

# Mad-MPI: Installation and User's Guide

The PM2 Team<sup>1</sup>

July 2, 2008 at 10:54

<sup>1</sup>runtime@labri.fr

## **Abstract**

Mad-MPI is a light implementation of the MPI standard. This simple, straightforward proof-of-concept implementation is a subset of the MPI API, that allows MPI applications to benefit from the NewMadeleine communication engine. Mad-MPI is based on the point-to-point nonblocking posting (`isend`, `irecv`) and completion (`wait`, `test`) operations of MPI, these four operations being directly mapped to the equivalent operations of NewMadeleine.

Mad-MPI also implements some optimizations mechanisms for derived datatypes. MPI derived datatypes deal with noncontiguous memory locations. The advanced optimizations of NewMadeleine allowing to reorder packets lead to a significant gain when sending and receiving data based on derived datatypes.

The latest version of this document is available from the following URL:

<http://runtime.bordeaux.inria.fr/MadMPI/doc/>

# Contents

<b>1</b>	<b>Installing Mad-MPI</b>	<b>2</b>
1.1	Getting Access to the Source Code . . . . .	2
1.2	Required Development Tools . . . . .	2
1.3	Prerequisites . . . . .	3
1.4	Installing PM2 . . . . .	3
1.5	Installing Mad-MPI . . . . .	3
1.6	Network Protocols . . . . .	4
1.7	Summary of Environment Variables . . . . .	5
<b>2</b>	<b>Running MPI Applications</b>	<b>6</b>
2.1	Basic Use . . . . .	6
2.2	Advanced Use . . . . .	7
2.3	MPI Compilers . . . . .	8
<b>3</b>	<b>Developping MPI Applications</b>	<b>9</b>
3.1	The API . . . . .	9
3.2	Debugging Mad-MPI applications . . . . .	9
3.3	Debugging Mad-MPI . . . . .	9

# Chapter 1

## Installing Mad-MPI

### 1.1 Getting Access to the Source Code

Mad-MPI files are hosted by the project PM2 on the InriaGforge at <https://gforge.inria.fr/projects/pm2/>. The source code is managed by a Subversion server hosted by the InriaGforge. To get the source code, you need:

1. To install the client side of the software Subversion if it is not already available on your system. The software can be obtained from <http://subversion.tigris.org/>.
2. To become a member of the project pm2. For this, you first need to get an account to the gForge server. You can then become a member of the project by contacting one of the project administrators. The list of the project administrators is available from the main project page.

You can also choose to check out the project's SVN repository through anonymous access. In this case, you do not need to become a member, and you will have limited access to the repository.

More information on how to get a gForge account, to become a member of the specified project, or on any other related task can be obtained from the InriaGforge at <https://gforge.inria.fr/>. The most important thing is to upload your public SSH key on the gForge server (see the FAQ at <http://siteadmin.gforge.inria.fr/FAQ.html#Q6> for instructions).

### 1.2 Required Development Tools

The following development tools are required to compile Mad-MPI:

- GNU C Compiler `gcc` (version 3.2 and higher).
- GNU `make` (version 3.81 and higher).
- GNU Bison (<http://www.gnu.org/software/bison/>) and Flex (<http://www.gnu.org/software/flex/>).

## 1.3 Prerequisites

To use Myrinet and Quadrics, you must get the appropriate drivers. Mad-MPI supports Myrinet networks through the MX driver and Quadrics networks through the ELAN driver.

```
% export MX_DIR=/softs/mx
% export SISCI_PATH=/opt/DIS
```

## 1.4 Installing PM2

Here are the required steps to install PM2.

1. Choose an installation directory for PM2, and set the environment variable `PM2_ROOT` to this directory.

```
% export PM2_ROOT=$HOME/soft/pm2
```

2. Go in the parent directory of the directory `PM2_ROOT` and check out the latest version from the Subversion server

- using the anonymous access.

```
% cd $HOME/soft
% svn checkout svn://scm.gforge.inria.fr/svn/pm2/trunk pm2
```

- or using your gForge account.

```
% cd $HOME/soft
% svn checkout svn+ssh://<login>@scm.gforge.inria.fr/svn/pm2/trunk pm2
```

You should end up with a directory named `pm2`.

3. You need to add to your `PATH` the directory `pm2/bin`.

```
% export PATH=${PM2_ROOT}/bin:${PATH}
```

4. PM2 is installed.

## 1.5 Installing Mad-MPI

Here are the required steps to install Mad-MPI.

1. Create the PM2 flavor tuned for Mad-MPI.

```
% pm2-create-sample-flavors nmad-mpi
```

By default, the only network protocol selected is TCP. To enable more protocols, see Section 1.6.

## 2. Compile Mad-MPI.

```
% make -C $PM2_ROOT/nmad FLAVOR=nmad-mpi
```

## 3. You need to provide a network configuration file, located in

```
`${PM2_ROOT}/nmad/examples/mpi/networks.cfg
```

This file is mandatory but users can select another location by defining the appropriate environment variable (see Section 2.2). When loading a Mad-MPI application, the network configuration file is read to determine which network links to use between the different machines involved in the execution. The sample file

```
`${PM2_ROOT}/leonie/examples/networks.cfg
```

can be used as a model to describe your network configuration.

## 4. Users should add to their PATH the element nmad/protocols/proto\_mpi/scripts/ in order to easily access the commands mpicc and mpirun.

```
% export PATH=${PM2_ROOT}/nmad/protocols/proto_mpi/scripts:${PATH}
```

## 1.6 Network Protocols

**This section can be ignored if you plan to use the TCP protocol only.**

By default, the flavor for Mad-MPI only enables the TCP network protocol. To enable more protocols such as MX or Quadrics, you need to modify the flavor. This can be done easily by starting the tool ezflavor.

```
% make -C $PM2_ROOT initnoflavor
% ezflavor
```

Once the graphical interface ezflavor is started, click on drop-down menu in the Flavors panel, select the flavor nmad-mpi and click on the Load button in the same panel.

The flavor is now loaded and can be modified. You can see in the right-hand side panel Modules all the modules involved in the implementation of Mad-MPI as well as all the options selected for these modules.

In the Modules panel, select the nmad tab, the options for this module will show up in the underneath panel. By using the scrollbar on the right-hand side of this panel, you can scroll down to the section showing the network protocols. Select the ones which are available to you.

Click on the Save button in the Flavors panel to record your modifications. You can now quit ezflavor by pressing Ctrl-Q, or by selecting in the toolbar menu, Flavor, and Quit.

## 1.7 Summary of Environment Variables

Here the list of environment variables you need to set (preferably in your login files such as `$HOME/.bashrc`) before being able to compile and execute Mad-MPI applications.

```
## Directory where PM2 is installed
export PM2_ROOT=$HOME/soft/pm2

## Access to the PM2 commands
export PATH=${PM2_ROOT}/bin:${PATH}

## Access to the Mad-MPI commands
export PATH=${PM2_ROOT}/nmad/protocols/proto_mpi/scripts:${PATH}
```

## Chapter 2

# Running MPI Applications

### 2.1 Basic Use

The directory `$PM2_ROOT/nmad/examples/mpi/basics` contains basic examples which can be used to test your installation of Mad-MPI. We will go through this section by using the program `cpi.c`.

- On top of the environment variables defined in Section 1.7, you need to define the following environment variable which is needed by PM2.

```
% export LEO_RSH="ssh -n -f"
```

- Compile your application with `mpicc`.

```
% cd $PM2_ROOT/nmad/examples/mpi/basics
% mpicc -o cpi cpi.c
```

- You need to make sure you can connect to the machines defined in your channel using the SSH protocol. Mad-MPI applications are launched using the PM2's mechanisms to remotely connect to the machines, this requires a SSH access to the machines without password or pass-phrase. You might need to change the definition of the environment variable `LEO_RSH` to match the configuration of your SSH connection. Here an example to test your configuration is working:

```
% echo $LEO_RSH
ssh -n -f
% $LEO_RSH jack uname -a
% Linux jack 2.6.4-fkt #14 SMP Thu Sep 23 2004 i686 GNU/Linux
%
```

- We suppose for now the network configuration file `networks.cfg` is present in the directory ``${PM2_ROOT}`/nmad/examples/mpi/` as explained in Section 1.5. Look at the contents of the file to see which machines you can use for your application. The machines can be specified by:

- Using the parameter `-machinefile` to specify the file containing the list of machines you wish to use.
- Using a OAR reservation, the application will be launched on the machines reserved by OAR.
- By setting the environment variable `MPI_HOST_FILE` to a file containing the list of the machines you wish to use.

```
% cat mymachines
node-16.bordeaux.grid5000.fr
node-41.bordeaux.grid5000.fr
node-47.bordeaux.grid5000.fr
% mpirun -machinefile mymachines ...
```

```
% cat $OAR_NODEFILE
node-16.bordeaux.grid5000.fr
node-41.bordeaux.grid5000.fr
node-47.bordeaux.grid5000.fr
% cat $MPI_HOST_FILE
node-16.bordeaux.grid5000.fr
node-41.bordeaux.grid5000.fr
%
```

- Start your application using the command `mpirun` taking at least in parameter the number of processes to launch by using the standard MPI option `-np` and the name of the application. The number of processes does not have to be the number of machines in the machine file (as specified above). Mad-MPI will use a subset of these machines, or will start more than one process per machine accordingly.

```
% mpirun -np 2 cpi
```

- You should obtain an output similar to:

```
##### sweetie
##### rantanplan
Process 0 of 2 on rantanplan
pi is approximately 3.1415926544231318, Error is 0.0000000008333387
wall clock time = 0.000237
Process 1 of 2 on sweetie
```

**Remark:** The default behavior of Mad-MPI is to launch processes only on the nodes specified in the channel configuration file. No process is spawned on the local machine (unless it is listed in the channel configuration file)

## 2.2 Advanced Use

- The default network file used for launching jobs is

```
${PM2_ROOT}/nmad/examples/mpi/networks.cfg
```

This file has to be provided by the person installing Mad-MPI. However, users can select another network file by defining the environment variable `MPI_NET_FILE` with the location of the new file, e.g.:

```
% export MPI_NET_FILE=$HOME/networks.cfg
```

- If the machines you wish to use to run your application are interconnected with more than 1 network, then Mad-MPI will by default select the first network to connect the nodes. To use another network, you can set the environment variable `MPI_NMAD_PROTOCOL` to the name of this network.

```
% export MPI_NMAD_PROTOCOL=mx
```

- You can create your own network configuration file by calling the script `mpiconfig`.

```
% mpiconfig -h
Syntax: ../mpiconfig [-f <machine file>] [-net <communication network>]
      Default machine file: OAR node file
      Default communication network: tcp
%
```

When started within an OAR reservation, the script will automatically use the reserved machines for the configuration. You can then start your application with the newly created files.

```
% oarsub -l nodes=2 -I
Host:Port = frontale.sophia.grid5000.fr:56926
IdJob = 257224
Interactive mode : waiting
% mpiconfig
Generation of networksMadMpi.cfg and machineMadMpi.cfg done
- nodes:      node-69.sophia.grid5000.fr,node-70.sophia.grid5000.fr
- net:        tcp
You can now start your MPI application:
  mpirun -config networksMadMpi.cfg -machinefile machineMadMpi.cfg -np ...
% mpirun -config networksMadMpi.cfg -machinefile machineMadMpi.cfg -np 4 cpi
...
%
```

## 2.3 MPI Compilers

Mad-MPI provides by default a C and C++ compiler. The C compiler is invoked by calling `mpicc` and the C++ compiler by calling `mpicxx`. This name was chosen instead of `mpicc` to avoid problems with operating systems not being case-sensitive.

Fortran compilers are also available by setting the PM2 flavor accordingly. One can use the tools `ezflavor` or `pm2-menu-config` to modify the flavor `nmad-mpi` (see Section 1.5). Setting the option `fortran_target_gfortran` or `fortran_target_ifort` of the module `common` will compile the Fortran support within Mad-MPI. Fortran compilers can then be invoked by calling `mpif77` or `mpif90`.

## Chapter 3

# Developping MPI Applications

### 3.1 The API

Mad-MPI implements almost all of the functionalities of the MPI standard. Some additional functionalities have been added to provide users with the facilities of the underlying New-Madeleine communication engine. The complete API is available at the following URL:

```
http://pm2.gforge.inria.fr/newmadeleine/html/group__mpi__interface.html
```

### 3.2 Debugging Mad-MPI applications

- If you wish to start your application under the debugger, you can use the option `-dbg`. All the processes of the MPI application will be launched through `gdb`, the GNU debugger.

```
% mpirun -dbg -np 2 cpi
```

- If you wish to start your application under `valgrind`, you can use the option `-valgrind`.

```
% mpirun -valgrind -np 2 cpi
```

### 3.3 Debugging Mad-MPI

The implementation of Mad-MPI and NewMadeleine are made up of modules. Each of these modules can be traced or logged. The PM2 flavor for Mad-MPI first needs to be tuned to enable the tracing and debugging facilities. This can be done easily by starting the tool `ezflavor`.

```
% make -C $PM2_ROOT initnoflavor  
% ezflavor
```

Once the graphical interface `ezflavor` is started, click on drop-down menu in the `Flavors` panel, select the flavor `nmad-mpi` and click on the `Load` button in the same panel.

The flavor is now loaded and can be modified. You can see in the right-hand side panel Modules all the modules involved in the implementation of Mad-MPI as well as all the options selected for these modules.

In the toolbar menu, select View, Common Options, Display Panel. In the new panel Common options, tick the option Set in the debug panel, and the option Unset in the opt panel. By clicking on the Apply to all button, the selection of these two options will be modified for all the modules at once.

Click on the Save button in the Flavors panel to record your modifications. You can now quit ezflavor by pressing Ctrl-Q, or by selecting in the toolbar menu, Flavor, and Quit.

You can now recompile Mad-MPI.

```
% make -C $PM2_ROOT/nmad FLAVOR=nmad-mpi clean
% make -C $PM2_ROOT/nmad FLAVOR=nmad-mpi
```

And recompile your application as explained in Section 2.1.

```
% cd $PM2_ROOT/nmad/examples/mpi/basics
% mpicc -o cpi cpi.c
```

Debug parameters allow to trace specific modules. The general format of a debug parameter is `--debug:<MODULE_NAME>-<TRACE_LEVEL>`. For example, the debug parameter `--debug:nmad-trace` will display all the trace messages within the module nmad. **Note that the module must have been compiled with the option debug.** More debug parameters are available, you can print the list as follows:

```

% mpirun -machinefile mac -np 2 cpi --debug:register
##### bolero.labri.fr
##### bolero.labri.fr
register debug name: register [default] (show=5)
register debug name: mpi_nmad_log [default] (show=DEFAULT (2))
register debug name: default [default] (show=2)
register debug name: log [default] (show=DEFAULT (2))
register debug name: nmad-log [log] (show=DEFAULT (2))
register debug name: tbx-log [log] (show=DEFAULT (2))
register debug name: ntbx-log [log] (show=DEFAULT (2))
register debug name: trace [default] (show=DEFAULT (2))
register debug name: nmad-trace [trace] (show=DEFAULT (2))
register debug name: ntbx-trace [trace] (show=DEFAULT (2))
register debug name: register [default] (show=5)
register debug name: mpi_nmad_log [default] (show=DEFAULT (2))
register debug name: default [default] (show=2)
register debug name: log [default] (show=DEFAULT (2))
register debug name: nmad-log [log] (show=DEFAULT (2))
register debug name: tbx-log [log] (show=DEFAULT (2))
register debug name: ntbx-log [log] (show=DEFAULT (2))
register debug name: trace [default] (show=DEFAULT (2))
register debug name: nmad-trace [trace] (show=DEFAULT (2))
register debug name: ntbx-trace [trace] (show=DEFAULT (2))
register debug name: nm_so_sr_log [default] (show=DEFAULT (2))
register debug name: nm_so_sr_log [default] (show=DEFAULT (2))
register debug name: mpi_nmad_trace [default] (show=DEFAULT (2))
register debug name: mpi_nmad_transfer [default] (show=DEFAULT (2))
register debug name: nm_so_trace [default] (show=DEFAULT (2))
register debug name: nm_so_sr_trace [default] (show=DEFAULT (2))
register debug name: mpi_nmad_trace [default] (show=DEFAULT (2))
register debug name: mpi_nmad_transfer [default] (show=DEFAULT (2))
register debug name: nm_so_sr_trace [default] (show=DEFAULT (2))
register debug name: nm_so_trace [default] (show=DEFAULT (2))
pi is approximately 3.1415926544231318, Error is 0.0000000008333387
wall clock time = 0.003131
register debug name: init-log [log] (show=DEFAULT (2))
register debug name: init-log [log] (show=DEFAULT (2))
%

```

For example, if you want to trace all the calls to communications routines made by Mad-MPI, you can specify the debug parameter `mpi_nmad_transfer`.

```

% mpirun -machinefile mac -np 2 cpi --debug:mpi_nmad_transfer
##### bolero.labri.fr
##### bolero.labri.fr
[mpir_isend_wrapper] Sent --> gate 0 : 4 bytes
[mpir_isend_wrapper] Sent finished
[MPI_Wait] Calling nm_so_sr_swait
[MPI_Wait] Returning from nm_so_sr_swait
[mpir_irecv_wrapper] Recv --< gate 0: 4 bytes
[mpir_irecv_wrapper] Recv finished, request = 0xbf84118c
[MPI_Wait] Calling nm_so_sr_rwait for request=0xbf84118c
[MPI_Wait] Returning from nm_so_sr_rwait
[mpir_isend_wrapper] Sent --> gate 0 : 8 bytes
[mpir_isend_wrapper] Sent finished
[MPI_Wait] Calling nm_so_sr_swait
[MPI_Wait] Returning from nm_so_sr_swait
[mpir_irecv_wrapper] Recv --< gate 0: 4 bytes
[mpir_irecv_wrapper] Recv finished, request = 0xbf84118c
[MPI_Wait] Calling nm_so_sr_rwait for request=0xbf84118c
[mpir_irecv_wrapper] Recv --< gate 0: 8 bytes
[mpir_irecv_wrapper] Recv finished, request = 0x892d4a0
[MPI_Wait] Calling nm_so_sr_rwait for request=0x892d4a0
[MPI_Wait] Returning from nm_so_sr_rwait
pi is approximately 3.1415926544231318, Error is 0.0000000008333387
wall clock time = 0.004468
[mpir_isend_wrapper] Sent --> gate 0 : 4 bytes
[mpir_isend_wrapper] Sent finished
[MPI_Wait] Calling nm_so_sr_swait
[MPI_Wait] Returning from nm_so_sr_swait
[MPI_Wait] Returning from nm_so_sr_rwait
%

```