

MPI Walkthrough using the UTSA Sun Blade Cluster

0.1 Intro

Our goal here is to get MPI working for you. We will use the Sun blade cluster, and the installed version of MPICH. To see the valid nodes to use, scope the first lines of the provided `blade.mach`. Note that only those first machines before the blank line have been tested to work together. Presently, the first 9 have been tested, so you can see the available machines to use by:

```
head -9 ~whaley/classes/spring10/cs6643/DMB/blade.mach
```

As I get more machine to work, I will change this file, so go ahead and cat the file most of the time.

Log into one of the machines given in the above file. I am going to assume you are using a `cs` variant in the directions below, `bash` users should adapt directions (`.bashrc` for `.cshrc`, `export` instead of `set`, etc. Edit your `.cshrc` (or `.tcshrc`, whichever you use), and add the following line at the bottom (or wherever you modify your path):

```
set path = (/usr/local/mpich-1.2.5/bin $path)
```

We will now attempt to run a simple MPI 'Hello World' program:

```
source ~/.cshrc
mpicc -o xmpitst ~whaley/classes/spring10/cs6643/DMB/mpihello.c
mpirun -machinefile ~whaley/classes/spring10/cs6643/DMB/blade.mach -np 2 \
    ./xmpitst
```

This should give you the expected message. However, likely it prompted you for a password for the remote machine. This is doable for a one-time 2-node run, but quickly gets old when debugging or running 32 nodes. If you did not have to type a password, you can skip the next section, otherwise you must generate some `ssh` keys so that you don't have to type your password when moving from machine to machine, as explained in the next section.

You will want to copy these files from my directory into yours so you don't have to keep filling in the full path, just copy the whole directory.

0.2 Ssh Key Generation

On one of the OKed machines, type the following:

```
ssh-keygen -b 1024 -t rsa
```

You will see something like:

```
Generating public/private rsa key pair.
Enter file in which to save the key (/home/<username>/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/<username>/.ssh/id_rsa.
Your public key has been saved in /home/<username>/.ssh/id_rsa.pub.
The key fingerprint is:
ab.11 ... ef <username>@ten10.cs.utsa.edu
```

Now:

```
>cd ~/.ssh
>ls -l
total 3
-rw----- 1 rcwhaley students 887 Feb 20 14:29 id_rsa
-rw----- 1 rcwhaley students 236 Feb 20 14:29 id_rsa.pub
-rw-r--r-- 1 rcwhaley students 485 Feb 20 14:11 known_hosts
> cp id_rsa.pub authorized_keys
```

Now, go back to the directory where you built `xmpitst`, and issue:

```
>mpirun -np 4 -machinefile \
    ~whaley/classes/spring10/cs6643/DMB/blade.mach ./xmpitst
```

You may need to answer "yes" to the question `add to list of known hosts` (one-time only for each node) if this is your first login, but should no longer need to type your password for each spawned node.

0.3 Playing around

You should now be able to run both MPI 'Hello World' programs with up to 8 nodes. You can get some usage info with `mpirun -h`. You can edit the `blade.mach` to spawn to only the nodes you want. If you don't want your login machine to be used as the first node in the parallel machine, add the flag `-nolocal` to `mpirun`. Note that any flags given after the executable name are assumed to be command-line arguments for the parallel program, so `mpirun ./xtst -np 4` is likely to produce unexpected results. You may want to experiment a bit, scoping the provided two routines, maybe writing your own and scoping the argument flags, etc.