Asynch_3d

This is the first release of the Asynch_3d proxy. This kernel is an attempt at establishing a small but meaningful proxy/benchmark for notification based 1-sided communication. Conceptually this kernel is an asynchronous, hybrid, 3d version of the Synch_p2p benchmark from Intel Parallel Research Kernels (PRK).

The kernel calculates a recursive stencil via pipelined parallelism. While this works surprisingly well for a shared memory architecture

```
  array_ELEM(i,j,k) = (array_ELEM(i-1,j,k)
                     + array_ELEM(i,j-1,k)
                     + array_ELEM(i,j,k-1)) * ONE_THIRD + 1;
```

the corresponding implementation for distributed memory systems is rather challenging.

Implementation details

The kernel is implemented as a hybrid MPI/GASPI/OpenMP Kernel. While the shared memory implementation parallelizes along the (j)-direction (with a single (i,k) plane per thread), the distributed memory implementation parallelizes along the (i)-direction. In both (i) and (j) directions the respective next neightbour elements can be processed, once the corresponding preceeding element has been calculated. If the resulting pipeline is completely filled all calculations will happen in parallel.

We note that this implies that there will be substantial load imbalance in the beginning and in the end of this calculation.

We also note that the first thread on the first rank has no dependencies to (i-1,j-1) and hence can progress completely asynchronously. Both the shared memory implementation and the GASPI implementation make use of the resulting asynchronous pipeline. Whereas in the latter case GASPI notificatios are used, the shared memory implementation uses volatile thread counters (flush before read, flush after write). It is not quite clear how this concept of asynchronous pipelining can be mapped to MPI-3. Correspondingly our MPI implementation uses conventional MPI_Send/MPI_Recv pairs.

Results

We have evaluated the proxy on a Infiniband FDR Ivy Bridge cluster and on Intel Xeon Phi in a weak scaling scenario, where we have used 12/60 threads/cores per socket respectively with up to 4 sockets/Xeon Phi cards. Given the fact, that this is a recursive kernel, both shared memory implementation and the hybrid GASPI implementation scale rather well. The MPI implementation is clearly limited by the fact that it can not leverage this asynchronous pipelining mechanism. We note that finding scalable grid dimensions is non-trivial – in addition to the usual side effects related to cache blocking, TLB misses, etc, there is also a strong impact on node-local load imbalance (between threads) and on load balance between the ranks. (if. e.g the (i) dimension is too small, the first rank will have completed the entire local grid before the neighbouring rank has even started. In that case we do get our desired pipelining effect, but zero scalability)

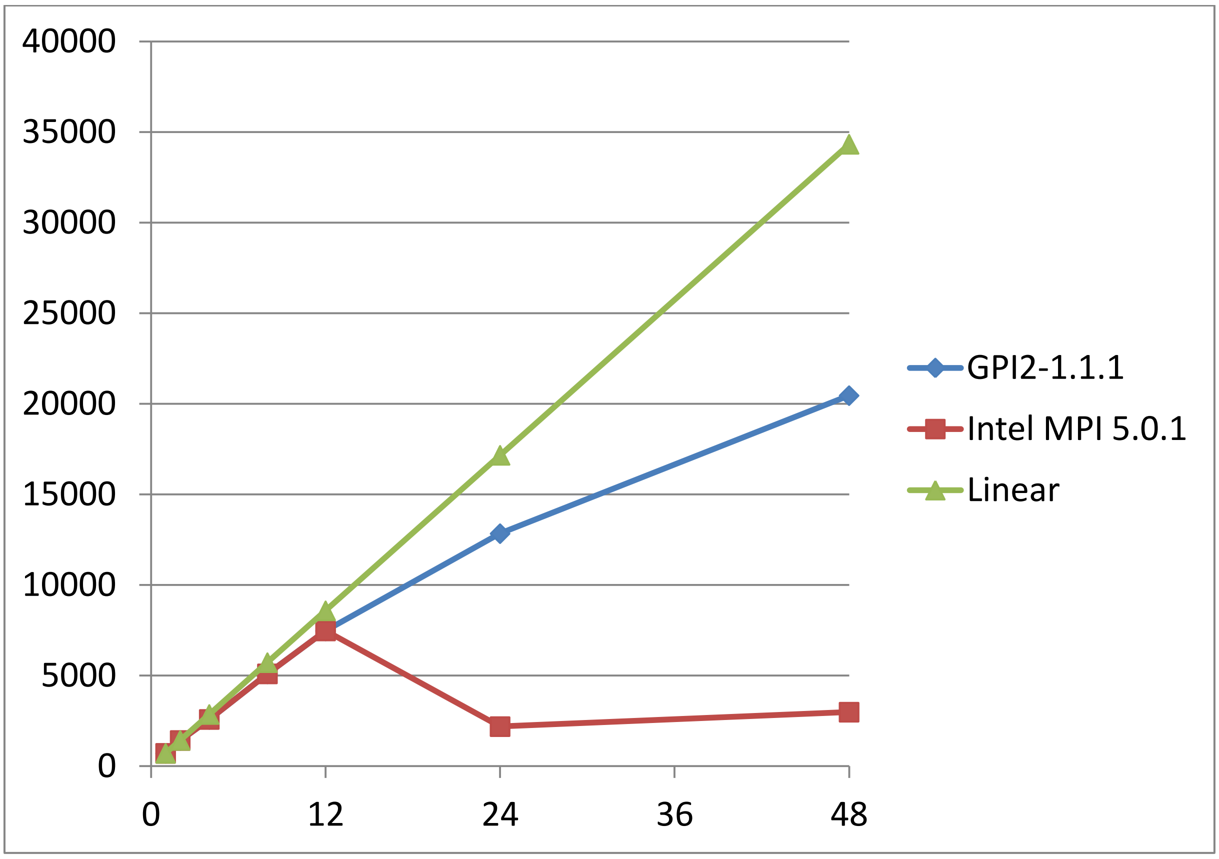

Figure 1. Speedup (Mflop/sec) vs. number of cores, 12 threads/cores per socket, 4 sockets. Grid dimension 2048x12x2048.

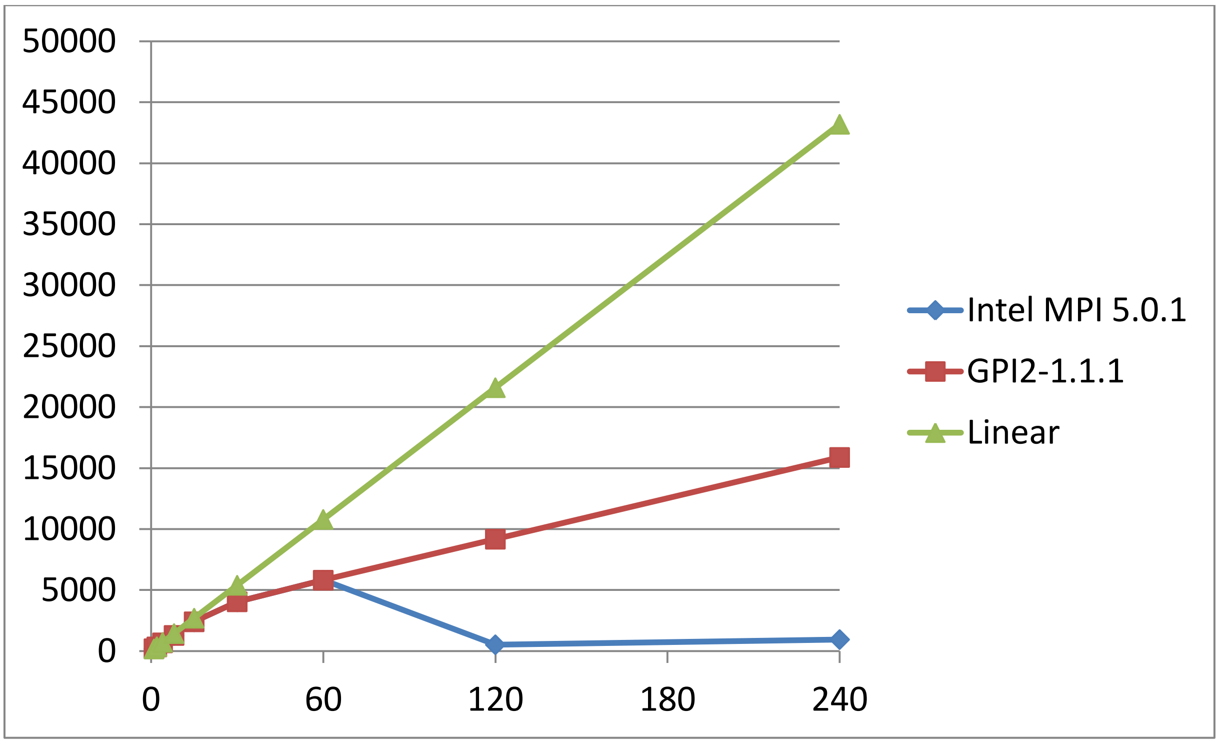

Figure 2. Speedup (Mflop/sec) vs. number of cores, 60 threads/cores per Xeon Phi, 4 Xeon Phi Cards. Grid dimension 16384x60x64.

Community involvement

We encourage the HPC community to provide new patterns or improve existing ones. No restrictions apply for either communication API or programming languages used in these improved patterns. Bonus points are granted for highly scalable implementations which also feature better programmability.

We are happy to include all those patterns/implementations in this Asynch_3d Release. We especially are interested in an MPI implementaion which *does* allow to use this form of asynchronous pipeline parallelism.

Related Documentation

MPI
http://www.mpi-forum.org/docs/mpi-3.0/mpi30-report.pdf (MPI API)
http://www.open-mpi.org/ (MPI Release)
http://mvapich.cse.ohio-state.edu/news/https://computing.llnl.gov/tutorials/mpi/ (Tutorial)

GASPI
http://www.gpi-site.com/gpi2/gaspi/ (GASPI API)
https://github.com/cc-hpc-itwm/GPI-2 (GPI2 release, implements GASPI)
https://github.com/cc-hpc-itwm/GPI-2/tree/next/tutorial (Tutorial)