

# GPU peer-to-peer techniques applied to a cluster interconnect

Davide Rossetti  
on leave from INFN Roma



APE group

# Credits



- APEnet design and development by INFN APE team
- Application development by CNR IAC team
- GPU support developed in collaboration with Massimiliano Fatica et al.
- Partially supported by EU EURETILE project (FP7 FET-ICT-2009.8.1 247846 [euretile.roma1.infn.it](http://euretile.roma1.infn.it))





# In summary

Question: what is GPU peer-to-peer good for ?

LATENCY!

BANDWIDTH.... not yet!

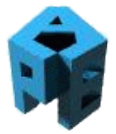
# Outlook



*APE group*

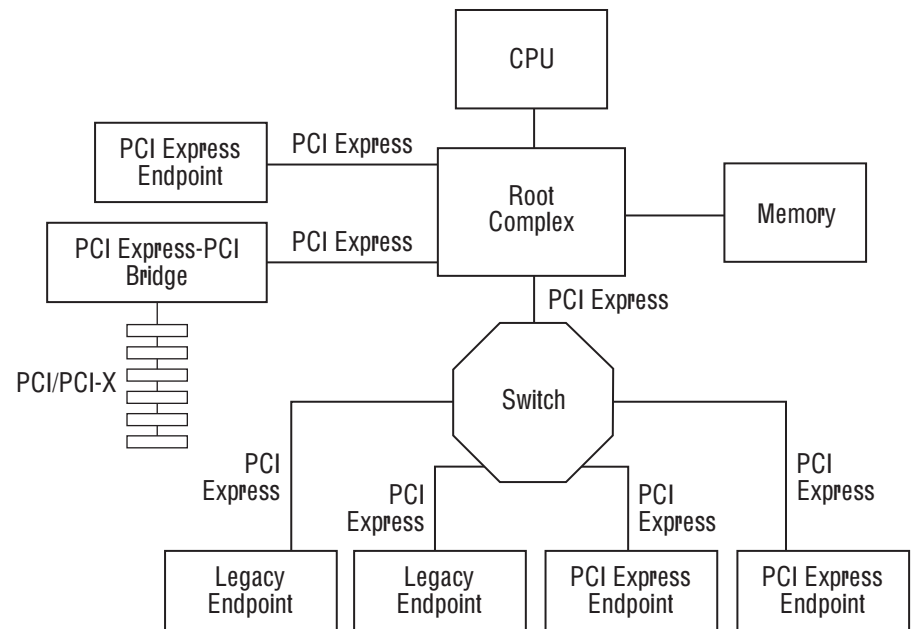
- GPU peer-to-peer on NVIDIA GPUs
- APEnet+ interconnect
- Synthetic Benchmarks
- Multi-GPU Applications





# Peer-to-peer transfers on PCIe

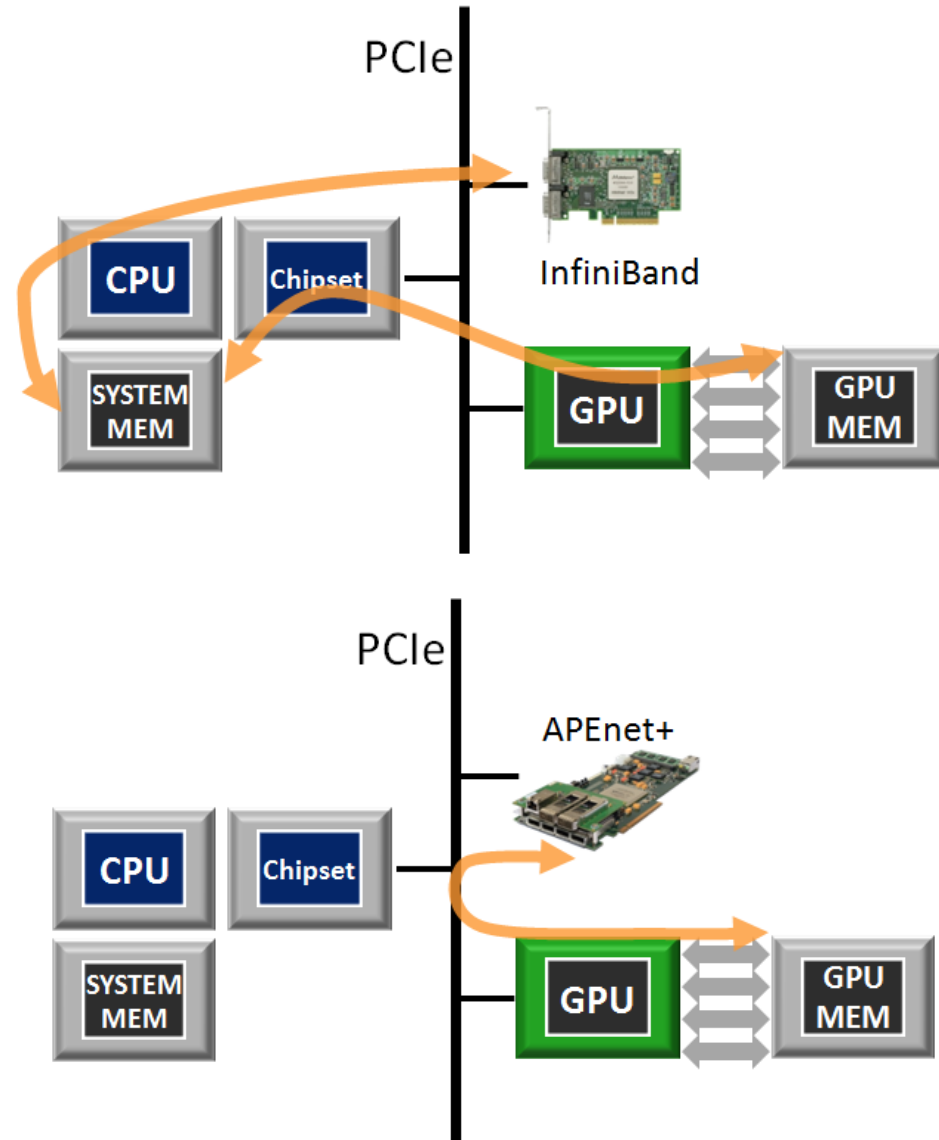
- Technically easy
- Mentioned on PCIe spec
- Usually host is src or dest
- PCIe device can generate transactions among each-other
- More popular on embedded/SoC





# GPU P2P: rationale

- Staging to host memory:
  - 7-10us penalty for cudaMemcpy (mostly syncs)
  - < 1us NIC HW latency
  - SW complexity
    - Pipelined algorithm
    - MVAPICH2, OpenMPI
- Direct path:
  - GPU-GPU path
    - Single fat-node workloads
  - GPU-3<sup>rd</sup> party dev path
    - HPC scaling
    - Other applications: HEP, Astronomy



# P2P: caveats



APE group

- PCIe topology
  - Balanced flow thru branches
- IOH/ICH
  - Crippled BW (by design :)
  - Chipset bugs
  - Bus crossings (SB QPI)
  - Switch (PLX, ...)

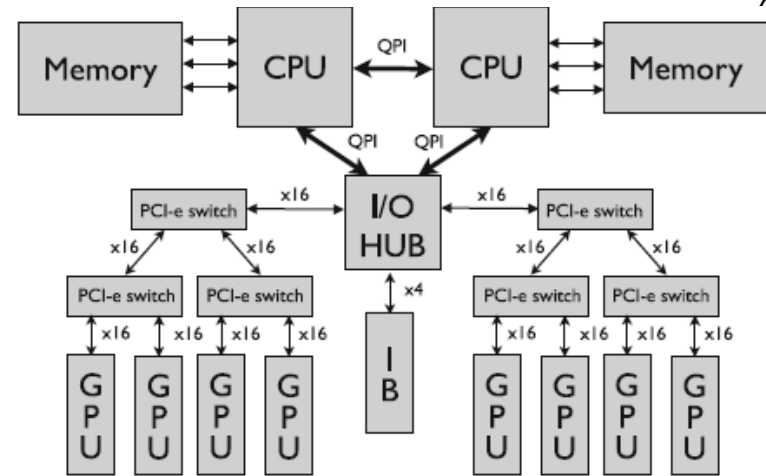
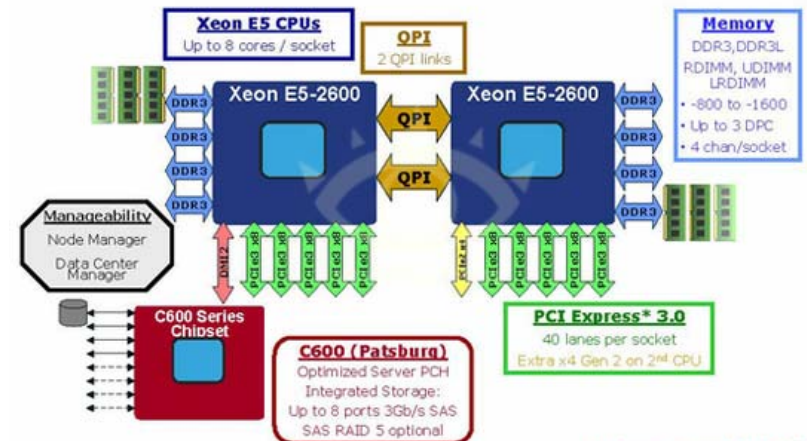


Fig. 6. PCIe topology for the system with 8 GPUs.

Intel® Xeon® Processor E5-2600 Product Family Architecture, Power Efficiency, and Performance

## Romley EP Platforms



Under Embargo Until March 6, 2012 9am PST





# P2P: NVIDIA GPUs

- NVP2P *GPU Direct v2*
  - For GPU-GPU
  - Proprietary (NDA)
  - *Custom* protocol  
no split trans!
  - Chip dependent (GF100, GK104, GK110)
- BAR1 *GPU Direct RDMA*
  - For 3<sup>rd</sup> party devices
  - Public support
  - Fermi: good WR, bad RD
  - Kepler: good RD/WR

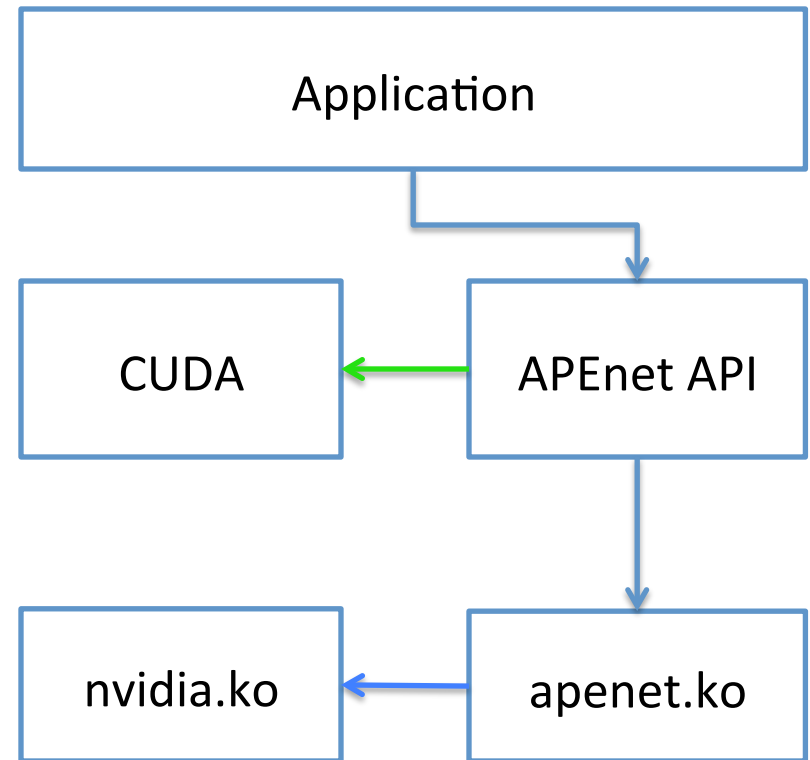
	NVP2P	BAR1
3 <sup>rd</sup> party HW complexity	High	Low
Mem map overhead	Low	High
GPU Mem avail	All	BIOS related
Read BW	Intermediate	Bad
Chipset bug resiliency	High	Low
K20 BAR1 Gen2*	MLNX ConnectX-3 Gen3 GPU Read	GPU Write
Tylesburg Gen2	1.4 GB/s	3.2 GB/s
Sandy Bridge Gen3	0.8 GB/s	~ 5 GB/s

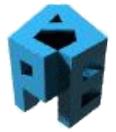
[\*] latest results from NVIDIA/Mellanox/OSU joint developments (GTC 2013)



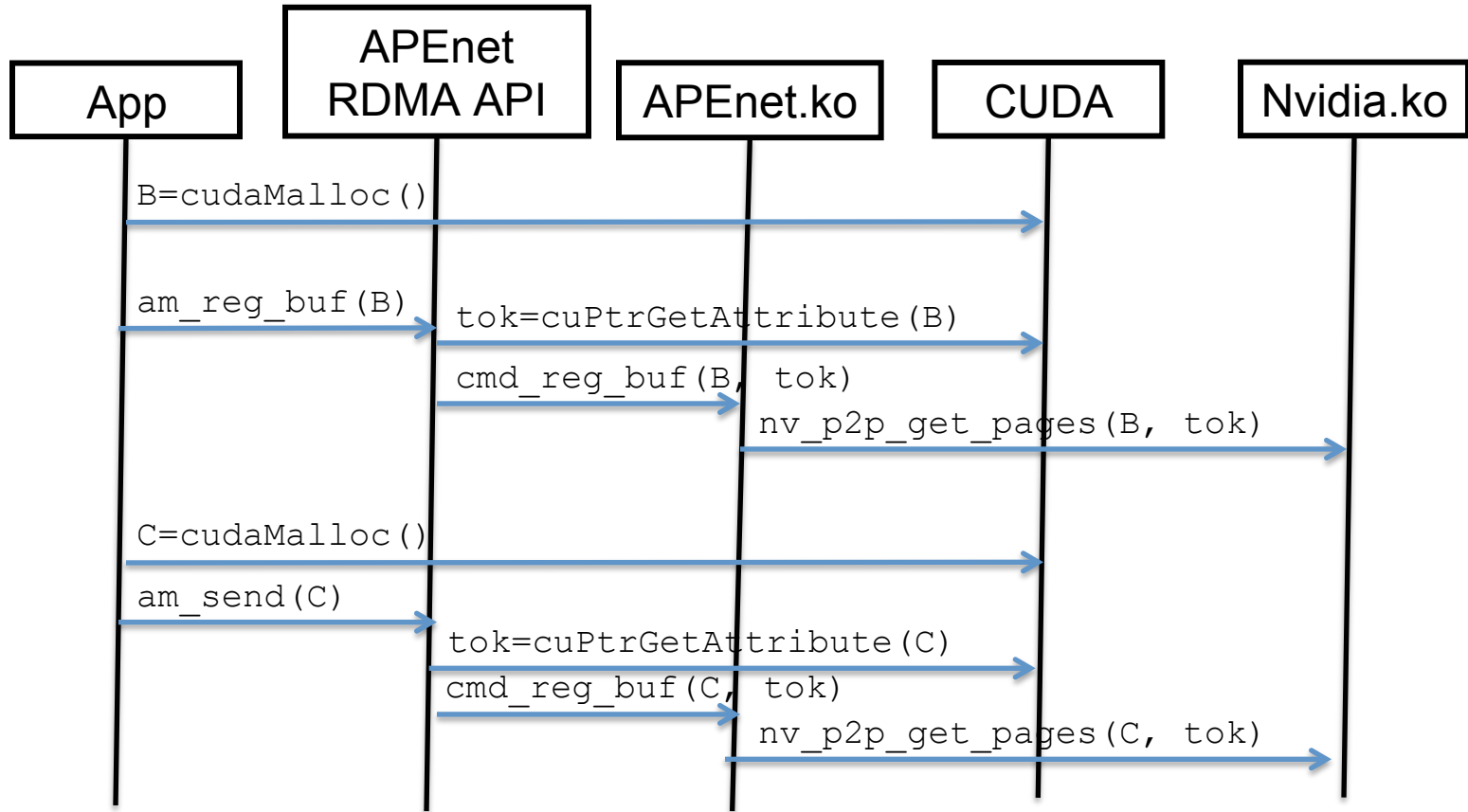
# P2P: SW interface

- BAR/NVP2P choice at run-time
- split user/kernel:
  - `cuPointerGetAttr(TOK)`  
[side effects !!]
  - `nv_p2p_get/put_pages()`
  - `nv_p2p_init_mappings()`





# P2P: SW interface



# Flash adv



*APE group*

Interested in GPU Direct RDMA ?

Contact Duncan Poole <[dpoole@nvidia.com](mailto:dpoole@nvidia.com)>

# APEnet+: architecture

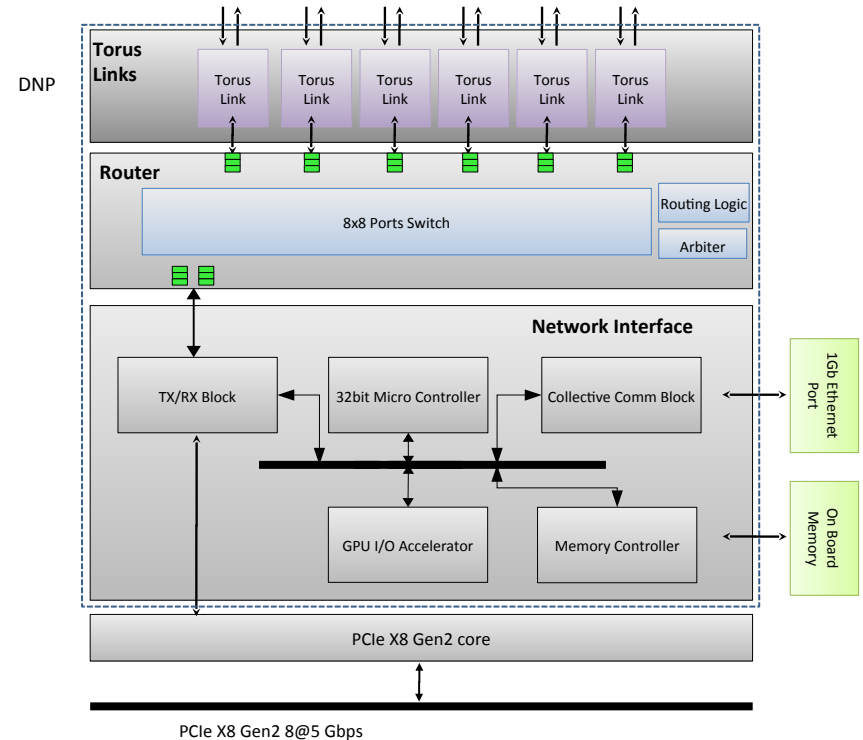
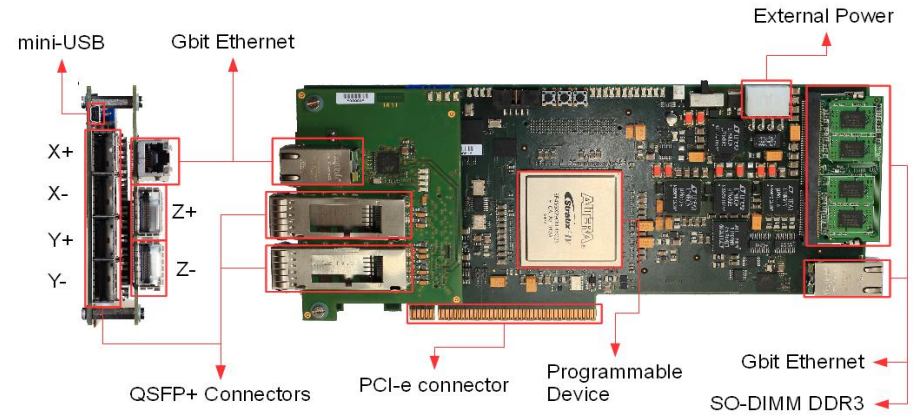


APE group

- APEnet+ card:
  - FPGA based
  - 8-ports switch
  - 6 bidirectional links (max 34 Gbps)
  - PCIe X8 Gen2 in X16 (4+4 GB/s)
  - Network Processor, SoC design
    - RISC 32bit
    - Accelerators

## • Features:

- Zero-copy RDMA host interface
- GPU P2P interface





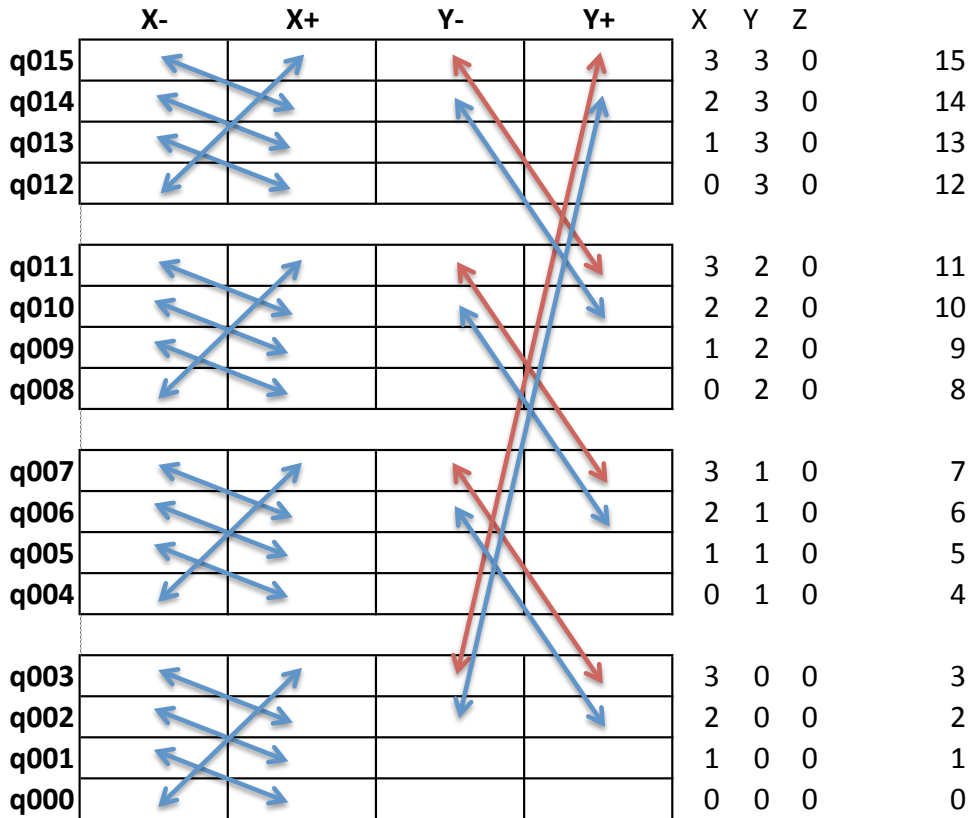


# APEnet+: QUonG

Current system:

16 nodes, 32 2075

4x4x1 topology



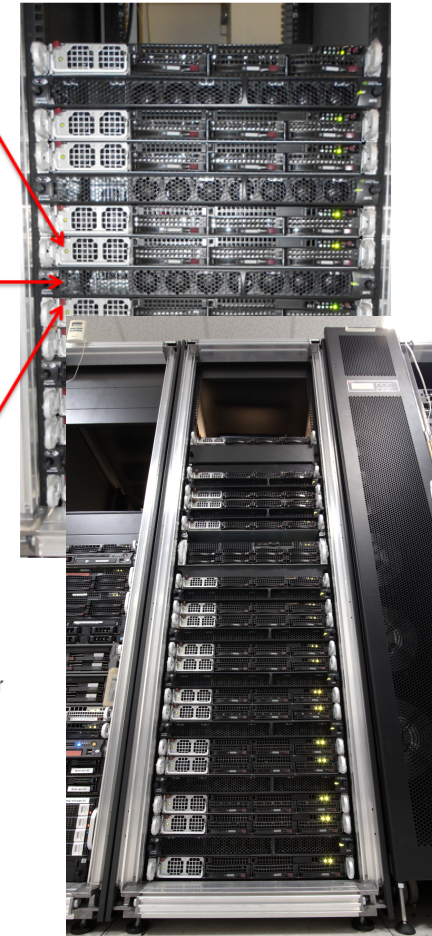
1U, two multi-core INTEL server equipped with APEnet+ card



1U, S2075 NVIDIA system packing 4 Fermi-class GPUs (~4 Tflops)



1U, two multi-core INTEL server equipped with APEnet+ card



# APEnet+: GPU peer-to-peer

NVP2P/BAR1 fit nicely with RDMA model

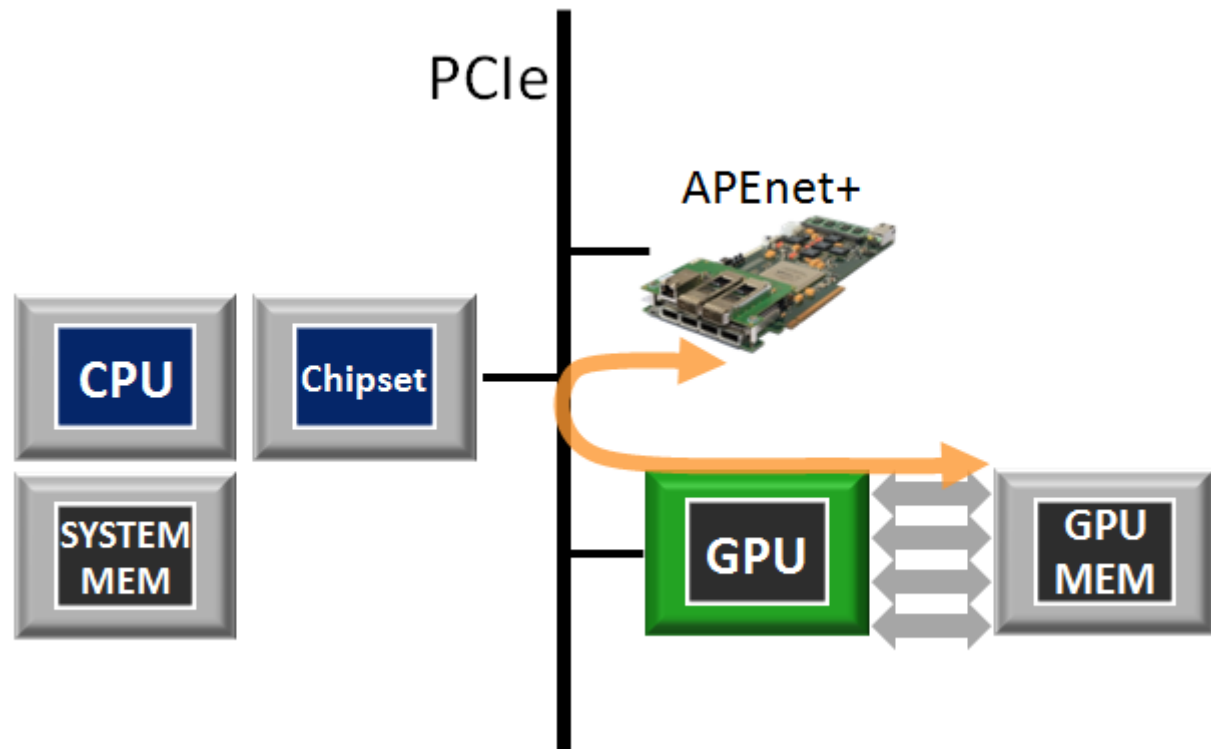
NVP2P WR: +/- easy

NVP2P RD: hard!

BAR1 WR: easy

BAR1 RD: easy

APEnet+ supports both



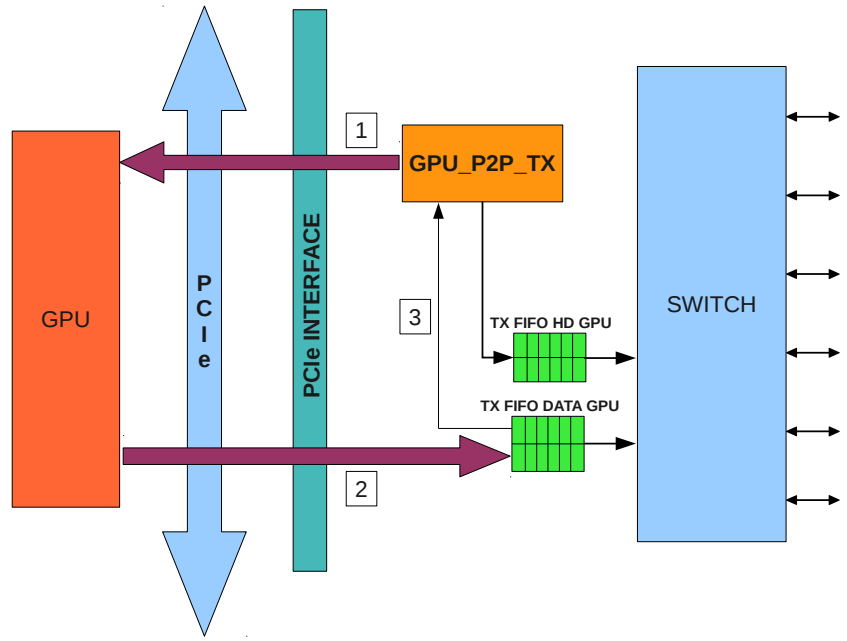


# GPU\_P2P\_TX

## GPU P2P TX flow

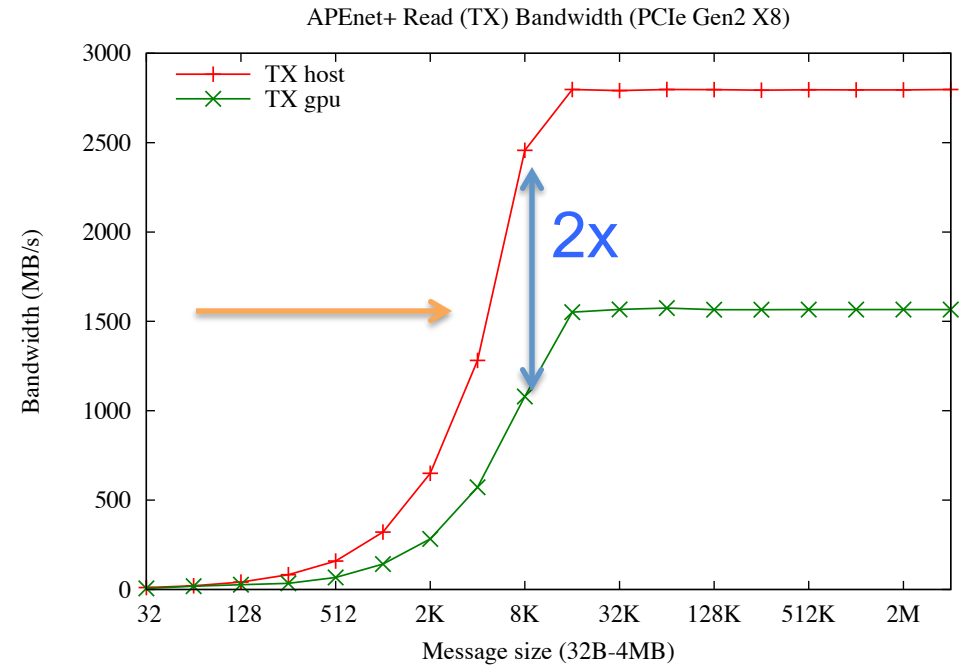
### Basic arch

### HW+SW(V2P)



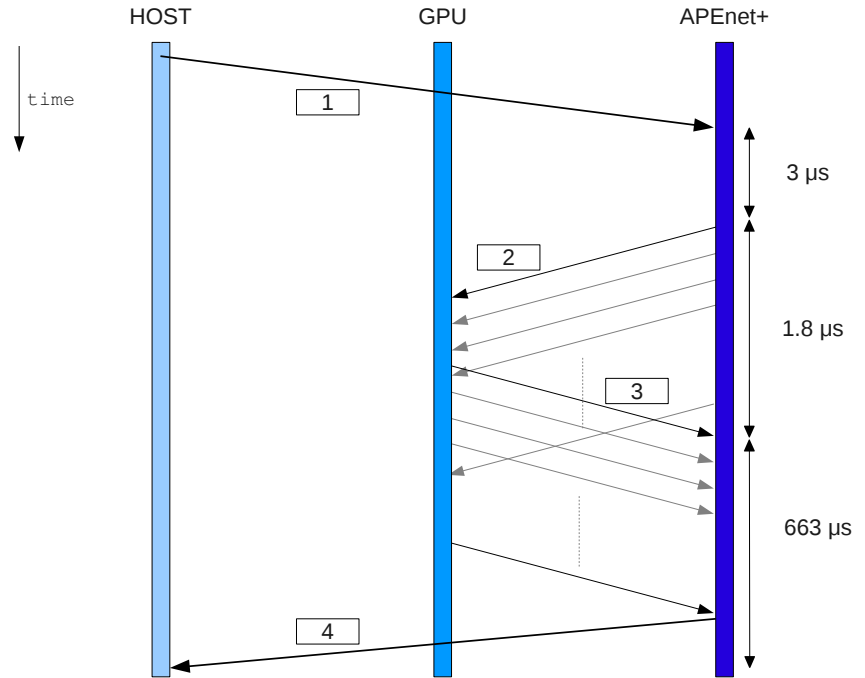
## GPU reading subperfs!

- More overhead
- Lower peak

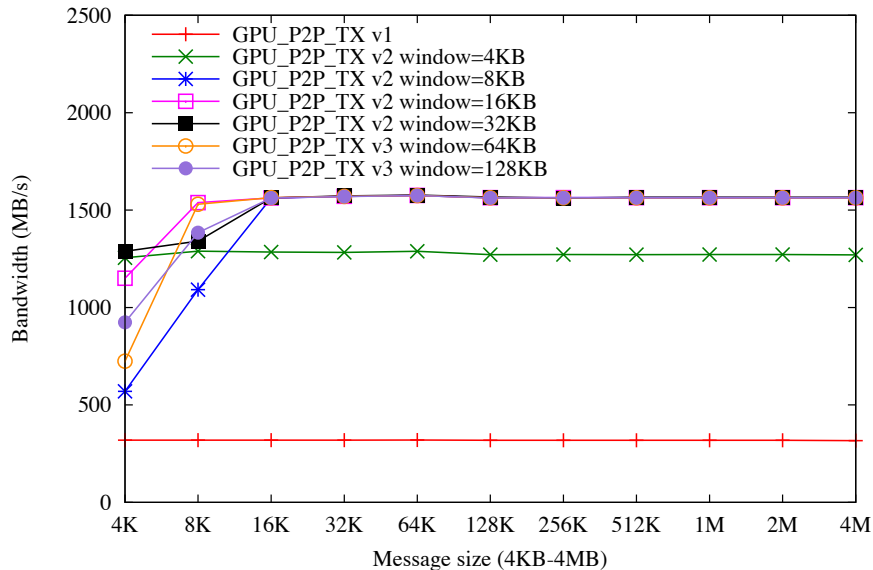


# GPU\_P2P\_TX

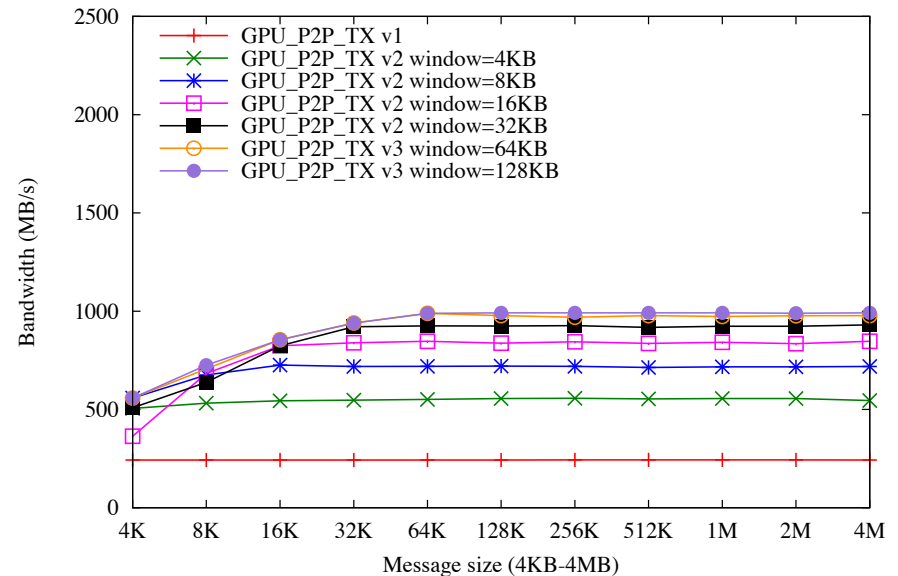
- GPU latency → overhead
- Hide w/ prefetching:
  - V1 nov 2011
  - V2 apr 2012 prefetch
  - V3 lug 2012 new flow-ctrl



GPU Read Bandwidth -- GPU read prefetch effect (PCIe Gen2 X8)



G-G loopback bandwidth -- GPU read prefetch effect (PCIe Gen2 X8)





# Benchmarks: single node BW

1.5-1.6 GB/s cap ?

- GPU memory latency
- APEnet+ DMA HW engine (num. out. read req)
- IOH/ICH

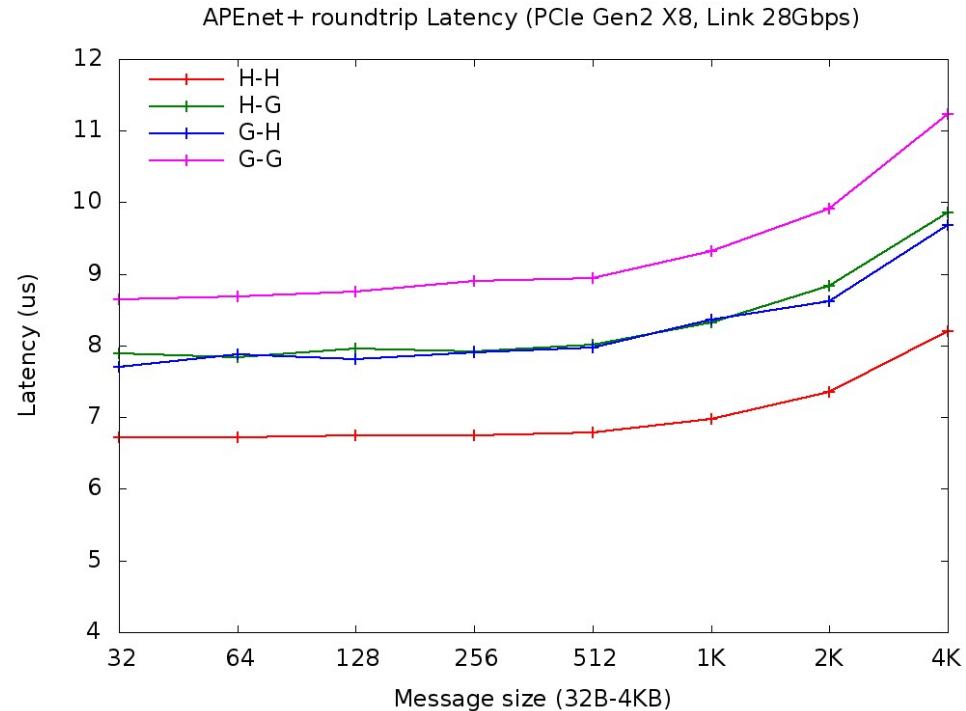
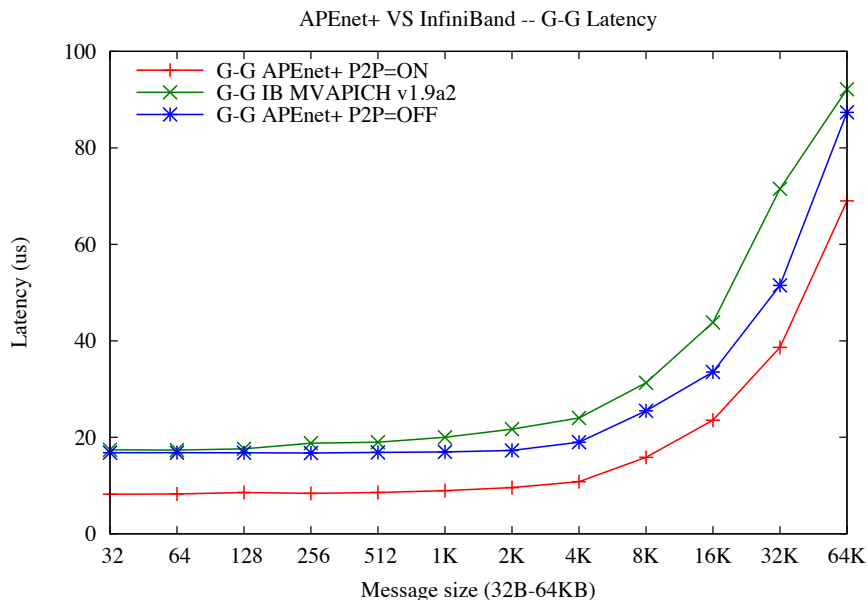
	BW	GPU/method	Platform	Active block
Host mem read	2.7 GB/s		X56xx/Tylesburg	TX
GPU mem read	1.5 GB/s	Fermi NVP2P	X56xx/Tylesburg	GPU_P2P_TX
GPU mem read	150 MB/s	Fermi BAR1	X56xx/Tylesburg	TX
GPU mem read	1.6 GB/s	Kepler NVP2P	I7/PLX	GPU_P2P_TX
GPU mem read	1.6 GB/s	Kepler BAR1	I7/PLX	TX
GPU-GPU loopback	1.3 GB/s	Fermi NVP2P	X56xx/Tylesburg	GPU_P2P_TX +RX
Host-Host loopback	1.5GB/s		X56xx/Tylesburg	RX



ASST

# Synthetic Benchmarks: 2 nodes latency

- GPU adds latency on both sides
- Additive (RX+TX)

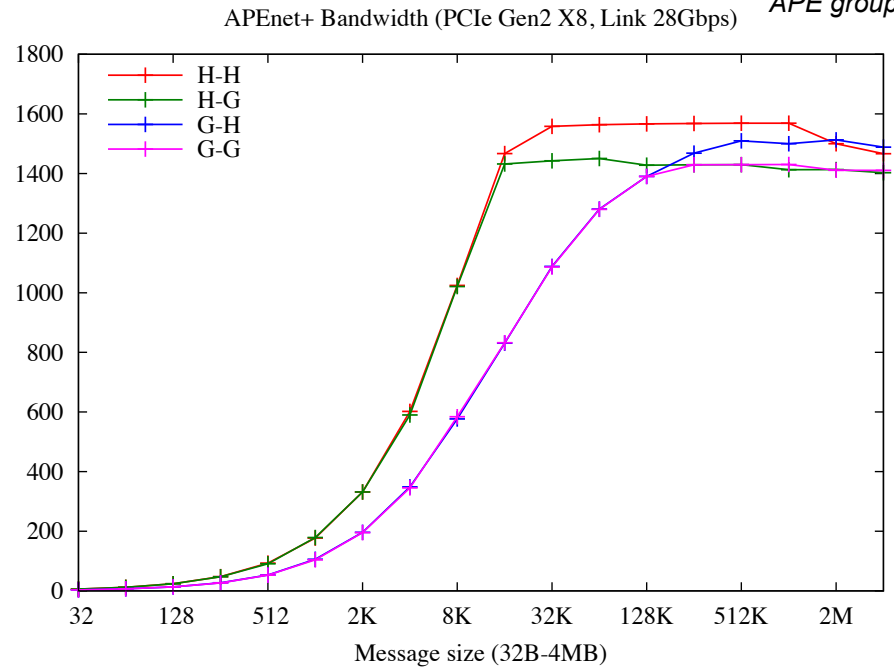
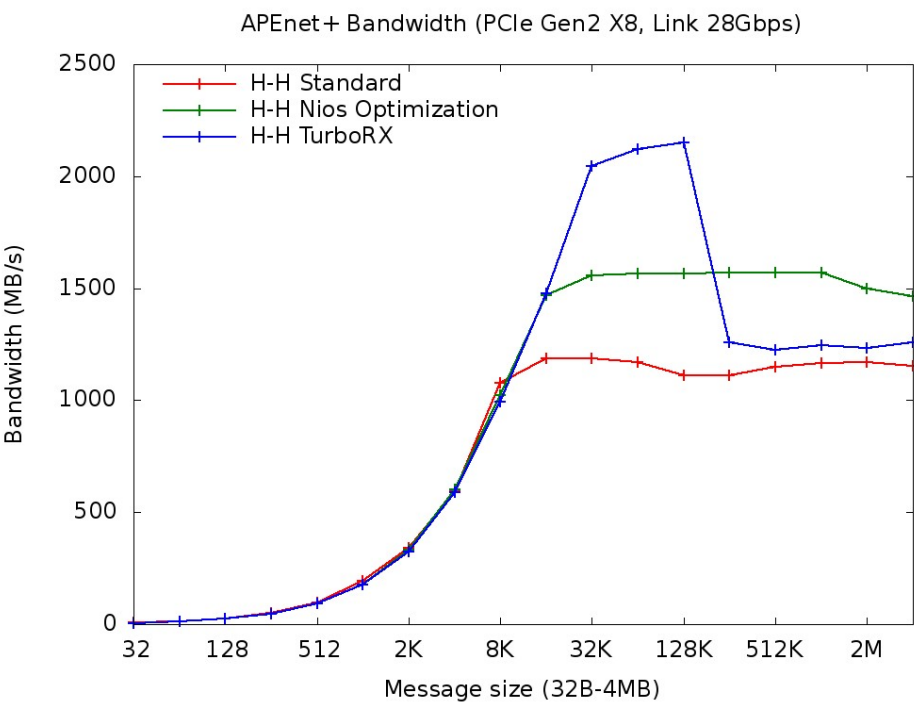


- Good anyway
- 50% lower than staging



# Synthetic Benchmarks: 2 nodes bandwidth

- H-x limited by APEnet+ RX
- G-x limited by GPU RD BW



## Improving RX

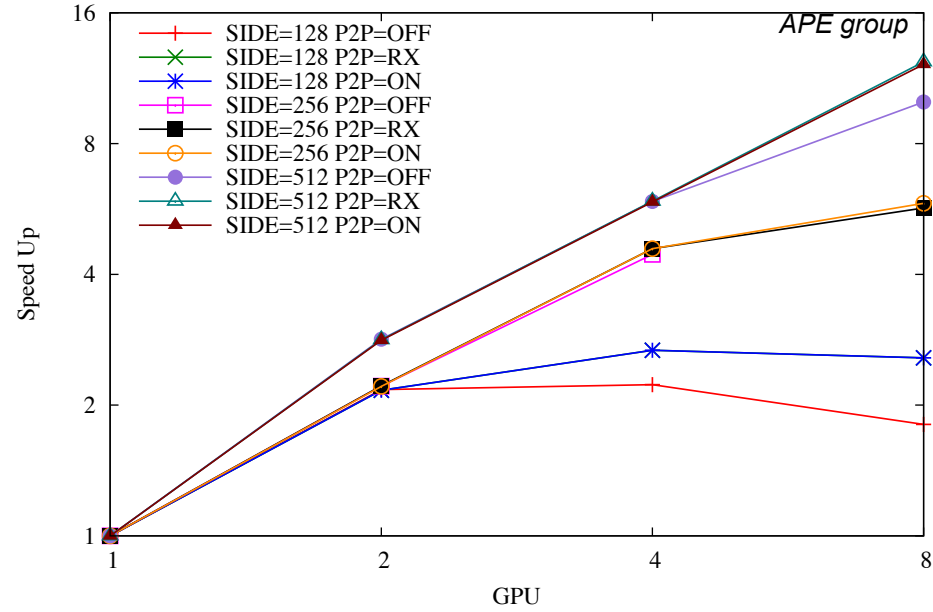
- Original code
- Optimized code
- HW accelerated



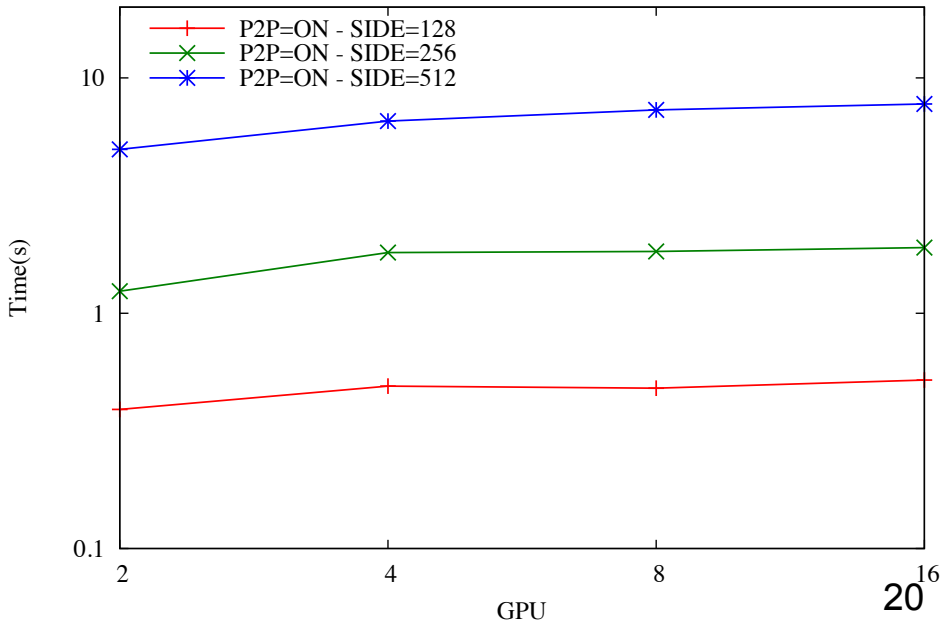
# Multi-GPU Apps: HSG

- Heisenberg SG
  - 3D spins
  - 3D lattice
  - +/-1 random couplings
- Full bag of tricks
  - Even-odd parallel overrelax
  - Domain decompose on Z w/ halo cells
  - Comp-comm overlap

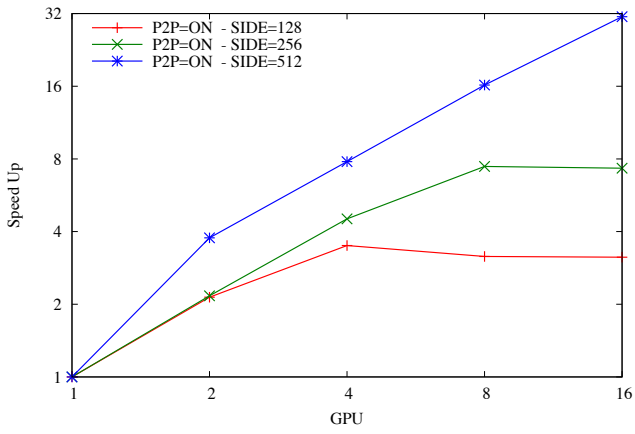
HSG scaling (PCIe Gen2 X8, Link 20Gbps)



HSG scaling - memcopy and network communications (PCIe Gen2 X8, Link 20Gbps)



HSG scaling (PCIe Gen2 X8, Link 20Gbps)



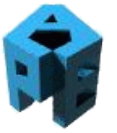




# Game Over

Very open to collaborations...

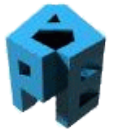
Thanks !!!



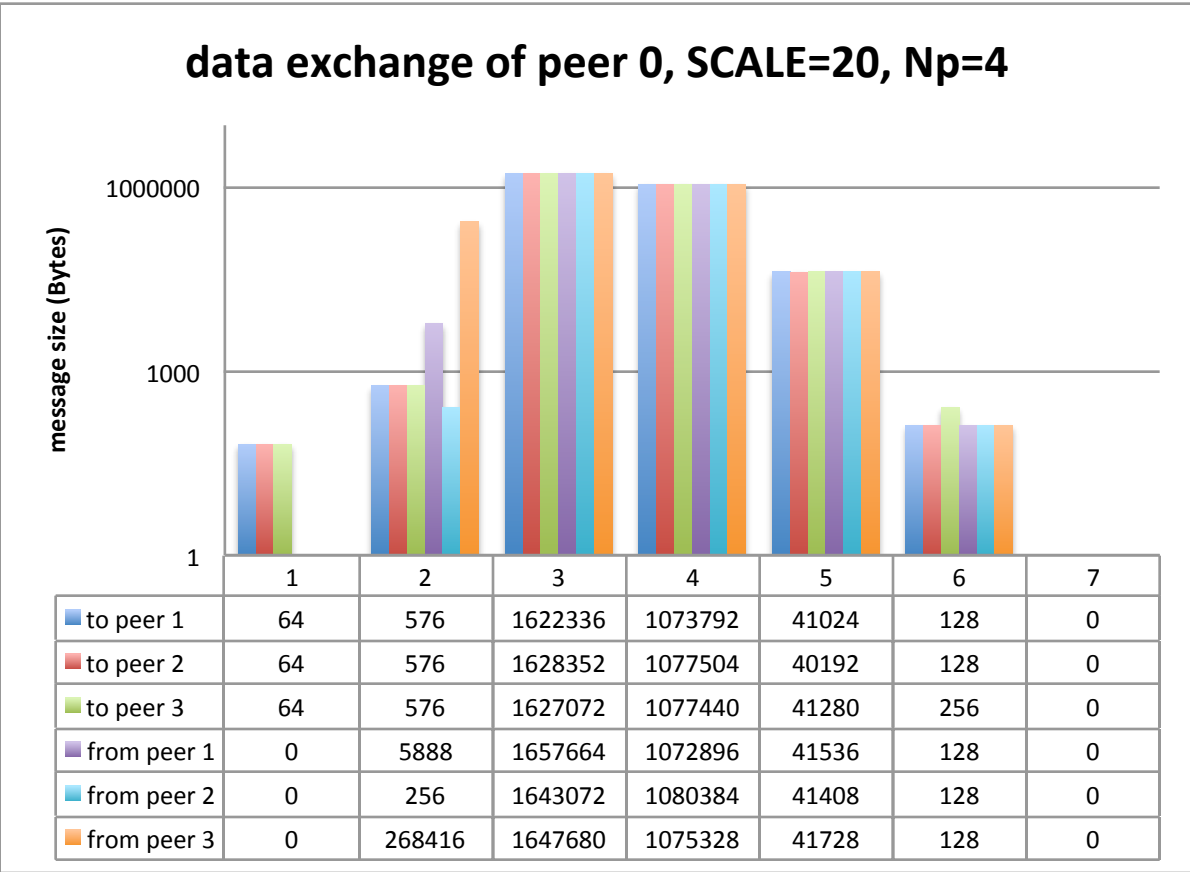
*APE group*

# Backup slides

# Multi-GPU Applications: GRAPH



*APE group*





# P2P among GPUs

- thru PLX: S2070, S2075



P2P great for HPC  
anything beyond HPC ?



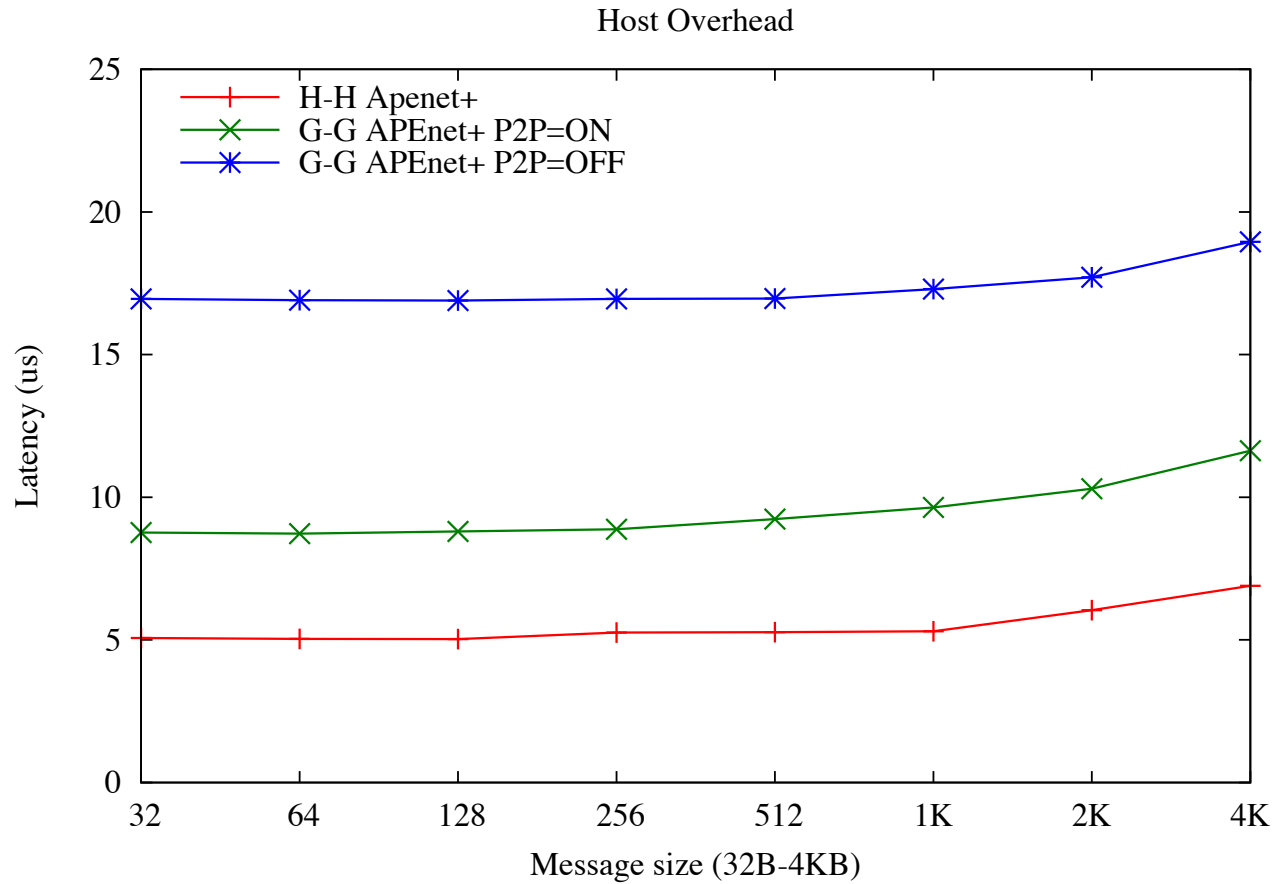
- Overlapping computation and communication
  - HSG 50% better
  - CUDA streams (boundary/bulk)
  - CUDA Async primitives
  - Black magic: `cudaStreamQuery*`
- Multiple GPUs per node with peer-to-peer (P2P)
  - HSG 50% better with 2 GPUs per node
  - Need proper PCIe architecture: GPUs connected to the same PCIe root complex

\* M.Bernaschi, M.Bisson, D.Rossetti “Benchmarking of communication techniques for GPUs” to appear in Journal of Parallel and Distributed Computing

# Host overhead



APE group

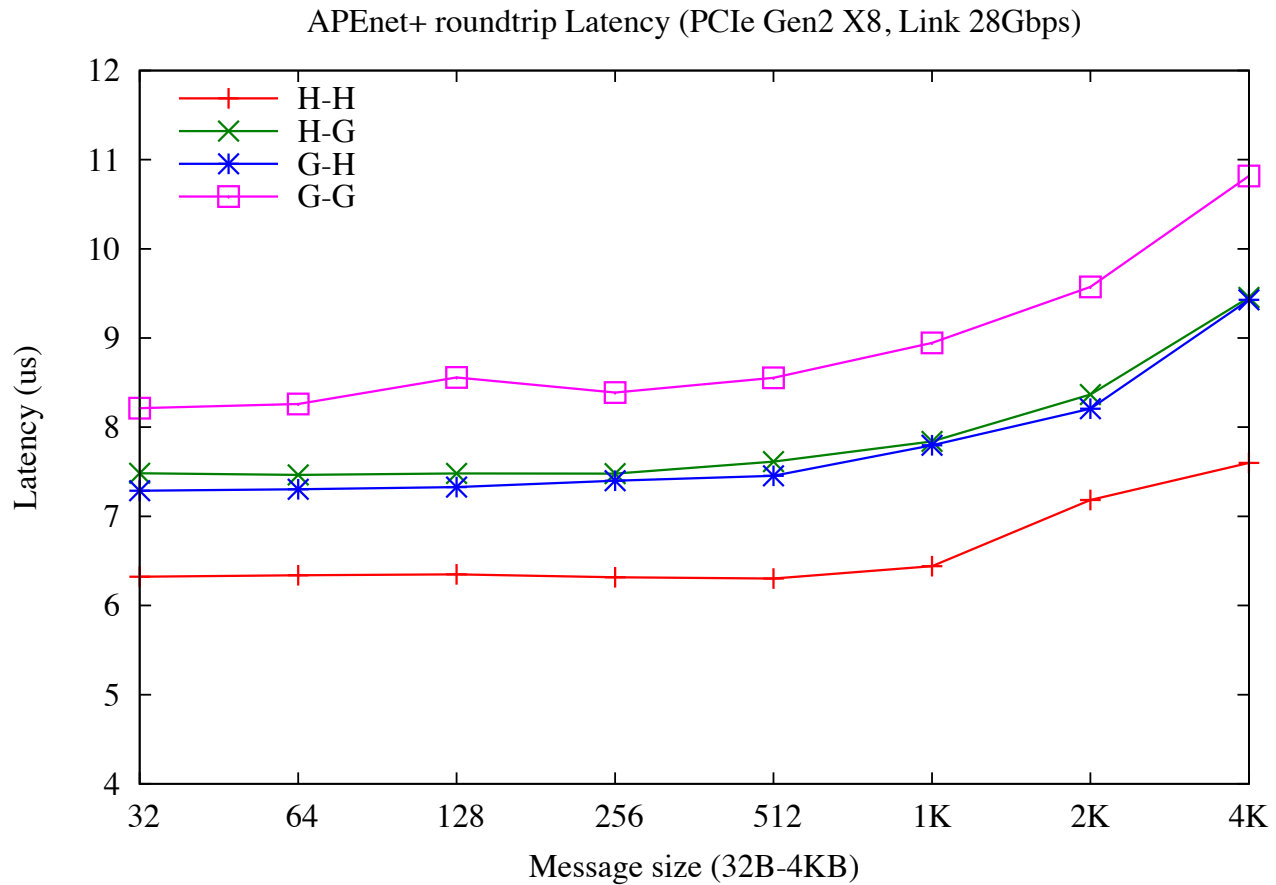




# A40



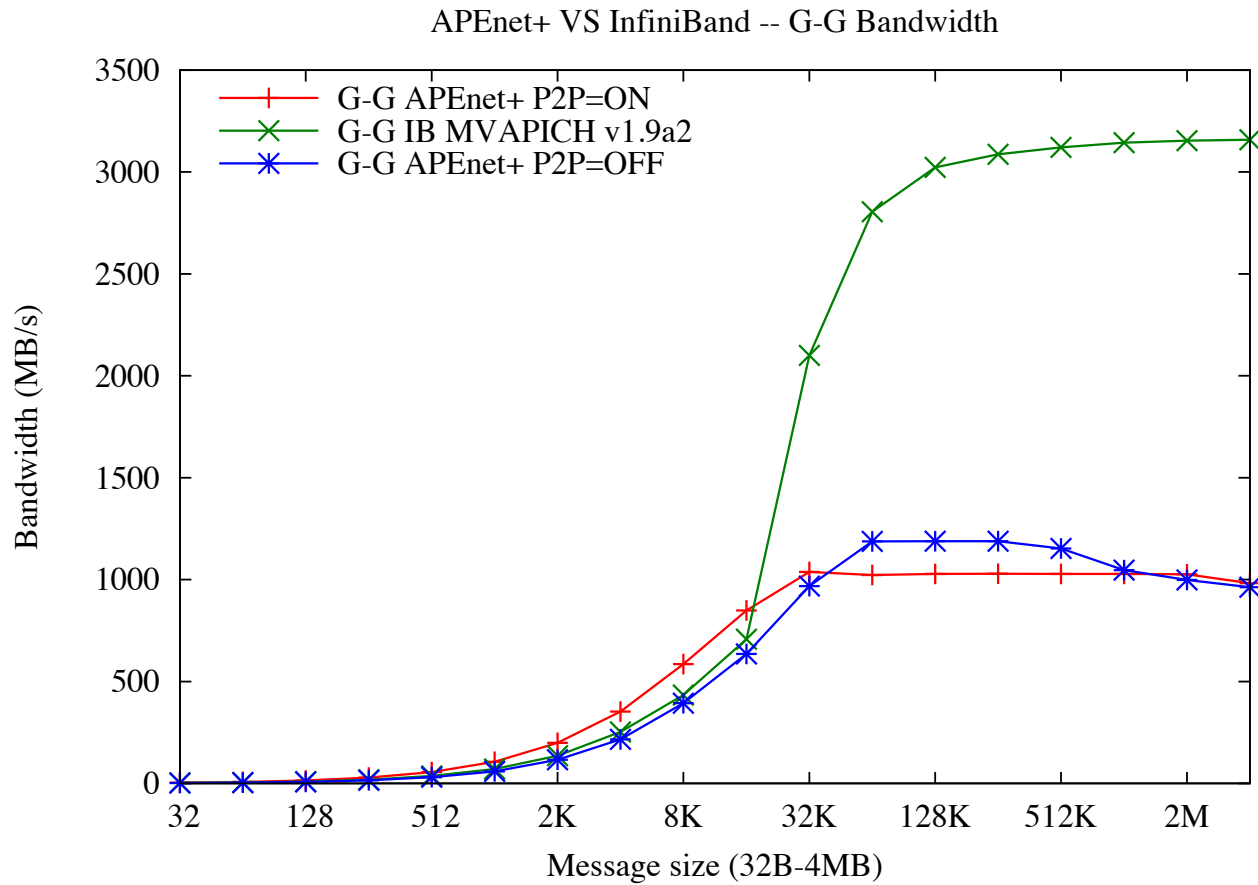
APE group



# A40



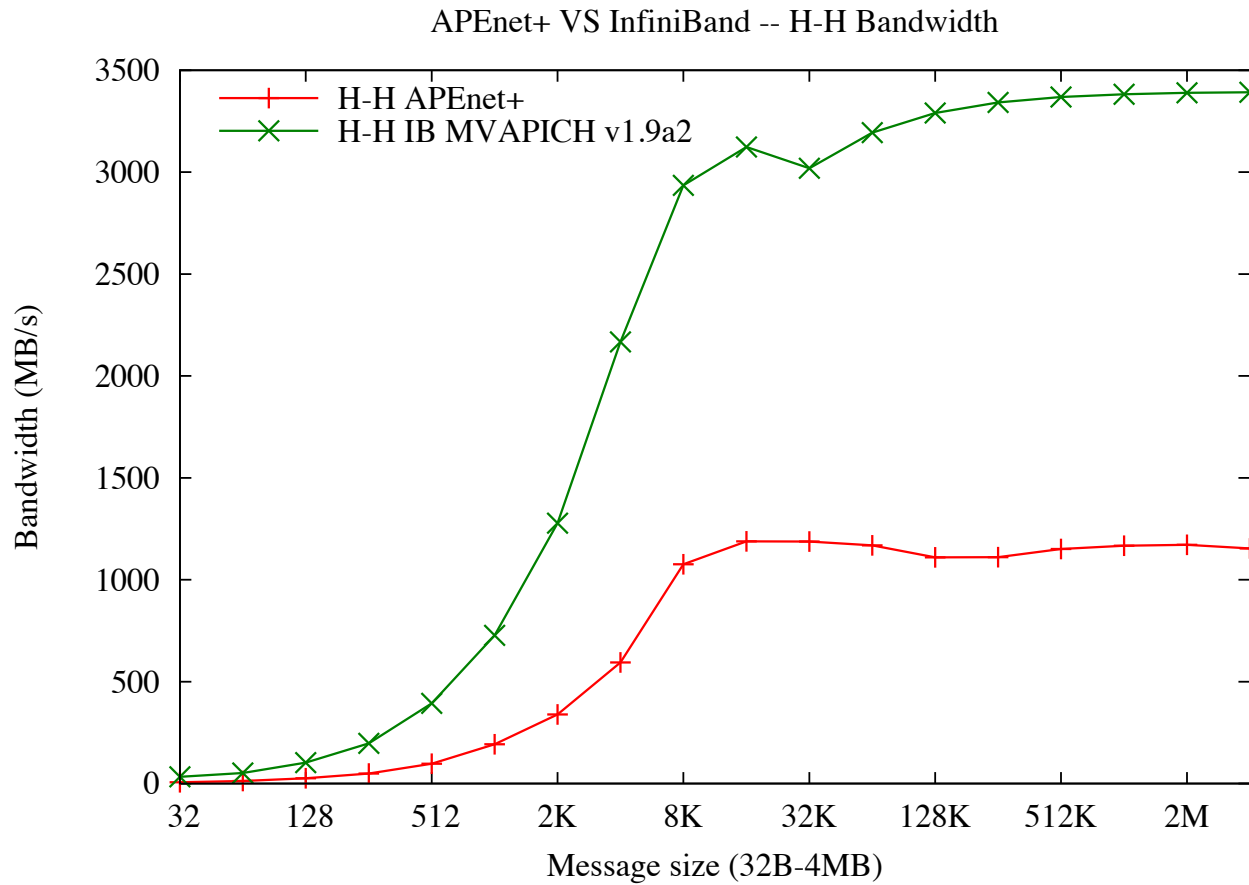
APE group



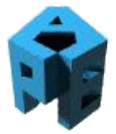
# A40



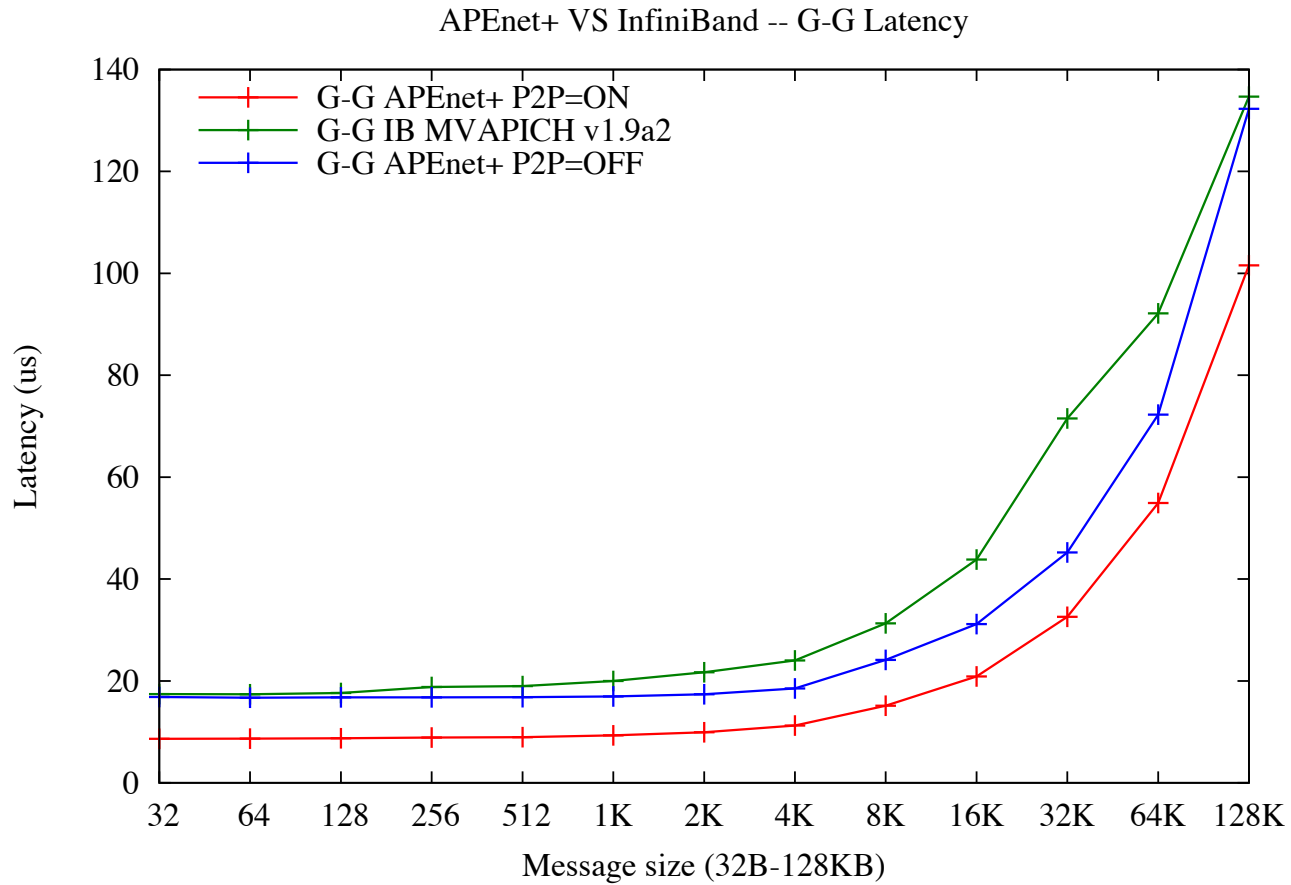
APE group



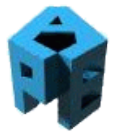
# A56



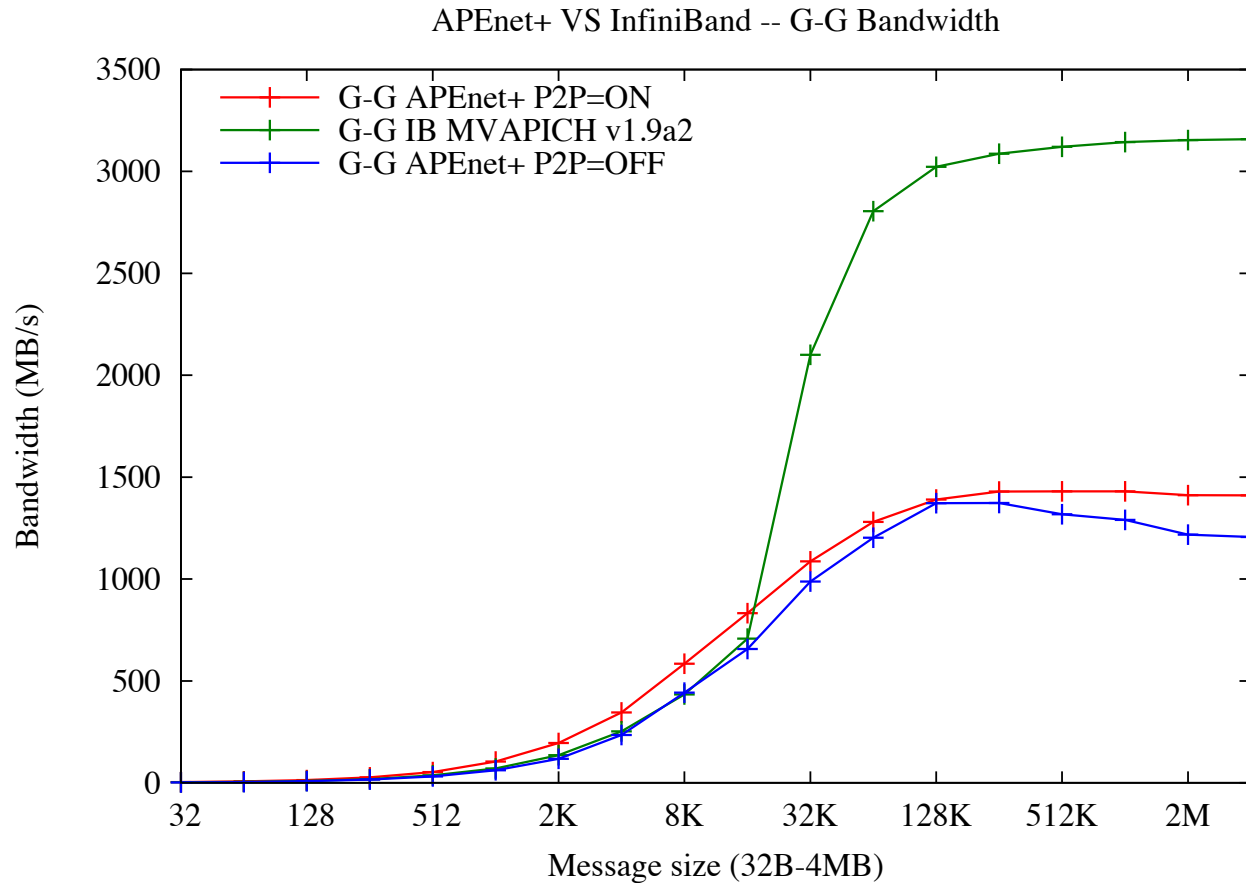
APE group



# A56



APE group



# A56



APE group

