2024 CESM Tutorial

Porting and Validating CESM 2.1.x

Jim Edwards,
NCAR CGD Software Engineer

Aug 5-9, 2024

This material is based upon work supported by the NSF National Center for Atmospheric Research, which is a major facility sponsored by the U.S. National Science Foundation under Cooperative Agreement No. 1852977.

Porting to a new system

Define the new machine locally in your $HOME/.cime directory

or

Define the machine in the your code sandbox to share with colleagues and potentially the CESM repositories.

Suggestion: Find a similar machine in the cesm/cime/config/cesm/machines/config_machines.xml file and copy it to create your new machine definition then edit to adapt to your system.

Defining a new machine in $HOME/.cime

- Create a directory $HOME/.cime.
- Add files config_machines.xml, config_batch.xml, config_compilers.xml
- The file config_machines.xml should contain an entry similar to those found in cime/config/cesm/machines/config_machines.xml
  - a minimal example is located in cime/config/cesm/machines/userdefined_laptop_template/config_machines.xml
- Repeat this process with config_compilers.xml and config_batch.xml

Suggestion: The contents of config_compilers.xml and config_batch.xml are cumulative - most of what you need to define for a particular compiler (such as intel) or a particular batch system (pbs or slurm) are defined near the top of the file in cime/config/cesm/machines use these settings and only overwrite or add settings particular to your system.

Defining a new machine in your sandbox

Same as above but add your settings directly to cime/config/cesm/machines in your sandbox. When your port is complete and you are ready to share:

1. Create a git branch in your sandbox:
   a. git checkout -b maint_5.6_port/*my_machine_name*
2. Create a fork of cime in github. (google it)
3. Open a pull request to the maint-5.6 branch of cime in the repository at https://github.com/ESMCI/CIME

The machine definition files

The 3 files of interest are config_machines.xml, config_batch.xml and config_compilers.xml

- config_machines.xml contains:
  - The path to cesm inputdata (this should be shared with other users if possible)
  - The location of case directories as well as run and build directories.
  - The names of compilers, batch systems and mpi libraries to be used
  - Module load and environment definitions
- config_batch.xml contains:
  - definitions of queues used on your system
  - submit arguments and batch submit flags
- config_compilers.xml contains:
  - Link commands for support libraries
  - compiler flags that may be unique to your system

```xml
<machine MACH="archer2">
            <DESC>two CrayAMD EPYC Zen2, 128 pes/node, batch system is SLURM</DESC>
            <NODENAME_REGEX>(ln\d{2}$|nid\d{6}$)</NODENAME_REGEX>
            <OS>CNL</OS>
            <COMPILERS>gnu,cray</COMPILERS>
            <MPILIBS>mpich,mpi-serial</MPILIBS>
            <CIME_OUTPUT_ROOT>$ENV{CESM_ROOT}/runs</CIME_OUTPUT_ROOT>
            <DIN_LOC_ROOT>$ENV{CESM_ROOT}/cesm_inputdata</DIN_LOC_ROOT>
            <DIN_LOC_ROOT_CLMFORC>${DIN_LOC_ROOT}/atm/datm7</DIN_LOC_ROOT_CLMFORC>
            <DOUT_S_ROOT>$ENV{CESM_ROOT}/archive/$CASE</DOUT_S_ROOT>
            <BASELINE_ROOT>$ENV{CESM_ROOT}/ccsm_baselines</BASELINE_ROOT>
            <CCSM_CPRNC>$ENV{CIMEROOT}/tools/cprnc/cprnc</CCSM_CPRNC>
            <GMAKE_J>8</GMAKE_J>
            <BATCH_SYSTEM>slurm</BATCH_SYSTEM>
            <SUPPORTED_BY>leeds.ac.uk</SUPPORTED_BY>
            <MAX_TASKS_PER_NODE>128</MAX_TASKS_PER_NODE>
            <MAX_MPITASKS_PER_NODE>128</MAX_MPITASKS_PER_NODE>
            <PROJECT_REQUIRED>TRUE</PROJECT_REQUIRED>
            <mpirun mpilib="default">
            <executable>srun</executable>
            <arguments>
            <arg name="cpubind"> --distribution=block:block --hint=nomultithread</arg>
            <!--<arg name="cpubind"> -ZZ-cpu-bind=cores</arg> -->
            </arguments>
            </mpirun>
            <module_system type="module"  allow_error="true">
            <init_path lang="perl">/usr/share/lmod/lmod/init/perl</init_path>
            <init_path lang="python">/usr/share/lmod/lmod/init/env_modules_python.py</init_path>
            <init_path lang="csh">/usr/share/lmod/lmod/init/csh</init_path>
            <init_path lang="sh">/usr/share/lmod/lmod/init/sh</init_path>
            <cmd_path lang="perl">/usr/share/lmod/lmod/libexec/lmod perl</cmd_path>
            <cmd_path lang="python">/usr/share/lmod/lmod/libexec/lmod python</cmd_path>

            <cmd_path lang="sh">module</cmd_path>
            <cmd_path lang="csh">module</cmd_path>

            <modules compiler="gnu">
            <command name="load"> PrgEnv-gnu</command>
            <command name="load"> cray-hdf5-parallel</command>
            <command name="load"> cray-netcdf-hdf5parallel</command>
            <command name="load"> cray-parallel-netcdf</command>
            <command name="load"> cray-libsci</command>
            </modules>
            <modules mpilib="mpi-serial">
            <command name="rm"> cray-netcdf-hdf5parallel</command>
            <command name="rm"> cray-hdf5-parallel</command>
            <command name="rm"> cray-parallel-netcdf</command>
            <command name="load"> cray-hdf5</command>
            <command name="load">cray-netcdf</command>
            </modules>
            </module_system>

            <environment_variables>
            <env name="PERL5LIB">/work/n02/shared/perl/5.26.2</env>
            <env name="OMP_NUM_THREADS">{{ thread_count }} </env>
            <env name="OMP_PLACES">cores </env>
            <env name="OMP_STACKSIZE">2G</env>
            <!--<env name="PATH">/work/n02/n02/csymonds/sw/conda/cesmenv2/bin:$ENV{PATH}</env>-->
            </environment_variables>

            <resource_limits>
            <resource name="RLIMIT_STACK">-1</resource>
            </resource_limits>
 </machine>
```

```xml
 <batch_system MACH="bluewaters" type="pbs" >
        <jobid_pattern>(\d+.bw)$</jobid_pattern>
        <directives>
        <directive>-l nodes={{ num_nodes }}:ppn={{ tasks_per_node }}:xe</directive>
        <directive default="/bin/bash" > -S {{ shell }} </directive>
        </directives>
        <queues>
        <queue walltimemax="24:00:00">normal</queue>
        <queue walltimemax="00:30:00" nodemin="1" nodemax="16" default="true">debug</queue>
        </queues>
</batch_system>
```

```xml
<compiler MACH="athena" COMPILER="intel">
  <CFLAGS>
          <append>  -xHost </append>
  </CFLAGS>
  <CPPDEFS>
          <append> -DINTEL_MKL -DHAVE_SSE2 </append>
  </CPPDEFS>
  <FFLAGS>
          <append>  -xHost </append>
  </FFLAGS>
  <FFLAGS>
          <append MODEL="nemo"> $(FC_AUTO_R8) -O3 -assume norealloc_lhs </append>
  </FFLAGS>
  <SLIBS>
          <append> $SHELL{${NETCDF_PATH}/bin/nc-config --flibs}</append>
  </SLIBS>
  <MPICXX MPILIB="mpich2">mpiicpc</MPICXX>
  <MPICC MPILIB="mpich2">mpiicc</MPICC>
  <MPIFC MPILIB="mpich2">mpiifort</MPIFC>
  <SCC>icc</SCC>
  <SFC>ifort</SFC>
  <TRILINOS_PATH MPILIB="mpich2">$ENV{TRILINOS_PATH}</TRILINOS_PATH>
</compiler>
```

Testing your port

You’ve made an initial attempt at a system port - how do you test it?

1. Start with a simple case:
    a. cd cime/scripts
    b. ./create_newcase –case foo –compset X –res f19_g17 –machine *my_machine_name*
    c. cd foo
    d. ./case.setup
    e. ./case.build
    f. ./case.submit

Almost certainly these commands won’t all work the first time. Refine your settings in the xml files and iterate on the above steps until all of them work without error. If you get stuck try the cesm forum: https://bb.cgd.ucar.edu/cesm/

Suggestion: CESM2.1 uses python versions between 3.7 and 3.9 - you may need to create a virtual environment using pip or conda if your python version is newer.

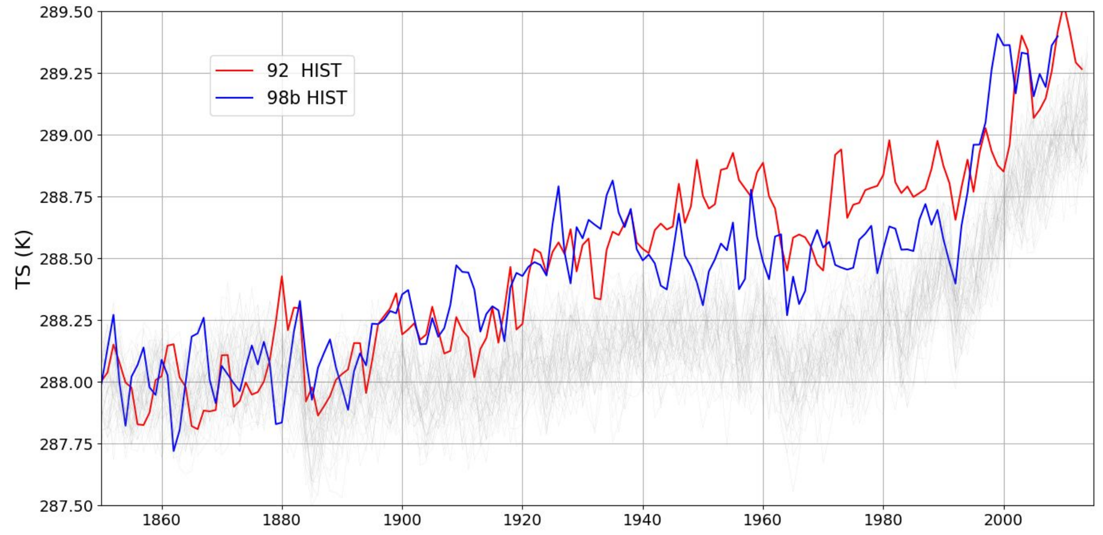
Testing your port: scripts_regression_tests

Once you have successfully built and submitted a cesm case:

- Build the cprnc tool used for comparing netcdf files.
  - git clone https://github.com/ESMCI/cprnc
  - cd cprnc
  - Follow instructions in README file.
  - Put the resulting cprnc executable in the directory specified by CCSM_CPRNC in config_machines.xml
- You are now ready to run the scripts_regression_tests
  - cd cime/scripts/tests
  - ./scripts_regression_tests.py

Testing your port: Running the ensemble consistency test

- See the README in cesm/cime/tools/statistical_ensemble_test
- The statistical ensemble test compares results of 3 runs done on your machine to an ensemble of runs computed on NCAR's cheyenne (retired).
- If your runs fit within the ensemble the test passes



- Upload completed runs to https://docs.cesm.ucar.edu/models/cesm2/verification/

How to Get Help

Post to the CESM forum: https://bb.cgd.ucar.edu/cesm/

Ask a question in slack: cesm2.slack.com

Open an issue in github:

- https://github.com/ESMCI/cime/issues
- https://github.com/ESCOMP/CESM/issues/
- … (for each component model)

Please only use github issues with authorization by a CESM developer.

Questions ?

Image courtesy Kolya Dols