FRIDAY: Run CCSM Tutorial (Jen, Jack)

Overview: You will be running the latest CCSM tag (ccsm4_0_beta19) with 4x5 degree atmosphere and a ~3 degree ocean. We recommend doing the cases in the order listed below. At the very least, make sure you understand the steps to run the ccsm_vanilla case. For the other cases, don't worry about getting through all of these runs/analysis. Complete in order and get as far as you can get.

CASE1) ccsm_vanilla. The goal of this case is to get you familiar with configuring and running CCSM using the standard CCSM scripts. No modifications are made to the scripts at all. Follow the instructions on page 2 of this handout " I: STEPS TO RUN A CCSM CASE - NO MODIFICATIONS".

CASE2) ccsm_baseline. The goal of this case is to teach you how to make minor modifications to the CCSM scripts (e.g., run length, automatic restart, changing the CAM namelist to save output at every timestep). Follow the more detailed instructions on starting on page 4 of this handout " II: STEPS TO RUN A CCSM CASE - MODIFICATIONS".

CASE3) ccsm_codemod The goal of this case is to teach you how to make CAM source code modifications within the framework of the CCSM scripts. You will change rhminl and compare the results to ccsm_baseline using simple tools (ncview, ncdiff). Follow the more detailed instructions on starting on page 4 of this handout "II: STEPS TO RUN A CCSM CASE - MODIFICATIONS".

CASE4,CASE5) ccsm_long, **ccsm_long_doublec02** The goal of these cases is to get you familiar with using the CAM diagnostics package on CCSM output. To save time, Jack and Jen ran these CCSM cases in advance ("like a cooking show..." says Dani). ccsm_long has 2000 C02 concentrations while ccsm_long_doubleco2 has doubled 2000 C02 concentrations. Follow the more detailed instructions on starting on page 7 of this handout " III: RUNNING THE CAM DIAGNOSTICS PACKAGE ON CCSM OUTPUT".

I. STEPS TO RUN A CCSM CASE - NO MODIFICATIONS

1) COPY CCSM CODE TO YOUR HOME DIRECTORY

The following command will create a directory with the CCSM code in ~$USER (e.g., mine is ~jenkay). You only need to copy the code once.

```
cp -R /fs/cgd/csm/collections/ccsm4_0_beta19 ~$USER
```

2) CREATE A NEW CASE

First, go to the script directory.

```
cd ~$USER/ccsm4_0_beta19/scripts
```

Then, run the create_newcase script to generate a new case. The options in the create_newcase script are:
-case = home directory for case. For this example, "/ptmp/$USER/ccsm_vanilla".
-mach = machine you are running on. For this tutorial, "bluefire".
-res = resolution. For this tutorial, "4x5_gx3v5".
-compset = model configuration For this tutorial, "B_2000_TRACK1", which is a fully coupled CCSM with 2000 gases and TRACK1 physics. For reference, fully coupled CCSM runs are called "B cases" while an "F case" is a CAM stand-alone run.

```
./create_newcase -case ~$USER/ccsm_vanilla -mach bluefire -res 4x5_gx3v5 -
compset B_2000_TRACK1 -skip_rundb
```

3) CONFIGURE THE MODEL (generate the namelist, prestage, and build scripts)

First, go to the case directory.

```
cd ~$USER/ccsm_vanilla
```

Then configure the case. If successful, prints "Successfully configured the case for bluefire"

```
configure -case
```

4) COMPILE CODE, AND BUILD NAMELISTS

To compile CCSM and build the required namelists, the following command is used: If successful, prints "CCSM BUILDEXE SCRIPT HAS FINISHED SUCCESSFULLY".

```
./ccsm_vanilla.bluefire.build
```

5) RUN CCSM

Before running CCSM, you should check and if necessary modify the project number (-P, 37591047), queue (-q), and wall clock (-W, 6 hour limit in regular). Don't change the MPI settings (e.g., -n, ptile, -a etc.).

```
less ccsm_vanilla.bluefire.run
nedit ccsm_vanilla.bluefire.run &
```

Next, submit the job using:

bsub -U RESERVATIONID < ccsm_vanilla.bluefire.run

You can monitor the job by typing: **bjobs**

JOBID USER STAT QUEUE FROM_HOST EXEC_HOST JOB_NAME SUBMIT_TIME
726687 $USER PEND regular be1005en *_baseline Jul 23 15:09

You can kill the job by typing: **bkill JOBID**

Job <726675> is being terminated

You can see the bluefire queue by typing: **batchview**

Output files will appear in the run directory, in this case /ptmp/$USER/ccsm_vanilla/run. There are atmosphere model output files (e.g., ccsm_vanilla.cam2.r.0001-01-06-00000.nc), ice model outputs (ccsm_vanilla.cice.r.0001-01-06-00000), land model outputs (ccsm_vanilla.clm2.r.0001-01-06-00000.nc), and ocean model outputs (ccsm_vanilla.pop.rh.0001-01-06-00000.nc). Because the run only went for 5 days, the only output files are restarts and no monthly history files were produced.

II. STEPS TO RUN A CCSM CASE WITH MODIFICATIONS

1) COPY CCSM CODE TO YOUR HOME DIRECTORY

You only need to copy the code once. If you did this for the ccsm_vanilla case - there is no need to do it again. You can create multiple cases from this set of CCSM code.

```
cp -R /fs/cgd/csm/collections/ccsm4_0_beta19 ~$USER
```

2) CREATE A NEW CASE

First, go to the script directory.

```
cd ~$USER/ccsm4_0_beta19/scripts
```

Then, run the create_newcase script to generate a new case. This will be done for each case.

```
./create_newcase -case ~$USER/ccsm_baseline -mach bluefire -res 4x5_gx3v5 -
compset B_2000_TRACK1 -skip_rundb
```

```
./create_newcase -case ~$USER/ccsm_codemod -mach bluefire -res 4x5_gx3v5 -
compset B_2000_TRACK1 -skip_rundb
```

3) CONFIGURE THE MODEL (generate the namelist, prestage, and build scripts)

First, go to the case directory. For example,

```
cd ~$USER/ccsm_baseline
```

Then, check or modify settings. To check the files, use the "less" command.

```
less env_conf.xml
less env_mach_pes.xml
```

You will need to use the xmlchange command or a text editor (vi, nedit, emacs) to change values in these files. For example, when Jack and I ran the ccsm_long case, we changed the run start date in env_conf.xml from "<entry id="RUN_STARTDATE" value="0001-01-01" />" to "<entry id="RUN_STARTDATE" value="0001-12-01" />".

```
nedit env_conf.xml &
xmlchange -file env_conf.xml -id RUN_STARTDATE -val 0001-12-01
```

If you are running on bluefire, there is no need to change the settings in env_mach_pes.xml. The MPI settings have been optimized for bluefire by professionals. After you have made modifications, configure the case. If successful, prints "Successfully configured the case for bluefire"

```
configure -case
```

4) MODIFY NAMELISTS OR CAM SOURCE CODE

Changes to the namelist or the CAM code must be made before compiling and building the namelists.

For CASE2 (ccsm_baseline) and CASE3 (ccsm_codemod), use a text editor to modify the CAM namelist so that several instantaneous values are saved. Type,

```
nedit Buildconf/cam.buildnml.csh &
```

Add the following three lines of code to the cam_inparm section of the code starting at line 72 of cam.buildnml.csh. fincl2 sets the variables to save in the instantaneous output file. nhtfrq(2) sets the frequency at which the values are saved, "1" means save the values at every timestep. mfilt(2) sets the number of values saved in each instantaneous output file. Here, we set it to a large value so that all of the output is saved in one file.

```
fincl2 = 'PSL','CLDTOT','CLDLOW','CLDHGH','CLOUD'
nhtfrq(2) = 1
mfilt(2) = 1000
```

For CASE3 (ccsm_codemod), modify the CAM source code. The CAM source code for the tag we are using is located at: /blhome/$USER/ccsm4_0_beta19/models/atm/cam/src/physics/cam. Here, we will make a change to a parameter in the cloud fraction parameterization code (cloud_fraction.F90). First, copy the code you wish to modify into the SourceMods directory.

```
cp
/blhome/$USER/ccsm4_0_beta19/models/atm/cam/src/physics/cam/cloud_fraction.F90
~$USER/ccsm_codemod/SourceMods/src.cam/
```

Then, use a text editor to modify the code. For example,

```
nedit SourceMods/src.cam/cloud_fraction.F90 &
```

Modify the rhminl parameter in cloud_fraction.F90 at line 224. rhminl sets the minimum relative humidity for cloud formation. If you increase (decrease) the rhminl value, the model will have less (more) cloud cover.

5) COMPILE CODE, AND BUILD NAMELISTS

To compile CCSM and build the required namelists, type:

```
./ccsm_baseline.bluefire.build
```

Note that compiling and building the namelists are done one step as opposed to two steps in CAM stand-alone. As the model is compiling, you can check status by monitoring text printed to the screen. When things get far enough along, you can also look at the individual model component log files by typing:

```
cd run
tail -f ocn.bldlog.*
```
(The "-f" gives the latest end of the file, control C to exit).

Eventually the log files are zipped (type "gunzip *.gz" to unzip them). When the script is done, the following is printed to the screen: "CCSM BUILDEXE SCRIPT HAS FINISHED SUCCESSFULLY"

6) CHANGE THE RUN SETTINGS

Use a text editor or xmlchange to make modifications to env_run.xml. For example,

```
nedit env_run.xml  &
```

In env_run.xml, you can change the run length. STOP_N sets the run length. STOP_OPTION sets the units for STOP_N. For example, when Jack and I ran the ccsm_long case, we increased the run length to 14 months by changing STOP_OPTION to "nmonths" and STOP_N to "14". You can also set up the run so that it automatically resubmits. Note: The run length variables are not in the namelist in CCSM, which is different than the CAM stand-alone scripts that you used earlier in the week.

For CASE2 (ccsm_baseline) and CASE3 (ccsm_codemod), try changing the resubmit value from "0" to "1". This will cause the model to run for 5 days two times for a total of 10 days of model integration.

7) RUN CCSM

Before running CCSM, you should check and if necessary modify the project number (-P), queue (-q), and wall clock (-W, 6 hour limit in regular, make it as short as you can because this moves you through the queue faster). Don't change the MPI settings (e.g., -n, ptile, -a etc.). For example,

```
less ccsm_baseline.bluefire.run
nedit ccsm_baseline.bluefire.run &
```

When everything is ready, submit the CCSM run. For example, type:

```
bsub -U RESERVATIONID < ccsm_baseline.bluefire.run
```

Output files will appear in the run directory. For example, in /ptmp/$USER/ccsm_baseline/run.

8) LOOK AT AND POST-PROCESS THE OUTPUT

There are several quick ways to look at the model output. To look at the file header and overall contents on bluefire, try:

```
cd /ptmp/$USER/ccsm_baseline/run/
ncl_filedump ccsm_baseline.cam2.h1.0001-01-01-00000.nc | less
ncdump -h ccsm_baseline.cam2.h1.0001-01-01-00000.nc | less
```

You can also experiment with doing some simple post-processing. First, log into the post-processing machine (storm1 or storm4). Luckily, the bluefire /ptmp directories are mounted on the storm computers so no file transfer is required.

```
ssh -Y storm1.scd.ucar.edu
```

You can use ncview to look at the file contents. For example,

```
ncview /gpfs/ptmp/$USER/ccsm_baseline/run/ccsm_baseline.cam2.h1.0001-01-01-
00000.nc
```

A set of useful tools for simple post-processing netcdf files are the ncoperators. More information on ncoperators is available at http://nco.sourceforge.net/nco.html. For example, you can use ncdiff to difference two files. For this tutorial, it would be useful to ncdiff ccsm_baseline and ccsm_codemod to look at changes in CLDTOT, CLDLOW, etc. that result from modifying rhminl. Use ncview the look at diff.nc.

```
cd /gpfs/ptmp/$USER/ccsm_baseline/run
ncdiff ccsm_baseline.cam2.h1.0001-01-01-00000.nc
/gpfs/ptmp/$USER/ccsm_codemod/run/ccsm_codemod.cam2.h1.0001-01-01-00000.nc
diff.nc
```

III. RUNNING THE CAM DIAGNOSTICS PACKAGE ON CCSM OUTPUT

1) If you don't have it already, copy the CAM diagnostics script to your storm /ptmp/ directory from

```
cp /fs/home/andrew/diag/diag_tutorial.csh /ptmp/$USER/
```

2) If it doesn't already exist, make a directory for the diagnostics.

```
mkdir /ptmp/$USER/diag
```

3) Modify the diagnostics script using a text editor so that produces diagnostics that compare ccsm_long and ccsm_long_doubleC02. Note: nedit is not available on storm so you will have to use either emacs or vi.

```
emacs diag_tutorial.csh &
```

Modify the script to set the work directory to your own /ptmp directory on storm:

```
setenv WKDIR /ptmp/$USER/diag
```

Set the case names for comparison:

```
set test_prefix = ccsm_long
set cntl_prefix = ccsm_long_doublec02
set CNTL=USER
```

Point the script to the history files for ccsm_long and ccsm_long_doubleC02 from Jen's /ptmp directory on blueice. For example, the monthly h0 files for ccsm_long should be specified in test_path.

```
test_path = /gpfs/ptmp/jenkay/archive/ccsm_long/atm/hist/
```

The monthly h0 files for ccsm_long_doublec02 should be specified in cntl_path.

```
cntl_path = /gpfs/ptmp/jenkay/archive/ccsm_long_doublec02/atm/hist/
```

Also note that the model was run from December of year one to January of year 3. So the beginning year and number of years will also have to be changed:

```
set test_begin = 2
set test_nyrs = 1
set cntl_begin = 2
set cntl_nyrs = 1
```

Also make sure that both sets of climatologies will be calculated:

```
set test_calcc = 0
set cntl_calcc = 0
```

4) Run the diagnostics script on your storm /ptmp directory by typing:

```
./diag_tutorial.csh
```

4) Change to a unique name and transfer the diagnostics output .tar file to your own machine and look at the diagnostics using a web browser.

On storm, type

```
cd /ptmp/$USER/diag
mv ccsm_long-ccsm_long_doublec02.tar ccsm_long-ccsm_long_doublec02_$USER.tar
ftp ftp.cgd.ucar.edu (login: anonymous, password: your email address)
cd incoming
put ccsm_long-ccsm_long_doublec02_$USER.tar
quit
```

On your personal machine

```
ftp ftp.cgd.ucar.edu (login: anonymous, password: your email address)
cd incoming
get ccsm_long-ccsm_long_doublec02_$USER.tar
quit

tar -xvf ccsm_long-ccsm_long_doublec02_$USER.tar
```

JEN NOTES:

AT THE START OF A RUN: MAKE SURE THAT THIS IS CHANGED TO "TRUE" in env_run.xml
<entry id="CONTINUE_RUN"   value="TRUE" />