ANL/MCS-TM-ANL-98/xx

User's Guide for MPE: Extensions for MPI Programs

by

Contents

2.2 Logfile Formats

You can find an example of the use of the MPE graphics library in the directory mpich/mpe/contrib/mandel. Enter

```
make
mpirun -np 4 pmandel
```

to see a parallel Mandelbrot calculation algorithm that exploits several features of the MPE graphics library.

2.4 Other MPE Routines

Sometimes during the execution of a parallel program, you need to ensure that only a few (often just one) processor at a time is doing something. The routines MPE_Seq_begin and MPE_Seq_end allow you to create a "sequential section" in a parallel program.

The MPI standard makes it easy for users to define the routine to be called when an error is detected by MPI. Often, what you'd like to happen is to have the program start a debugger so that you can diagnose the problem immediately. In some environments, the error handler in MPE_Errors_call_dbx_in_xterm allows you to do just that. In addition, you can compile the MPE library with debugging code included. (See the -mpedbg

memory buffers are collected and merged in parallel during `MPI_Finalize`. During execution, `MPI_Pcontrol`

2.5.4 Real-Time Animation

The third library does a simple form of real-time program animation. The MPE graphics library contains routines that allow a set of processes to share an X display that is not associated with any one specific process. Our prototype uses this capability to draw arrows that represent message tra□c as the program runs. Note that MPI programs can generate

- F2CMPI_LIBS - The compiler flag needed to link Fortran to C MPI wrapper library with all the above mentioned libraries. For `MPICH`, this should be `-lfmpich`. Otherwise, it could be `-lmpe_f2cmpi`, MPE's own Fortran to C MPI wrapper library.
- FLIB_PATH - The full compiler flag needed to link Fortran MPI programs with the logging As part of the

We generate these routines by writing the “do something” parts only once, in schematic form, and then wrapping them around the PMPI_

3 Using MPE

The Multi-Processing Environment (MPE) attempts to provide programmers with a complete suite of performance analysis tools for their MPI programs based on post processing

lib/ contains all the libraries that user program needs to link with.

bin/ contains all the utility programs that user needs to use.

sbin/ contains the MPE uninstall script to uninstall the installation.

share/ contains user read-only data. Besides 'share/examples/', user usually does not

3.4.1 Log Format Converters

clogTOslog2:

```
mpiexec -default -env MPE_LOGFILE_PREFIX <output-logname-prefix> \
                 -env TMPDIR <local-tmp-dir> : -n 32 <executable-name>
```

For other MPI implementations, how environmental variables are passed remains unchanged. User needs to get familar with the environment and set the environmental variables accordingly.

3.5.3 Viewing Logfiles

MPE's install directory structure is the same as MPICH's and MPICH2's. So all MPE's utility programs will be located in 'bin/

`--with-fflags=MPE_FFLAGS` Specify extra FFLAGS to the F77 and `MPI_F77` compilers, e.g. `-64` for IRIX64 F77 compiler

`--with-mpiinc=MPI_INC` Specify compiler's include flag for MPI include directory, e.g. `-I/pkgs/MPI/include` for '`mpi.h`'

`--with-mpilibs=MPI_LIBS` Specify compiler's library flag for MPI libraries, e.g.

`--with-java2=JAVA_HOME` Specify the path of the top-level directory of the Java, JDK, installation for Jumpshot-3 only. If this option or `--with-java` is not given, Jumpshot-3's configure will try to locate JDK for you to build Jumpshot-3. For performance reason, it is recommended to use the latest Java 2, i.e. JDK-1.3.x, to build Jumpshot-3.

`--with-wishloc=WISHLOC`

```
${MPE_SRC_DIR}/configure --with-mlilibs=-lmli \
                         --with-java=/usr/java-1.1.6/usr/java
make
make install PREFIX=${MPE_INSTALL_DIR}mn.9[(m9.98␣13.for)-333(the)-334(64)-334(bit)
```

Using MPE with LAM for fortran MPI program is not working until recently. Configure options listed above enable MPE's internal Fortran to C MPI library. To use LAM's Fortran to C MPI library in LAM 6.3.3 or later, 'liblamf77mpi.a', do

```
setenv MPI_CC ${LAM_INSTALL_DIR}/bin/mpicc
setenv MPI_f77 ${LAM_INSTALL_DIR}/bin/mpif77
${MPE_SRC_DIR}/configure --with-mpilibs="-L${LAM_INSTALL_DIR}/lib -lpmpi" \
                         --with-f2cmpilibs=-llamf77mpi \
                         --with-java1=/sandbox/jdk117_v3 \
                         --with-java2=/sandbox/jdk1.3.1
make
make install PREFIX=${MPE_INSTALL_DIR}
```

LAM 6.5.6 to 6.5.9 has a bug that interferes with MPE's configure and make in enabling byte swapping on little endian machine, e.g. intel box. For details, see

http://www.lam-mpi.org/MailArchives/lam/msg04894.php

http://www.lam-mpi.org/MailArchives/lam/msg04896.php

Solution: Upgrade to newer version of LAM.

LAM 7.0 has included 'libpmpi.a' into 'libmpi.a', so –with-mpilibs may not be needed.

- For an existing MPICH, do

```
setenv MPI_CC ${MPICH_INSTALL_DIR}/mpicc
setenv MPI_F77 ${MPICH_INSTALL_DIR}/mpif77
${MPE_SRC_DIR}/configure  --with-f2cmpilibs=-lfmpich \
                          --with-mpelibname=newMPE \
                          --with-java1=/sandbox/jdk117_v3 \
                          --with-java2=/sandbox/jdk1.3.1
make
make install PREFIX=${MPE_INSTALL_DIR}
```

used by itself. This is only optional and is of use only if you wish to install the MPE

from Blackdown. Using SuperX instead of Exceed causes Jumpshot-2

might expand to:

```
static int MPI_Send_ncalls_2;
static int MPI_Recv_ncalls_2;
```

(end of example)

```
{{fnall <function name escape> <function A> <function B> ... }}
  ...
  {{callfn}}
  ...
{{endfnall}}
```

{{fnall}} defines a wrapper to be used on all functions except the functions named. Wrappergen will expand into a full function definition in traditional C format. The {{callfn}} macro tells wrappergen where to insert the call to the function that is being profiled. There must be exactly one instance of the {{callfn}} macro in each wrapper definition. The macro specified by <function name escape> will be replaced by the name of each function.

Within a wrapper definition, extra macros are recognized.

{{vardecl <type> <arg> <arg> ... }}

Use vardecl to declare variables within a wrapper definition. If nested macros request variables through vardecl with the same names, wrappergen will create unique names by adding consecutive integers to the end of the requested name (var, var1, var2, ...) until a unique name is created. It is unwise to declare variables manually in a wrapper definition, as variable names may clash with 7ther wrappers, and

Acknowledgments

The work described in this report has benefited from conversations with and use by a large number of people. We also thank those that have helped in the implementation of MPE, particularly Edward Karrels.

Debbie Swider had maintained MPE in MPICH-1.2.0 and before. Omer Zaki did the first implementation of Jumpshot, Jumpshot-2. Abhi Shandilya did the original clog2slog converter. Anthony Chan implemented SLOG-API, extended Jumpshot-2 to Jumpshot-3 to

SLOG-API clog2slog Jumpshot-2

Ju