The Message Passing Interface (MPI) is a standard interface for scientific computing on clusters. It handles the sticky system specific issues of data marshaling and communication substrates.

MPI operates with an abstraction of processes. Each process has an integer rank which can be used to identify the process within the computation.

Memory is not shared between MPI processes.

MPI does not really provide visibility into the location of a process in the physical topology.
int main(int argc, char** argv) {
    int err; // var to catch MPI errors... Could be useful if we checked
    err = MPI_Init(&argc, &argv);
    int rank;
    err = MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    int tot;
    err = MPI_Comm_size(MPI_COMM_WORLD, &tot);
    printf("Hello World. I'm %d of %d\n", rank, tot);
    MPI_Finalize();
    return 0;
}
if(rank==0) {
    sprintf(text,"Hello from rank %d",rank);
    MPI_Send(text,strlen(text)+1,MPI_CHAR,recver,0,MPI_COMM_WORLD);
    MPI_Recv(text,255,MPI_CHAR,sender,0,MPI_COMM_WORLD,&mpistat);
    sprintf(outText,"Rank %d received: %s",rank,text);
} else {
    MPI_Recv(text,255,MPI_CHAR,sender,0,MPI_COMM_WORLD,&mpistat);
    sprintf(outText,"Rank %d received: %s",rank,text);
    sprintf(text,"Hello from rank %d",rank);
    MPI_Send(text,strlen(text)+1,MPI_CHAR,recver,0,MPI_COMM_WORLD);
}

MPI_Gather(outText,255,MPI_CHAR,allText,255,MPI_CHAR,0,MPI_COMM_WORLD);

In: mpi_ring.c
Next Steps

Go to the shell and pull from the repository.

Use ACISS to run the code on a few nodes.