Using GPI for low latency datatransfer on a GPU-Cluster

Lena Oden, Fraunhofer Institute

Abstract

Data transfer between multiple GPUs in a cluster is one of the main bottlenecks in GPU-based High performance computing. Especially codes with many short messages suffer from high latency of MPI for data transfer between GPUs. The Global address space Programming Interface (GPI) is an Industry-quality programming interface for PGAS (Partitioned global address space). As it has proven its superiority against MPI in many industrial and scientific applications in a CPU domain, we extend this concept to GPU domains to provide a high scalability and low latency communication interface for GPUs. GPI for GPUs supports the new GPU-Direct RDMA Technology, which allows direct data transfer between the device memories of two GPUs on different nodes by using RDMA-capable hardware. We improve the latency from 35us for data transfer between two GPUs with MPI to 3.3us with GPI for GPUs on an Infiniband Interconnect. The bandwidth could be improved from 2.2GB/s with MPI to up to 2.9 GB/s with GPI.

GPI for GPUs

Technologies

To create global memory segments on GPU-Device memory, the memory must be pinnable and accessible for RDMA-technique. Since no official release for GPU-Direct RDMA on Infiniband is available until now, we developed our own patch to use this technique. In previous releases of GPI, the shared part is one virtual continuous memory segment, which is created during setup and has the same size on all nodes. In future releases, which come along with the new GASPI (Global Address Space Programming Interface) standard, multiple shared memory segments on each node are supported. GPI for GPUs allows placing a shared memory segment either on host memory or on GPU-Device memory. Once a shared memory segment is created, it makes no difference, if it’s located on host memory or on device memory. This allows data transfer between all hosts and GPUs in the system with the same set of instructions.

Results

The results of a latency and bandwidth tests using GPU for GPUs are shown in Figure 4 and 5. In GPI, we developed two different protocols for data transfer between GPUs. Using a normal RDMAWrite instruction, the data are directly copied between two GPUs. In this case, the data transfer is completely handled by the devices. Still, this leads to a low bandwidth for big messages. Our special RDMAWrite operation uses buffered host copies on the local side for bigger message sizes. This leads to a better bandwidth, but on the writing side a GPU-Thread is required to control the data stream. In Figure 6 are the results of a stencil-benchmark are shown. A lower z-size means less computation, while the communication quantity stays constant.

Figure 1: The GPI programming Model

GPI is a PGAS interface for C, C++ and Fortran, which was developed at the Fraunhofer Institute for Industrial Mathematics (Fraunhofer ITWM). PGAS (Partitioned Global Address Space) is a programming model, where a global address space is created. This address space is logical partitioned and each part is local to one node. Still each thread on each node can directly read and write the parts of a remote node.

In GPI, remote read and write instructions are non-blocking and one-sided, which allows overlapping of communication and computation. By using one sided communication instead of message passing, communication overhead is minimized and data transfer latency can be reduced to a minimum.

Figure 2: The GPI-GPU programming Model

To extend the concept of GPI and PGAS to GPU memory, at least a thread for host memory or on device memory. This allows data transfer between all hosts and GPUs in the system with the same set of instructions.

Figure 3: GPU Direct Bandwidth and Latency

If a memory transfer requires the Infiniband HCA reading data from the GPU, the Bandwidth scalded down from 3.3 GB/s to maximum 1GB/s. Copying data from host memory to remote GPU memory results in the full Infiniband bandwidth, since in this case the Infiniband HCA only has to write data to the GPU.