The Multi-Processor Communications library

CARRIBAULT Patrick, PÉRACHE Marc

June 26, 2009

Contents

1 Introduction

This document describes the Multi-Processor Communications library (MPC). The *Getting Started* section describes the steps to setup the library and some simple way of using it. The following sections detail the API and the different types of executions.

The following article contains more detailed information about the MPC framework: "MPC: a unified parallel runtime for clusters of NUMA machines," Europar 2008.

2 Getting Started

2.1 Standard

The following instructions take you through a sequence of steps to get the default configuration MPC up and running. Alternate configuration options are described later, in the section

Alternative configurations.

1. The following prerequisites are required:

   - The main archive file MPC_1.1_rc7.tar.gz
   - A C compiler (e.g., gcc)
   - A GNU C compiler
   - An optional Fortran compiler if Fortran applications are to be used (e.g., g77 or gfortran)

   The configure will check for these prerequisites and will try to find alternate ways if possible.

2. Unpack the main archive (`tar` format) and go to the top-level directory:

   ```
   tar xfz MPC_1.1_rc7.tar.gz
   cd MPC_1.1_rc7
   ```

   If your tar program does not accept the 'z' option, use the following commands

   ```
   gunzip MPC_1.1_rc7.tar.gz
   tar xf MPC_1.1_rc7.tar
   cd MPC_1.1_rc7
   ```

3. Choose an installation directory (the default is `/usr/local/`):

   ```
   mkdir /home/you/mpc-install
   ```

   The most convenient choice is a directory shared among all the machines you want to use the library on. If such a directory is not accessible, you will have to deploy MPC on every machine after the installation.

4. Configure MPC, specifying the installation directory:

   ```
   ./configure --prefix=/home/you/mpc-install
   ```

5. Build MPC:

   ```
   make
   ```

6. Install the MPC commands and library:

   ```
   make install
   ```

7. Add the `bin/` subdirectory of the installation directory to your path.
   For csh and tcsh:

   ```
   setenv PATH /home/you/mpc-install/bin:\$PATH
   ```

For bash and sh:

```
PATH=/home/you/mpc-install/bin:\$PATH ; export PATH
```

Check that everything is working well at this point by running

```
which mpcrun
which mpc_cc
```

8. To compile your first MPC program, you may execute the `mpc_cc` compiler:

```
mpc_cc MPC_Tests/parallel/MPC_Message_Passing/hello_world.c \
       -o hello_world
```

or use your favorite compiler thanks to access to cflags and ldflags:

```
$CC MPC_Tests/parallel/MPC_Message_Passing/hello_world.c \
     -o hello_world ‘mpc cflags‘ ‘mpc ldflags‘
```

9. To execute your MPC program, use the mpcrun command:

```
mpcrun -m=ethread      -n=4 hello_world
mpcrun -m=ethread_mxn  -n=4 hello_world
mpcrun -m=pthread      -n=4 hello_world
```

See the section *Thread types* for details on the '-m' option.

2.2 Alternatives

The previous section described the default installation and configuration of MPC. But other alternatives are available. You can find out more details on the configuration by running:

```
./configure --help
```

3 MPC user APIs

3.1 Message Passing

3.1.1 API

MPC is MPI 1.3 compliant except for the following functions that are not implemented:

- MPI_Comm_test_inter
- MPI_Comm_remote_size
- MPI_Comm_remote_group
- MPI_Intercomm_create
- MPI_Intercomm_merge
- MPI_Get_elements
- MPI_Cancel

See the document *MPI: A Message-Passing Interface Standard* version 1.3 (May 2008) for more details.

3.1.2 Warning to users

Remove global variables! In MPC, every MPI task is a thread and thus all tasks share global variables with each other.

3.2 Threads

3.2.1 API

MPC provide a POSIX Thread 2003 compatible API.

3.2.2 Thread types

The main command `mpcrun` accepts the '-m' option to choose between several kind of threads. Here is a list of the current available thread types:

1. Ethread: Mx1 user level thread model.
2. Ethread_mxn: MxN user level thread model.
3. Pthread: underlying POSIX Thread library.

The article *MPC: a unified parallel runtime for clusters of NUMA machines* (EuroPar 2008) contains more details on the different thread types and their characteristics.

3.2.3 Warning to users

It is dangerous to mix MPC POSIX Threads and system POSIX Threads! This mix may lead to an undefined behavior.

4 Details on mpcrun

The `mpcrun` script allows to launch MPC programs with different types of parallelism. Its usage is defined as follows:

```
Usage mpcrun [option] [--] binary [user args]

Informations:
    --help,-h Display this help
    --show, Display command line
    --version-details, Print version of each module used
    --report, Print report
    --tmp_dir=dir, Directory to store mpc files
    --verbose,-v Verbose mode

Topology:
    --task-nb=n,-n=n Total number of tasks
    --process-nb=n,-p=n Total number of processes
    --cpu-nb=n,-c=n Number of cpus per process
    --node-nb=n,-N=n Total number of nodes
    --disable-smt Disable SMT capabilities
    --share-node Restrict CPU number to share node
```

```
Multithreading:
    --multithreading=n,-m=n Define multithreading mode
        modes: pthread ethread_mxn ethread

Network:
    --network=n,-net=n Define Network mode
        modes:  none tcp ...
        modes (experimental): ...

Checkpoint/Restart and Migration:
    --checkpoint Enable checkpoint
    --migration Enable migration
    --restart Enable restart

Launcher:
    --launcher=n,-l=n Define launcher
    --opt=<options> launcher specific options
    --launch_list print available launch methods

Debugger:
    --dbg=<debugger_name> to use a debugger
```

4.1 Mono-process job

In order to run an MPC job in a single process, you should use on of the following methods (depending on the thread type you want to use).

```
mpcrun -m=ethread      -n=4 hello_world
mpcrun -m=ethread_mxn  -n=4 hello_world
mpcrun -m=pthread      -n=4 hello_world
```

4.2 Multi-process job on a single node

In order to run an MPC job in a 2-process single-node manner with *TCP* inter-process communications, you should use on of the following methods (depending on the thread type you want to use).

```
mpcrun -m=ethread      -n=4 -p=2 -net=tcp hello_world
mpcrun -m=ethread_mxn  -n=4 -p=2 -net=tcp hello_world
mpcrun -m=pthread      -n=4 -p=2 -net=tcp hello_world
```

There are different implementations of inter-process communications. A call to `mpcrun --help` details all the available implementations.

4.3 Multi-process job on multiple nodes

In order to run an MPC job on 2 node with 8 processes communicating with *TCP*, you should use one of the following methods (depending on the thread type you want to use).

```
mpcrun -m=ethread      -n=4 -p=8 -net=tcp -N=2 -l=ssh hello_world
mpcrun -m=ethread_mxn  -n=4 -p=8 -net=tcp -N=2 -l=ssh hello_world
mpcrun -m=pthread      -n=4 -p=8 -net=tcp -N=2 -l=ssh hello_world
```

There are different implementations of inter-process communications and launch methods. A call to `mpcrun --help` detail all the available implementations and launch methods.

4.3.1 Launch with ssh

In order to execute an MPC job on multile nodes using *ssh*, you need to fill the `$HOME/.mpcrun_tcp_host` file with the list of nodes to use.

5 Contacts

- CARRIBAULT Patrick patrick.carribault@cea.fr
- PÉRACHE Marc marc.perache@cea.fr