

Table of Contents

Setup a CESM 1.0.x benchmark run on a generic system.....	1
System requirements.....	1
Compile NETCDF (Requirement).....	1
Download CESM source code.....	1
Adapt configuration files.....	2
Create a new case.....	3
Change the layout.....	3
Set simulation length.....	4
Configure the case.....	5
Change history file output frequency of atmosphere model (cam).....	5
Build the case.....	5
Run the model.....	5
Change the resolution.....	6
Produce a summary.....	6

Setup a CESM 1.0.x benchmark run on a generic system

This is a **cookbook** to setup a CESM 1.0 benchmark run on a generic (Linux) system.

See also porting CESM in the CESM user's guide:

http://www.cesm.ucar.edu/models/cesm1.0/cesm/cesm_doc/c2161.html

System requirements

- **Compilers** known to work: intel 10.1, pgi 7.2, 8.0, 9.0, (pathscale 3.2)
- **MPI** implementations known to work: openmpi 1.4, 1.5, mvapich2 1.4, 1.5
- **tcs** CESM setup scripts are written in tcs

Compile NETCDF (Requirement)

- See also http://www.ncl.ucar.edu/Download/build_from_src.shtml#NetCDF

- Download NETCDF

```
wget http://www.unidata.ucar.edu/downloads/netcdf/ftp/netcdf-4.1.1.tar.gz
tar xfvz netcdf-4.1.1.tar.gz
cd netcdf-4.1.1
```

- Compile netcdf with the compiler you use later to compile the model. For example for intel compiler do:

```
export FC=ifort
export F77=ifort
export F90=ifort
export CPPFLAGS="-fPIC -DpgiFortran"
./configure --prefix=/usr/local/netcdf-4.1.1-intel --disable-netcdf-4 --disable-dap
make
make test
```

- Install NETCDF

```
make install
```

Download CESM source code

- Download from NCAR SVN. A password is needed to checkout. Please register at: http://www.cesm.ucar.edu/models/cesm1.0/register/register_cesm1.0.cgi In case you don't have time to register ask urs.beyerle@env.ethz.ch

```
svn export https://svn-ccsm-release.cgd.ucar.edu/model_versions/cesm1_0_2 cesm1_0_2
```

Adapt configuration files

- Change to **Machines** directory

```
cd cesm1_0_2
cd scripts/ccsm_utils/Machines/
```

- Meaning of filenames

Filename	Purpose
env_machopts.*	Set environment: Can be used to set paths to compiler, MPI library, NETCDF library
Macros.*	Set compiler name and paths to MPI library, NETCDF library. Set compiler options
mkbatch.*	Setting for queuing system

where * corresponds to a machine.

- As starting point take configuration files of a machine that is close to your environment. For example if you run on Linux, have a look at brutus_io, brutus_im, brutus_po or brutus_pm where i=intel, p=pgi, o=openmpi, m=mvapich2. You can get a list of pre-configured machines with

```
cd cesm1_0_2/scripts
./create_newcase -l
```

- Let's assume you use intel and openmpi, start with *.brutus_io files:

```
cp env_machopts.brutus_io env_machopts.your_machine
cp Macros.brutus_io      Macros.your_machine
cp mkbatch.brutus_io     mkbatch.your_machine
```

- Add to **config_machines.xml** a configuration tag for your machine (your_machine) - only the important lines are listed below

```
<machine MACH="your_machine"
DESC="Test System"
EXEROOT="/scratch/$CCSMUSER/$CASE"
OBJROOT="$EXEROOT"
INCROOT="$EXEROOT/lib/include"
DIN_LOC_ROOT_CSMDATA="/scratch/cesm1/inputdata"
DIN_LOC_ROOT_CLMQUAN="/scratch/cesm1/inputdata/atm/datm7/atm_forcing.datm7"
BATCHQUERY="qstat -f"
BATCHSUBMIT="qsub"
GMAKE_J="4"
MAX_TASKS_PER_NODE="4"
MPI SERIAL_SUPPORT="FALSE" />
```

please set the following variables:

```
EXEROOT=                                # working directory, final location of binary and output
DIN_LOC_ROOT_CSMDATA=                   # input data, data will be downloaded on the fly
DIN_LOC_ROOT_CLMQUAN=                   # input data, data will be downloaded on the fly
MAX_TASKS_PER_NODE=                     # define cores per node
```

- Configure **mpirun** execution: Search in **mkbatch.your_machine** for the line starting the executable **ccsm.exe** and replace it with the correct mpirun command for your system, for example something like

```
mpirun -np ${maxtasks} ./ccsm.exe >&! ccsml.log.\$LID
# or
mpirun -x LD_LIBRARY_PATH -np ${maxtasks} ./ccsm.exe >&! ccsml.log.\$LID
```

Create a new case

- Change to **scripts** directory

```
cd cesml_0_2
cd scripts
```

- Define the machine type (MACH), resolution (RES), compset (COMP=B: fully coupled model)

```
MACH=your_machine
COMP=B
RES=1.9x2.5_gx1v6
```

Possible values for RES are **T31_gx3v7** (~3°), **1.9x2.5_gx1v6** (2°) or **0.9x1.25_gx1v6** (1°).

- Define a case name (in principle, can be any name)

```
CASE=$RES-$COMP-benchmark
```

- Create case

```
./create_newcase -res $RES -compset $COMP -mach $MACH -case $CASE
```

Change the layout

- The layout will define which model component (ATM, LND, ICE, OCN, CPL) will run on how many cores.
- To **change the layout** you don't have to recreate the case (but you can if you like).
- Change into case directory

```
cd $CASE
```

- Clean the case

```
./configure -cleanmach
```

- Configure layout (all components will run on all cores)

```
NTASKS=16 #or# NTASKS=32 #or# NTASKS=64 #or# NTASKS=128 #or# NTASKS=256 #or# NTASKS=512

./xmlchange -file env_mach_pes.xml -id NTASKS_ATM -val $NTASKS
./xmlchange -file env_mach_pes.xml -id NTASKS_LND -val $NTASKS
./xmlchange -file env_mach_pes.xml -id NTASKS_ICE -val $NTASKS
./xmlchange -file env_mach_pes.xml -id NTASKS_OCN -val $NTASKS
./xmlchange -file env_mach_pes.xml -id NTASKS_CPL -val $NTASKS
./xmlchange -file env_mach_pes.xml -id NTASKS_GLC -val 1
./xmlchange -file env_mach_pes.xml -id TOTALPES -val $NTASKS
./xmlchange -file env_mach_pes.xml -id ROOTPE_OCN -val 0
```

- **Exception:** For resolution **T31_gx3v7** with **NTASKS=32**, **NTASKS=64**

```
NTASKS=32 #or# NTASKS=64

NTASKS_ATM=24
NTASKS_OCN=8
NTASKS_LND=24
NTASKS_ICE=20
NTASKS_CPL=24
NTASKS_TOTAL=32
F=$(( $NTASKS / $NTASKS_TOTAL ))
./xmlchange -file env_mach_pes.xml -id ROOTPE_GLC -val 0
./xmlchange -file env_mach_pes.xml -id NTASKS_GLC -val 1
./xmlchange -file env_mach_pes.xml -id ROOTPE_ATM -val 0
./xmlchange -file env_mach_pes.xml -id NTASKS_ATM -val $(( $NTASKS_ATM * $F ))
./xmlchange -file env_mach_pes.xml -id ROOTPE_LND -val 0
./xmlchange -file env_mach_pes.xml -id NTASKS_LND -val $(( $NTASKS_LND * $F ))
./xmlchange -file env_mach_pes.xml -id ROOTPE_ICE -val 0
./xmlchange -file env_mach_pes.xml -id NTASKS_ICE -val $(( $NTASKS_ICE * $F ))
./xmlchange -file env_mach_pes.xml -id ROOTPE_OCN -val $(( $NTASKS_ATM * $F ))
./xmlchange -file env_mach_pes.xml -id NTASKS_OCN -val $(( $NTASKS_OCN * $F ))
./xmlchange -file env_mach_pes.xml -id ROOTPE_CPL -val 0
./xmlchange -file env_mach_pes.xml -id NTASKS_CPL -val $(( $NTASKS_CPL * $F ))
./xmlchange -file env_mach_pes.xml -id TOTALPES -val $NTASKS
```

Set simulation length

- In general, CESM is hardwired to generate monthly average data. In principle this can be turned off but needs a lot of code changes. Therefore it's not considered here. The following two cases are suggested instead:

- **CASE 1:** Run a short simulation (10 days) with producing no restart files (REST_OPTION=never) and no archiving (DOUT_S=FALSE)

```
./xmlchange -file env_run.xml -id STOP_OPTION -val 'ndays'
./xmlchange -file env_run.xml -id STOP_N -val '10'
./xmlchange -file env_run.xml -id REST_OPTION -val 'never'
./xmlchange -file env_run.xml -id DOUT_S -val 'FALSE'
```

- **CASE 2:** Run a longer simulation (1 months) with producing restart files at the end

```
./xmlchange -file env_run.xml -id STOP_OPTION -val 'nmonths'
./xmlchange -file env_run.xml -id STOP_N      -val '1'
./xmlchange -file env_run.xml -id REST_OPTION -val '$STOP_OPTION'
./xmlchange -file env_run.xml -id DOUT_S      -val 'FALSE'
```

Configure the case

- Configure case

```
./configure -case
```

Change history file output frequency of atmosphere model (cam)

- **CASE 1:** leave the default which is monthly output data
- **CASE 2:** change history file output frequency to **one hour**. Edit **Buildconf/cam.buildnml.csh** and set just after **&cam_inparm** the variable **nhtfrq = -1**

```
cat Buildconf/cam.buildnml.csh
...
&cam_inparm
  nhtfrq  = -1
  absems_data          = '$DIN_LOC_ROOT/atm/cam/rad/abs_ems_factors_fastvx.c030508'
...
```

In case you change **Buildconf/cam.buildnml.csh**, you can not go back by just deleting the lines. **You have configure an new case with with `./create_newcase` !**

Build the case

- Build and compile the model

```
./$CASE.$MACH.build
```

Run the model

- Run the model, for example with LSF **queuing system**

```
bsub < $CASE.$MACH.run
```

- To start without a queuing system just execute:

```
./$CASE.$SMACH.run
```

- Timing results can be found after the run has been successfully completed in folder **timing**

```
cat timing/ccsm_timing.$CASE.*
...
Model Throughput:          6.39    simulated_years/day
...
```

Change the resolution

- Recommended resolutions are T31_gx3v7 (~3°), 1.9x2.5_gx1v6 (2°), 0.9x1.25_gx1v6 (1°)
- Create for each resolutions a separate cases !

Produce a summary

- Create performance matrix for **CASE 1** and **CASE 2**. Fill in Model Throughput in **simulated_years/day**
- **CASE 1:** Run a short simulation with producing almost no output (I/O) - Values are from **brutus.ethz.ch**

resolution / layout (NTASKS)	16	32	64	128	192	256	512
T31_gx3v7	12.0	18.3	26.9	--	--	--	--
1.9x2.5_gx1v6	--	2.27/2.60					--
0.9x1.25_gx1v6	--	--	1.88				

- **CASE 2:** Run a longer simulation with producing hourly output data - Values are from **brutus.ethz.ch**

resolution / layout (NTASKS)	16	32	64	128	192	256	512
T31_gx3v7	7.9	10.8	13.6	--	--	--	--
1.9x2.5_gx1v6	--						--
0.9x1.25_gx1v6	--	--					



Copyright © by the contributing authors. All material on this collaboration platform is the property of the contributing authors.

Ideas, requests, problems regarding Wiki? Send feedback