RUPRECHT-KARLS-
UNIVERSITÄT
HEIDELBERG

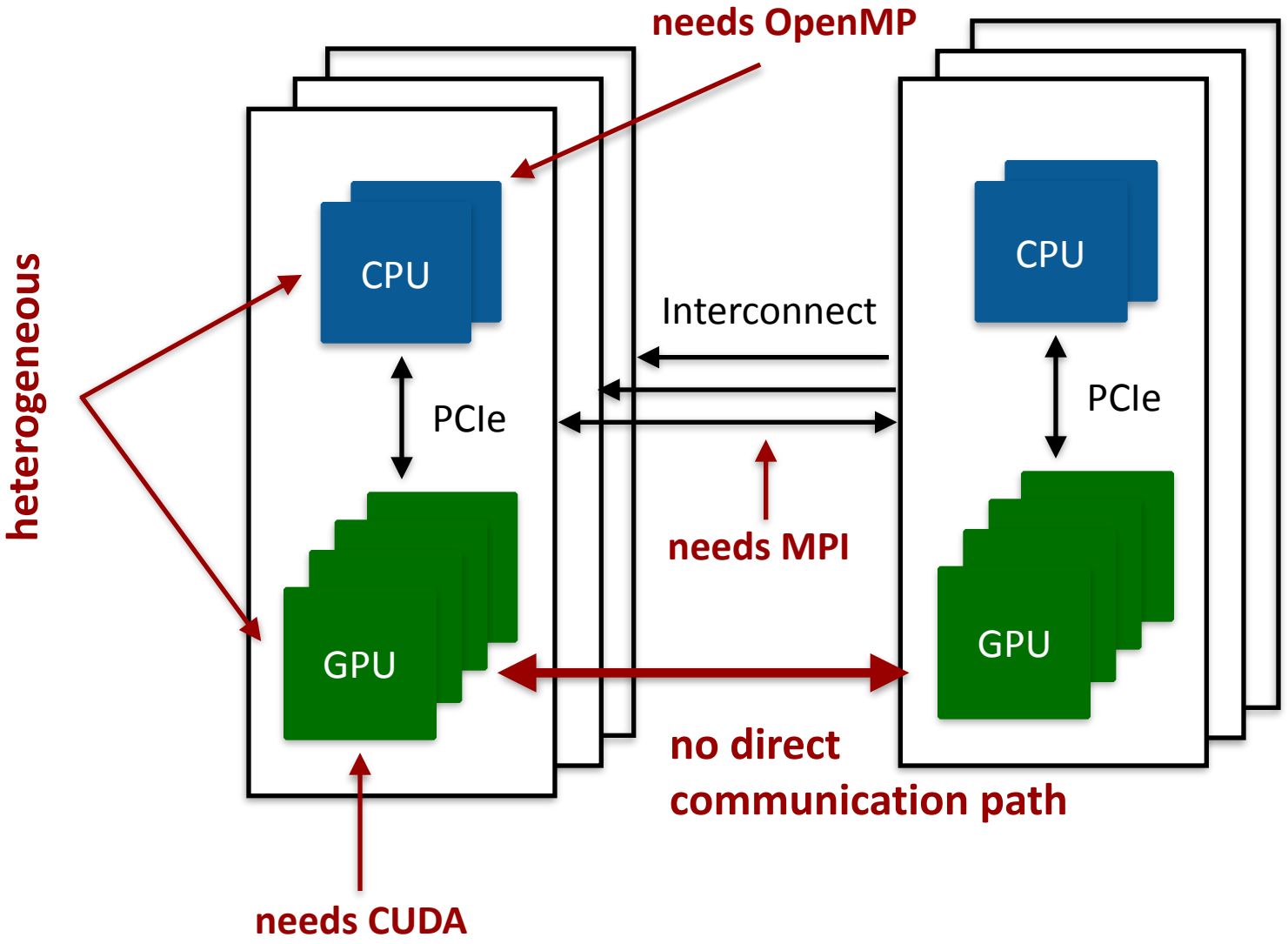
Analyzing Put/Get APIs for Thread-Collaborative Processors

Benjamin Klenk, Lena Oden, Holger Fröning
Institute for Computer Engineering
University of Heidelberg, Germany

HUCAA Workshop, ICPP 2014, Minneapolis, MN, USA

September 12, 2014

Today's systems



Motivation

Complexity

- Programming effort becomes huge
- Where to start with optimizations?

Power

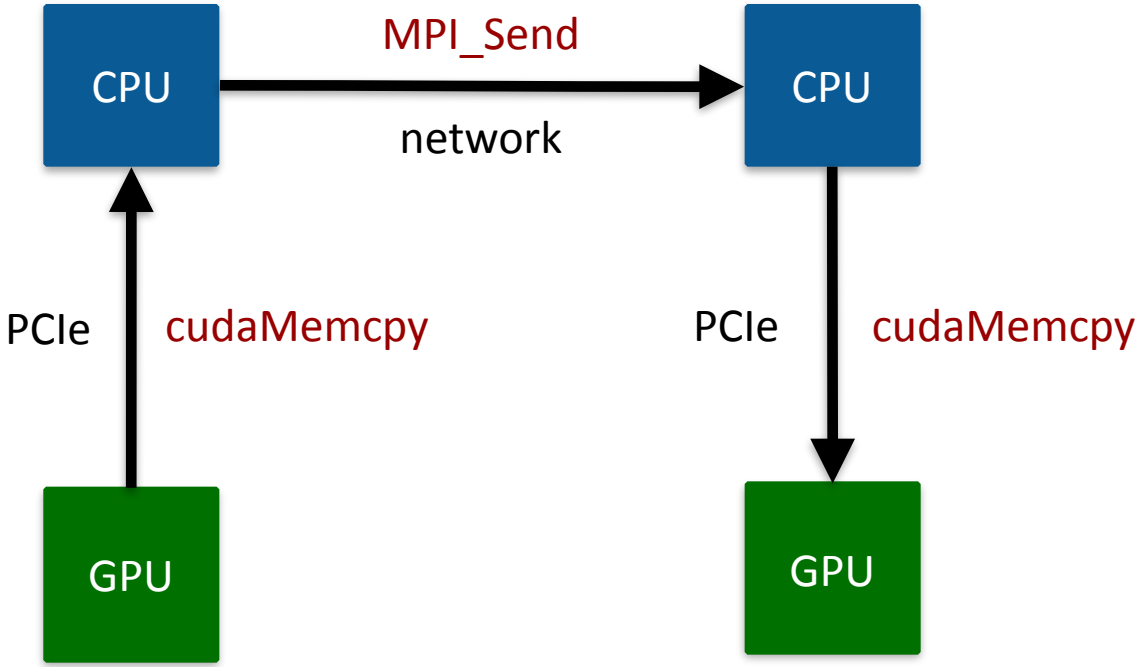
- End of Dennard Scaling
- CPU handles entire communication

Performance

- Additional data copies via PCIe
- Adds latency, especially for small messages

State of the Art Communication

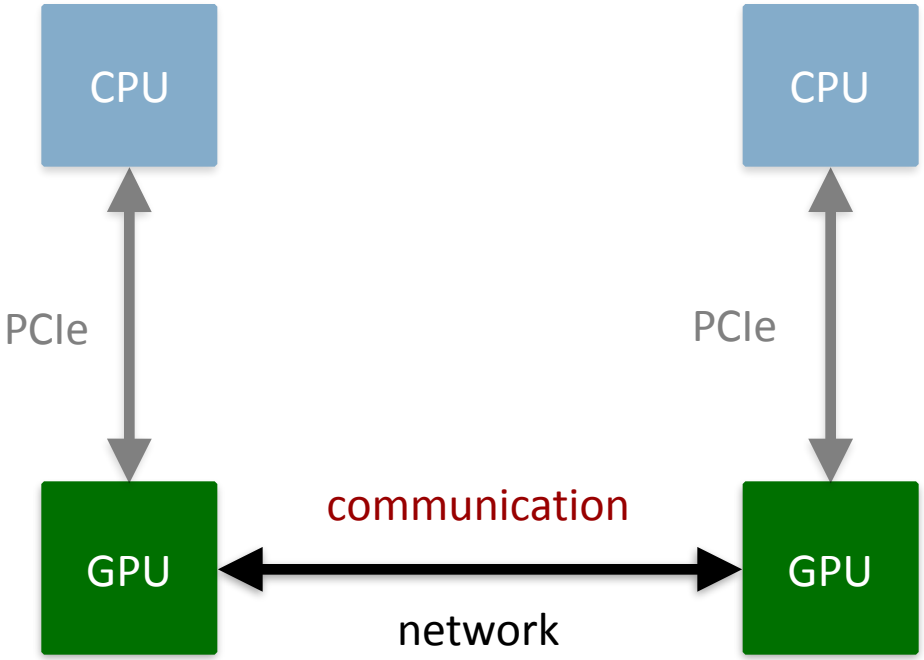
- There is no MPI on the GPU / within CUDA kernels
- Context switches for communication!



- Optimizations:
  - GPUDirect RDMA (in theory)
  - Host memory staging
  - Pipelining

Why bother the CPU?

- Free the CPU → less power consumption / perform other work
- Avoid context switches → reduce latency and simplifies programming!
- GPU controls entire communication!



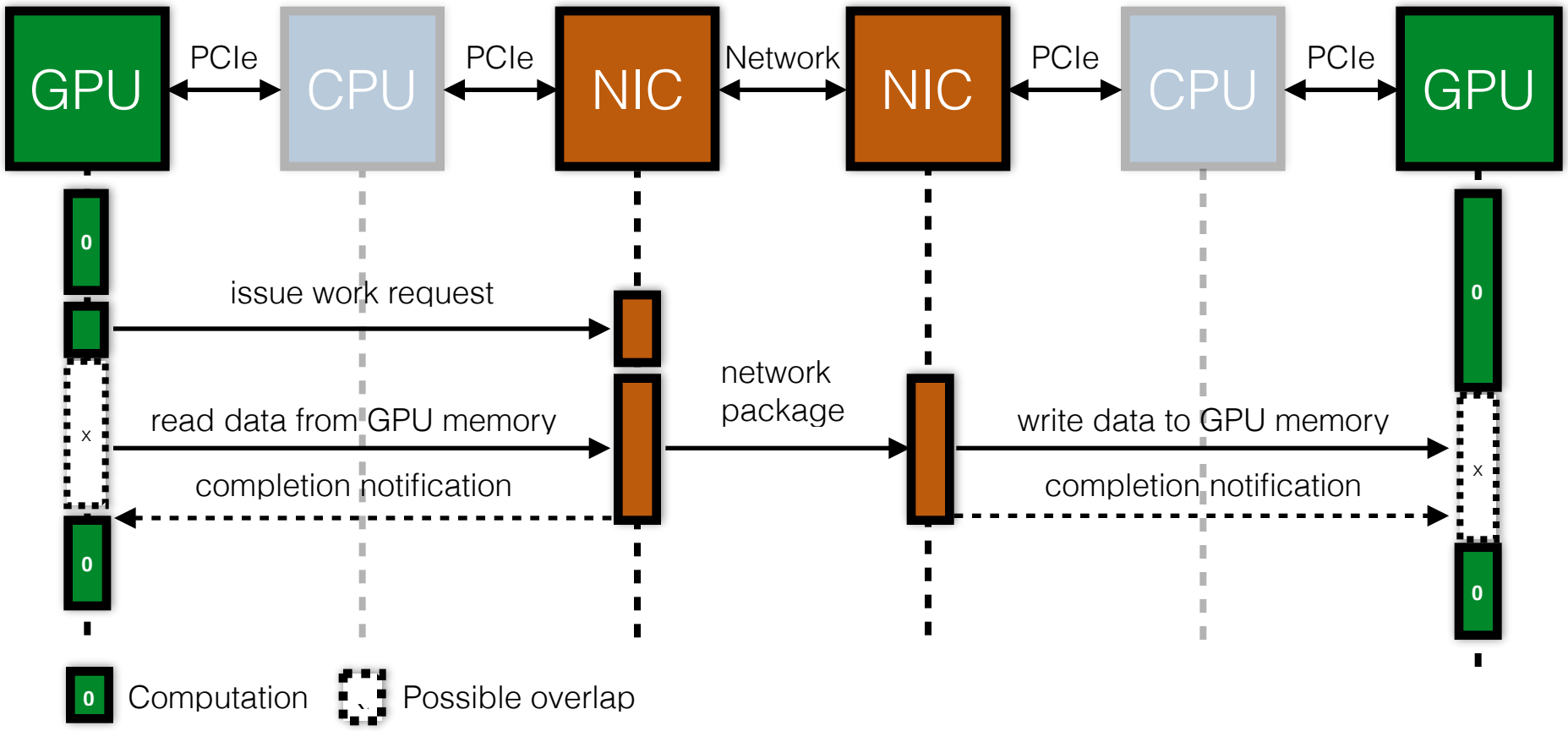
- Examples:
  - GGAS [2]
  - DCGN (CPU helper threads) [3]
  - MVAPICH (CPU controlled) [4]

What about PUT/GET semantics?

Outline

- Background
  - Put/Get
  - EXTOLL
  - Infiniband (IB)
- Performance
- Analysis
- Future Work

Put/Get Communication



Only <u>one</u> thread is required to generate WRs and consume notifications!

EXTOLL

- Focus: low latency communication
- Different functional units
  - VELO: message based communication [5]
  - SMFU: shared virtual memory [6]
  - **RMA: put/get semantics** [7]

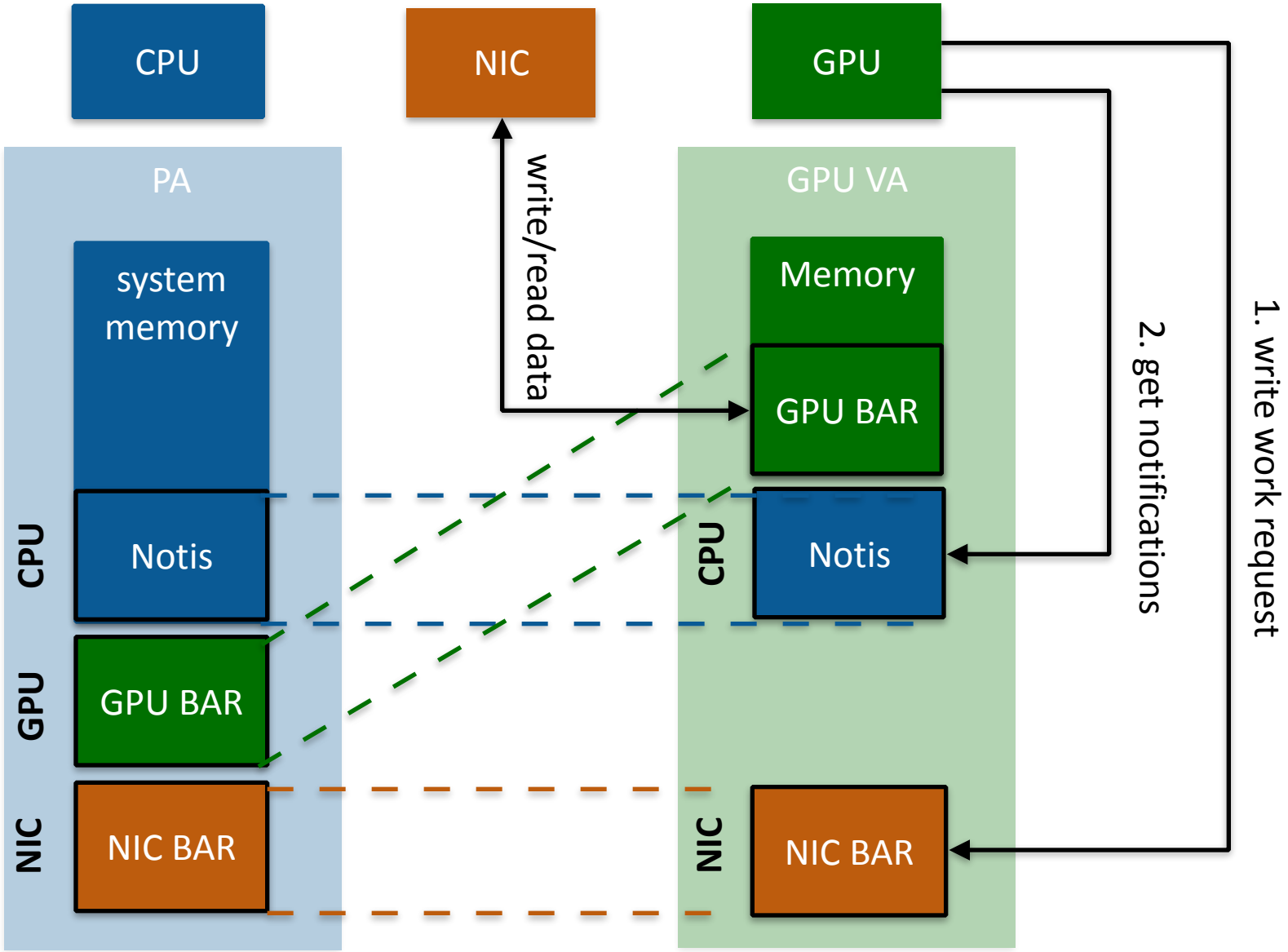
  - Some technical features:
    - Direct network, collective operation offload
    - **FPGA** @ 157 MHz (4-6 links, 3D Torus), ~16Gbit/s per direction
    - ASIC @ 800 MHz (7 links) under test, ~120Gbit/s per direction

Infiniband

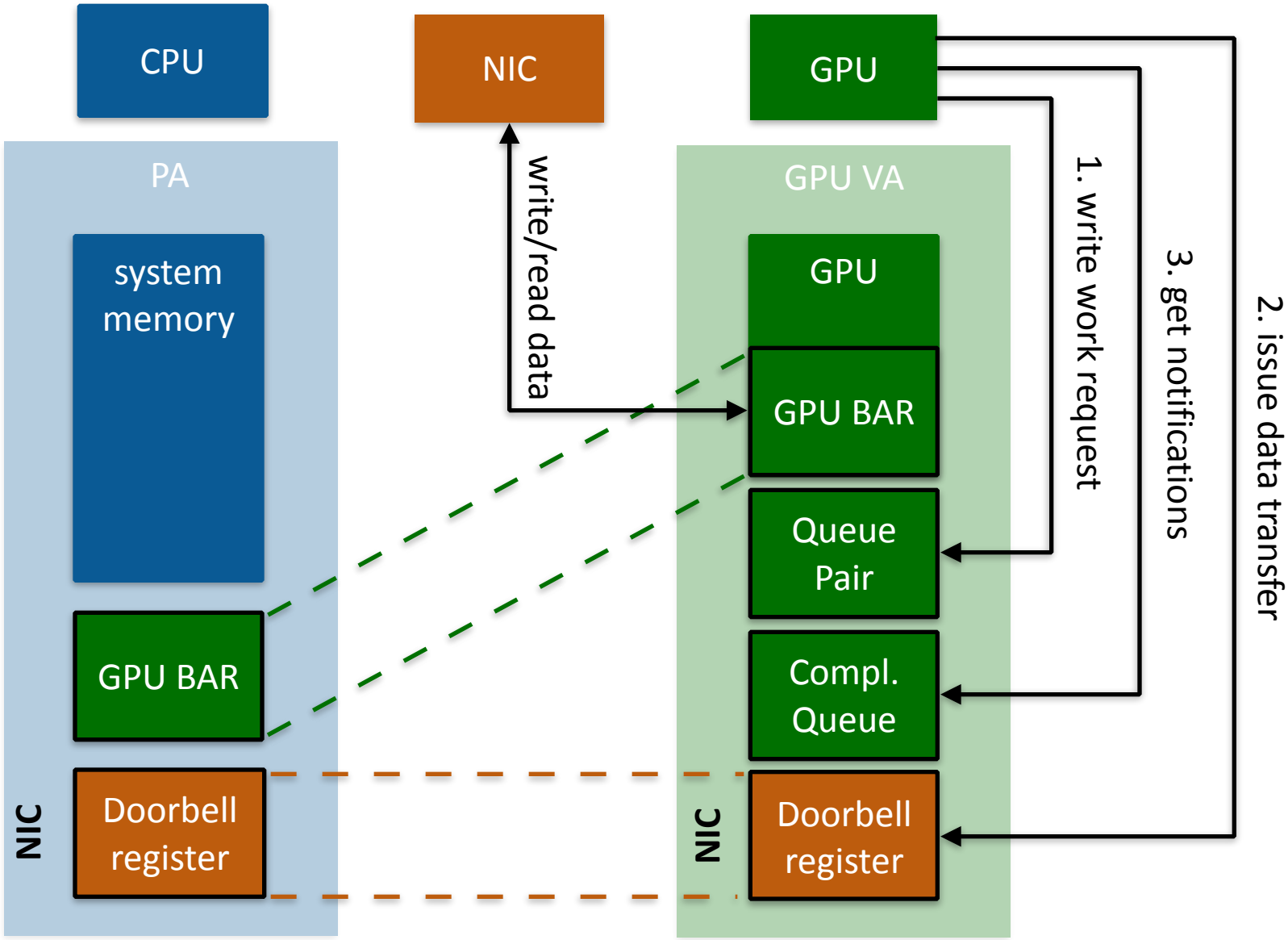
- State-of-the-art HPC network
- Main vendor: Mellanox Technologies Ltd.
- Switched fabric topology
- **Remote Direct Memory Access (RDMA) support**

- Technical features:
  - **ASIC** implementation
  - We used QDR (Quad Data Rate)
  - 4xLinks @ 8 Gbit/s (=32Gbit/s)

EXTOLL PUT/GET API



IB Verbs PUT/GET API



Performance

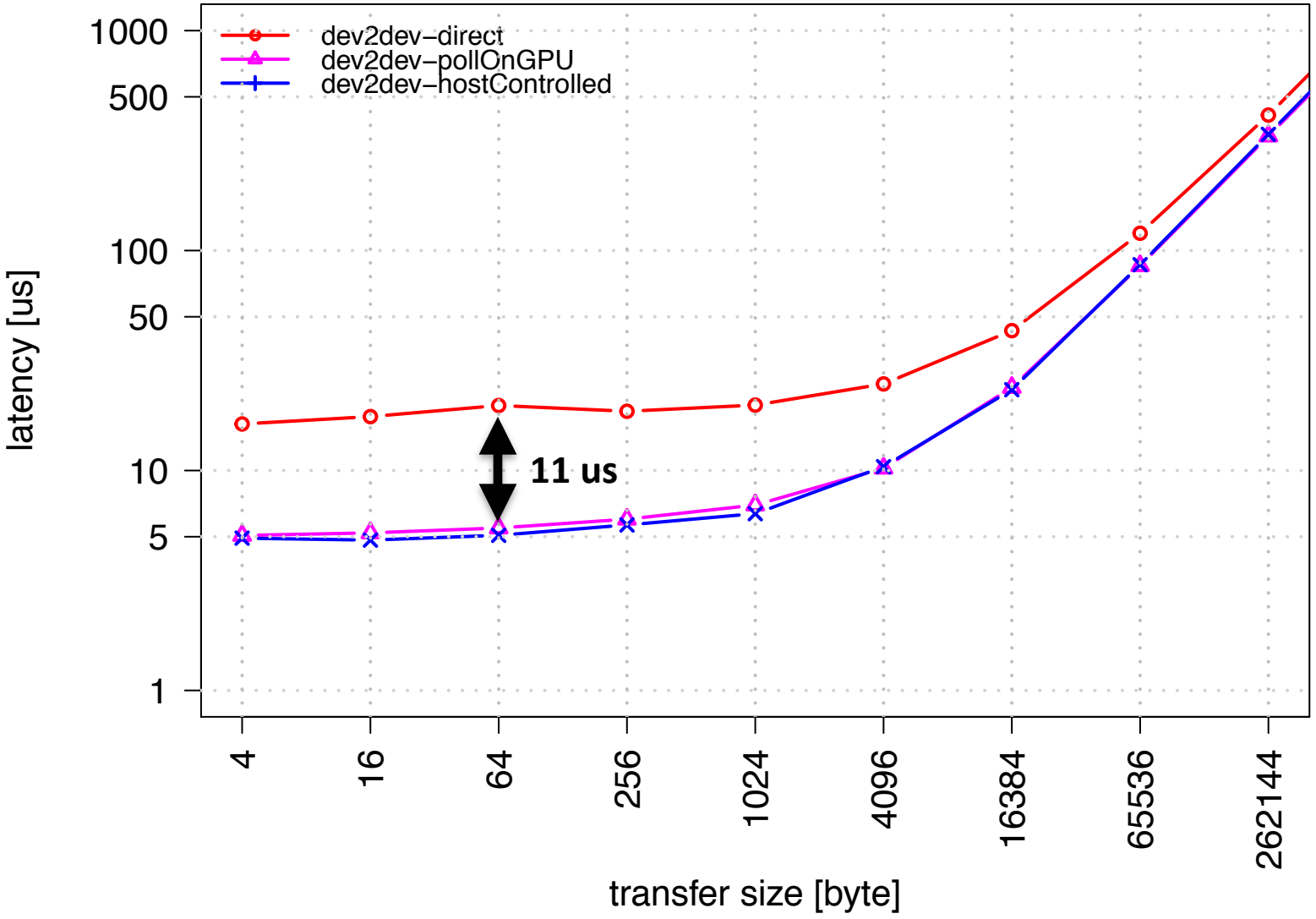
Let’s talk about numbers!

Let's talk about numbers: performance

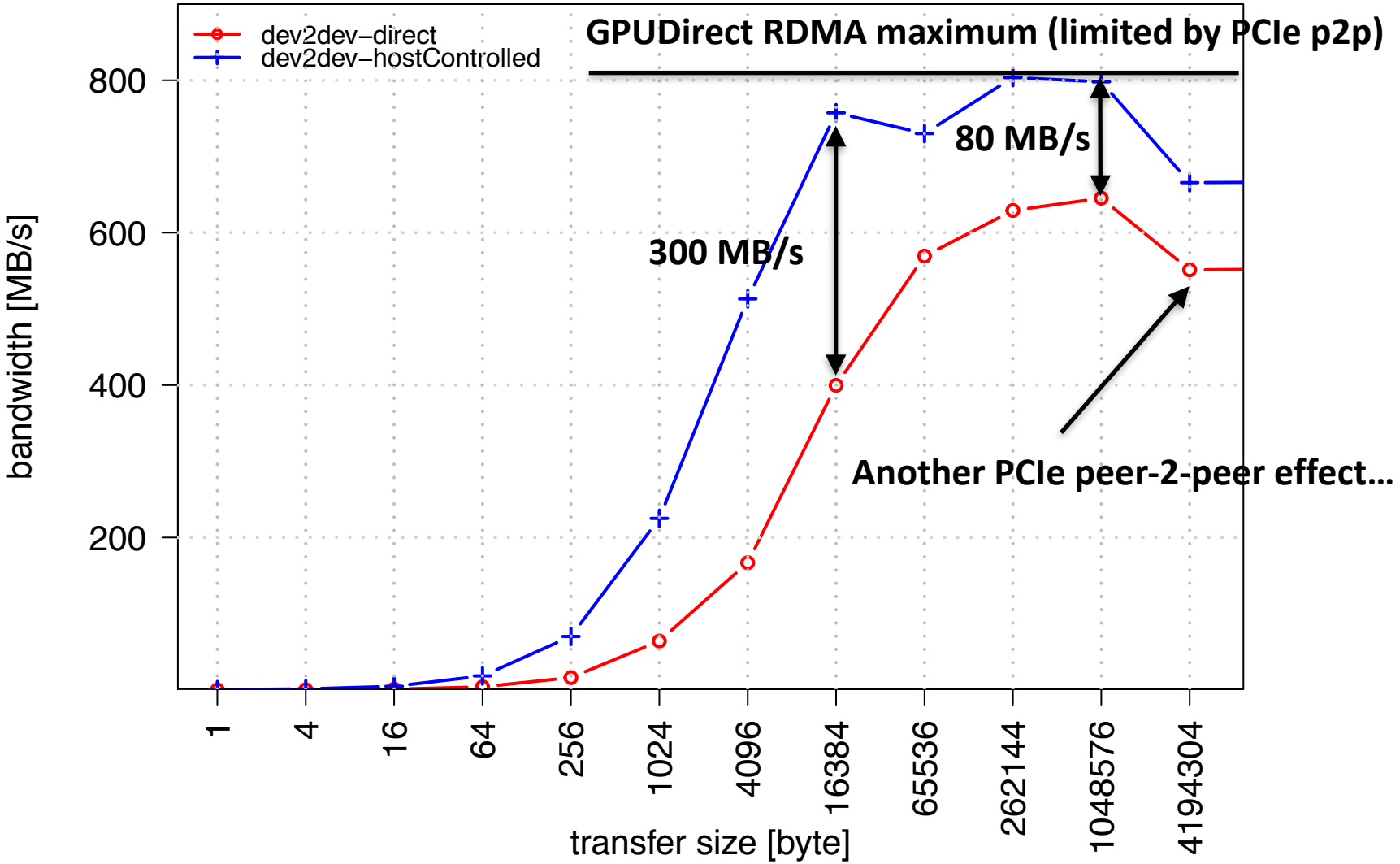
- We measured following metrics to determine the performance:
  - **Latency** (ping-pong)
  - **Bandwidth**

- We don't compare EXTOLL and IB directly
  - EXTOLL → FPGA @ 156 MHz ; ~9.8 Gbit/s
  - IB → ASIC (QDR); ~32 Gbit/s

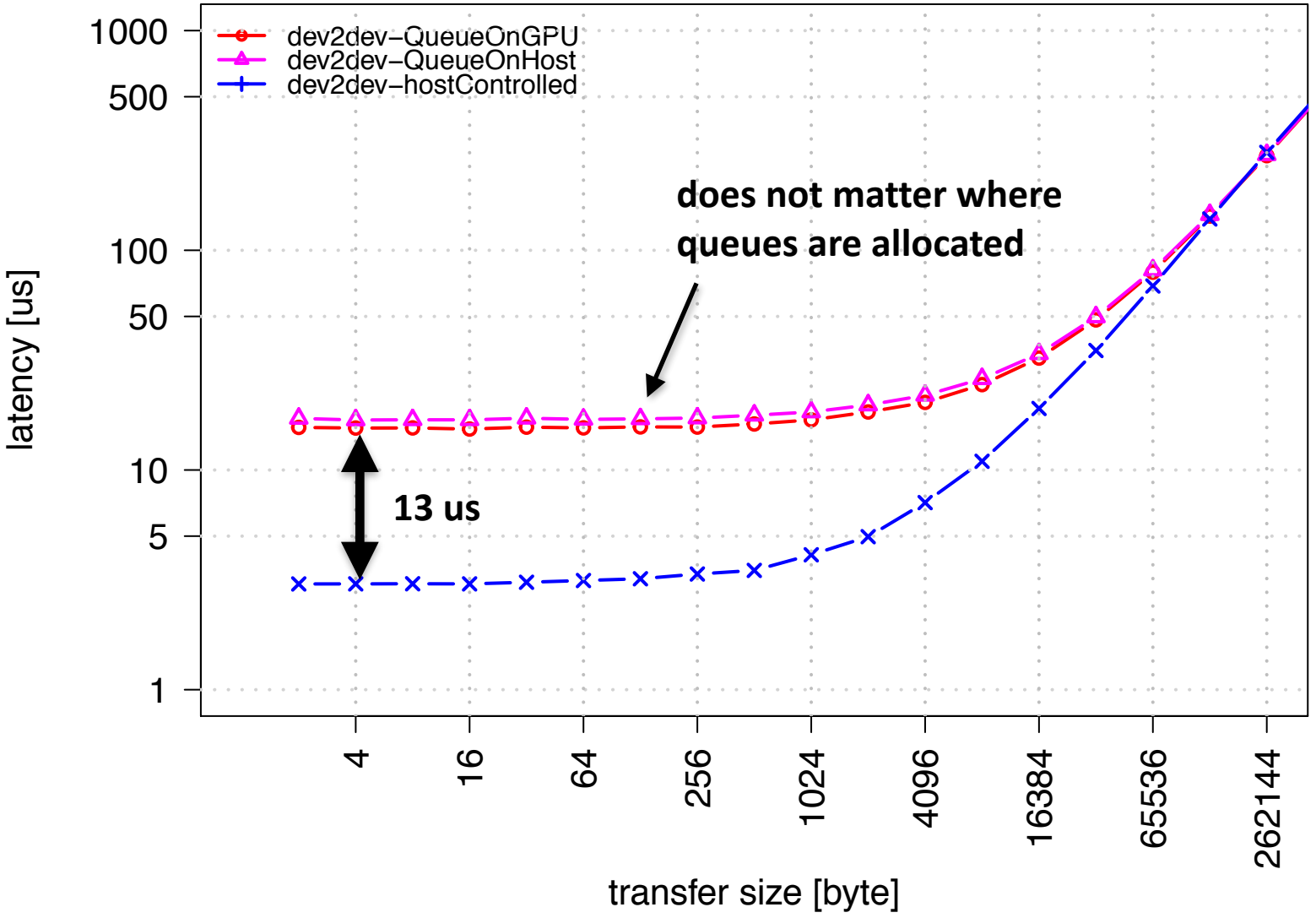
EXTOLL - Latency (Half-Roundtrip)



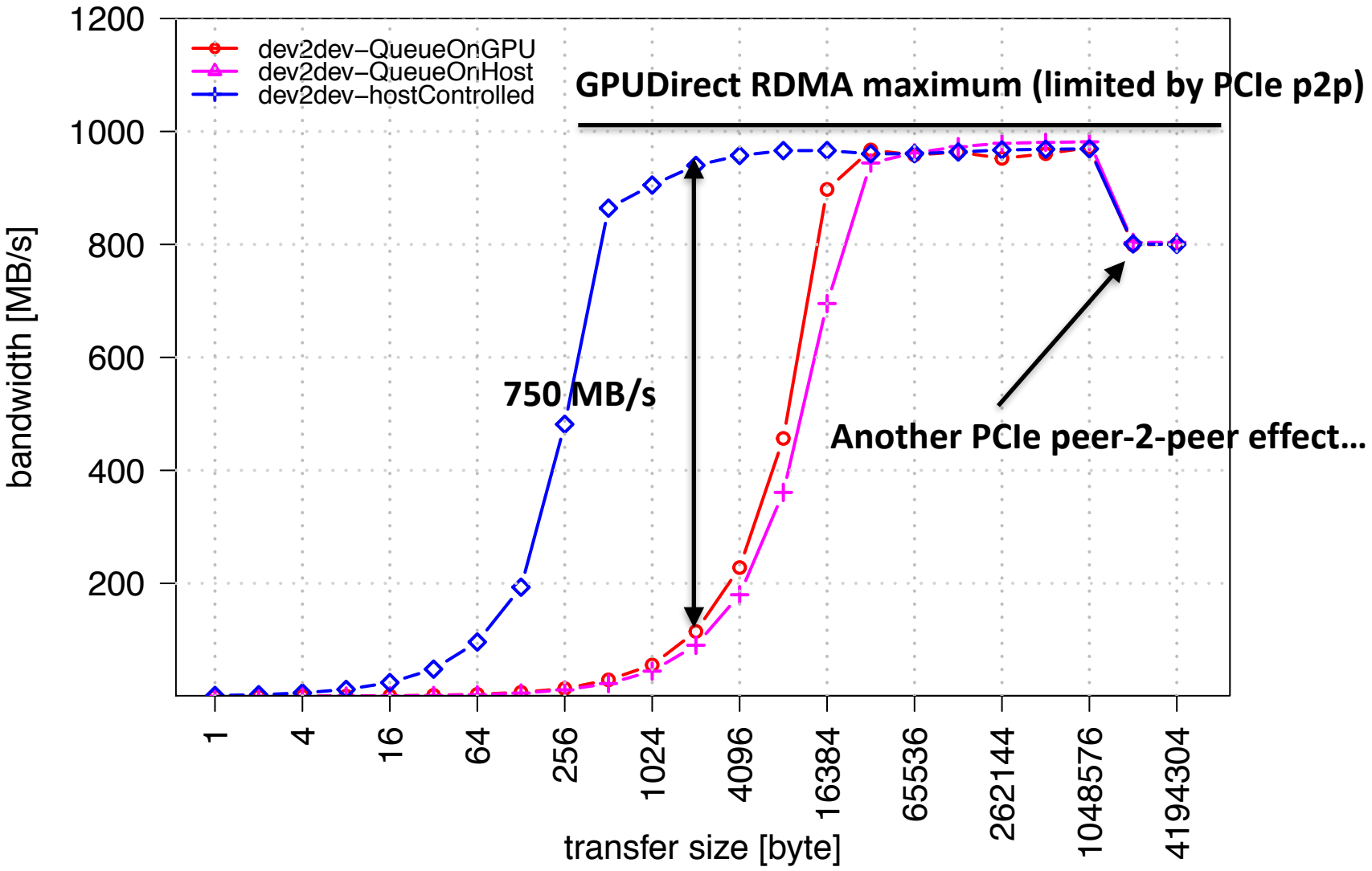
EXTOLL - Bandwidth



Infiniband - Latency



Infiniband- Bandwidth



Analysis

What's behind all this?

Observations

EXTOLL

- Polling on local GPU memory reduced latency significantly
- CPU-controlled peak bandwidth cannot be reached

Infiniband

- It does not matter where the queues are allocated at
- Almost CPU-controlled peak bandwidth can be achieved

Why?

Let's take a deeper look!

GPU Performance Counter for EXTOLL

Use notification system (CPU memory)

Polling on the last received element

metric	system memory	device memory
sysmem reads (32B accesses)	4,368	0
sysmem writes (32B accesses)	2,908	303
globmem64 reads (accesses)	0	1,314
globmem64 writes (accesses)	500	400
l2 read hits	0	3,143
l2 read requests	4,822	2,970
l2 write requests	5,268	404
memory accesses (r/w)	6,788	1714
instruction executed	46,413	22,491

GPU Performance Counter for Infiniband

QP and CQ allocated in CPU memory

QP and CQ allocated in GPU memory

metric	Buffer on Host	Buffer on GPU
sysmem reads (32B accesses)	772	80
sysmem writes (32B accesses)	670	316
l2 read misses	999	1,405
l2 read hits	16,647	14,575
l2 read requests	16,657	15,110
l2 write requests	1,990	1,885
memory access (r/w)	59,937	58,905
instruction executed	123,297	110,463

Insights

EXTOLL

- Polling on local GPU memory reduced latency significantly
- Notifications cause too much PCIe Traffic
- CPU-controlled peak bandwidth cannot be reached
- Also due to notifications (to ensure data has been sent before the next command is issued)

Infiniband

- It does not matter where the queues are allocated at
- Performance is limited due to WR generation and notification consuming (plenty of instructions)
- Almost CPU-controlled peak bandwidth can be achieved
- Work request generation and processing can be pipelined

Conclusion/Future Work

What did we learn?

Final Remarks

- In general, two performance aspects:
  - Work Request generation → Fast with EXTOLL
  - Notification handling → Notifications have to be placed in GPU memory

- What can we do better?
  - Get rid of PCIe… :)
  - EXTOLL: Move notifications to the GPU
  - IB: Make WR generation leaner (too many instructions are required)
  - IB: Large memory footprint due to queue pairs
  - More work has to be done in notification mechanisms

Future Work

- We are going to …
  - move the notifications to the GPU memory
  - optimize WR and notification formats
  - create a communication library with suitable abstractions
  - evaluate application performance

Questions?

Thank you very much for your attention!

Contributors:
Holger Fröning (Supervisor)
Lena Oden (PhD Student)

Special thanks to:
EXTOLL
NVIDIA

References

[1] John Shalf, Sudip Dosanjh, and John Morrison. 2010. Exascale computing technology challenges. In Proceedings of the 9th international conference on High performance computing for computational science (VECPAR'10), José M. Laginha M. Palma, Michel Daydé, Osni Marques, and João Correia Lopes (Eds.). Springer-Verlag, Berlin, Heidelberg, 1-25.

[2] Oden, L.; Froning, H., "GGAS: Global GPU address spaces for efficient communication in heterogeneous clusters," Cluster Computing (CLUSTER), 2013 IEEE International Conference

[3] Stuart, J.A; Owens, J.D., "Message passing on data-parallel architectures," Parallel & Distributed Processing, 2009. IPDPS 2009. IEEE International Symposium on , vol., no., pp.1,12, 23-29 May 2009

[4] S. Potluri, K. Hamidouche, A. Venkatesh, D. Bureddy, and D. K. Panda, Efficient Inter-node MPI Communication using GPUDirect RDMA for InfiniBand Clusters with NVIDIA GPUs Int'l Conference on Parallel Processing (ICPP '13), October 2013. , October 2013

[5] Litz, H.; Froening, H.; Nuessle, M.; Bruening, U., "VELO: A Novel Communication Engine for Ultra-Low Latency Message Transfers," Parallel Processing, 2008. ICPP '08. 37th International Conference on , vol., no., pp.238,245, 9-12 Sept. 2008

[6] Holger Fröning and Heiner Litz, Efficient Hardware Support for the Partitioned Global Address Space, 10th Workshop on Communication Architecture for Clusters (CAC2010), co-located with 24th International Parallel and Distributed Processing Symposium (IPDPS 2010), April 19, 2010, Atlanta, Georgia.

[7] Mondrian Nüssle, Martin Scherer, and Ulrich Brüning. A resource optimized remote-memory- access architecture for low-latency communication. 38th International Conference on Parallel Processing (ICPP-09), 2009.