Message Passing Interface
Lecture 3

Manfred Liebmann
Technische Universität München
Chair of Optimal Control
Center for Mathematical Sciences, M17
manfred.liebmann@tum.de

November 24, 2015

MPI Communicators and Groups

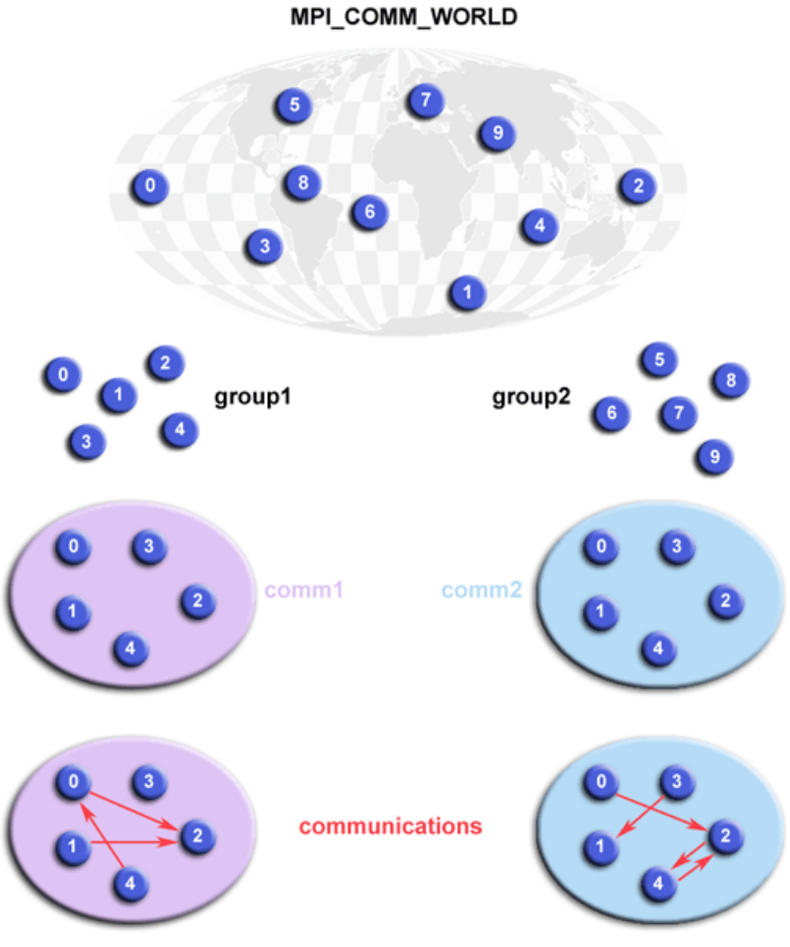

Figure 1: MPI communicators and groups

One-Sided Communication

Only one process is involved actively in the communication. Put and Get operations must be synchronized with a fence. Data is written to or read from the foreign address space defined by the window.

- One-sided communication
  - MPI_Win_create, MPI_Win_free (window into remote address space)
  - MPI_Win_fence (synchronize)
  - MPI_Put (writes into remote memory)
  - MPI_Get (reads from remote memory)
  - MPI_Accumulate (updates remote memory)

Scalable synchronization with MPI_Win_start, MPI_Win_complete and MPI_Win_post, MPI_Win_wait. Advanced Post/Start/Complete/Wait synchronization model.

MPI Parallel File I/O

```
#include <mpi.h>
int main(int argc, char** argv)
{
    MPI_Init(&argc, &argv);

    int rank, size;
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &size);

    MPI_File fh;
    MPI_Status status;

    int n = 256, dsp = (rank * n) / size, cnt = n / size, buf[cnt];

    MPI_File_open(MPI_COMM_WORLD, "data.bin", MPI_MODE_CREATE | MPI_MODE_WRONLY, MPI_INFO_NULL, &fh);
    MPI_File_write_at(fh, dsp * sizeof(int), buf, cnt, MPI_INT, &status);
    MPI_File_close(&fh);

    MPI_File_open(MPI_COMM_WORLD, "data.bin", MPI_MODE_RDONLY, MPI_INFO_NULL, &fh);
    MPI_File_read_at(fh, dsp * sizeof(int), buf, cnt, MPI_INT, &status);
    MPI_File_close(&fh);

    MPI_Finalize();
    return 0;
}
```