PROCEEDINGS OF SCIENCE

APEnet+ project status

Roberto Ammendola[a], **Andrea Biagioni**[b], **Ottorino Frezza**[b], **Francesca Lo Cicero**[b], **Alessandro Lonardo**[b], **Pier Stanislao Paolucci**[b], **Davide Rossetti***[b], **Francesco Simula**[b], **Laura Tosoratto**[b] **and Piero Vicini**[b]

[a]*INFN Roma 2 Tor Vergata*
[b]*INFN Roma 1 La Sapienza*
E-mail: davide.rossetti@roma1.infn.it

We present an update of the status of the APEnet+ project, which is a custom 3D Torus interconnect system for GPU-accelerated computing clusters. The network host adapter hosts an FPGA-based programmable Network Processor, with one PCIe x8 Gen2 host link, on-board routing & switching logic, and up to six bidirectional off-board links each running up to 34 Gbps. A simple RDMA protocol and an experimental direct network-to-GPU interface bring a significant latency reduction for inter-node data transfers, even in those cases where one or more GPUs are involved. A justification for APEnet+ design choices on LQCD algorithms and some preliminary performance results for node-to-node synthetic communication tests and a will be shown, together with an overview of ongoing developments.

XXIX International Symposium on Lattice Field Theory
July 10 - 16 2011
Squaw Valley, Lake Tahoe, California

*Speaker.

PoS(Lattice 2011)045

1. Introduction

The APEnet project aims at developing a network fabric which allows assembling an HPC cluster *à la* APE with off-the-shelf components. Its latest iteration is the APEnet+ board [1]: a FPGA-based PCIe board with 6 fully bidirectional off-board links with 34 Gbps of raw bandwidth per direction, state-of-the-art signaling capabilities — up to X8 Gen2 bandwidth towards the host PC — and a Remote Direct Memory Access (RDMA) protocol that leverages upon peer-to-peer (P2P) capabilities of Fermi-class NVIDIA GPUs [2] to obtain real zero-copy, GPU-to-GPU low latency transfers.

2. The QUonG reference platform

This line of research is now driving a deployment initiative called QUonG (lattice QUantum chromodynamics ON GPU) [3]; a comprehensive effort to provide a hybrid, GPU-accelerated x86_64 cluster with a 3D toroidal mesh topology, able to scale up to $10^4/10^5$ nodes, with bandwidths and latencies balanced for the requirements of modern LQCD codes. It is a flexible and modular platform that can be tailored to different application requirements, *e.g.* by adjusting the GPUs vs CPUs and GPUs vs APEnet+ cards ratios, developing new features hardware-accelerated by the embedded reconfigurable logic, off-loading some processing onto the card itself, *etc.*.

The QUonG elementary mechanical assembly is a 3U stack consisting of two servers, each one equipped with two multi-core CPUs and a APEnet+ card, which *sandwich* one NVIDIA S2075 multiple GPU system with 4 NVidia M2075 GPUs. This assembly collates two QUonG *elementary computing units* — simply *units* from now on — where each server hosts one APEnet+ board and one interface board to drive 2 (out of the 4) GPUs inside the S2075; a QUonG unit is topologically equivalent to two vertexes of the APEnet 3D mesh.

We started at the beginning of 2012 integrating 14 QUonG elementary mechanical assemblies into a 42U standard rack enclosure; the aggregated peak performances are 74 and 37 TFlops in single and double precision respectively, with an estimated power consumption around 25 KW and at an estimated cost around 300 K€. Sixteen such systems are a viable configuration for a QUonG PFlops-scale installation within a short timeframe. This system will be characterized by a GFlops/W ratio of 3.4 with an estimated total power consumption of the order of 400 KW and a GFlops/€ratio of 0.29.

3. The APEnet+ hardware architecture

3.1 Distributed Network Processor: internals

The APEnet+ network architecture has, at its core, the Distributed Network Processor (DNP): it performs inter-node data transfers and acts as an off-loading network engine for the computing node. The current DNP implementation is deployed on a high performance Altera® FPGA. The DNP hardware blocks structure is split into three blocks:

- *Torus Links* — managing packet-encapsulation of data for DC-balanced, CRC-protected, full-duplex point-to-point connections of each node with 6 neighbours in a 3D toroidal topology, with 34 Gbps of raw bandwidth per link;

- *Router* — establishing dynamic links among the 8 ports cross-bar switch, the 6 Torus links and 2 internal links, according to dimension-ordering;
- *Network Interface* — implementing in hardware the PUT and GET semantics for the Remote Direct Memory Access (RDMA) on the receive data path and scatter-gathering data between the PCIe port (both from host or GPU) and the relevant destination ports on the transmit data path.

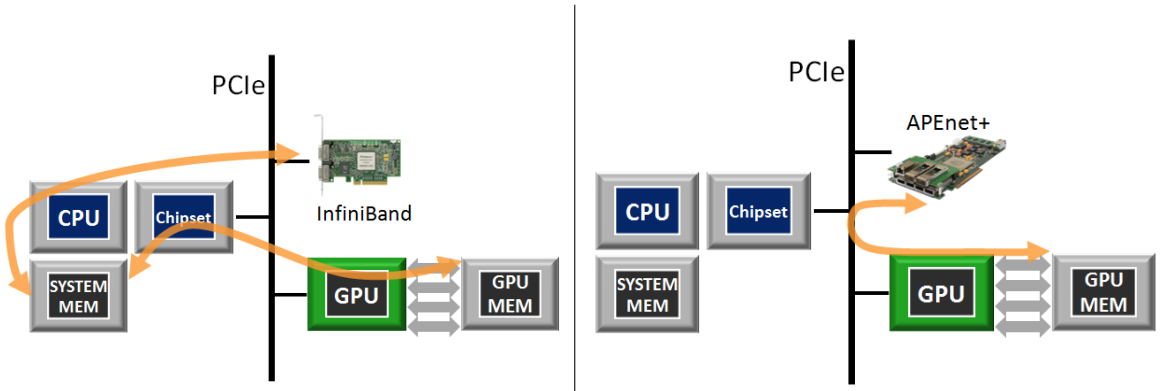
The RDMA protocol on a x86/x86_64 Linux OS requires virtual memory management for application-registered buffers. The tasks of managing a Look-Up Table (LUT) with addresses and lengths of registered buffers and a Page Table (PT) needed for virtual-to-physical address translations is off-loaded to a firmware running on a soft-core μC on the FPGA, with a LUT/PT instance allocated once per OS process and once per GPU; each APEnet+ can be used by up to 4 OS processes at a time.

3.2 GPU support

One of the strongest points of the APEnet+ design stands in its being able to take part in the so-called PCIe peer-to-peer (P2P) transactions. For instance, pushing data from GPU memory to the network with a Mellanox InfiniBand adapter needs a number of stages; the transmission of data residing on GPU memory requires the CPU to:

- wait for current GPU kernel to finish;
- copy data from GPU to an intermediate CPU memory buffer;
- issue network transfer command on this memory buffer,

and vice-versa on the receive side (see Fig. 1a). On the other hand, GPUDirect technology allows direct data exchange between GPUs – when they both are behind the same I/O Controller Hub (ICH) — without any CPU action or going through an intermediate buffer. APEnet+ is the first non-NVidia device with specialized hardware blocks [7] to support the NVidia GPUdirect peer-to-peer inter-GPU protocol; the APEnet+ board can target GPU memory by ordinary RDMA semantics with no CPU involvement and dispensing entirely with intermediate copies (see Fig. 1b). In this way, real zero-copy, inter-node GPU-to-host, host-to-GPU or GPU-to-GPU transfers can be



(a) Traditional transfer.

(b) APEnet+ transfer.

Figure 1: Left, 2-steps GPU mem TX/RX; right, APEnet+ zero-copy GPU mem TX/RX via P2P.

achieved, with substantial reductions in latency; preliminary results on such reductions are very promising (see section 5).

4. Performance model

We envision a node equipped with $N_{GPU} = 1,2$ or 4 GPUs; each GPU is assigned a lattice chunk of volume $T \times L^3$ so that each node works on a $T \times L^3 \times N_{GPU}$ sublattice volume.

The most onerous task for standard parallel LQCD codes is the sparse matrix-vector product on the lattice which is domain-decomposed over the nodes. Such operator employs 1320 floating point operations per lattice site; in this way, each application needs (the factor 1/2 comes from even/odd preconditioning):

$$f(L,T,N_{GPU}) = (1320\ \frac{\text{Flop}}{\text{site}}) \times (TL^3 N_{GPU}\ \text{sites}) \times \frac{1}{2}(\text{for E/O})$$

Actual floating point performance on a GPU-equipped node is an involved function of L, T and many details of the calculation; for lattices not too small and with a number of tricks to reduce the pressure on the GPU memory bandwidth (temporal gauge fixing, appropriate choice of basis for the γ matrixes, 8 parameter representation for $SU(3)$ matrixes, *etc.*), a sustained performance of approximately 150 GFlop/s per GPU in single precision is within reach [6].

Assuming for the sublattice a length along the 4^{th} dimension of TN_{GPU}, slicing the sublattice over the GPU's along that dimension means that the frame of the sublattice is made of $6TL^2N_{GPU} + 2L^3$ sites. For such decomposition to work, the hypersurfaces belonging to the interfacing neighbouring nodes sublattices — the haloes, as they are called in numerical PDE jargon — must be kept updated; with the above assumptions, these data amount to:

$$r(L,T,N_{GPU}) = \begin{cases} \text{if } N_{GPU} = 4 & 6TL^2N_{GPU} \\ \text{if } N_{GPU} = 1,2 & 6TL^2N_{GPU} + 2L^3 \end{cases} \text{sites} \times (96\ \tfrac{\text{bytes}}{\text{site}}) \times \tfrac{1}{2}(\text{for E/O}) \times \tfrac{1}{2}(\text{for } \gamma\text{-proj.})$$

With $N_{GPU} = 4$ we imagine to be able to exhaust within one node the whole 4^{th} dimension; this is why the halo to be swapped reduces to $6TL^2N_{GPU}$ sites. The first factor $\frac{1}{2}$ reckons with the fact that the operator can be restricted to work only on sites of given parity because of even/odd preconditioning, the second one for the fact that the two halves of a γ-projected spinor are linearly dependent so that only one needs to be communicated. With such performance figures, a hard lower limit for execution time is the ratio $\frac{f(L,T,N_{GPU})}{N_{GPU}} \cdot 150GFlop/s$.

When we put communications into the picture, the better this can be overlapped with computation, the nearer to this limit the system can approach; in this case, a reasonable request is that the system be *balanced*, *i.e.* times spent computing and communicating are mostly equal.

Since the single precision floating point performance is known, defining BW as the balanced network bandwidth of a node requires:

$$\frac{f(L,T,N_{GPU})}{N_{GPU} \cdot 150GFlop/s} = \frac{r(L,T,N_{GPU})}{BW}$$

In this way, the bandwidth requirement for the APEnet+ on a single node is:

$$BW(L,T,N_{GPU}) = \frac{r(L,T,N_{GPU})}{f(L,T,N_{GPU})} \times N_{GPU} \times 150GFlop/s \tag{4.1}$$

As a reference of a typical LQCD application, we take the case study of a $64^3 \times 128$-sized lattice of single precision Wilson fermions, which is more or less the biggest volume for current

production simulations. In table 1 we make a run down of different configurations able to accommodate such a lattice with the corresponding figures for needed nodes, GPUs and bandwidth.

Unit local lattice	GPUs/unit	Req. BW (GB/s)	Total GPUs
$16^3 \times 32$	2	4.3	512
$16^3 \times 64$	2	4.0	256
$32^3 \times 64$	2	2.1	32
$16^3 \times 128$	4	7.4	256
$32^3 \times 128$	4	3.7	32

Table 1: Network bandwidth required for balanced strong scaling of QUonG systems on a $64^3 \times 128$ LQCD lattice, varying the numbers of GPUs per node and the per-node local lattice. A GPU with 3GB of memory is assumed.

The hard limit for APEnet+ is the PCIe interface bandwidth, which can be estimated as $\simeq 3GB/s$, accounting for 8b/10b encoding and a 75% effective bus utilization. Comparing the listed numbers with the APEnet+ PCIe host interface peak bandwidth, 3 GB/s, we foresee no major impediments to systems composed of a few GPUs per APEnet+ $(1 \div 4)$. Actually, looking at table 1 we identify a sweet spot on the third row, *i.e.* $N_{GPU} = 2$ per APEnet+.

Therefore, our envisioned cluster node is made up of a multi-core CPU, two NVIDIA GPUs and one APEnet+ card. If applied to the $64^3 \times 128$ lattice, a large QUonG system should be optimally divided into partitions composed of 32 GPUs.

5. Benchmarks

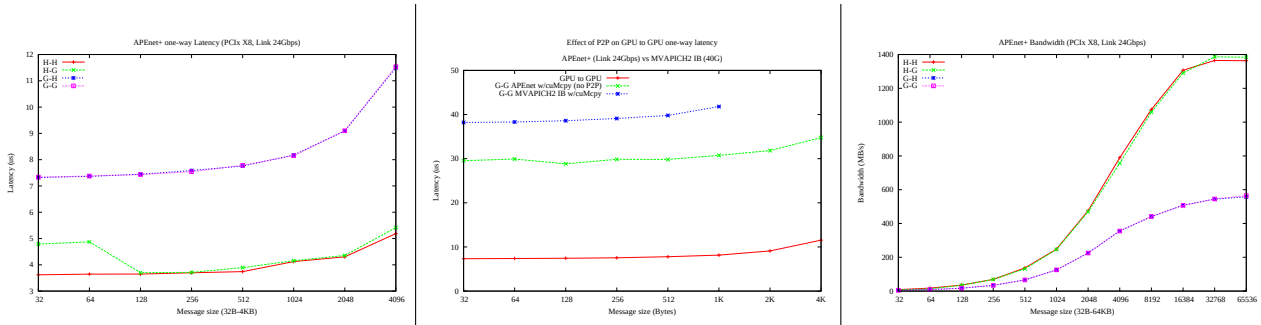
The APEnet+ benchmarks were performed on a QUonG platform mounting SuperMicro servers with dual Xeon 56xx processors, 24GB system memory, running CentOS 5.7 x86_64, one NVIDIA C2050 GPU in an x16 Gen2 slot. The APEnet+ cards used were preliminary and used a reduced link speed of 13 Gbps.

Our benchmark program is coded using the APEnet RDMA APIs and is basically a one-way point-to-point test involving two nodes, similar in spirit to the OSU MPI bandwidth test [1]. Basically, the receiver node 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 and ends by sending back an acknowledgment (ACK) packet. The transmitter node waits for an initialization packet containing the receiver node buffer (virtual) memory address, writes that buffer N times in a loop with RDMA PUT, then waits for a final ACK packet.

In fig. 2a we plot the timings of our benchmark test for small message sizes, which is our estimate of the one-way latency[2]. H stands for Host and G for GPU; red and green lines are for TX of Host memory, while blue and purple ones are for GPU memory TX. The test does not take advantage of any special optimization trick for small messages, like copying of data in temporary buffers; we still need work for the pipelining capability of the APEnet+ HW, so we expect it to perform not very differently from a round-trip test. In the near future, we plan to analyze and optimize the round-trip as well as the bidirectional bandwidth benchmarks. From the plot, a factor

[1] `http://mvapich.cse.ohio-state.edu/benchmarks/`

[2] Note that this is not the half round-trip latency as usually reported in the literature.



(a) APEnet+ Latency. (b) APEnet+ vs. MVAPICH over IB. (c) APEnet+ Bandwidth.

Figure 2: Characterization of the APEnet+ board.

two penalty is evident when the transmitting buffer is located on the GPU. Anyway 7 μs is already a good result, as shown in the plot in Fig. 2a. We are aware of the shortcoming; it depends on details of the P2P protocol that we are not entitled to disclose and which will be addressed in the near future.

Figure 2b compares the improvements brought about by the P2P support: without the P2P optimizations, the results are those on the green line; timings must include two `cudaMemcpy()`'s to pull data from GPU memory to host memory and vice-versa on the receiving side. Enabling the P2P feature, the results are those on the red curve. From the plot it is clear that the difference between the two curves is about 20μs, which is approximately twice the cost of a single `cudaMemcpy()`. The APEnet+ card shows a latency improvement roughly by a factor 4. To give an idea, the points on the blue line come from a bi-directional latency test of MVAPICH2, MPI over InfiniBand from the Ohio State University [3], which does not exploit the P2P feature to boost the transfers.

The plot in Fig. 2c is the result of a very preliminary bandwidth benchmark. It should be noted that there is no such difference between lines having the same source buffer memory type, *i.e.* the type of destination memory has no significant impact. As mentioned previously, the discriminating factor is the use of the GPU as a transmitter. The upper curves in figure 2c rise as expected up to a message size of 16 KB, where a plateau appears due to the limited bandwidth of the torus links — 1150 MB/s are roughly equivalent to a raw link speed of 11 Gbps. The lower curves — where GPU memory is the transfer source — show a low asymptotic bandwidth of roughly 600 MB/s. This is mainly due the high impact of the P2P read protocol, similar to the effect seen in figure 2a, which adds a constant overhead to the transmission of each 4 KB packet; in the current preliminary implementation, where the protocol is fully implemented in software on the μC firmware, the overhead is not overlapped among subsequent packet transmissions, so that it basically acts as multiple barriers, preventing the pipelining of the packet flow.

6. Future work

As shown in section 4, with two GPUs per node a sustained bandwidth of roughly 2 GB/s is required to fully overlap the communication with the computation on a $64^3 \times 128$ global lattice; in other words, the strong scaling of the application should be perfect up to 32 GPUs.

[3] `http://mvapich.cse.ohio-state.edu/performance/mvapich2/inter_gpu.shtml`

With the definitive hardware in our hands and the QUonG commissioning phase in progress, we have conducted a preliminary analysis of our interconnect; they show that our GPU optimizations based on direct PCIe P2P access to GPU memory provide a factor three reduction in one-way latency for small message sizes against what is obtained without P2P. The regime of large message size is currently negatively impacted by the external link speed cap of 13 Gbps, a downside which is currently being addressed. On top of this, the current TX-from-GPU path implementation has to be improved to allow for packet flow pipeling.

In the next months we will focus on a few key items: accelerating the software stack by moving most of the latency sensitive code to user-space. Besides, we plan on improving the TX-from-GPU data path by implementing some pre-fetching techniques to hide some latency and by speeding up the processing in the μC firmware. Of course, moving towards the final 34 Gbps link speed is decisive to meet the good scaling on LQCD. The hardware support for the RDMA GET primitive is still in progress and could be used to reduce the initial round-trip traffic in point-to-point MPI primitives for large buffer sizes [8]. OpenMPI support for GPU buffers in point-to-point primitives is really needed to get the benefits of our work in standard application with minimal code refactoring.

Starting from Q3 2012, in coincidence with delivery of PCIe Gen3 capable host platforms and GPUs and market availability of new FPGAs, we plan to upgrade the APEnet+ host interface to PCIe Gen3, which in principle could provide for an additional factor two in bandwidth, improving the scalability threshold of the QUonG platform.

References

[1] R. Ammendola et al., *APEnet+: a 3D toroidal network enabling petaFLOPS scale Lattice QCD simulations on commodity clusters*, Proceedings of *Lattice 2010 conference*, PoS LAT2010 (2010) 022

[2] NVidia GPUDirect technology `http://developer.nvidia.com/object/gpudirect.html`.

[3] R. Ammendola et al., *QUonG: A GPU-based HPC System Dedicated to LQCD Computing*, Symposium on Application Accelerators in High-Performance Computing (SAAHPC), 113-122, (2011).

[4] D.J. Holmgren, *PC Clusters for Lattice QCD*, Nucl. Phys. B - Proc. Suppl. **140**:183-189, March 2005, [arXiv:hep-lat/0410049v2].

[5] `http://lattice.github.com/quda`

[6] R. Babich, *Hardware developments for lattice QCD*, `http://www.physik.uni-regensburg.de/STRONGnet/documents/STRONGnet2010/babich.pdf`

[7] R. Ammendola et al., *Mastering Multi-GPU Computing on a Torus Network*, Poster @ NVIDIA GPU Technology Conference 2010, San Jose (CA).

[8] S. Sur et al., *RDMA Read Based Rendezvous Protocol for MPI over InfiniBand: Design Alternatives and Benefits*, Symposium on Principles and Practice of Parallel Programming (PPOPP'06), March 29-31, 2006, Manhattan, New York City.