MPI for multi-core, multi socket, and GPU architectures: Optimised shared memory allreduce

Andreas Jocksch[1] and Jean-Guillaume Piccinali[1]

[1]CSCS, Swiss National Supercomputing Centre, Via Trevano 131

6900 Lugano, Switzerland

Motivation

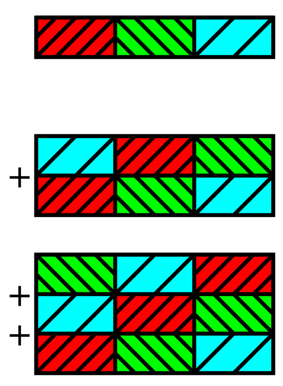
- Today's supercomputers have a growing number of cores per socket and more and more sockets per node
- Intranode communication needs to be efficient also as part of more complex internode communication
- MPI persistent collective communication [1, 2] provides new interface
- Our focus is on reduction operations allreduce (and reduce_scatter)

Contributions

- Utilisation of a shared memory segment [3] invisible to the user
- Copy in with reduction algorithm with chunks of the total message
- Reduction in shared memory using a tree algorithm with integrated barrier [4]
- Consideration of multiple sockets per node but also multiple GPUs

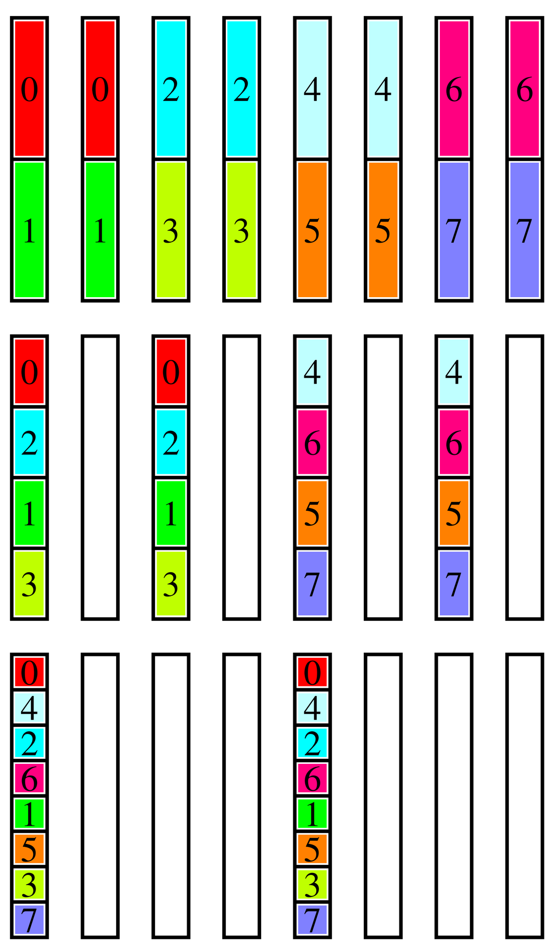
Algorithms

- Copy in



Copy in (top), first reduction (middle), and second reduction (bottom), + equals reduction operator, colours indicate the different MPI tasks, horizontal data vectors

- Tree reduction



Tree reduction in shared memory, colours (and numbers 0-7) indicate the different MPI tasks, vertical data vectors

- Algorithm corresponds to matrix vector multiplication (GEMV) with a vector of only unity entries

Implementation and tuning

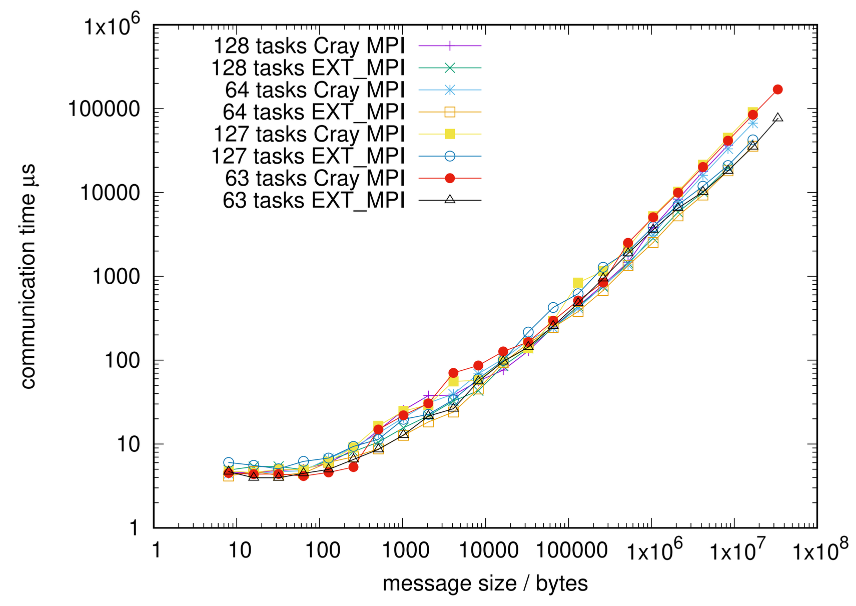
- For short messages, data and barrier flags on the same cache line
- Non-binary tree for non 2^n tasks, first reduction step which reduces to 2^n tasks, or no special treatment for short messages
- Benchmark for best algorithm at initialisation time: MPI_Allreduce_init
- Bytecode generated in initialisation phase for repeated execution
- Multiple sockets per node implemented with one shared memory segment per socket
- Prototype library which implements part of the persistent collective communication of the MPI 4.0 standard and blocking collective communication

`https://github.com/eth-cscs/ext_mpi_collectives`

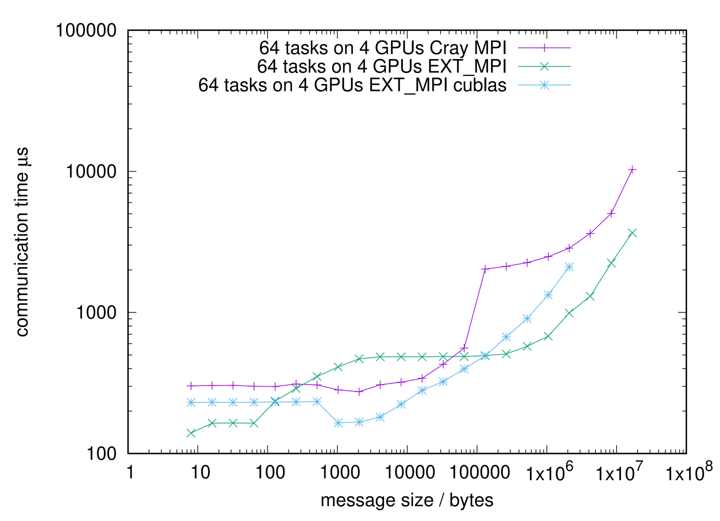
GPU support

- One CUDA-kernel for multiple reductions and copy operations
- Alternatively call to cublas matrix-vector multiplication

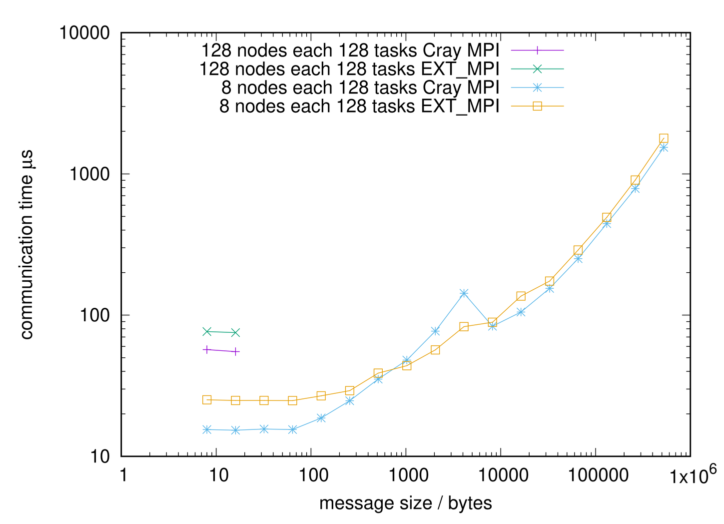
Benchmarks



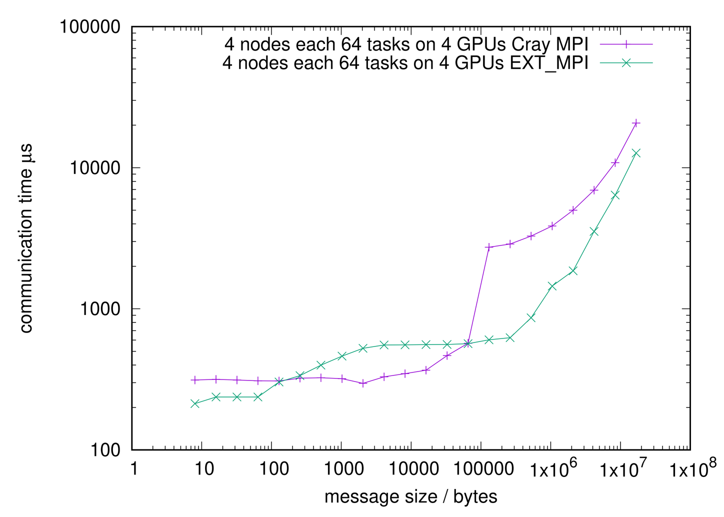
Allreduce, 128 MPI tasks on one node with two AMD EPYC 7742 CPUs, HPE (Cray) MPI and EXT_MPI



Allreduce, 64 MPI tasks on one node with four NVIDIA A100, HPE (Cray) MPI and EXT_MPI



Allreduce, 8 and 128 nodes each with 128 MPI tasks on two AMD EPYC 7742 CPUs, HPE Slingshot network, HPE (Cray) MPI and EXT_MPI



Allreduce, 4 nodes each with 64 MPI tasks on four NVIDIA A100, HPE Slingshot network, HPE (Cray) MPI and EXT_MPI

- On the CPU our routines mostly outperform the reference library, HPE(Cray) MPI
  - All algorithmic options are used in the benchmark
  - For short messages one shared memory segment is chosen (node is assumed to be one socket), for long messages multiple ones (CPU and GPU)
  - For 127 tasks (prime number) not ideal performance, further tuning required
- Problem: MPI point-to-point communication is slow with shared memory (shmget) therefore low performance of our library for multiple nodes with multiple MPI tasks per node

Conclusions

- For single node CPU communication and single or multiple node GPU communication very efficient implementation
- The persistent collective MPI communication interface allows for highly optimised algorithms
- Our shared memory algorithm could be used for the blas level 1 operation GEMV for speeding it up with multiple MPI tasks per node

Future work

- Pipelining for overlap between computation and communication (NCCL provides this feature).
- Comparison with NCCL

References

[1] Bouhrour, S., Pepin, T., Jaeger, J.: Towards leveraging collective performance with the support of MPI 4.0 features in MPC. Parallel Computing 109, 102860 (2022)

[2] Jocksch, A., Ohana, N., Lanti, E., Koutsaniti, E., Karakasis, V., Villard, L.: An optimisation of allreduce communication in message-passing systems. Parallel Computing 107, 102812 (2021)

[3] Li, S., Hoefler, T., Hu, C., Snir, M.: Improved MPI collectives for MPI processes in shared address spaces. Cluster computing 17(4), 1139–1155 (2014)

[4] Mohamed El Maarouf, A.K., Giraud, L., Guermouche, A., Guignon, T.: Combining reduction with synchronization barrier on multi-core processors. Concurrency and Computation: Practice and Experience p. e7402 (2023)