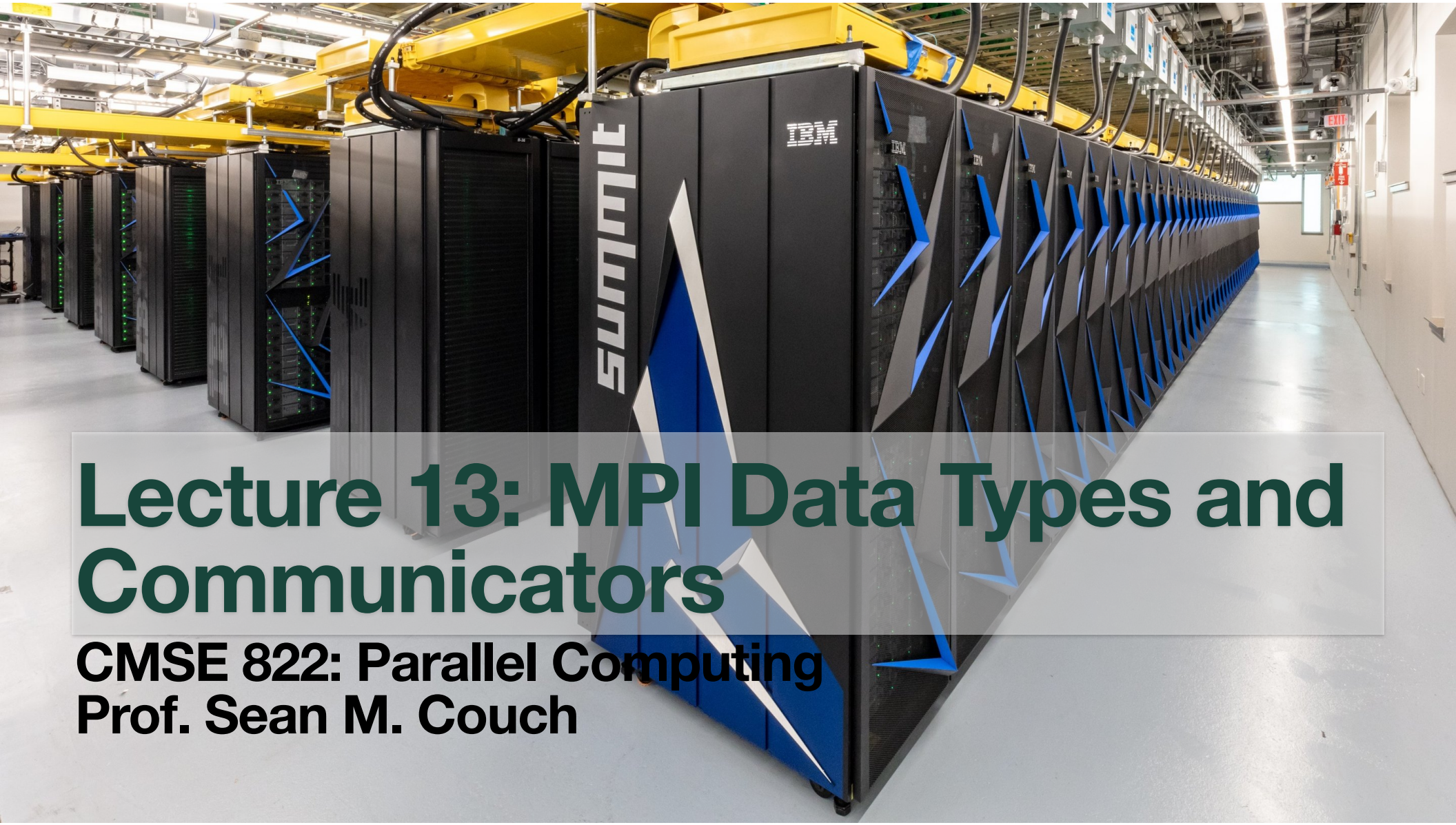


Lecture 13: MPI Data Types and Communicators

CMSE 822: Parallel Computing

Prof. Sean M. Couch

Puppy Time!

MPI Groups and Communicators

- See Chapter 6 of “Parallel Programming in Science and Engineering.”
- See also LLNL MPI Tutorial.

MPI Groups and Communicators

Groups vs. Communicators:

- A group is an ordered set of processes. Each process in a group is associated with a unique integer rank. Rank values start at zero and go to N-1, where N is the number of processes in the group. In MPI, a group is represented within system memory as an object. It is accessible to the programmer only by a "handle". A group is always associated with a communicator object.

- A communicator encompasses a group of processes that may communicate with each other. All MPI messages must specify a communicator. In the simplest sense, the communicator is an extra "tag" that must be included with MPI calls. Like groups, communicators are represented within system memory as objects and are accessible to the programmer only by "handles". For example, the handle for the communicator that comprises all tasks is MPI_COMM_WORLD.

- From the programmer's perspective, a group and a communicator are one. The group routines are primarily used to specify which processes should be used to construct a communicator.

MPI Groups and Communicators

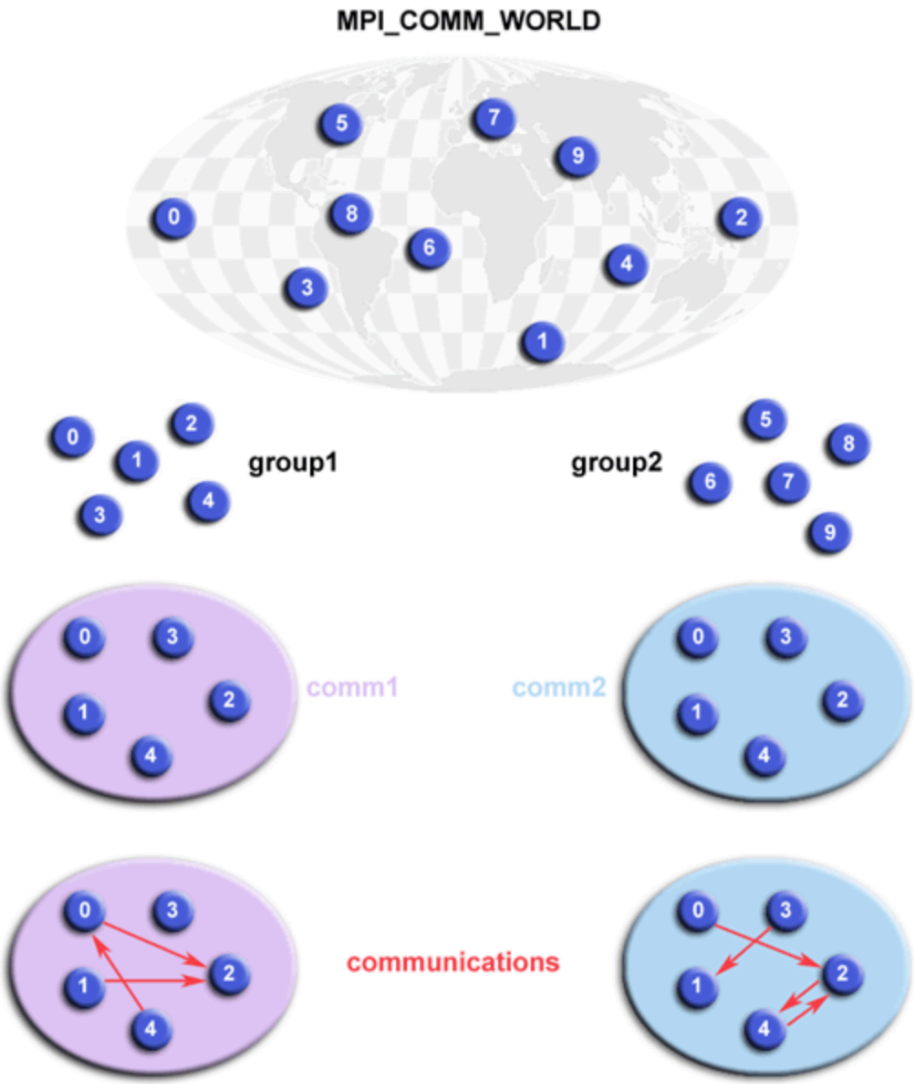
- **Primary Purposes of Group and Communicator Objects:**

1. Allow you to organize tasks, based upon function, into task groups.
2. Enable Collective Communications operations across a subset of related tasks.
3. Provide basis for implementing user defined virtual topologies
4. Provide for safe communications

MPI Groups and Communicators

Programming Considerations and Restrictions:

- Groups/communicators are dynamic - they can be created and destroyed during program execution.
- Processes may be in more than one group/communicator. They will have a unique rank within each group/communicator.
- MPI provides over 40 routines related to groups, communicators, and virtual topologies.
- Typical usage:
  1. Extract handle of global group from MPI_COMM_WORLD using MPI_Comm_group
  2. Form new group as a subset of global group using MPI_Group_incl
  3. Create new communicator for new group using MPI_Comm_create
  4. Determine new rank in new communicator using MPI_Comm_rank
  5. Conduct communications using any MPI message passing routine
  6. When finished, free up new communicator and group (optional) using MPI_Comm_free and MPI_Group_free



- More useful, perhaps, is “splitting” an existing communicator
- See Section 6.4 in “Parallel Programming.”

C Language - Group and Communicator Example

```c
#include "mpi.h"
#include <stdio.h>
#define NPROCS 8

main(int argc, char *argv[])  {
int        rank, new_rank, sendbuf, recvbuf, numtasks,
           ranks1[4]={0,1,2,3}, ranks2[4]={4,5,6,7};
MPI_Group  orig_group, new_group;   // required variables
MPI_Comm   new_comm;   // required variable

MPI_Init(&argc,&argv);
MPI_Comm_rank(MPI_COMM_WORLD, &rank);
MPI_Comm_size(MPI_COMM_WORLD, &numtasks);

if (numtasks != NPROCS) {
  printf("Must specify MP_PROCS= %d. Terminating.\n",NPROCS);
  MPI_Finalize();
  exit(0);
  }

sendbuf = rank;

// extract the original group handle
MPI_Comm_group(MPI_COMM_WORLD, &orig_group);

//  divide tasks into two distinct groups based upon rank
if (rank < NPROCS/2) {
  MPI_Group_incl(orig_group, NPROCS/2, ranks1, &new_group);
  }
else {
  MPI_Group_incl(orig_group, NPROCS/2, ranks2, &new_group);
  }

// create new new communicator and then perform collective communications
MPI_Comm_create(MPI_COMM_WORLD, new_group, &new_comm);
MPI_Allreduce(&sendbuf, &recvbuf, 1, MPI_INT, MPI_SUM, new_comm);

// get rank in new group
MPI_Group_rank (new_group, &new_rank);
printf("rank= %d newrank= %d recvbuf= %d\n",rank,new_rank,recvbuf);

MPI_Finalize();
}
```

MPI Data Types

- See Chapter 5 of “Parallel Programming in Science and Engineering.”

MPI Built-in Types

C Data Types		Fortran Data Types	
MPI_CHAR	char	**MPI_CHARACTER**	character(1)
MPI_WCHAR	wchar_t - wide character		
MPI_SHORT	signed short int		
MPI_INT	signed int	**MPI_INTEGER** **MPI_INTEGER1** **MPI_INTEGER2** **MPI_INTEGER4**	integer integer*1 integer*2 integer*4
MPI_LONG	signed long int		
MPI_LONG_LONG_INT **MPI_LONG_LONG**	signed long long int		
MPI_SIGNED_CHAR	signed char		
MPI_UNSIGNED_CHAR	unsigned char		
MPI_UNSIGNED_SHORT	unsigned short int		
MPI_UNSIGNED	unsigned int		
MPI_UNSIGNED_LONG	unsigned long int		
MPI_UNSIGNED_LONG_LONG	unsigned long long int		

MPI Built-in Types

C Data Types		Fortran Data Types	
MPI_FLOAT	float	**MPI_REAL** **MPI_REAL2** **MPI_REAL4** **MPI_REAL8**	real real*2 real*4 real*8
MPI_DOUBLE	double	**MPI_DOUBLE_PRECISION**	double precision
MPI_LONG_DOUBLE	long double		
MPI_C_COMPLEX **MPI_C_FLOAT_COMPLEX**	float _Complex	**MPI_COMPLEX**	complex
MPI_C_DOUBLE_COMPLEX	double _Complex	**MPI_DOUBLE_COMPLEX**	double complex
MPI_C_LONG_DOUBLE_COMPLEX	long double _Complex		
MPI_C_BOOL	_Bool	**MPI_LOGICAL**	logical
MPI_INT8_T **MPI_INT16_T** **MPI_INT32_T** **MPI_INT64_T**	int8_t int16_t int32_t int64_t		

MPI Built-in Types

C Data Types		Fortran Data Types	
MPI_UINT8_T **MPI_UINT16_T** **MPI_UINT32_T** **MPI_UINT64_T**	uint8_t uint16_t uint32_t uint64_t		
MPI_BYTE	8 binary digits	**MPI_BYTE**	8 binary digits
MPI_PACKED	data packed or unpacked with MPI_Pack()/ MPI_Unpack	**MPI_PACKED**	data packed or unpacked with MPI_Pack()/ MPI_Unpack

Notes:

- Programmers may also create their own data types (see Derived Data Types).
- MPI_BYTE and MPI_PACKED do not correspond to standard C or Fortran types.
- Types shown in **GRAY FONT** are recommended if possible.
- Some implementations may include additional elementary data types (MPI_LOGICAL2, MPI_COMPLEX32, etc.). Check the MPI header file.

MPI Derived Data Types

- MPI also provides facilities for you to define your own data structures based upon sequences of the MPI primitive data types. Such user defined structures are called derived data types.
- Primitive data types are contiguous. Derived data types allow you to specify non-contiguous data in a convenient manner and to treat it as though it was contiguous.
- MPI provides several methods for constructing derived data types:
  - Contiguous
  - Vector
  - Indexed
  - Struct

Group Exercises

- Read through Homework 7 README
- Discuss prompts in your groups and complete the exercises