Getting Started

October 2009

Table of Contents:

I. Configuration

You are a registered CDP mozart4 user and have successfully downloaded, "gunzipped", and "untarred" the files mz4_release.tar.gz and mozart_data.tar.gz from the NCAR Community Data Portal (CDP). Now you are ready to begin preparing and running MOZART-4 simulations. Before doing anything else you need to configure mozart for your computing environment. Once configuration is successfully completed it should be relatively simple and straightforward to build and run the supplied standard simulation. And unless your computing environment changes you only need to successfully configure once. The standard simulation is not a lightweight "toy" model. In fact, it is the present production simulation.

In the following MZ4ROOT will refer to the "top" level mozart4 directory that has the following subdirectories:

- configuration
- model
- run
- preproc
- utils

To configure mozart get into the MZ4ROOT/configuration directory and issue the command:

cfg_mz4 data_dir=MZ_DATA_DIR/mozart_data

where MZ_DATA_DIR is the top level directory where you gunzipped and untarred the mozart input dataset file mozart_data.tar.gz. As an example lets say you placed the input dataset file in /data1/mozart_data. Then in the above command you would set:

data_dir=/data1/mozart_data

If all goes well cfg_mz4 will successfully complete and you should see something like the following if your operating system is Linux:

```
compiler is : pgf90

... Writing configuration file : MZ4.cfg

mz4 configuration summary
----------------------------------------------------------
mz4 configuration is SUCCESSFUL
mz4 can build executable for all modes
(hybrid, mpi, openmp, single cpu)


FYI: this computing environment does NOT have batch queuing
----------------------------------------------------------

... mz4 configuration details are in file cfg.log
```

This is taken from a machine that has the Portland Group fortran90 compiler pgf90 installed as the default fortran compiler. The key information is contained in the two lines:

"mz4 configuration is SUCCESSFUL"
"mz4 can build executable for all modes"

If you don't see this output then configuration will either have partially succeeded or failed altogether. What exactly is configuration doing? cfg_mz4 is checking your computing environment for the following:

- An **mpif90** script which automatically sets include files and libraries needed to compile programs using MPI
- A fortran90 compiler from either Portland Group, Intel, or Lahey in that order for systems with the Linux operating system
- The netcdf include file **netcdf.inc** and library **libnetcdf.a**

If you are running mozart in an IBM environment or on a larger LINUX cluster then it is almost certain all the listed steps will succeed (IBM installations running the AIX operating system have their own compiler and MPI script). If you are running on a smaller machine such as the Cray CX1 or the SGI OctaneIII or a workstation or even a laptop then configuration can be a little less straightforward depending on how your computing environment has been setup.

If cfg_mz4 finds a valid compiler it attempts to compile a simple fortran90 test program. If cfg_mz4 either can't find a valid compiler in your PATH environment variable or for some reason the test compilation fails then configuration will fail. If you believe you have one of the above listed fortran90 compilers on your system but it is not in your PATH environment variable then altering your PATH environment variable to include the fortran90 directory should cure the problem. After altering your PATH environment variable rerun "cfg_mz4" with the "data_dir=" option. If you do not have a listed fortran90 compiler then you need to acquire one of the above compilers.

As with the fortran90 compiler, if cfg_mz4 finds the **mpif90** script in your PATH or in one of a set of "usual" directories it will check the mpif90 script by compiling a short MPI program. Just as with the compiler, if you know that mpif90 is on your system and cfg_mz4 can't find it then the best course of action is to add the directory containing the mpif90 script to your PATH environment variable and rerun cfg_mz4. Alternatively you may issue the cfg_mz4 command with the "**mpi_dir=**" option pointing to the directory where the mpif90 script resides. It is very rare for the mpif90 compilation test to fail. If this happens it usually indicates there is something wrong with the MPI installation. In that case you should consult your system administrator and/or a computer savvy colleague. If for some reason cfg_mz4 can't successfully complete the mpif90 configuration phase it will mean that you will only be able to run mozart in single cpu or OpenMP mode.

Finally, no matter what your computing environment, mozart needs to confirm the existence of valid netcdf include and library files. cfg_mz4 will try to find the netcdf files in a set of "usual" directories. If netcdf is installed then cfg_mz4 will test netcdf with a small test compilation. If the compilation test fails there is almost surely something amiss with the installed netcdf. If cfg_mz4 fails to find the netcdf files and you know netcdf is installed on your system then you should specify the path to the directory containing the netcdf library file, libnetcdf.a, by using the "**ncd_dir=**" command option (you can get a brief description of cfg_mz4, its use and options, by issuing the command "**cfg_mz4 help**"). Usually the netcdf directory is a full file path ending in /lib. If you don't have netcdf installed on your machine then acquire version 3.6.2 from the website http://www.unidata.ucar.edu/downloads/netcdf and follow the installation directions in "Installation and Porting Guide" on website http://www.unidata.ucar.edu/software/netcdf/docs.

In addition to the critical checks listed above cfg_mz4 also sets the:

- full path to the mozart4 root directory
- full path to the input data directory
- default model "mode" for compiling (default is HYB or OMP)
- computing site name (defaults to Local)
- batch queue command (set to NONE if there is no batch queuing system)

At the time of this distribution the only batch queuing system that is supported in the mozart "framework" is the NCAR supercomputer site. If you are using mozart at NCAR on the IBM supercomputer (bluefire) you should have the "site=NCAR" option as a cfg_mz4 argument. If you would like another batch queuing system directly supported send email to stacy@ucar.edu. With your help we will try to assist you but there are no guarantees.

These settings and others pertaining to the compiler, netcdf, and the mpif90 script are stored in the ascii file **MZ4.cfg**. You can find a listing of the configuration settings at the end of the **cfg.log** file which contains a slightly more detailed summary of the configuration process.

II. Preprocessor

Although not required to build and run the "standard" simulation supplied with mozart you will need to use the preprocessor to modify basic simulation components such as chemistry, boundary conditions, wet removal, dry deposition, history file output, etc. To build the preprocessor, get into the MZ4ROOT/preproc directory and issue the command:

make_mozpp

That should build the mozart preprocessor executable, moz_pp, in the directory MZ4ROOT/preproc/bin. The make_mozpp command uses the compiler stored in the configuration file MZ4.cfg to compile and link moz_pp.

To use the preprocessor get into the MZ4ROOT/preproc/inputs directory and issue the command:

mozpp *inputfile*

where *inputfile* is any mozart preprocessor input file. Only the file mz4_synoz_ncept42.inp is supplied with the release. Traditionally preprocessor input files have a .inp suffix but this is not a requirement. The preprocessor produces three files:

- **mozpp.subs.tar** (contains fortran90 source code required to build an executable)
- mz4_synoz_ncept42.doc (contains the "document" describing the preprocessed *inputfile*)
- mz4_synoz_ncept42.dat (contains history file(s) output controls)

The last two file names can be set in the preprocessor input file and will default to mz4.doc, mz4.dat otherwise. The first file name, **mozpp.subs.tar**, is fixed and can't be overridden.

Creating a modified simulation starts with the preprocessor and should proceed in the following fashion:

- copy an existing preprocessor file
- edit the copy
- preprocess the copy

Before making modifications, we encourage you run the preprocessor on the mz4_synoz_ncept42.inp file via the command:

mozpp mz4_synoz_ncept42.inp

and examine both the input file, mz4_synoz_ncept42.inp, and the "document" file mz4_synoz_ncept42.doc in the MZ4ROOT/preproc/output directory.

III. Building an executable

As distributed mozart does not include an executable. To build the "standard" executable get into the MZ4ROOT/model directory and issue the command:

make_mz4

This command relies on information in the configuration file. As with all mozart commands you can get command information by including the single option "help" as in:

make_mz4 help

If make_mz4 is successful, which should be the case, you will find the executable file **mz4_hyb** in the MZ4ROOT/model/bin directory. The hyb suffix in mz4_hyb reflects the default model mode setting in the configuration file. You can override the default naming with the "**exe=**" command option. The default mozart executable, mz4_hyb, has a t42 horizontal spatial grid. Suppose you wanted a separate simulation with a t63 horizontal grid. In this case you would create a new preprocessor input file, via the steps outlined above in the preprocessor section, with the spatial grid inputs set to t63 values (see MOZART4_preprocessor.pdf for details). Then, in the MZ4ROOT/model directory, create a new directory, pp_t63. Get into the pp_t63 directory and move the mozpp.subs.tar file from the MZ4ROOT/preproc/output directory to pp_t63. Then issue the command:

```
tar –xf mozpp.subs.tar
```

Alternately, in the MZ4ROOT/model/pp_t63 directory you could issue the command:

```
tar –xf ../../preproc/output/mozpp.subs.tar
```

and skip the step of moving the mozpp.subs.tar file. Once you have "untarred" the mozpp.subs.tar files in the pp_t63 directory get back into the MZ4ROOT/model directory and issue the command:

make_mz4 exe=mz4_t63_hyb usr_src=pp_t63

This will produce the executable mz4_t63_hyb in the MZ4ROOT/model/bin directory using the source files in the MZ4ROOT/model/pp_t63 directory. You will now have a t63 resolution companion to the standard t42 executable mz4_hyb. What would happen if you had forgotten to use the "exe=" option? Nothing catastrophic. However, you would overwrite the "standard" executable, mz4_hyb , since the executable name defaults to mz4_hyb for a hybrid executable. In this example specific file and directory names are for illustrative purposes only: you may use valid names of your choice.

IV. Running a simulation

Thus far you have configured mozart for your computing platform, built and used the preprocessor, and built the "standard" mozart executable. Now you're ready to run the "standard" simulation. The following assumes you will run a simulation from the command line, specifically you are not submitting the simulation to a batch queuing system. To run the standard simulation, get into the MZ4ROOT/run directory and issue the command:

run_mz4 &

Note that the simulation will run in background. If you run the simulation in foreground all the simulation ascii "log" file output will go to stdout, most likely your screen. In that case it will be hard for you to do anything else while the simulation runs. By default the simulation will run on one half of the available cpus on your workstation. If you have an eight cpu machine, the standard simulation, which only runs three time steps, should complete in a few minutes. What if you want to run a three day "standard" simulation with output to the history files once every simulation day? You don't need to recompile or even run the preprocessor in this case. However, you do need to create a new "simulation" namelist file. Every mozart simulation reads a "simulation" fortran namelist file during initialization. The namelist file specifies simulation temporal controls such as the basic time step, start date, and duration as well as defining dataset file specifications. The default simulation namelist is **mz4.tpl.nml**. As with procedures for creating new preprocessor input files you should make a copy of mz4.tpl.nml, edit the copy, and then use the "nml=" option on the mz4_run command. If the copy namelist is named mz4.3d.nml then you would edit mz4.3d.nml implementing the following alterations:

Namelist variable	old value	new value
sim_duration	‘1h’	‘3d’
hstfrq(1)	‘1h’	‘1d’
hstfrq(2)	‘1h’	‘1d’

Just before the end of the workday you could run the three day simulation via:

```
run_mz4 nml=mz4.3d.nml full_node &
```

Remember, by default run_mz4 will use one half of the available cpus. On an eight cpu MacPro the three day simulation would take about 90 minutes. Since you aren’t going to be working on your MacPro until tomorrow you decide to use all eight cpus and shorten the simulation wall clock time by two. Hence the “full_node” option.

As with cfg_mz4, make_mozpp, mozpp, and make_mz4 you can get command information by issuing the command with the “help” option as in:

```
run_mz4 help
```

In building the executable we saw that it was very possible to overwrite existing executables when command defaults are not superceded with explicit command options. This is also the case for the history files with the run_mz4 command. Unfortunately, at the time of this release, there are no options to handle redirecting history and restart file output to directories other than the default **hist** and **rest** subdirectories in MZ4ROOT/run. Thus you could finish one simulation, start a different simulation, and inadvertently write over the prior history files! The best way to avoid this trap is to move all the files in the **res**t and **hist** subdirectories to a uniquely named directories before beginning a new simulation. In fact, there is no harm in moving the entire directory as in:

```
mv hist /data1/mozart/mz4_t63/hist
mv rest /data1/mozart/mz4_t63/rest
```

V. Summary

Assuming configuration goes without incident you can run a short, “standard” mozart simulation by issuing the following commands:

- **cfg_mz4** in MZ4ROOT/configure
- **make_mz4** in MZ4ROOT/model
- **run_mz4** in MZ4ROOT/run

That’s all it takes to get started with mozart. Unless you are content to just run the “standard” simulation as supplied with the history outputs supplied then you will want to build and use the mozart preprocessor.

Don’t forget that each of the above commands and the **make_mozpp** and **mozpp** commands have a built in help facility invoked via:

command help

where **command** is a place holder for the actual mozart command.

Additional documentation will be posted as it is prepared on the MOZART-4 website:
    http://gctm.acd.ucar.edu/
and on the MOZART-4 wiki page:
    https://wiki.ucar.edu/display/mozart4/Home
Please consult these pages for additional information.