## Chapter 3.

# Using mpirun to Execute Applications

The mpirun(1) command is the primary job launcher for the MPT implementations of MPI. The mpirun command must be used whenever a user wishes to run an MPI application on IRIX or UNICOS systems (on IRIX systems, XMPI can be used in place of mpirun). On IRIX or UNICOS systems, you can run an application on the local host only (the host from which you issued mpirun) or distribute it to run on any number of hosts that you specify. Use of the mpirun command is optional for UNICOS/mk systems and currently supports only the -np option. Note that several MPI implementations available today use a job launcher called mpirun, and because this command is not part of the MPI standard, each implementation's mpirun command differs in both syntax and functionality.

# 3.1 Syntax of the mpirun Command

The format of the mpirun command for UNICOS and IRIX is as follows:

mpirun [*global_options* ] *entry*[*: entry ...* ]

The *global_options* operand applies to all MPI executable files on all specified hosts. The following global options are supported:

| Option | Description |
|---|---|
| -a[rray]*array_name* | Specifies the array to use when launching an MPI application. By default, Array Services uses the default array specified in the Array Services configuration file, arrayd.conf. |
| -d[ir]*path_name* | Specifies the working directory for all hosts. In addition to normal path names, the following special values are recognized:<br>. Translates into the absolute path name of the user's current working directory on the local host. This is the default.<br>~ Specifies the use of the value of $HOME as it is defined on each machine. In general, this value can be different on each machine. |
| -f[ile]*file_name* | Specifies a text file that contains mpirun arguments. |
| -h[elp] | Displays a list of options supported by the mpirun command. |
| -p[refix]*prefix_string* | Specifies a string to prepend to each line of output from stderr and stdout for each MPI process. Some strings have special meaning and are translated as follows:<br>·%g translates into the global rank of the process producing the output. (This is equivalent to the rank of the process in MPI_COMM_WORLD.)<br>·%G translates into the number of processes in MPI_COMM_WORLD.<br>·%h translates into the rank of the host on which the process is running, relative to the mpirun(1) command line.<br>·%H translates into the total number of hosts in the job.<br>·%l translates into the rank of the process relative to other processes |

running on the same host.

·%L translates into the total number of processes running on the host.

·%@ translates into the name of the host on which the process is running.

For examples of the use of these strings, first consider the following code fragment:

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

{

   MPI_Init(&argc, &argv);

   printf("Hello world\n");

   MPI_Finalize();

}
```

Depending on how this code is run, the results of running the `mpirun` command will be similar to those in the following examples:

```
mpirun -np 2 a.out

Hello world

Hello world

mpirun -prefix ">" -np 2 a.out

>Hello world

>Hello world

mpirun -prefix "%g" 2 a.out

0Hello world

1Hello world

mpirun -prefix "[%g] " 2 a.out

[0] Hello world

[1] Hello world

mpirun -prefix "<process %g out of %G> " 4 a.out

<process 1 out of 4> Hello world

<process 0 out of 4> Hello world

<process 3 out of 4> Hello world

<process 2 out of 4> Hello world
```

```
mpirun -prefix "%@: " hosta,hostb 1 a.out

hosta: Hello world

hostb: Hello world
```

```
mpirun -prefix "%@ (%l out of %L) %g: " hosta 2, ho
stb 3 a.out

hosta (0 out of 2) 0: Hello world

hosta (1 out of 2) 1: Hello world

hostb (0 out of 3) 2: Hello world

hostb (1 out of 3) 3: Hello world

hostb (2 out of 3) 4: Hello world
```

```
mpirun -prefix "%@ (%h out of %H): " hosta,hostb,ho
stc 2 a.out

hosta (0 out of 3): Hello world

hostb (1 out of 3): Hello world

hostc (2 out of 3): Hello world

hosta (0 out of 3): Hello world

hostc (2 out of 3): Hello world

hostb (1 out of 3): Hello world
```

`-v[erbose]`
Displays comments on what `mpirun` is doing when launching the MPI application.

The *entry* operand describes a host on which to run a program, and the local options for that host. You can list any number of entries on the `mpirun` command line.

In the common case (same program, multiple data (SPMD)), in which the same program runs with identical arguments on each host, usually only one entry needs to be specified.

Each entry has the following components:

- One or more host names (not needed if you run on the local host)
- Number of processes to start on each host
- Name of an executable program
- Arguments to the executable program (optional)

An entry has the following format:

*host_list local_options program program_arguments*

The *host_list* operand is either a single host (machine name) or a comma−separated list of hosts on which to run an MPI program.

The *local_options* operand contains information that applies to a specific host list. The following local options are supported:

| Option | Description |
|---|---|
| `-f[ile]`*file_name* | Specifies a text file that contains `mpirun` arguments (same as *global_options*.) For more details, see Section 3.2. |
| `-np`*np* | Specifies the number of processes on which to run. (UNICOS/mk systems support only this option.) |
| `-nt`*nt* | On UNICOS systems, specifies the number of tasks on which to run in a multitasking or shared memory environment. On IRIX systems, this option behaves the same as `-np`. |

The *program program_arguments* operand specifies the name of the program that you are running and its accompanying options.

# 3.2 Using a File for `mpirun` Arguments (UNICOS Or IRIX)

Because the full specification of a complex job can be lengthy, you can enter `mpirun` arguments in a file and use the `-f` option to specify the file on the `mpirun` command line, as in the following example:

```
mpirun -f my_arguments
```

The arguments file is a text file that contains argument segments. White space is ignored in the arguments file, so you can include spaces and newline characters for readability. An arguments file can also contain additional `-f` options.

# 3.3 Launching Programs on the Local Host Only

For testing and debugging, it is often useful to run an MPI program on the local host only without distributing it to other systems. To run the application locally, enter `mpirun` with the `-np` or `-nt` argument. Your entry must include the number of processes to run and the name of the MPI executable file.

The following command starts three instances of the application `mtest`, to which is passed an arguments list (arguments are optional).

```
mpirun -np 3 mtest 1000 "arg2"
```

You are not required to use a different host in each entry that you specify on the `mpirun(1)` command. You can launch a job that has two executable files on the same host. On a UNICOS system, the following example uses a combination of shared memory and TCP. On an IRIX system, both executable files use shared memory:

```
mpirun host_a -np 6 a.out : host_a -nt 4 b.out
```

## 3.3.1 Using `mpirun(1)` to Run Programs in Shared Memory Mode (UNICOS Or IRIX)

For running programs in MPI shared memory mode on a single host, the format of the `mpirun(1)` command is as follows:

`mpirun -nt[`*nt*`]`*progname*

The `-nt` option specifies the number of tasks for shared memory MPI, and can be used on UNICOS systems only if you have compiled and linked your program as described in Section 2.1.1. A single UNIX process is run with multiple tasks representing MPI processes. The *progname* operand specifies the name of the program that you are running and its accompanying options.

The `-nt` option to `mpirun` is supported on IRIX systems for consistency across platforms. However, since the default mode of execution on a single IRIX system is to use shared memory, the option behaves the same as if you specified the `-np` option to `mpirun`. The following example runs ten instances of `a.out` in shared memory mode on host_a:

```
mpirun -nt 10 a.out
```

### 3.3.2 Using the mpirun(1) Command on UNICOS/mk Systems

The mpirun(1) command has been provided for consistency of use among IRIX, UNICOS, and UNICOS/mk systems. Use of this command is optional, however, on UNICOS/mk systems. If your program was built for a specific number of PEs, the number of PEs specified on the mpirun(1) command line must match the number that was built into the program. If it does not, mpirun(1) issues an error message.

The following example shows how to invoke the mpirun(1) command on a program that was built for four PEs:

```
mpirun -np 4 a.out
```

### 3.3.3 Executing UNICOS/mk Programs Directly

Instead of using the mpirun(1) command, you can choose to launch your MPI programs on UNICOS/mk systems directly. If your UNICOS/mk program was built for a specific number of PEs, you can execute it directly, as follows:

```
./a.out
```

If your program was built as a *malleable* executable file (the number of PEs was not fixed at build time, and the -Xm option was used instead), you can execute it with the mpprun(1) command. The following example runs a program on a partition with four PEs:

```
mpprun -n 4 a.out
```

## 3.4 Launching a Distributed Program (UNICOS Or IRIX)

You can use mpirun(1) to launch a program that consists of any number of executable files and processes and distribute it to any number of hosts. A host is usually a single Origin, CRAY J90, or CRAY T3E system, or can be any accessible computer running Array Services software. Array Services software runs on IRIX and UNICOS systems and must be running to launch MPI programs. For available nodes on systems running Array Services software, see the /usr/lib/array/arrayd.conf file.

You can list multiple entries on the mpirun command line. Each entry contains an MPI executable file and a combination of hosts and process counts for running it. This gives you the ability to start different executable files on the same or different hosts as part of the same MPI application.

The following examples show various ways to launch an application that consists of multiple MPI executable files on multiple hosts.

The following example runs ten instances of the a.out file on host_a:

```
mpirun host_a -np 10 a.out
```

When specifying multiple hosts, the -np or -nt option can be omitted with the number of processes listed directly. On UNICOS systems, if you omit the -np or -nt option, mpirun assumes -np and defaults to TCP for communication. The following example launches ten instances of fred on three hosts. fred has two input arguments.

```
mpirun host_a, host_b, host_c 10 fred arg1 arg2
```

The following example launches an MPI application on different hosts with different numbers of processes and executable files, using an array called test:

```
mpirun -array test host_a 6 a.out : host_b 26 b.out
```

The following example launches an MPI application on different hosts out of the same directory on both hosts:

```
mpirun -d /tmp/mydir host_a 6 a.out : host_b 26 b.out
```