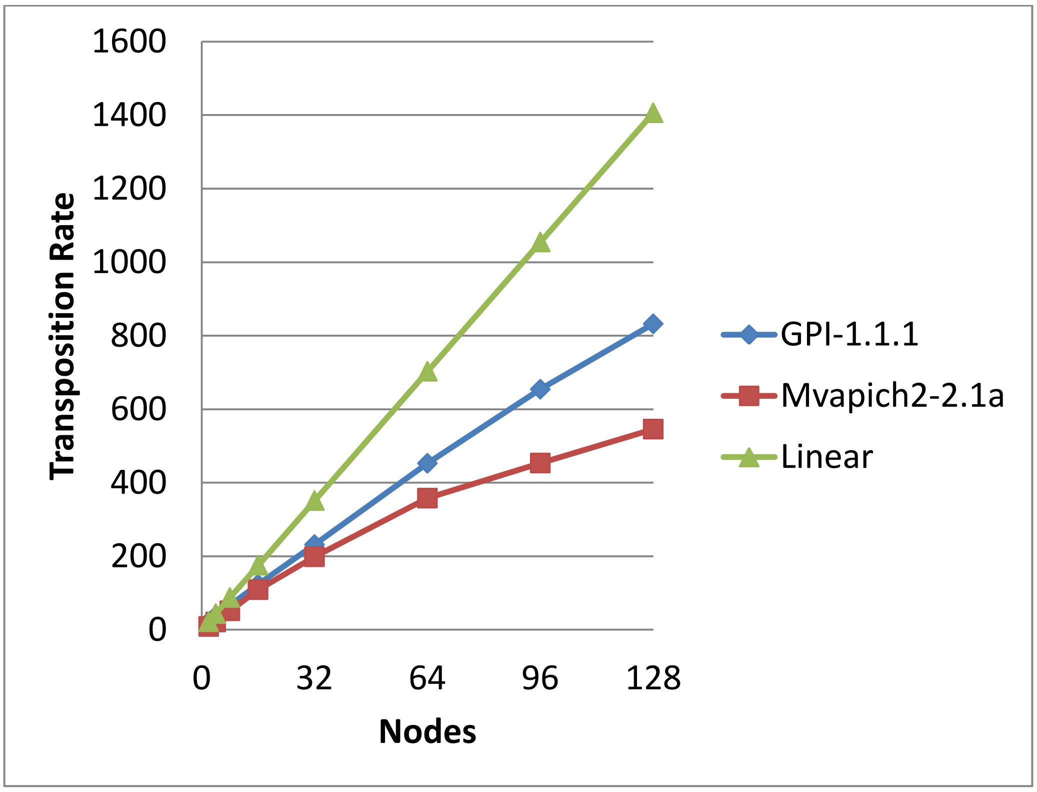
Pipelined Transpose

This is the initial release of a pipelined transpose proxy. This kernel is an attempt at establishing a small but meaningful proxy/benchmark for notification based 1-sided communication.

The kernel calculates a global transpose of a matrix for a column-based matrix distribution. This is a hybrid implementation where a global transpose is followed by a local transpose. While the MPI implementation uses an MPI alltoall for global communication, the GASPI implementation uses a single communication step, in which all required communication to all target ranks is issued in a single loop. The notification based one sided communication requests are scheduled (target rank sequence) such that we minimize network congestion, i.e. we always aim for bidirectional communication even though the actual communication is entirely one-sided.

GASPI here can (and does) leverage pipelining, namely the overlap of a local transpose with the communication of the global transpose. To that end GASPI uses the concept of notifications which allows for a fine-grain overlap of computation of local transposition (per source rank) with the global transposition. In GASPI notifications are true (epoch free) one-sided puts, where notifications are bundled with the put such that at the target we can test for completion of individual puts.

We note that this (fairly naïve) implementation appears to scale higher than a corresponding existing MPI implementation which make use of MPI_Alltoall.



Strong scaling for Pipelined Transpose (12288x12288 matrix), 24 Threads/Cores per node.

We also note that MPI potentially allows for (coarse grain) pipelining via tiling of the matrix and testing/waiting for multiple corresponding non-blocking alltoall collectives. For pipelining a simple local transpose and for a strong scaling scenario however it is not quite clear how this concept can be implemented efficiently. For comparison: To achieve the same fine-grain pipeline granularity (N-1, where N == num procs) as our simple GASPI implementation, MPI would require N-1 simultaneous alltoall communications, which is clearly not feasible. Reducing N to the number of threads T in our hybrid environment will not solve the problem either – especially since in strong scaling the tile size will shrink with N^2 and additional tiling would accelerate this effect.

The GASPI implementation allows for fine-grain pipelining per source rank, does not require communication tiling and keeps communication buffers at their maximal size.

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.

Even though we are a bit sceptical with respect to tiling on a communication level we remain especially interested in implementations which can leverage tiling for communication (in whatever form) such that tiling yields corresponding performance gains.

We are happy to include all those patterns/implementations in this Pipelined_transpose Release.

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)