Scalable Communication Architectures for GPU-centric Systems

Benjamin Klenk and Holger Fröning
Computer Engineering Group, Ruprecht-Karls University Heidelberg
Mannheim, Germany
{benjamin.klenk,holger.froening}@ziti.uni-heidelberg.de

1. INTRODUCTION

Accelerators are an essential component in today's High Performance Computing (HPC) systems and are now evolving to become peer devices with an increasing number of features and tighter integration.
As peers, accelerators need to sink and source networking traffic, which seems promising since costly interactions with the CPU are avoided.
GPU-sided communication can be implemented by two different models: offloading communication to dedicated networking hardware or performing communication tasks on the GPU on top of shared memory models like Nvidia's NVLink. This work shows that for GPU-to-GPU traffic the offloading option is always superior to CPU-controlled communication. We also present some early results and insights of communication being processed on the GPU without additional networking hardware.

2. COMMUNICATION MODELS AND GPU-NIC INTERACTIONS

Traditionally, systems deploy Network Interface Controllers (NICs) to offload communication to specialized hardware, which provides high performance message processing and address translation capabilities.
With NVIDIA's GPUDirect Remote Direct Memory Access (RDMA) feature the GPU is able to directly access other PCIe devices and vice versa, allowing to access the NIC and trigger data transfers. If the NIC is able to receive and forward memory operations, a global address space (GAS) can be implemented, allowing for fine-grain communication between GPUs[1].
In previous work [3], we showed that bandwidth of GPU-controlled communication is favorable. Fine-grain communication, specifically, can get close to the network's peak bandwidth, even for small data transfers.
In the same work we also compared disparate communication models on application level, showing that GPU-sided communication outperforms CPU-controlled communication for any application we looked at. While fine-grain communication is best for applications with small messages, GPU-sided RDMA is superior for applications that allow to overlap computation with communication, such as the stencil-based Himeno. Fine-grain communication requires more resources since each thread issues a memory operation across the network, reducing the availability of resource for computation.
Besides performance, another important metric is energy consumption, which is also studied in [3]. It became apparent that GPU-controlled communication results in tremendous energy savings, mainly due to shorter execution times and the CPU being in idle state during communication phases.

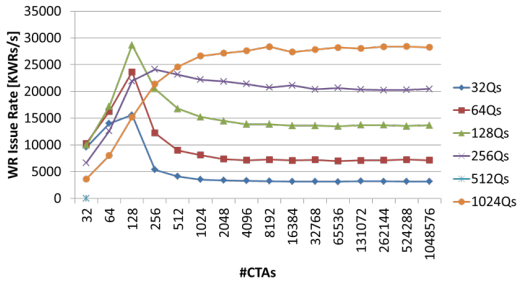

Figure 1: Work Request (WR) issue rate between two CTAs. A WR amounts to 4B and the queue can hold 4K WRs. Experiments were run on a Kepler K80 GPU.

3. SOFTWARE-DEFINED GPU-CENTRIC COMMUNICATION

Current trends point towards integrated solutions, where networking capabilities are added to processors. This applies to both CPUs and many-core architectures, like Intel's integrated OmniPath, and also to GPUs, like NVIDIA's Pascal architecture which implements an NVLink interface.
NVLink enables a shared virtual address space across GPUs and provides fine-grain communication based on load/store memory operations. However, such a plain load/store model is complicated at scale since buffer management has to be explicitly handled by the user and the entire network's memory is difficult to be mapped into the local virtual address space. To combat

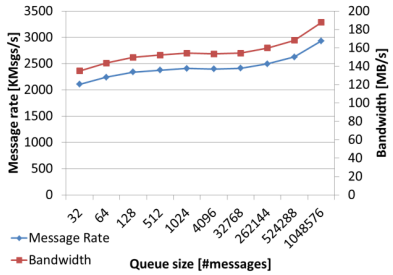

Figure 2: Single-warp inter-GPU message rate with 4B per message. Experiments were run on two K80s, connected via PCIe.

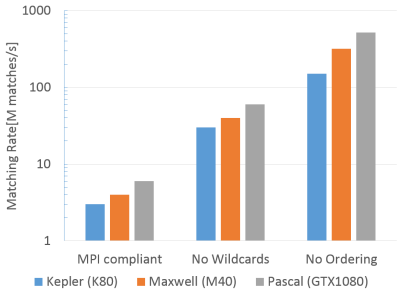

Figure 3: Message/receive request matching performance on three generations of GPUs, applying different types of messaging relaxations.

these issues, we have begun to explore managed communication on top of the GAS, such as message passing or one-sided communication. In addition, message passing is a well known programming model with clearly identifiable data transfer costs.
Designing a message-passing system on top of NVLink's GAS has challenges. Some of them are addressed in the following.

Efficient Queue Structures: Queues are essential for communication as they allow multiple processes to share a memory allocation. However, queuing on GPUs is mainly limited by the required atomic memory operations to enqueue elements and the arithmetic to determine the current read and write indices. Specialized atomic operations can improve these mechanisms with little or no impact on the hardware as it seems that only microcode changes are required.
Fig. 1 shows a work request (WR) rate up to 30M WRs/s for inter-CTA queuing, with 4B-sized WRs, and 32 WRs per CTA being enqueued. Fig. 2 depicts the performance for a single-warp that enqueues messages into a remote GPU's queue, with a rate of up to 3M messages/s (4B messages).

Message/Receive Request matching: Another important aspect is message/receive request matching. MPI's matching semantics allow for wildcards and maintain ordering, which seems to limit performance on highly parallel processors as it creates dependencies. As Fig. 3 shows, by simply prohibiting source wildcards a speedup of 10x over the MPI-compliant matching can be achieved. Hash tables, which are only possible if ordering is relaxed, perform even more remarkably with up to 500M matches/s (80x).
We also analyzed Exascale proxy applications [4] and found that such relaxations of the matching semantics are feasible. Most applications would at least support 20 queues and wildcards are rarely used. Furthermore, queue lengths are moderate with most less than 1024 entries, allowing queues to be processed by a single CTA. Additionally, many applications seem to follow a strict bulk-synchronous model where ordering could be given up to yield higher performance.

Scheduling and dynamic resource allocations: Scheduling and the lack of CTA-level preemption is another challenge. However, NVIDIA recently introduced instruction-level preemption with their latest Pascal architecture [6] and we believe future generations of GPUs will continue to improve in this regard.
Similarly, dynamic allocation of memory on kernel level has been shown to be possible [5], but costs are still high. The push towards a unified memory between CPU and GPU [6] makes us confident that future GPU generations will provide more performant ways to allocate memory on GPUs.

4. FUTURE WORK

In future work we plan to design a communication library that provides optimized building blocks, such as queues, message/receive matching structures, topology-aware collective operations, in-kernel memory management, and active messages to support autonomous communication between GPUs. This facilitates the design and implementation of GPU-centric communication models, optimized for applications and the highly parallel nature of GPUs.

5. REFERENCES

[1] L. Oden and H. Fröning, "GGAS: Global GPU address spaces for efficient communication in heterogeneous clusters," in IEEE Cluster, 2013.

[2] B. Klenk, L. Oden, and H. Fröning, "Analyzing put/get APIS for thread-collaborative processors," in HUCAA Workshop in conjunction with ICPP, Minneapolis, MN, USA, 2014.

[3] B. Klenk, L. Oden, H. Fröning, "Analyzing Communication Models for Distributed Thread-Collaborative Processors in Terms of Energy and Time", in International Symposium on Performance Analysis of Systems and Software (ISPASS), Philadelphia, PA, 2015

[4] NERSC, "Characterization of the DOE Mini-apps." Retrieved July 14, 2016 from `https://portal.nersc.gov/project/CAL/doe-miniapps.htm`.

[5] A. V. Adinetz, and D. Pleiter. "Halloc: A fast and highly scalable GPU dynamic memory allocator," Retrieved July 14, 2016 from `https://github.com/canonizer/halloc`.

[6] NVIDIA, "NVIDIA Tesla P100," whitepaper, 2016. Retrieved July 14, 2016 from `https://images.nvidia.com/content/pdf/tesla/whitepaper/pascal-architecture-whitepaper.pdf`.