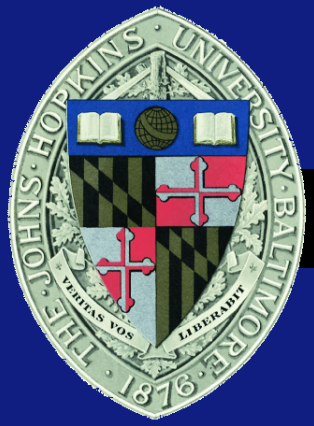
Lecture 6.3
MPI Collective Operations

EN 600.320/420

Instructor: Randal Burns

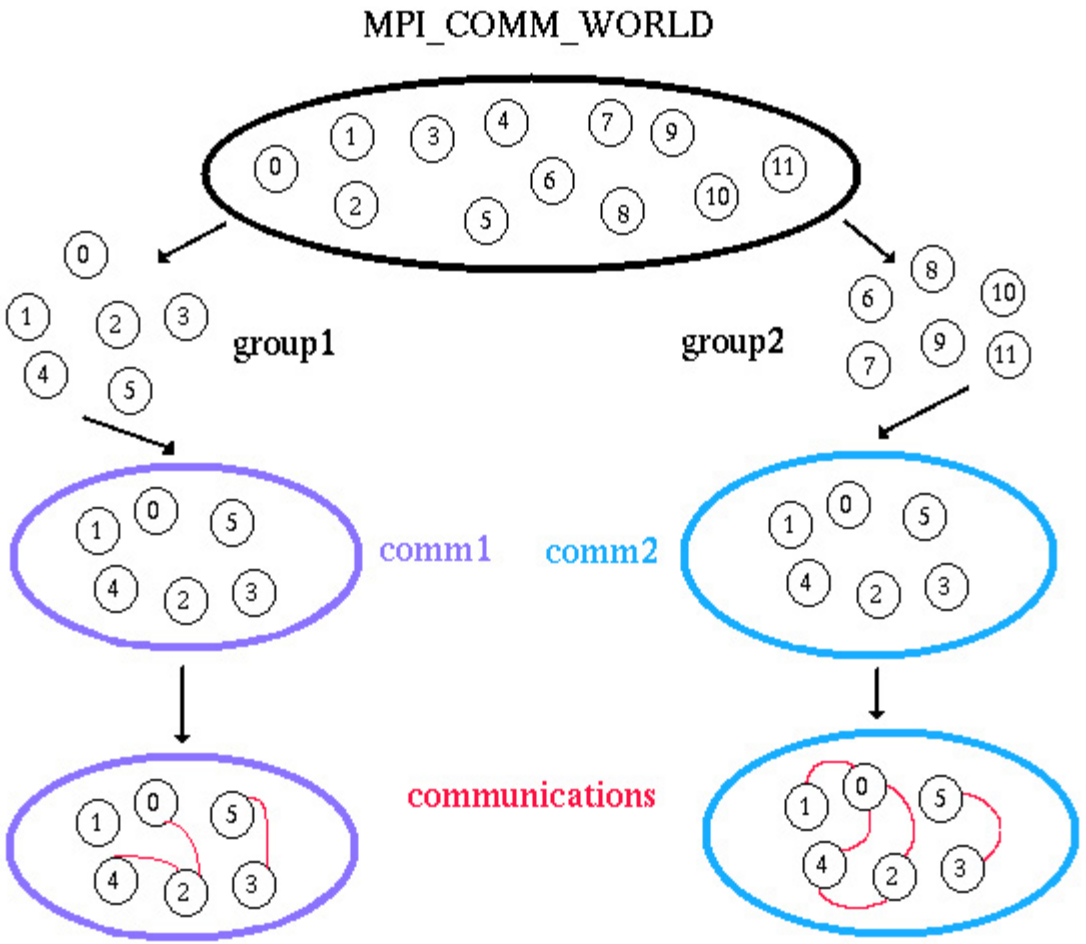
15 February 2017



Department of Computer Science, *Johns Hopkins University*

Process Groups

- Ordered group of processes
- Scope communication for collective and point to point operations
- Defined dynamically (at runtime)



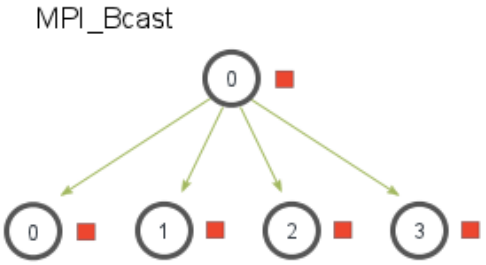
https://www.msi.umn.edu/content/mpi-group-management-communicator

Collective Operations

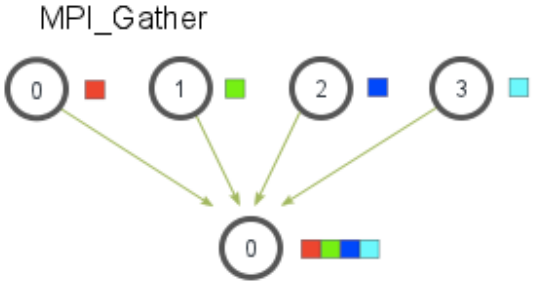
- Collective = uses all processes to accomplish the task
  - i.e. the whole communicator
- Collective allows the runtime to optimize the communication pattern

- Operations:

```
MPI_Broadcast ( sendbuf, recvbuf,
   count, datatype, op, root, comm )
```



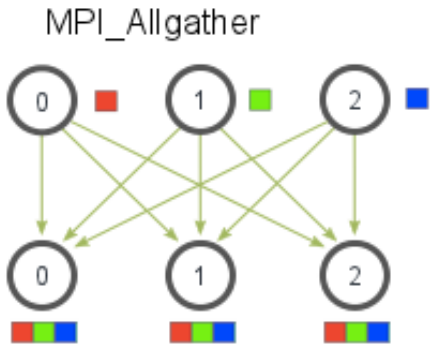

```
MPI_Gather(sendbuf,sendcount,
sendtype, recvbuf, recvcount,
recvtype, root, comm)
```



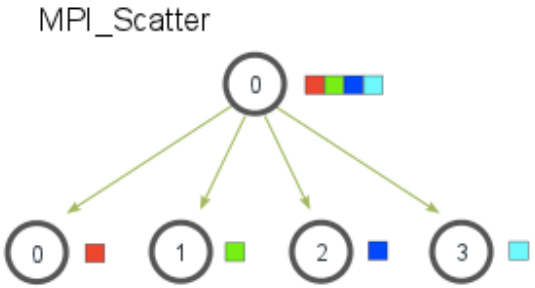
http://mpitutorial.com/tutorials/mpi-broadcast-and-collective-communication/

Collective Operations II

- Operations
  - All-to-all communication
    - Provide unified global view



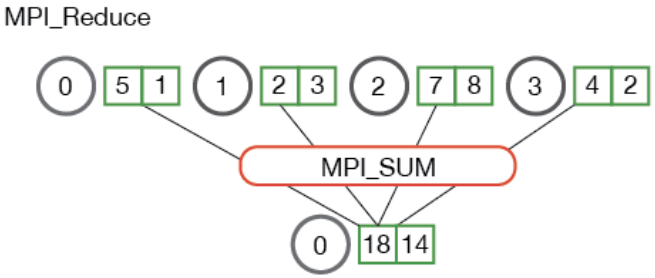
  - Scatter
    - Disseminate from a single process



http://mpitutorial.com/tutorials/mpi-scatter-gather-and-allgather/

Using MPI in Simple Programs II

```
MPI_Reduce ( sendbuf, recvbuf,
             count, datatype, op, root, comm )
```



- Using the specified operation
  - Aggregates: mean, sum
  - Extrema: min, max
  - User defined functions via MPI_OP_CREATE
    - Required property of function: Associativity

Reduce Functions

- All operations are “algebraic” in that they can be applied in any order.

Representation	Operation
MPI_MAX	Maximum
MPI_MIN	Minimum
MPI_SUM	Sum
MPI_PROD	Product
MPI_LAND	Logical and
MPI_BAND	Bit-wise and
MPI_LOR	Logical or
MPI_BOR	Bit-wise or
MPI_LXOR	Logical exclusive or
MPI_BXOR	Bit-wise exclusive or
MPI_MAXLOC	Maximum value and corresponding index
MPI_MINLOC	Minimum value and corresponding index

Using MPI in Simple Programs III

- Using MPI_Reduce

```
h = 1.0 / (double) n;
sum = 0.0;
for (i = rank + 1; i <= n; i += size) {
    x = h * ((double)i - 0.5);
    sum += (4.0 / (1.0 + x*x));
}
mypi = h * sum;

MPI::COMM_WORLD.Reduce(&mypi, &pi, 1, MPI::DOUBLE,
                       MPI::SUM, 0);
if (rank == 0)
    cout << "pi is approximately " << pi
         << ", Error is " << fabs(pi - PI25DT)
         << endl;
```