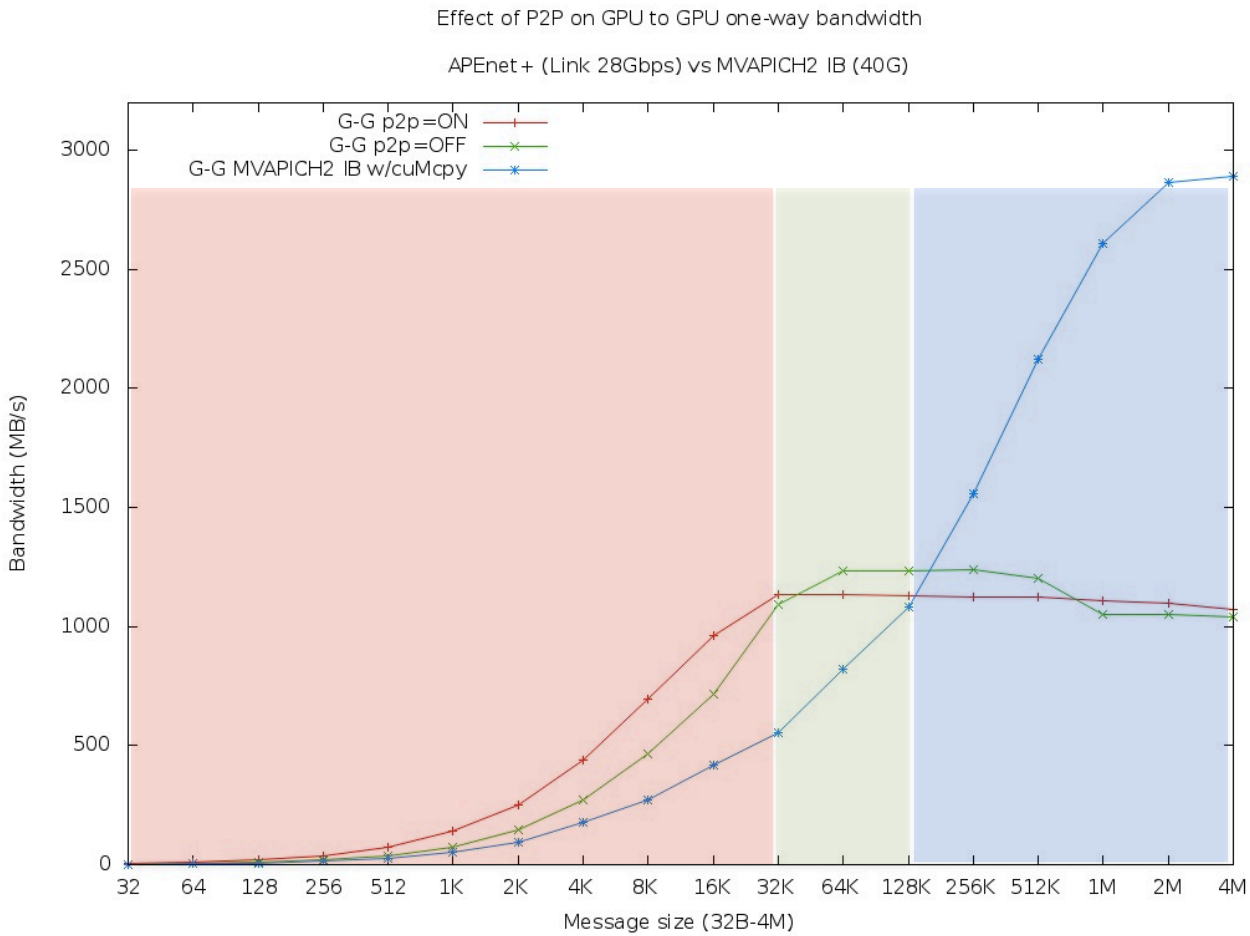
APEnet+ vs InfiniBand – g2g Latency

- 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

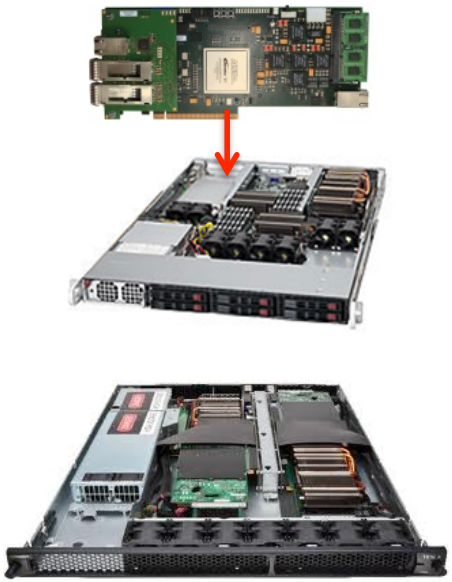


APEnet+ VS rest of the World

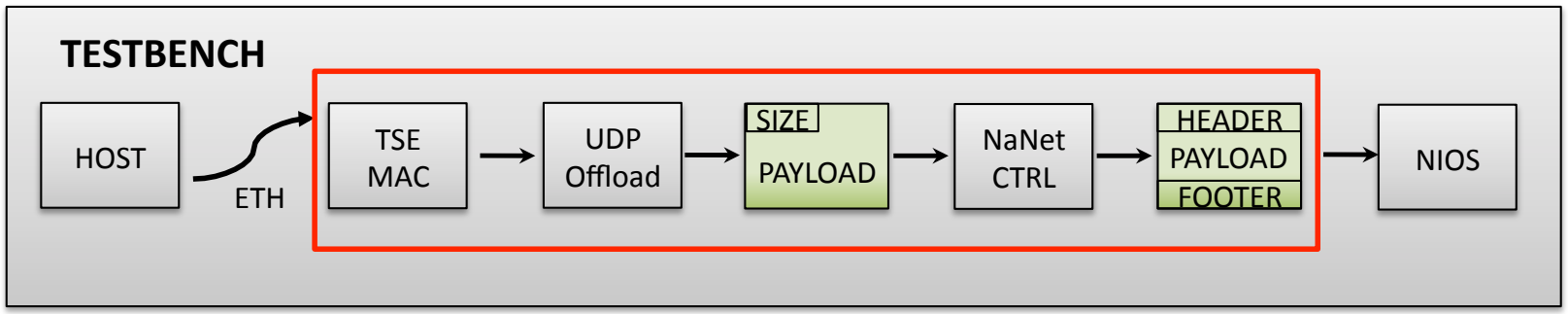
- Below 32 KB P2P wins
- 32 kB – 128 KB P2P shows limits
- over 128 KB Pure bandwidth wins



TEST



- Benchmarking Platform:
  - 1U, two multi core INTEL server equipped with APEnet+ card.
  - 1U, S2075 NVIDIA system packing 4 Fermi-class GPUs (~4 Tflops).
- UDP offload and NaNet CTRL test:
  - The host generates a data stream of 10^5 32bit word (packet size is 4KB).
  - The packets follow the standard path.
  - The Nios II reads the packet and checks whether the data correspond to those sent by the host.
- Integration of UDP offload and NaNet CTRL in Network Interface completed:
  - Debugging stage.
  - Latency measures.



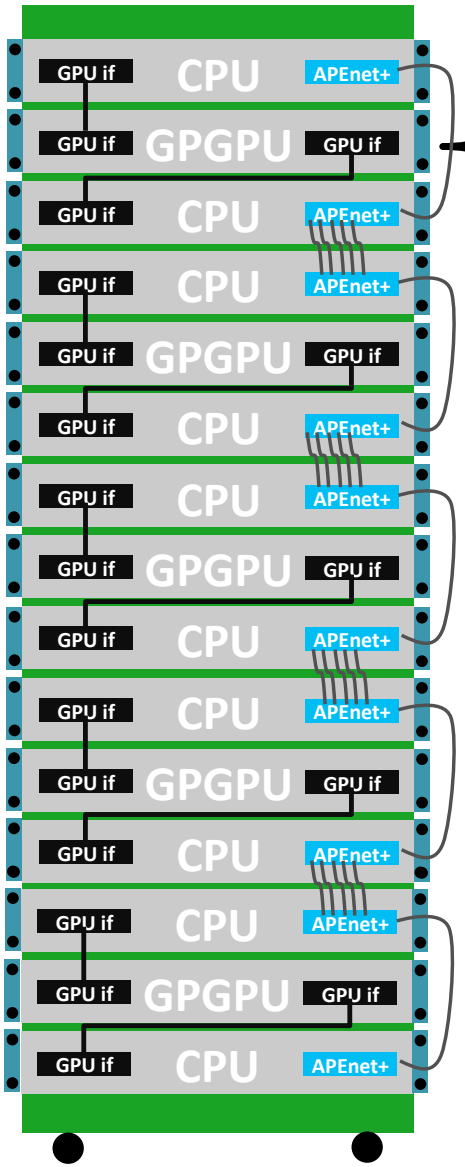
APEnet+ board production and test

- 4 APEnet+ boards produced during 2011.
- 15 APEnet+ boards on 2Q/12 and 10 more to complete the QUonG rack for 4Q/12.
- Preliminary technical test performed by the manufacturer.
- Deeper functional tests:
  - Clock Generators
    - Fixed frequency oscillators measured through a digital oscilloscope.
    - Programmable clock (si570) firmware have been validated.
  - JTAG Chain
    - Stratix IV and MAX2 (EPM2210). 64MB Flash memory. Master controller EPM240 CPLD.
    - Windows OK, complete functionality on Linux obtained by-passing the EPM240 firmware.
  - PCIe
    - Altera Hard IP + PLDA IP.
    - PLDA test-bench adapted and implemented. Successfull.
  - Memory
    - SODIMM DDR3. FPGA acts as memory controller.
    - NIOS + Qsys environment (read and write). Still in progress.
  - Ethernet
    - 2 Ethernet RJ45 connectors (1 main board + 1 daughter board)
    - NIOS + Qsys environment. Still in progress.
  - Remote Links
    - 6 links (4 main board + 2 daughter board)
    - Tranceiver Toolkit by Altera (bit error rate with random pattern) to find the best parameters (400MHz / 32GB main board – 350MHz / 28GB daughter board)

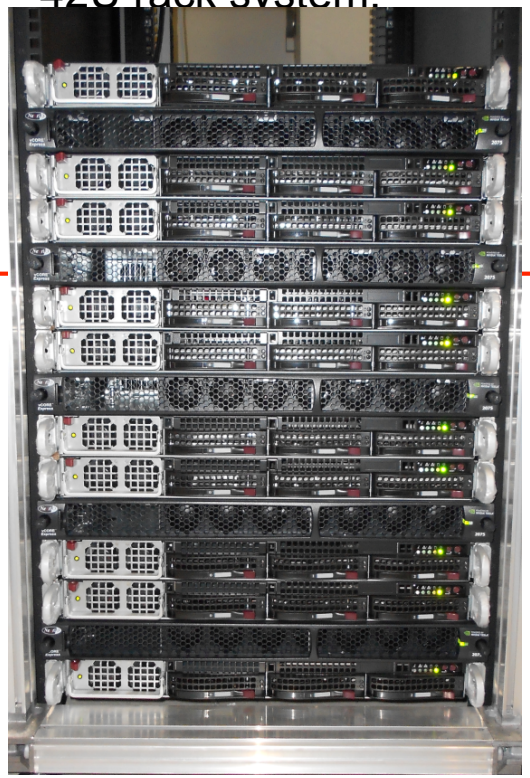
Benchmarking Platform

- 3 slightly different servers
  - SuperMicro motherboards.
  - CentOS 5.6/5.7/5.8 x86_64.
  - Dual Xeon 56xx.
  - 12GB – 24GB DDR3 memory.
  - Nvidia C2050/M2070 on X16 Gen2 slots.
- Preliminary benchmarks:
  - Coded with APEnet RDMA API.
  - CUDA 4.1.
  - One-way point to point test involving two nodes.
    - Receiver node tasks:
      - Allocates a buffer on either host or GPU memory.
      - Registers it for RDMA.
      - Sends its address to the transmitter node.
      - Starts a loop waiting for N buffer received events.
      - Ends by sending back an acknowledgement packet.
    - Transmitter node tasks:
      - Waits for an initialization packet containing the receiver node buffer (virtual) memory address
      - Writes that buffer N times in a loop with RDMA PUT
      - Waits for a final ACK packet.

QUonG status and next future

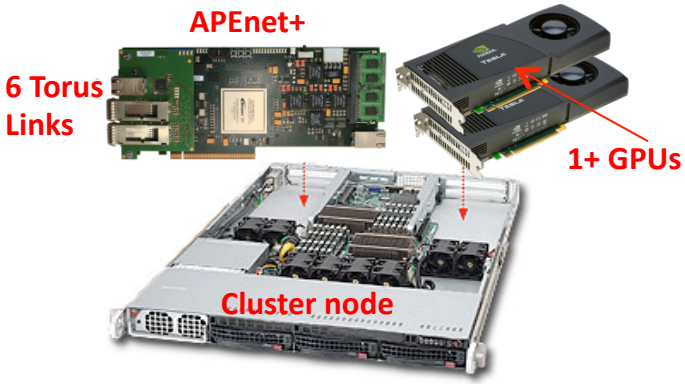


- QUonG elementary mechanical assembly:
  - multi-core INTEL (packed in 2 1U rackable system)
  - S2090 FERMI GPU system (5 TFlops)
  - 2 APEnet+ board
- 42U rack system:



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,…)



GPU support: P2P

- CUDA 4.0:
  - Uniform address space
  - GPUdirect 2.0 aka P2P among up to 8 GPUs
  - CUDA 4.1: P2P protocol with *alien* devices
- P2P between Nvidia Fermi and APEnet+
  - First non-Nvidia device to support it!!!
  - Joint development with NVidia
  - APEnet+ card acts as a peer
  - APEnet+ I/O on GPU FB memory
- Problems:
  - work around current chipset bugs
  - Exotic PCIe topologies
  - Sandy Bridge Xeon