Data Movement Options in Accelerated Clusters

Holger Fröning

Institute of Computer Engineering, Ruprecht-Karls University of Heidelberg, Germany
holger.froening@ziti.uni-heidelberg.de

Abstract. The concurrency galore is currently defining computing at all levels. At the high end, it is leading to clusters composed of hundreds or thousands of nodes, each integrating 10-100s of computing cores, and fed with instructions by up to two orders of magnitudes more threads. Technology constraints prohibit a reversal of this trend, and the still unsatisfied need for more computing power has led to a pervasive use of accelerators to speed up computations. Both in terms of energy and time, communication is more expensive than computation. Being amplified by the advent of Big Data, we are observing a fundamental transition to communication-centric systems composed of heterogeneous computing units. This talk will review current techniques for data movements in clusters. This is followed by an introduction to the Global GPU Address Spaces (GGAS) project, which we use to explore data movement optimizations in heterogeneous environments. The talk will conclude with a summary of synergies regarding related research projects.

1 Motivation

We can clearly see that the multi-/many-core trend won't revert. The main reasons include technological constraints, like the end of Dennard scaling, which previously ensured that total chip power stays constant. On the other hand, we can observe a vast increase in technology diversity. For processing tasks, various options exist including x86, ARM, GPUs and MICs. On the storage side, we can choose between spinning disks, SSD-based and PCIe-based FLASH, and DRAM for in-memory computations. While this diversity is a huge research opportunity, the concurrency galore together with the raising interest in heterogeneity will amplify complexity, forming a research challenge. We are facing multiple *Instruction Set Architectures (ISAs)* within one system, different and maybe incompatible task, execution and communication models, and different types of memory and NUMA levels. In addition, energy costs for data movements will dominate the overall energy consumption of future systems, while computations with the help of specialization will only state a tiny fraction in terms of energy [1] [2] [3]. As energy costs for data movements are in first order a constant multiple of the distance, any optimization in energy will require an optimization of locality [3].

In this work we will shortly summarize current trends, observations and insights that lead to another fundamental transition, leading towards communication-centric heterogeneous systems. Data movement optimizations are essential, and we will provide a solution for tailored for GPUs.

2 A Fundamental Transition

Our work to explore data movement optimizations in accelerated clusters is motivated by several observations. The following list summarizes them, before we conclude our key finding.

- **Multi-/Many-core** is already pervasive, and – although highly debated – Big Data is present and won't revert. While memory resources are scarce anyway, the transition to multi-core amplifies the problem in terms of memory capacity per core. DRAM is still the first choice for performance-critical workloads, as can be seen looking at the various in-memory and in-core solutions. Big Data pushes the requirements regarding capacity way beyond what single systems can deliver. However, a possibility to overcome capacity limitations might be sharing at cluster level.
- **Hard power constraints** are the main limitation of computing systems, due to technical, ecological and economic reasons. This applies to devices as small as smartphones or other mobile devices, but also to large-scale facilities like clusters, supercomputers or datacenters. These energy constraints have also led to a huge interest in heterogeneous processing, using accelerators to speed-up certain workloads. This specialization significantly improves energy-efficiency in terms of performance-pre-Watt, however also dramatically increases complexity as different ISAs are employed.
- The **technology diversity** is in-line with this trend towards specialization, and now users have various possibilities to compose a computing system, including processors (CPU, GPU, MIC), memory (DRAM, GDDR, PCM, Hybrid Memory Cube), storage (SSD-based or PCIe-based FLASH, spinning disks), network (multiple Ethernet variants, Infiniband, specialized cluster networks) and even complete systems (Cray, IBM Blue Gene, Seamicro). This diversity can help to maximize the energy efficiency of a system; however certain workload characteristics will likely be favored then. In addition the complexity of a system is increasing.
- **Performance disparities** in terms of throughput are diminishing. For instance, looking at different interconnect technologies we can see that PCIe Gen3 peaks at 16GB/s bandwidth, but cluster interconnects are reaching similar levels: current Infiniband FDR dual-port adapter cards achieve about 12GB/s, the EXTOLL Tourmalet network card will reach about 10GB/s and 100 Gigabit Ethernet is expected to yield similar values. Noteworthy, a single channel DRAM connection translates to about 17GB/s (DDR3-1033). As we can see, the difference in terms of bandwidth between local and remote accesses is diminishing, in particular for accelerators located behind a PCIe interface. This is a huge opportunity to aggregate scarce resources by sharing at cluster level, to overcome capacity constraints and thus to fulfill the demands fed by Big Data [3].

As a result, the multi-/many-core revolution in combination with Big Data, technology diversity, energy constraints, the need for specialization and the diminishing disparities will lead to *a fundamental transition to communication-centric systems composed of heterogeneous computing units*.

3 An Optimized Data Movement Method for GPU Clusters

As it is unfeasible to explore all communication options for heterogeneous systems in a single work, we focus here on data movement optimizations for GPU clusters. GPUs differ significantly in their execution model from CPUs, and experiments have shown that GPUs perform quite poorly when initiating traditional communication methods directly and without any help of the CPUs.

If we shortly review the typical GPU architecture, we can find that it was already described pretty well by Valiant using the *Bulk-Synchronous Parallel (BSP)* model [4]. This model introduces a super-step that is composed of three phases for computation, communication, and synchronization, respectively. GPUs are a prime example for the parallel slackness that is leveraged to schedule and pipeline computation and communication efficiently. GPUs excel in performance if they perform their tasks in such a thread-collaborative way.

If GPUs are employed at cluster level for acceleration purposes, existing communication models do not match this BSP model, resulting in hybrid solutions like MPI+CUDA, MPI+OpenCL or similar. While the increase in complexity is obvious, communication performance can degrade, too. GPUs are not designed to excel in performance when controlling networking hardware, in particular regarding descriptor creation, network device interaction, and synchronization to ensure consistency. The programmer has to use the CPU for these tasks, resulting in highly heterogeneous code and frequent context switches between the GPU and CPU domains.

The GGAS research project [5] uses global address spaces at cluster level to facilitate a communication model that is in-line with the GPU's task model. In particular, GGAS relies on thread-collective communication, minimizes branch divergence, avoids costly context switches to the CPU domain, and allows for zero-copy data movements. Best to our knowledge, GGAS is still the only approach that is in-line with the GPU's thread-collaborative execution model.

Using GGAS, communication is performed by a set of threads collaboratively, by using loads or stores to remote memory. The modern GPUs memory respectively system interface coalesces these fine-grain accesses into bulk transfers. These accesses are intercepted by a network device with appropriate support. Such a support is typically not present in network devices, but only small modifications are required. The network device forwards the request to its destination that is identified based on the used address. From a GPU's point of view, differences between regular host memory accesses and accesses to remote locations are not visible. This maintains the commodity aspect of GPUs, and limits hardware changes to the network interface.

4 Results

We assessed the performance of GGAS using multiple workloads. In [5] we have shown performance results for the Himeno benchmark, which is a stencil code that solves a 3D Poisson equation in generalized coordinates on a structured curvilinear mesh. In [6] we have optimized reduce and allreduce operations for GGAS, also showing performance and energy improvements.

Here, we would like to shortly focus on a workload that puts a very high load on the network interface and the network itself. In the HPCC RandomAccess benchmark [7] a large table is distributed among the different nodes. Each node generates a random stream of XOR updates to be performed on the table. These updates can be coalesced into buckets of up to 1024 entries, each bucket for a certain destination. The updates are then send in a single data movement to the node responsible for the given table partition. This results in many small messages that are exchanged in a push-like manner, making this benchmark highly depending on sustained message rate [8].

While this benchmark is initially developed for CPUs, we re-implemented it for GPUs and used it to analyze performance for different communication models. These include GGAS, a Put/Get model and MPI as the state-of-the-art. We can see that GGAS outperforms MPI by 1.9x in terms of updates-per-second on a 4-node test system. At the same point, GGAS is about 5% faster than a GPU-driven Put/Get communication model.

Noteworthy, the code for GGAS is much simpler compared to MPI and Put/Get. The GGAS model with its thread-collaborative approach for data movement is very easy to understand for an experienced GPU programmer. Furthermore, context switches back to the CPU domain are avoided and control flow can be completely confined to the GPU domain.

5 Conclusion

We can see that performance is at least on-par for most workloads, and often outperforms traditional solutions like MPI+CUDA significantly. At the same time, programming for GPU clusters is dramatically simplified as no hybrid communication model has to be used. The GPU's beauty of simplicity can be maintained at cluster level, avoiding penalties in terms of performance, productivity and usability.

In the future we plan to assess performance with more workloads with varying characteristics like computational intensity, communication patterns and others. In particular we are interested in graph processing, like breadth-first search, strongly-connected-components, or betweenness centrality. We also think that integer workloads like sorts, scans or reductions are becoming increasingly important. We also plan for a GPU-tailored communication library that is based on GGAS, among others.

We plan to embrace future activities in the Alleycat project, which in particular gears to address complexity, usability and productivity while maintaining similar performance levels. Such an approach can be a key to foster the use of heterogeneous processing for other domains besides HPC, including data-warehousing, decision support, data mining, high performance analytics and more.

Acknowledgements. The author would like to thank Sudhakar Yalamanchili and Mark Hummel for the various insightful discussions. Also, Lena Oden and Benjamin Klenk have contributed with discussions to many insights and performed numerous experiments.

We also acknowledge the sponsoring and continuing support by Nvidia, Xilinx and EXTOLL.

References

1. Dally, B.: Gpu computing: To exascale and beyond (keynote). In: International Conference for High Performance Computing, Networking, Storage and Analysis, SC (2013)
2. Shalf, J., Dosanjh, S., Morrison, J.: Exascale computing technology challenges. In: Palma, J.M.L.M., Daydé, M., Marques, O., Lopes, J.C. (eds.) VECPAR 2010. LNCS, vol. 6449, pp. 1–25. Springer, Heidelberg (2011)
3. Yalamanchili, S.: Scaling Resource Compositions in a Flatter World. In: Workshop on Unconventional Cluster Architectures and Applications (UCAA), in conjunction with ICPP 2012, Pittsburgh, US (2012)
4. Valiant, L.G.: A bridging model for parallel computation. Communications of the ACM 33(8), 103–111 (1990)
5. Oden, L., Fröning, H.: GGAS: Global GPU Address Spaces for Efficient Communication in Heterogeneous Clusters. In: IEEE International Conference on Cluster Computing 2013, Indianapolis, US, September 23-27 (2013)
6. Oden, L., Klenk, B., Fröning, H.: On Achieving High Message Rates. In: 14th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), Chicago, IL, US, May 26-29 (2014)
7. Aggarwal, V., Sabharwal, Y., Garg, R., Heidelberger, P.: HPCC RandomAccess benchmark for next generation supercomputers. In: IEEE International Symposium on Parallel & Distributed Processing (IPDPS 2009), pp. 1–11. IEEE Computer Society, Washington, DC (2009)
8. Fröning, H., Nüssle, M., Litz, H., Leber, C., Brüning, U.: On Achieving High Message Rates. In: 13th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid), May 13-16, Delft, The Netherlands (2013)