Appendix D: MPP MM5 - The Distributed Memory (DM) Extension

Overview -

Users may download additional components that support execution of the model on distributed memory (DM) parallel machines. The DM option to MM5 has been implemented so that users who do not wish to run on these types of machines may simply download and build the model as before without downloading the additional components for the DM option. Further, all additional components for the DM option reside in a new directory, MPP, within the top-level MM5 source directory and this MPP directory need exist only if the DM extension is to be employed. Therefore, the first step to obtaining and building the model for a distributed-memory parallel machine is to obtain and unarchive the main model distribution file (a compressed UNIX tar archive file) into a directory. Then, a secondary DM-specific distribution file (also a compressed UNIX tar archive file) is unarchived, creating the MPP directory. After downloading, uncompressing, and unarchiving the main model and the DM-specific components, the model may be compiled and run. The first section of this tutorial provides details and step by step instructions for downloading and unarchiving the main model and DM-specific components. The next section describes configuration and compilation of the model. This is followed by a section with information on running the model on a distributed memory parallel machine.

Additional sources of information on the DM-parallel option to MM5 include the on-line Helpdesk, which is an archive of support email, the Tips and Troubleshooting page, and the Benchmarking page.

These instructions have been updated for MM5 Version 3.

Downloading -

All the standard job decks and source code tar files needed to run the MM5 model and its pre- and post- processors are available via anonymous ftp to ftp://ftp.ucar.edu/mesouser/MM5V3. The model and the additional code for the DM-parallel option are distributed in two archive files:

- MM5.TAR.gz, which contains the standard distribution of the current release of MM5, and
- MPP.TAR.gz, the MPP directory and contents comprising the DM option to MM5.

If you do not intend to use the DM-parallel option to run on a distributed memory parallel machine, you do not need the second file nor do you need to read further in this documentation. Please refer to the standard MM5 documentation for instructions on running MM5 without the DM-parallel option.

If you intend to run the model on a distributed-memory parallel computer using the DM-option, download both these files onto your local machine. If you are using ftp directly, be sure to type “bin” at the ftp prompt before downloading, to make certain that the files are transferred in binary format. Otherwise, you may be unable to uncompress and unarchive the files.

Note: Version 3 test data is available in the TESTDATA subdirectory of the MM5V3 directory on the ftp site.

Once you have downloaded the files onto your local machine, they should be uncompressed and unarchived. The following sequence of UNIX shell commands will create a complete MM5 source directory directly below the one in which the MM5.TAR.gz and MPP.TAR.gz archive files are stored. The commands will then fill the directory with the necessary files and directories to compile the model with the DM-parallel option. Upon completion of the commands, the current working directory will be the just-created MM5 top-level source directory. The "%" (percent) characters are not to be typed; they represent the command line prompt, which may be different on your system.

```
% gzip -d -c MM5.TAR.gz | tar xf -
% cd MM5
% gzip -d -c ../MPP.TAR.gz | tar xf -
```

At this point, the MM5 code and the DM-option components have been unarchived into your MM5 directory. The next section describes how to configure and compile the model.

Configuration and compilation -

Configuring, compiling, and running the model with the distributed-memory parallel option involves the following steps:

1. Modify the configure.user file to reflect the machine-specific and scenario-specific settings for your run.
2. Compile the model to use the distributed memory parallel option.
3. Create the job deck for your run (optional).
4. Submit the job (platform specific).

1. Configuring the model

The configure.user file in the top level of the MM5 directory comprises sections for machine- and scenario-specific settings, some of which may need to be modified. The sections are as follows:

1. System variables: not affected
3. Fortran Options: not affected (these are ignored when compiling with "make mpp")
4. General commands: not affected
5. Model configuration: NHYDRO, FDDAGD, FDDAOBS, MAXNES, MIX, MJX, MKX
6. Physics options: IMPHYS, MPHYSTBL, ICUPA, IBLTYP, FRAD, ISOIL, ISHALLO
7. MPP options: this section begins with a number of important settings that apply to all DM-parallel platforms. Two important settings are PROCMIN_NS and PROCMIN_EW used to reduce the memory usage of the model on larger numbers of processors. (Please see "Notes on the PROCMIN variables" at the end of this tutorial for more details on the PROCMIN variables.) The remainder of the section is analogous to Section 3 and contains sets of compiler, linker, and other options for the computers on which the DM-parallel option is currently supported. The settings for your computer should be uncommented (remove the leading # character) and edited as necessary.

Please refer to the sample configure.user file for details.

Except for Section 7, the configure.user file that is distributed with the model is set up already for the "Storm of the Century" case whose data is available in the ftp TESTDATA directory. A configure.user file is provided with the largedomain case, with the following case-specific settings:

```
MAXNES = 1
MIX = 200
MJX = 250
MKX = 27
```

Note that MM5 Version 3 is distributed with a configure.user.linux file but this is not intended for use with the DM-parallel option. Use the settings in the appropriate subsection of Section 7 in the configure.user file for Linux Beowulf clusters of PCs.

2. Making the Executable

Issue the command "make mpp" from the main MM5 directory. Assuming a successful compile, the resulting executable file will be named mm5.mpp and it will appear in the directory named "Run" in the main MM5 directory. Note that the first time you execute this command after downloading and unarchiving the file in the MM5 directory, some additional installation will occur; specifically, software in the MPP directory will be compiled and some additional directories and links between files will be set up (e.g., MM5/MPP/build).

VERY IMPORTANT: Once the DM-parallel version has installed in a directory it will not compile properly if moved or copied to another location without first uninstalling and allowing the code to re-install itself in the new location. This is because there are certain absolute directory paths that are set up during installation; moving the model invalidates these paths. It is simple to uninstall the model by simply typing "make uninstall." To rebuild the code in a different configuration when it hasn't been moved from it's installed location, it is sufficient to type "make mpclean".

There is additional discussion of compilation issues for the DM-parallel version of the code at the end of this document.

3. Making the mm5.deck (optional)

Issue the command "make mm5.deck". This will produce a file named mm5.deck in your MM5 directory.

The original purpose of the mm5.deck in MM5 was to generate the namelist (mmlif) file, set up links to data files, and then execute the model and non-DM parallel users may still use it this way. However, because file systems, run commands, and queuing mechanisms vary from installation to installation, the mm5.deck files that are generated for DM-parallel platforms do little more than generate the namelist. Even then, the mm5.deck file created by "make mm5.deck" may still need to be edited to accomplish your specific run. In particular, length of the run in minutes (TIMAX), the length of the model time step in seconds (TISTEP), output frequency in minutes (TAPFRQ), whether to write model restarts (IFSAVE), the run time size of your domain(s) (NESTIX, NESTJX), their spatial location (NESTI, NESTJ), and time to start and end (XSTNES, XENNES) may need to be changed either in the mm5.deck or in the mmlif namelist file that

results from executing the mm5.deck.

The mm5.deck generated by default (with the configure.user file that is distributed with the model) is suitable for use with the Storm of the Century case data. The largedomain data case comes with an mm5.deck already included, however, in this case it may be necessary to edit the resulting mmlif file to change occurrences of the Fortran-77 style "&END" to the Fortran-90 style "/" delimiters (IBM).

Running the Model -

In the Run directory, or in some other run directory, the following files or symbolic links should exist:

mm5.mpp	Executable file
BDYOUT_DOMAIN1	Copy of or symbolic link to lateral boundary conditions file
mmlif	Copy of or symbolic link to mmlif namelist file
LOWBDY_DOMAIN1	Lower boundary conditions
LANDUSE.TBL	Land surface properties (distributed with MM5)
MMINPUT_DOMAIN1	Initial conditions file for coarse domain
MMINPUT_DOMAIN[2-9]	Initial conditions (IOVER=1)
TERRAIN_DOMAIN[2-9]	Terrain file (IOVER=2) for nests
MMINPUT2_DOMAIN1	Used only for FDDA on IBM, a copy of MMINPUT_DOMAIN1
restrts	Directory for restart files (only needed if using restarts)

Run the parallel model using the command for your system. Many parallel machines will use some version of the mpirun command. Here are some examples:

1) mpirun -np 16 mm5.mpp
2) mpirun -np 16 -machinefile machines mm5.mpp
3) dmpirun -pf procfile
4) mprun -np 4 mm5.mpp
5) poe mm5.mpp -rmpool 1 -procs 2 -pgmmodel spmd

Examples (1) and (2) run the code on 16 processors. The -machinefile option (compatible with MPICH) in example (2) allows you to specify processors in a file, here named "machines". Examples (3), (4), and (5) are for DEC MPI, Sun MPI, and IBM, respectively. Check the manual pages for the mpirun command on your system. You may also need to interact with a queuing system or batch scheduler. If you are running on a Linux cluster, please see "Note on Linux systems"

at the end of this tutorial.

When the job runs, it will create files in the run directory:

MMOUT_DOMAIN1	Output files from coarse domain
MMOUT_DOMAIN[2-9]	Output files from nests
rsl.out.0000	Standard output from processor zero
rsl.out.0001, rsl.out.0002, ...	Standard output from other processors
rsl.error.0000	Standard error from processor zero
rsl.error.0001, rsl.out.0002, ...	Standard error from other processors

Additional notes on compilation

As with the non-distributed memory versions of the model, the distributed-memory version is compiled using the UNIX make utility and controlled by settings within the configure.user file in the main MM5 directory. This section provides some additional detail that may be helpful in working with the DM-parallel model.

> *Design note:* A general note on design philosophy: users and developers will notice some additional complexity within the build mechanism for the DM option. There is a conservation of complexity principle at work here: we avoided introducing changes for DM parallelism (message passing, loop and data restructuring, etc.) into the MM5 Fortran source by transferring it, instead, to a library (RSL), a source translator (FLIC), and into the MM5 build mechanism. Thus, the changes for DM parallelism are effected automatically and transparently to the source code in a series of precompilation steps, without the necessity of a separate version of the model source code. Most of the complexity of the DM parallel build mechanism has been hidden from users within several commands that are close in look and usage to the commands that are used to build the non-DM version of MM5. Even so, some knowledge of what is going on behind the scenes when the model is built is important and is provided in the discussion that follows.

There are several DM-specific options to the “make” command. The command “make mpp” is used to build the model and this command is analogous to just typing “make” to build the non-DM version. The command “make mpclean” is used to put the code back to a non-configured state by deleting object (.o) files from the source tree; it is analogous to the “make clean” command for the non-DM version. The command “make uninstall” is unique to the DM version of the model, and it is used to return the code to a pristine, more or less as-downloaded state.

Command: **make mpp**

The “make mpp” command builds (compiles) the model based on the settings in the configure.user file and places the resulting executable, mm5.mpp, in the directory named Run in the top-

level MM5 directory. The first time after unarchiving the MM5.TAR.gz and MPP.TAR.gz files into the MM5 directory, the “make mpp” command installs software needed by the DM version in the directory named MPP in the MM5 directory prior to building the model. It also sets up the MPP/build directory and creates symbolic links to MM5 source files in the other directories of the MM5 source tree. The DM code is compiled only in the MPP/build directory, not in the various subdirectories of the MM5 source tree, where the non-DM version is compiled. This, and the fact that the executable file has a different name from the non-DM version, allows both the DM and non-DM versions of the model to be compiled and coexist within the same MM5 directory. Subsequent invocations of “make mpp” will only recompile the model and not re-install the MPP software.

Prior to compilation, the MPP/build directory will contain a Makefile.RSL and a large number of files, actually symbolic links, with names that end in .F and a few that end in .c .

> *Note* to programmers and developers: since these are symbolic links you may either edit the source files in their local directory (e.g. dynamics/nonhydro/solve3.F) or through the symbolic link (MPP/build/solve3.F). After compilation, the directory will also contain the intermediate object files with names that end in .o. The “make mpp” command will only recompile files that have been touched since their corresponding .o file was compiled. For debugging purposes, it is possible to also have the .f files retained in the MPP/build directory. To do this, uncomment (remove the # character) from the line “# RM = echo” in the file MPP/build/Makefile.RSL (the file will only exist in this directory if the code has been made using “make mpp” once). This will prevent intermediate files from being deleted between compiles.

Command: **make mpclean**

The “make mpclean” command is used to restore the model to a clean, un-compiled state. The “make mpclean” command causes all of the files ending with .o and any other intermediate files in the MPP/build directory to be removed. This should be done whenever the model is reconfigured in the configure.user file, prior to rebuilding the code with the “make mpp” command. Usually, when making code changes to MM5 source files, it is not necessary to “make mpclean” between each compile. However, any time code describing a COMMON block is altered, the model should be completely rebuilt using “make mpclean” first.

Command: **make uninstall**

The “make uninstall” command undoes what was done to install the DM option when the “make mpp” command was executed the first time. It uninstalls software needed by the DM option and it removes the MPP/build directory, including all symbolic links. The effect is to restore the code to an “as downloaded” state with respect to the DM option (it won’t undo changes you’ve made to the configure.user file or to model source files). Because the installation of the DM option sets up some non-relative directory path information within the MPP directory, the “make uninstall” command should be used whenever the code is moved to a different directory. Otherwise, there are very few instances for which the “make uninstall” command is necessary; “make mpclean” will usually suffice.

Notes on the PROCMIN variables

The PROCMIN_NS and PROCMIN_EW variables, which appear in the section 7 of the configure.user file, can be used to reduce the memory required by the DM-parallel model when running on a larger numbers of processors. These two variables determine the horizontal dimensions of MM5 arrays for each processor AT COMPILE TIME. Roughly speaking, PROCMIN_NS divides MIX and PROCMIN_EW divides MJX. (This is not exact, due to "pad" or "ghost" cells maintained in the DM-parallel arrays.) Therefore the larger you make PROCMIN_NS and PROCMIN_EW, the smaller the per processor memory. The product (PROCMIN_NS x PROCMIN_EW) specifies the MINIMUM number of processors for which the MM5 executable is valid. If you try running the model with LESS processors than this product, the internal array dimensions (determined at compile time!) will not be large enough to handle the effective subdomain sizes on each processor. If this happens, the model will die with the runtime abort message in the rsl.error.0000 file:

'MPASPECT: UNABLE TO GENERATE PROCESSOR MESH. STOPPING.'

In this case, you must either run with more processors, or reduce PROCMIN_NS and/or PROCMIN_EW and recompile.

An executable compiled with PROCMIN_NS=1 and PROCMIN_EW=1 uses the maximum per processor memory, but is valid for running with any number of processors. For the most efficient use of memory, set PROCMIN_NS and PROCMIN_EW so that their product equals the number of processors you will be using. For a given product (e.g., 16 processors), runtimes might vary up to 10 or possibly even 15 percent based on the ratio of PROCMIN_NS to PROCMIN_EW. For example, on most systems we have found that setting PROCMIN_NS=2 and PROCMIN_EW=8 (2x8 decomposition) is more efficient than a 4x4 or 8x2 decomposition.

Note on Linux systems

Users of Linux clusters should be aware that details for configuring, compiling, and running the MM5 model will vary based on their particular system configuration and installation. Processor chip, operating system, compilers, and interconnect are all variables that make your system unique. If you are experiencing problems compiling and/or running the model on your system, you should scan the on-line Helpdesk for Linux-specific notices.

Note on time-series output

If one sets IFTSOUT = .TRUE., and defines TSLAT and TSLON for the time-series locations, one will obtain time-series output in fort.26 for domain 1, fort.27 for domain 2 and so on for **serial** runs.

For MPI runs, the time-series output is not written to the fort.2? files, but rather (unfortunately) scattered in the various rsl.out.* files. The rsl.out.* file which will contains the time-series output will correspond to the processor that calculated the time-series. This means that if 2 time-series outputs were requested, they may be in written to two different rsl.out.* files, as their respective locations may have placed them on two different processors.

Note on restart runs

For serial runs, SAVE_DOMAINn files are created, which needs to be renamed to RESTART_DOMAINn before submitting a restart job.

For MPI runs, restart files are written to a directory MM5/Run/restrt
These files already have the correct naming convention for a restart, so no renaming is required.

A restart file will be written for each processor. As these restart files ONLY contain information pertinent to the processor that created them, it is essential to restart with EXACTLY the same number of processors that was used during the initial run.