Running MPI on the qstruct and qstruct clusters
updated April 19, 2014

In order to run MPI, the MPI runtime needs to know about what machines you will be running on, and to ensure that you have permission to run on these.

These actions should only need to be done once and are the same regardless of what *shell* you run in linux/unix.

1. create a file .mpd.conf in your home directory and change its permission to only be readable by the owner, i.e.

[smidkiff@qstruct01 ~/TEST]$ chmod 600 .mpd.conf

It should contain the line:

MPD_SECRETWORD=xxxxxx

The xxxxs should be replaced with a random string of letters and numbers -- this word lets MPI keep your processes separate from other's processes. DO NOT PUT YOUR PASSWORD IN HERE!

2. Create a file with the node names you want to use. Call this file mpd.hosts, and put it somewhere you can remember, like your home directory or the directory you run programs from. Each line should be of the form machinename:number_of_processors, thus for the qstruct machines each line should look like:

qstruct01:4
qstruct02:4
qstruct03:4
qstruct04:4
qstruct05:4
qstruct06:4
qstruct07:4
qstruct08:4
...
qstruct19:4

2. type the command

```
echo $SHELL
```

This will likely print out a path ending in one of `bash`, `tcsh` or `csh`

If it prints out `bash` do the following. Steps for `tcsh` or `csh` are given later.

bash shell instructions

3a. Add the following to the bottom of the `.bashrc` file which is in the home directory of the machine you log into with your career account.

```
export MPI_HOME=/opt/mpich2-gcc/1.4.1p1
export PATH=/$MPI_HOME/bin:$PATH
export LD_LIBRARY_PATH=$MPI_HOME/lib64
```

if there is already an `export LD_LIBRARY_PATH` command in your `.bashrc` file add the line:

```
export LD_LIBRARY_PATH=$MPI_HOME/lib64:$LD_LIBRARY_PATH
```

instead.

csh and tcsh shell instructions

3b. Add the following to the bottom of the `.cshrc` file which is in the home directory of the machine you log into with your career account.

```
setenv MPI_HOME /opt/mpich2-gcc/1.4.1p1
setenv PATH /$MPI_HOME/bin:$PATH
setenv LD_LIBRARY_PATH $MPI_HOME/lib64
```

if there is already an `export LD_LIBRARY_PATH` command in your `.cshrc` file add the line:

```
setenv LD_LIBRARY_PATH $MPI_HOME/lib64:$LD_LIBRARY_PATH
```

instead.

The following steps need to be done every time you start a session of running MPI jobs.

4. start up the MPI runtime system for your jobs. The command below will run on 4 nodes using the entries, from first to last (and wrapping around, if necessary) from the mpd.hosts file. You will be prompted for your password from each machine. Type it, hit enter and repeat for P times, where P is the argument to totalnum.

```
[smidkiff@qstruct01 ~/TEST]$ mpdboot --totalnum=4 --file=/home/dynamo/b/
smidkiff/mpd.hosts
```

5. You can run

```
mpdtrace
```

to see what machines MPI has set up to run on. It should be the first P entries in the `mpd.hosts` file specified on the `mpdboot` command.

The following actually runs your program. As long as you remain logged into the node you executed 3 - 6 on, you don't need to repeat them for each run.

6. To compile a program for execution under MPI, use the command:

```
[smidkiff@qstruct01 ~/TEST]$ mpicc your_mpi_program.c
```

7. run your program - this runs it on 8 processes using the 4 machines above.

```
[smidkiff@qstruct01 ~/TEST]$ mpirun -np 8 --hostfile mpd.hosts ./a.out
```

Note that when you do this you will be prompted at least 8 times for your password. It is a pain to deal with this every time you run a job, especially if you are running on many nodes. To not have to do this look at the document at https://engineering.purdue.edu/~smidkiff/ece563/files/sshGenKey.pdf

8. You can check if an environment variable is defined by typing "`echo $VARIABLE`", e.g., "`echo $LD_LIBRARY_PATH`", and seeing if anything is printed. If nothing is printed it is not defined.

8. The program I ran was an MPI hello world, and is shown on the next page. The output from running it is:

```
Greetings from process 1!
Greetings from process 2!
Greetings from process 3!
Greetings from process 4!
Greetings from process 5!
Greetings from process 6!
Greetings from process 7!
```

```
#include <stdio.h>
#include <string.h>     // this allows us to manipulate text strings
```

```
#include "mpi.h"        // this adds the MPI header files to the program

int main(int argc, char* argv[]) {
  int my_rank;        // process rank
  int p;              // number of processes
  int source;         // rank of sender
  int dest;           // rank of receiving process
  int tag = 0;        // tag for messages
  char message[100];  // storage for message
  MPI_Status status;  // stores status for MPI_Recv statements

  // starts up MPI
  MPI_Init(&argc, &argv);
  // finds out rank of each process
  MPI_Comm_rank(MPI_COMM_WORLD, &my_rank);
  // finds out number of processes
  MPI_Comm_size(MPI_COMM_WORLD, &p);

  if (my_rank!=0) {
    sprintf(message, "Greetings from process %d!", my_rank);
    dest = 0; // sets destination for MPI_Send to process 0
    // sends the string to process 0
    MPI_Send(message, strlen(message)+1, MPI_CHAR, dest, tag, MPI_COMM_WORLD);
  } else {
    for(source = 1; source < p; source++){
      // receives greeting from each process
      MPI_Recv(message, 100, MPI_CHAR, source, tag, MPI_COMM_WORLD, &status);
      printf("%s\n", message); // prints out greeting to screen
    }
  }
  MPI_Finalize(); // shuts down MPI
  return 0;
}
```

9. To print out the names of the nodes you program is running on you can use the code:

```
int len = 1024;
char hostname[1024];
hostname[1023] = '\0';
gethostname(hostname, 1023);
cout << hostname << endl;
```

For C programs you can just use `printf` to print the hostname.