LSU

LSU INFORMATION TECHNOLOGY SERVICES

Introduction to MPI Programming – Part 3

LSU CENTER FOR COMPUTATION & TECHNOLOGY

Outline

- Communicators
- Virtual topology

Communicators

- MPI_COMM_WORLD
  - Default communicator
  - Contains all processes
- Additional communicators
  - Useful when communications need to occur
    - Among a subset of the processes
    - In a specific pattern

Multiple Communicators

Communicator	P1	P2	P3	P4	P5	P6	P7	P8
MPI_COMM_WORLD	0	1	2	3	4	5	6	7
NewComm1					0	1	2	3
NewComm2	0		1		2		3	

Suppose a program is launched on Queen Bee with 8 processes, and two new communicators are created

Creating New Communicators

- Create from a process group
  - MPI provides functions to map communicators to groups and manipulate groups
- Split an existing communicator into sub-communicators

Creating Communicators from Groups

- Two or three steps
  - Map the old communicator to a group
    - `MPI_Comm_Group(Oldcomm,Oldgroup)`
  - Modify the group
    - Operations can be inclusion, exclusion, intersect, union etc.
  - Create a new communicator from the modified group
    - `MPI_Comm_Create(Oldcomm,Newgroup,Newcomm)`

Group Management Functions

- Create a new group out of two groups
  - `MPI_Group_union(Group1, Group2, Newgroup)`
  - `MPI_Group_intersection(Group1, Group2, Newgroup)`
  - `MPI_Group_difference(Group1, Group2, Newgroup)`
- Create a new group from one group
  - `MPI_Group_incl(Group, n, ranks[], Newgroup)`
    - `Newgroup` has the n members of `Group` specified by `ranks[]`
  - `MPI_Group_excl(Group, n, ranks[], Newgroup)`
    - `Newgroup` has all members of `Group` except the n members specified by `ranks[]`

Example: Create A New Communicator

```
! Find rank in the old communicator
call mpi_comm_rank(mpi_comm_wold,myoldrank,ierr)

! Map MPI_COMM_WORLD to the group “oldgroup”
call mpi_comm_group(mpi_comm_world,oldgroup,ierr)

! Create a new group which includes all processes but process 0 in the
old group
rank_excl=0
call mpi_group_excl(oldgroup,1,rank_excl,newgroup,ierr)

! Create a new communicator
call mpi_comm_create(mpi_comm_world,newgroup,newcomm,ierr)
call mpi_comm_rank(newcomm,newrank,ierr)
write(*,*) “My new rank is”,newrank
```

- What the screen output will be?

Example: Create A New Communicator

```
! Find rank in the old communicator
call mpi_comm_rank(mpi_comm_wold,myoldrank,ierr)

! Map MPI_COMM_WORLD to the group “oldgroup”
call mpi_comm_group(mpi_comm_world,oldgroup,ierr)

! Create a new                                          rocess 0 in the
old group
rank_excl=0
call mpi_group

! Create a new
call mpi_comm_                                          rr)
call mpi_comm_rank(newcomm,newrank,ierr)
write(*,*) “My new rank is”,newrank
```

```
[lyan1@qb563 ex]$ mpirun -np 4 ./a.out
0 - MPI_COMM_RANK : Null communicator
[0] [] Aborting Program!
           2           1
           3           2
           1           0
MPI process terminated unexpectedly
```

Example: Create A New Communicator

```
! Find rank in the old communicator
call mpi_comm_rank(mpi_comm_wold,myoldrank,ierr)

! Map MPI_COMM_WORLD to the group “oldgroup”
call mpi_comm_group(mpi_comm_world,oldgroup,ierr)

! Create a new group which includes all processes but process 0 in the
old group
rank_excl=0
call mpi_group_excl(oldgroup,1,rank_excl,newgroup,ierr)

! Create a new communicator
If (myoldrank.ne.rank_excl) then
call mpi_comm_create(mpi_comm_world,newgroup,newcomm,ierr)
call mpi_comm_rank(newcomm,newrank,ierr)
write(*,*) “My new rank is”,newrank
```

- The excluded process is not a member of the new group, so newcomm is actually a null communicator for it

Splitting An Existing Communicator

- Syntax: `MPI_Comm_split(Oldcomm, color, key, newcomm)`
  - Partition the group associated with `Oldcomm` into disjoint subgroups, one for each value of color
  - Each subgroup contains all processes of the same `color`
  - Within each subgroup, the processes are ranked in the order defined by the value of key, with ties broken according to their rank in the old group
  - A new communicator `newcomm` is created for each subgroup

Example: MPI_COMM_SPLIT

```
! Find rank in the old communicator
call mpi_comm_rank(mpi_comm_wold,myoldrank,ierr)

color=mod(myoldrank,2)
! Split MPI_COMM_WORLD
call mpi_comm_split(mpi_comm_world,color,myoldrank,newcomm,ierr)
! Find rank in the new communicator
call mpi_comm_rank(newcomm,mynewrank,ierr)

write(*,*) myoldrank,mynewrank
```

- What the output will be if this program is run with 4 processes?

Example: MPI_COMM_SPLIT

```
! Find rank in the old communicator
call mpi_comm_rank(mpi_comm_wold,myoldrank,ierr)

color=mod(myoldrank,2)
! Split MPI_COMM_WORLD
call mpi_comm_split(mpi_comm_worl
! Find rank in the new communicato
call mpi_comm_rank(newcomm,mynewra

write(*,*) myoldrank,mynewrank
```

```
[lyan1@qb563 ex]$ mpirun -np 4 ./a.out
           0           0
           3           1
           1           0
           2           1
```

- What the output will be if this program is run with 4 processes?

Example: MPI_COMM_SPLIT

```
! Find rank in the old communicator
call mpi_comm_rank(mpi_comm_wold,myoldran

color=mod(myoldrank,2)
! Split MPI_COMM_WORLD
call mpi_comm_split(mpi_comm_world,color,
! Find rank in the new communicator
call mpi_comm_rank(newcomm,mynewrank,ierr

write(*,*) myoldrank,mynewrank

call mpi_bcast(a,1,mpi_integer,0,newcomm,ierr)
```

Rank in MPI_COMM_WORLD	Value of a	
	Before	After
0	2	?
1	4	?
2	6	?
3	8	?

- What will be the outcome of this MPI_BCAST call (with 4 processors)?

Example: MPI_COMM_SPLIT

```
! Find rank in the old communicator
call mpi_comm_rank(mpi_comm_wold,myoldranl

color=mod(myoldrank,2)
! Split MPI_COMM_WORLD
call mpi_comm_split(mpi_comm_world,color,r
! Find rank in the new communicator
call mpi_comm_rank(newcomm,mynewrank,ierr

write(*,*) myoldrank,mynewrank

call mpi_bcast(a,1,mpi_integer,0,newcomm,ierr)
```

Rank in MPI_COMM_WORLD	Value of a	
	Before	After
0	2	2
1	4	4
2	6	2
3	8	4

- What will be the outcome of this MPI_BCAST call (with 4 processors)?

CENTER FOR COMPUTATION & TECHNOLOGY

Virtual Topology

- Situations that call for a ranking system beyond the linear one we have seen
  - When solving problems in parallel we often have 2-d or 3-d decomposition, then assign a sub-domain to each process
    - A process grid needs to be built to represent this layout
    - Using a multi-dimensional rank system would be more natural and clear
  - The parallel algorithm being implemented has an intrinsic topology, e.g. the tree structure used in parallel sorting

MPI Topology Functions

- MPI provides two types of topologies
  - Cartesian
  - Graph
- The topological information is stored with the communicator
  - Note the difference between communicator and group: we can create a communicator using a 2-d Cartesian topology that contains all processes – it has the same group of processes as MPI_COMM_WORLD

Creating a Cartesian Topology

- Syntax: `MPI_Cart_Create(Oldcomm, Ndims, Dim_size[], Periods[], Reorder, Newcomm)`
  - `Ndims`: number of dimensions
  - `Dim_size`: array of dimension size
  - `Periods`: array specifying periodicity of each dimension
  - `Reorder`: Whether or not reorder the processes

Query Topological Information

- MPI_Cart_Coords
  - Finds the coordinates of a process given its rank
- MPI_Cart_Rank
  - Finds the ranks of a process given its coordinates
- MPI_Cart_Get
  - Get information about a Cartesian grid
- MPI_Cart_Shift
  - Finds the resulting source and destination ranks, given a shift direction and amount
  - Can be used to get the (linear) rank of neighboring processes

Example: Setting up a 2-d Process Grid

```
! Describe the process grid
ngriddim=2
griddims(1)=4; griddims(2)=nprocs/griddims(1)
period(0)=.true.; period(1)=.true.
reorder=.false.

! Create the Cartesian grid
call mpi_cart_create(mpi_comm_world,ngriddim,griddims, &
                     period,reorder,gridcomm,ierr)

! Get the coordinates of a given rank
call mpi_comm_rank(gridcomm,myrank,ierr)
call mpi_cart_coords(gridcomm,myrank,2,mycoord,ierr)

! Set up the stencil
call mpi_cart_shift(gridcomm,0,1,above,below,ierr)
call mpi_cart_shift(gridcomm,1,1,left,right,ierr)
```

Exercise 5: Laplace Solver

- Goal: rewrite the 2-d version using a Cartesian topology

References

- MPI Standard (http://www.mcs.anl.gov/research/projects/mpi)
- Online tutorials
  - NCSA *Introduction to MPI* online course (http://www.citutor.org)
  - LLNL MPI tutorial page
- *Practical MPI Programming* (IBM Redbook)
- *Using MPI* by William Gropp, Ewing L. Lusk, Anthony Skjellum and Rajeev Thakur