
Using MPI in EE&CE at UWA

MPI is installed on the student network for the Suns in 2.71 and G50, the PC's in 1.51 and G01, and on us2, a dual processor (SMP) Linux machine. The default installation directory is `/usr/local/appl/mpich` on all machines. To use MPI you will need to have `/usr/local/appl/mpich/bin` in your `PATH` and `/usr/local/appl/mpich/man` in your `MANPATH`. Advanced Computer Architecture 319 students should ensure you have this, by including the following line in your `.cshrc` file (which is in your home directory)

```
if ( -f /usr/local/units/aca319/bin/pathadd.csh )
source /usr/local/units/aca319/bin/pathadd.csh
```

You will also need to have `ssh` configured so that you can `ssh` from machine to machine without requiring a password. Create `ssh` keys and store these in your `authorized_keys` file. Do the following, pressing enter whenever prompted for a passphrase or filename:

```
ssh-keygen -t dsa
cd ~/.ssh
touch authorized_keys
cat id_dsa.pub >> authorized_keys
```

Obviously if you have already generated some of these keys you will not need to re-generate them (nor add them to the `authorized_keys` file with the `cat` command)

Machines available on each architecture are listed in the files `/usr/local/appl/mpich/share/machines.ARCH` where `ARCH` is `solaris`, `solaris2`, `LINUX`, `LINUX2` or `us2`. When you use `mpirun` without explicitly selecting a machines file (using the `-machinefile` option) the machines which are used are

automatically selected from one of these files, corresponding to the architecture that MPI thinks you are running on. You can always over-ride this auto-detection with the `-arch` option for `mpirun`). Note that these files contain only separate listings for the separate labs. This is so that the MPI network traffic stays within a single lab, meaning it is confined to a single network switch so that: 1) your message latencies are minimized and 2) you have the least possible impact on other network users. It is possible to use machines from the other labs, but you must manually edit your machines file to do this.

Suns in 2.71 (listed in `machines.solaris`): `sun1 .. sun23`

Suns in G.50 (listed in `machines.solaris2`): `sun24 .. sun49`

PC's in 1.51 (listed in `machines.LINUX`): `pc01 .. pc31`

PC's in G.01 (listed in `machines.LINUX2`): `pc32 .. pc68`

Due to problems arising from machines being turned off (or in Windows XP) and unavailable for use by MPI, you should use the script `getmachines.sh` (which is in `~aca319/bin`) to produce a machines file which you then pass to `mpirun`. This program detects the architecture on which it is being run (`LINUX` or `solaris`) and produces an output machines file for that architecture (using the global machines file as a starting point).

You can get some brief information on `getmachines.sh` by typing `"getmachines.sh -h"`. The most common way to use it is to type

```
getmachines.sh -outfile=machines
```

which will output the list of available machines to the file `"machines"` in the current directory.

So, to compile and run your program:

```
mpicc -o myprog myprog.c
getmachines.sh -outfile=machines
mpirun -machinefile machines -np 2
cleanmachines.sh -machinefile=machines
```

Notes:

1. The `mpirun` command does not use '=' signs for its command line parameters whereas the `getmachines.sh` and `cleanmachines.sh` scripts do
2. The second step doesn't need to be done every time before you run - only when you have a problem with one of the machines in your machine file, or you want to see if more machines have become available
3. The fourth step is only necessary if something goes wrong with your program
4. If you want to use computers in another lab, eg you are in 1.51 which means the default is to use `machines.LINUX` and you want to use computers in G.01 which are listed in the `machines.LINUX2` file, pass the desired architecture descriptor to `getmachines.sh`, ie `"getmachines.sh -arch=LINUX -outfile=machines"`. If you want to use a combination of machines from different labs you will have to run `getmachines.sh` twice, producing two machines files and combining them.

Using MPI on us2

us2 is a dual-processor Linux machine which lives in the server room off G.50. Everyone should be able to log into it and use it just like any of the other Linux PC's. It has two different versions of MPICH installed on it, corresponding to two different communication mechanisms: `ch_P4` (the standard UNIX message-passing mechanism which is used in the Suns and Linux labs) which is installed in `/usr/local/appl/mpich/`, and `ch_shmem` (message-passing via shared memory) which is installed in `/usr/local/appl/mpich_shmem/`. When you use the commands `mpicc` and `mpirun` on us2, you are using the default `ch_p4` version. To use the `ch_shmem` versions you should use the commands `mpicc-shmem` and `mpirun-shmem`, which have been aliased to point to the `shmem` commands (type `"alias"` at the command prompt to see this). Note that if you try to use run a program using the `ch_p4` version of `mpirun` and it was compiled with the `ch_shmem` version of `mpicc` (or vice versa) you will run into problems.

When you use `mpirun` on `us2`, it thinks it has `ARCH=LINUX` when what you really want is `ARCH=us2`. To get around this you must specify the architecture explicitly on the `mpirun` and/or `mpirun-shmem` command lines:

```
mpirun -arch us2 -np 2 myprog
```

or

```
mpirun-shmem -arch us2 -np 2 myprog
```

You can of course run your program with more than two processes. All that will happen is that multiple processes will be allocated to each CPU.

You won't have the same fault tolerance problems with MPI on `us2` as on the lab networks so you do not need to use `getmachines.sh` to create a machine file.

Visualization Tools

For performance visualization, a couple of tools have installed on the different computers:

- `upshot` on the Linux machines
- `jumpshot v3` on the Suns

These are installed in `/usr/local/app1/mpich/share` and can be executed via the commands `upshot` and `jumpshot` (which are aliases I have set up)

Look [here](#) for information on using the MPE library for logging MPI events, and [here](#) for instructions on using `upshot` to view these logs.
