Running MPI

Cory Christensen

January 16, 2020

This report explains how to install MPI, compile and run MPI programs, and gives an example set of code and output for reference.

1 Installing MPI

My Computer runs on Windows 10 in order to install MPI it is best to install a Linux subsystem to compile and run programs from the command line. I have compiled a list of steps I used to get MPI to install and run on my computer.

1.1 Enable Windows Subsystem For Linux

1. Open System Settings
2. Go to "Update and Security"
3. Go to the "For Developers Tab" in the left column"
4. Enable "Developer Mode" by selecting the check box
5. Open Control Panel
6. Go to "Programs"
7. Go to "Turn Windows Features on or off"
8. Enable "Windows Subsystem For Linux" by selecting the check box

1.2 Install Linux

1. Go to the Microsoft Store
2. Install a linux subsystem (UBUNTU,CENTOS,DEBIAN, etc...). I chose "DEBIAN" to install
3. Open debian and type the following: sudo apt-get install openmpi-bin openmpi-common openssh-client openssh-server libopenmpi1.3

1.3 Running the Program

I used the round robin example from class to test MPI. I modified the program slightly by adding a sleep function to keep the code from printing the output before receiving others output. The code for this may be found in the next section. I used the following steps to compile and run the code.

1. Navigate to the directory where you program is saved using cd ¡path¿
2. runing this command compiles the code and generates an a.out file: mpic++ Assn1.cpp
3. run this command to run the program: mpirun -np 8 -oversubscribe a.out

```
The output of the program can be found in the output section. The code is expected
to output the following for each process

process * received a ** and sent it on

where * is the rank of the process and ** is a randomly generated number that is
passed to each process.
```

2 Code

```
#include <iostream>
#include <mpi.h>
#include <unistd.h>
#include <stdlib.h>

//#include "/usr/local/include/mpi.h"
#define MCW MPI_COMM_WORLD

using namespace std;

int main(int argc, char **argv){

    int rank, size;
    int data;
    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MCW, &rank);
    MPI_Comm_size(MCW, &size);

    if(rank==0){
        data = rand()%100;
        cout<< "process 0 generated a "<<data<<endl;
        MPI_Send(&data,1,MPI_INT,(rank+1)%size,0,MCW);
        MPI_Recv(&data,1,MPI_INT,MPI_ANY_SOURCE,0,MCW,MPI_STATUS_IGNORE);
        sleep(1); //to keep from printing out of order
        cout<< "process 0 received a "<<data<<endl;

    } else {
        MPI_Recv(&data,1,MPI_INT,MPI_ANY_SOURCE,0,MCW,MPI_STATUS_IGNORE);
        sleep(1); //to keep from printing out of order
        cout<<"process "<<rank<<" received a "<<data<<" and sent it on.\n";
        MPI_Send(&data,1,MPI_INT,(rank+1)%size,0,MCW);
    }


    MPI_Finalize();

    return 0;
}
```

3 Output

```
process 0 generated a 83
process 1 received a 83 and sent it on.
process 2 received a 83 and sent it on.
process 3 received a 83 and sent it on.
process 4 received a 83 and sent it on.
[Cory-C:00091] 7 more processes have sent help message help-btl-vader.txt / cma
-permission-denied
[Cory-C:00091] Set MCA parameter "orte_base_help_aggregate" to 0 to see all help /
error messages
process 5 received a 83 and sent it on.
process 6 received a 83 and sent it on.
process 7 received a 83 and sent it on.
process 0 received a 83
```