**Motivation**

- Today’s supercomputers have a growing number of cores per socket and more and more sockets per node.
- Intra-node communication needs to be efficient also as part of more complex inter-node 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.

**Implementations 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_int.
- 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.

**Benchmarks**

- 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.

**GPU support**

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

**References**