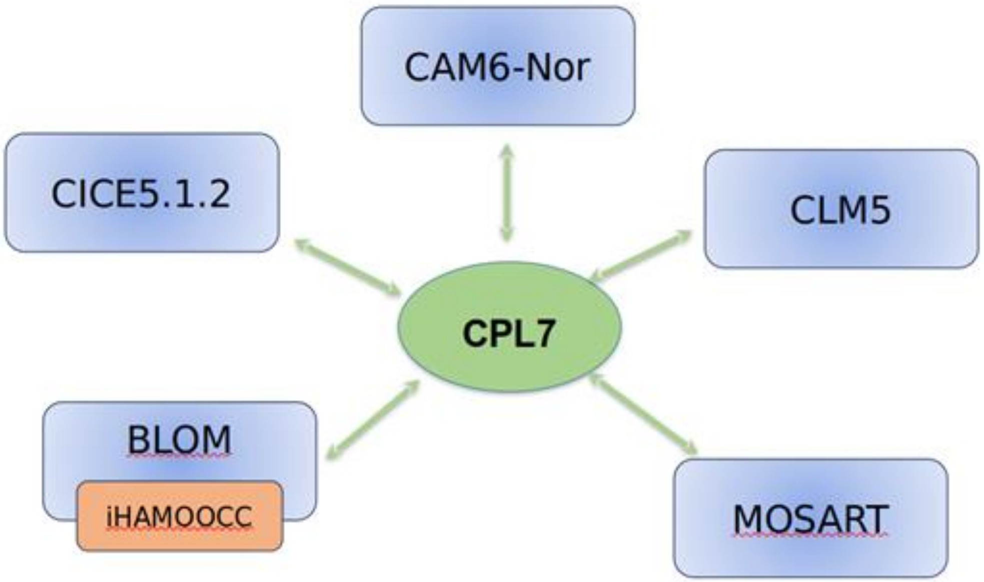
NorESM2.3 Architecture and Use

Mariana Vertenstein

January 16, 2026

What is the NorESM2.3 architecture

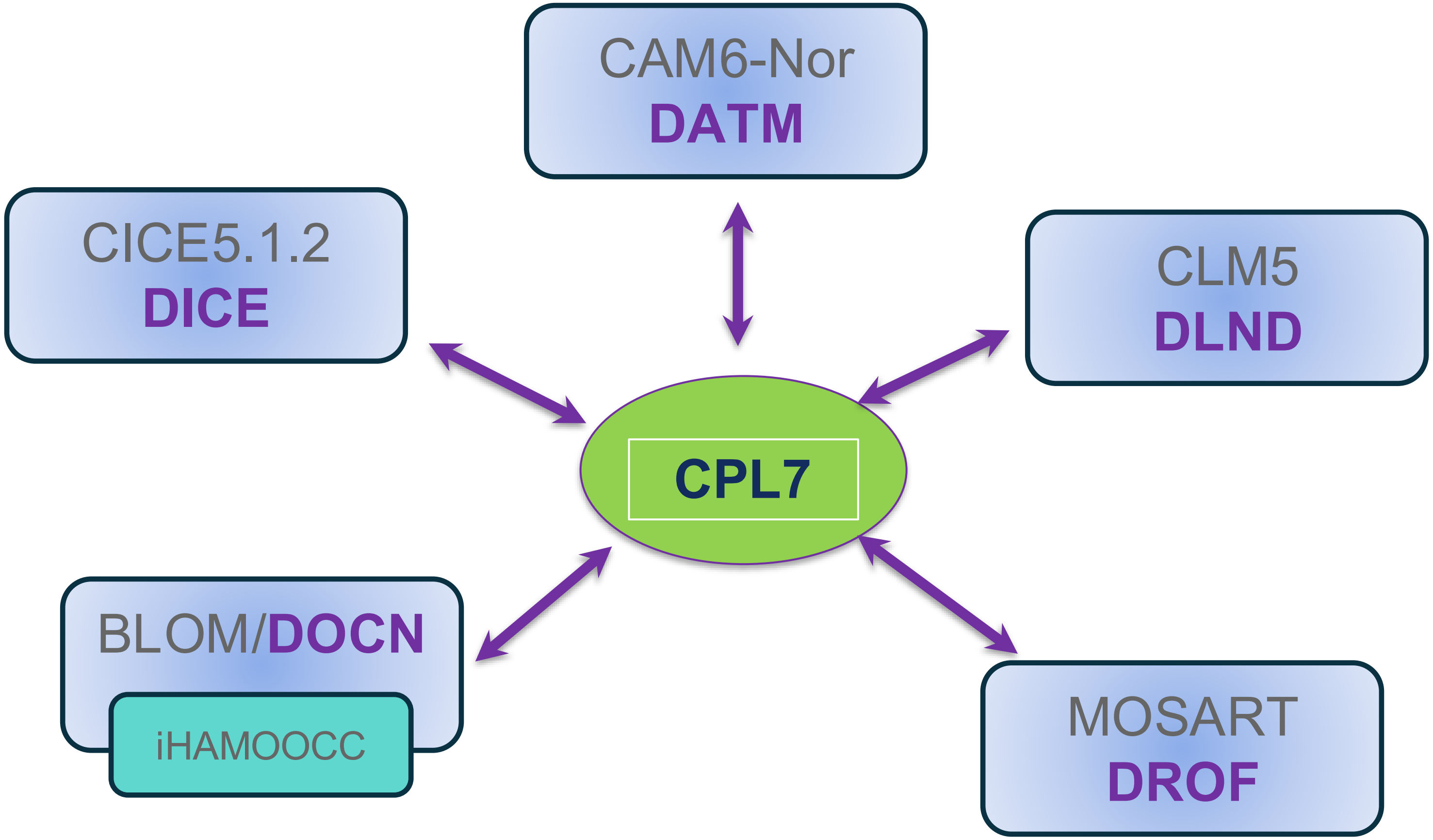
NorESM2 coupling and prognostic components



NorESM2.3 is based on the second version of the Community Earth System Model, CESM2, and share most of the CESM2 structure, but modifies model component.

Atmosphere model :
  CAM6-Nor replaces standard CAM
  OsloAero6 atmospheric chemistry
Ocean model :
  BLOM Isopycnic coordinate model
  iHAMOCC ocean biogeochemical model
Sea-ice model:
  Wind drift of snow

NorESM2 data components



Specifying the compset –
first independent part of experiment definition

Compset specifies:

- Which components should be used
- What physics each component should have

Creating a new case

A new case is created by running the script

$SRCROOT/cime/scripts/create_newcase

Where $SRCROOT is your root directory for a NorESM checkout

Red is required
Grey is optional

./create_newcase
-- **case** <full pathname of case>
-- **compset** *<long compset name or alias>*
-- **res** *<long resolution name or alias>*
-- **machine** <machine name>
-- **project** <project name>
-- pecount <label, e.g. M>
--user_mod_dirs <path to user mods>
--run-unsupported

First part

Second part

What is the compset longname syntax?

TIME_ATM[%phys]_**LND**[%phys]_**ICE**[%phys]_**OCN**[%phys]_**ROF**[%phys]_**GLC**[%phys]_ **WAV**[%phys]*][_BGC%phys]*

TIME = Time period (e.g. 1850, 2000, HIST, SSP126, SSP245, SSP370. SSP585)

ATM = [**CAM60**, DATM] ;

LND = [**CLM50**, DLND, SLND] ;

CICE = [**CICE**, DICE, SICE]

OCN = [**BLOM**, DOCN, SOCN];

ROF = [**MOSART**, RTM, DROF, SROF] ;

GLC = [**CISM2**, SGLC]

WAV = [**WW3**, DWAV, SWAV];

BGC = optional BGC scenario

Stub components denote a component that does nothing

OPTIONAL %phys attributes specify submodes of the given system

For example DOCN%DOM is the mode of DOCN where specified SST files are read in
ALL data models must have a %phys option that corresponds to the data model mode

More on compset specification

Compset specifies which components will be used *along with* possible component physics options and associated forcing files (time period)

- Fully coupled: (compset aliases beginning with N or B)
  - **1850_CAM60%NORESM_CLM50%BGC_CICE_BLOM%ECO_MOSART**_SGLC_SWAV
  - Run ./query_config --compsets allactive
- CAM (atm) with prescribed docn/cice forcing (alias NF, F or Q)
  - **1850_CAM60%NORESM_CLM50%BGC_CICE%PRES_DOCN%DOM_MOSART**_SGLC_SWAV
  - Run ./query_config --compsets cam
- CLM (lnd) with prescribed datm forcing (alias I)
  - **1850_DATM%GSWP3v1_CLM50%BGC_SICE_SOCN_MOSART**_SGLC_SWAV
  - Run ./query_config --compsets clm
- BLOM with prescribed datm and rof forcing (alias G and C)
  - **1850_DATM%NYF_SLND_CICE_BLOM%ECO_DROF%NYF**_SGLC_SWAV
  - Run ./query_config --compsets blom

Some NorESM2.3 fully coupled compsets

N1850frc2: (piControl)

1850_CAM60%NORESM%FRC2_CLM50%BGC-CROP_CICE%NORESM-CMIP6_BLOM%ECO_MOSART_SGLC_SWAV_BGC%BDRDDMS

NHISTfrc2: (historical)

HIST_CAM60%NORESM%FRC2_CLM50%BGC-CROP_CICE%NORESM-CMIP6_BLOM%ECO_MOSART_SGLC_SWAV_BGC%BDRDDMS

NSSP126frc2: (CMIP6 scenario)

SSP126_CAM60%NORESM%FRC2_CLM50%BGC-CROP_CICE%NORESMCMIP6_BLOM%ECO_MOSART_SGLC_SWAV_BGC%BDRDDMS

Specifying the model resolution –

Second independent part of experiment definition

Choosing the Model Grid

If atm/lnd/cice/docn are on the same grid need to specify an ocean/land mask in alias (mtn14)

- Grid specification for each component
  - **a%name_l%name_oi%name_r%name_m%mask_g%name_w%name**
- Fully coupled:
  - a%1.9x2.5_l%1.9x2.5_oi%tnx1v4_r%r05_g%null_w%null_m%tnx1v4
  - alias is f19_tn14
- CAM with prescribed docn/cice forcing
  - a%1.9x2.5_l%1.9x2.5_oi%1.9x2.5_r%r05_g%null_w%null_m%tnx1v4
  - alias is f19_f19_mtn14
- CLM with prescribed datm forcing
  - a%1.9x2.5_l%1.9x2.5_oi%null_r%r05_g%null_w%null_m%tnx1v4
  - alias is f19_f19_mtn14
- BLOM/CICE with prescribed datm and rof forcing
  - a%T62_l%null_oi%tnx1v4_r%rx1_g%null_w%null_m%tnx1v4
  - alias is T62_tn14

Standard model grids for NorESM2.3

In general:

- Ocean and Sea-ice MUST always have the same grid
- Atmosphere and land generally have the same grid - but usually different from active ocean/ice
- River, Glacier and Wave are on their own grids - and different from the above
- Coupler is used to interpolate/transfer fields from one grid to another grid and does not have its own grid

Grids for scientifically supported NorESM2.3 experiments include:

Atmosphere & land:
f19 : 1.9x2.5 = 1.9 degree latitude and 2.5 degree longitude resolution = 144x96
f09 : 0.9x1.25 = 0.9 degree latitude and 1.25 degree longitude resolution = 288x192

Ocean & sea ice:
tnx1v4 : tripolar 1 degree grid = 360x384

Data atmosphere: (OMIP experiments)
T62, **TL319** : approx. 2 degree Gaussian and 0.5 degree spectral grid

For experts:
All available grid definitions are in $SRCROOT/cime/config/cesm/config_grids.xml)

Understanding your case

- README.case
- xmlquery
- preview_namelist
- pelayout
- preview_run

Understanding your case –
README.case and xmlquery

```
> ./create_newcase  –case foo –compset N1850 –res f19_tn14
```

Compset longname is:

```
1850_CAM60%NORESM_CLM50%BGC-
CROP_CICE%NORESMCMIP6_BLOM%ECO_MOSART_SGLC_SWAV_BGC%BDR
DDMS
```

Model grid longmae is:

```
a%1.9x2.5_l%1.9x2.5_oi%tnx1v4_r%r05_g%null_w%null_m%tnx1v4
```

- **What to do first – look README.case**
- **What are are the available xml variables, their definition and their values?**

  Most information you need to know about the case is in **xmlquery!!!**

  How does xmlquery work?

  ```
  > ./xmlquery –help
  ```

  What are all the variables in the xml files (don’t need to look inside)

  ```
  > ./xmlquery –- listall
  ```

  What if I want to look at particular xml variables but don’t know the full name

  ```
  > ./xmlquery –-partial (or –p)
  > ./xmlquery –-partial –-full (or –p –-full)
  ```

Example: query the BLOM configuration

```
> ./xmlquery –p  BLOM
Results in group build_component_blom
            BLOM_TRACER_MODULES: iage ecosys
            BLOM_TURBULENT_CLOSURE: oneeq advection
            BLOM_UNIT: cgs
Results in group run_component_blom
            BLOM_COUPLING: full
            BLOM_NDEP_SCENARIO: 1850
            BLOM_N_DEPOSITION: TRUE
            BLOM_RIVER_NUTRIENTS: TRUE
            BLOM_VCOORD: isopyc_bulkml
```

```
> ./xmlquery --full BLOM_TRACER_MODULES
Results in group build_component_blom
            BLOM_TRACER_MODULES: value=iage ecosys
                        type: c
                        valid_values: ['iage', 'iage ecosys']
                        description: Optional ocean tracers.
                        Valid values are acombination of: iage ecosys
                        file: $CASEROOT/foo/env_build.xml
```

Understanding your case – preview_namelists

What are my resolved namelists?

The namelists that actually get read by the components.

```
> cd $CASEROOT
> ./preview_namelists
```

- Must be run after ./case.setup
- See the namelists:
  - Output namelists are in *$CASEROOT/CaseDocs* and *in $RUNDIR/run*
  - **CaseDocs/ is where you can quickly see component namelists – this is ONLY used for documentation**
- Customize the namelists
  - Edit the appropriate user_nl_XXX file
  - then run ./preview_namelists again
  - (DO NOT edit files in either CaseDocs or run – they will be overwritten)

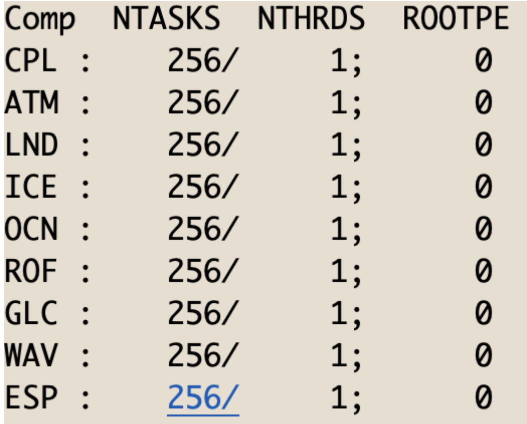
Understanding your case – pelayout

What is my processor layout?

NOTE: this might not be optimal for your experiment out of the box!

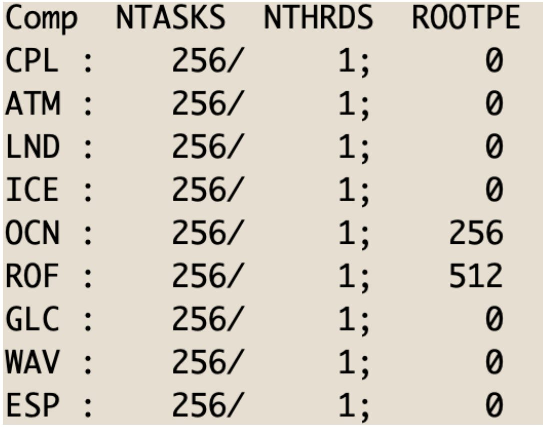
>./pelayout

NTASKS: number of MPI tasks for each component
NTHRDS: number of OpenMP threads for each component
ROOTPE: first MPI task for each component



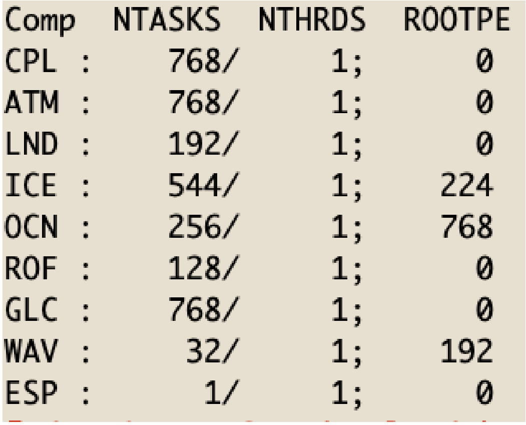
```
Comp   NTASKS   NTHRDS   ROOTPE
CPL :     256/      1;       0
ATM :     256/      1;       0
LND :     256/      1;       0
ICE :     256/      1;       0
OCN :     256/      1;       0
ROF :     256/      1;       0
GLC :     256/      1;       0
WAV :     256/      1;       0
ESP :     256/      1;       0
```

All components sharing the same MPI tasks – total tasks is 256



```
Comp   NTASKS   NTHRDS   ROOTPE
CPL :     256/      1;       0
ATM :     256/      1;       0
LND :     256/      1;       0
ICE :     256/      1;       0
OCN :     256/      1;     256
ROF :     256/      1;     512
GLC :     256/      1;       0
WAV :     256/      1;       0
ESP :     256/      1;       0
```

OCN and ROF on different tasks – total tasks is 768



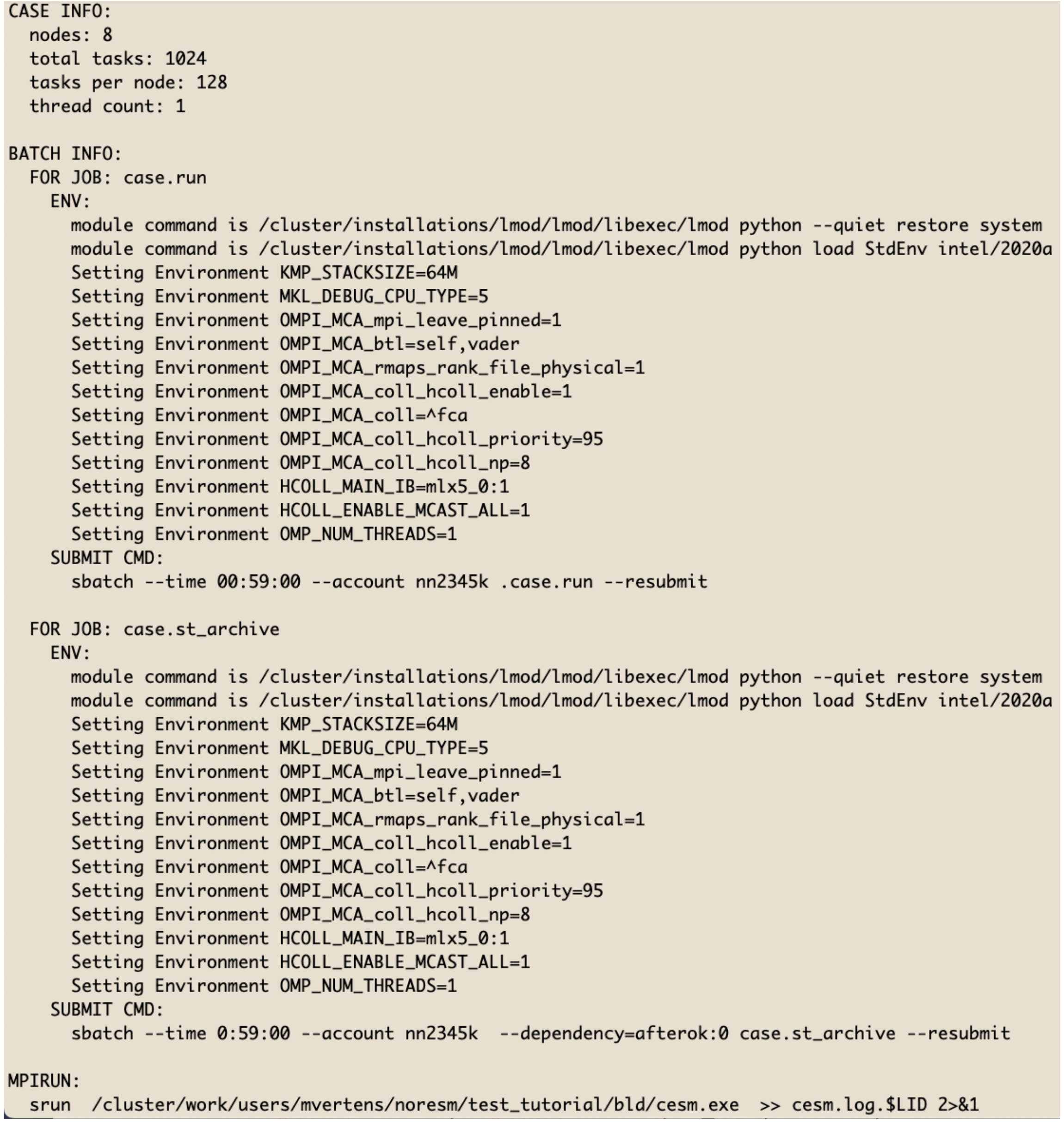
```
Comp   NTASKS   NTHRDS   ROOTPE
CPL :     768/      1;       0
ATM :     768/      1;       0
LND :     192/      1;       0
ICE :     544/      1;     224
OCN :     256/      1;     768
ROF :     128/      1;       0
GLC :     768/      1;       0
WAV :      32/      1;     192
ESP :       1/      1;       0
```

lnd and ice running on different tasks but sharing those with atm

Understanding your case
preview_run

- What batch information is being submitted?
- How is the short-term archiving used?
- What are the environment variables that are set?
- What are the batch queue settings?

➢ **./preview_run**



```
CASE INFO:
  nodes: 8
  total tasks: 1024
  tasks per node: 128
  thread count: 1

BATCH INFO:
  FOR JOB: case.run
    ENV:
      module command is /cluster/installations/lmod/lmod/libexec/lmod python --quiet restore system
      module command is /cluster/installations/lmod/lmod/libexec/lmod python load StdEnv intel/2020a
      Setting Environment KMP_STACKSIZE=64M
      Setting Environment MKL_DEBUG_CPU_TYPE=5
      Setting Environment OMPI_MCA_mpi_leave_pinned=1
      Setting Environment OMPI_MCA_btl=self,vader
      Setting Environment OMPI_MCA_rmaps_rank_file_physical=1
      Setting Environment OMPI_MCA_coll_hcoll_enable=1
      Setting Environment OMPI_MCA_coll=^fca
      Setting Environment OMPI_MCA_coll_hcoll_priority=95
      Setting Environment OMPI_MCA_coll_hcoll_np=8
      Setting Environment HCOLL_MAIN_IB=mlx5_0:1
      Setting Environment HCOLL_ENABLE_MCAST_ALL=1
      Setting Environment OMP_NUM_THREADS=1
    SUBMIT CMD:
      sbatch --time 00:59:00 --account nn2345k .case.run --resubmit

  FOR JOB: case.st_archive
    ENV:
      module command is /cluster/installations/lmod/lmod/libexec/lmod python --quiet restore system
      module command is /cluster/installations/lmod/lmod/libexec/lmod python load StdEnv intel/2020a
      Setting Environment KMP_STACKSIZE=64M
      Setting Environment MKL_DEBUG_CPU_TYPE=5
      Setting Environment OMPI_MCA_mpi_leave_pinned=1
      Setting Environment OMPI_MCA_btl=self,vader
      Setting Environment OMPI_MCA_rmaps_rank_file_physical=1
      Setting Environment OMPI_MCA_coll_hcoll_enable=1
      Setting Environment OMPI_MCA_coll=^fca
      Setting Environment OMPI_MCA_coll_hcoll_priority=95
      Setting Environment OMPI_MCA_coll_hcoll_np=8
      Setting Environment HCOLL_MAIN_IB=mlx5_0:1
      Setting Environment HCOLL_ENABLE_MCAST_ALL=1
      Setting Environment OMP_NUM_THREADS=1
    SUBMIT CMD:
      sbatch --time 0:59:00 --account nn2345k  --dependency=afterok:0 case.st_archive --resubmit

MPIRUN:
  srun  /cluster/work/users/mvertens/noresm/test_tutorial/bld/cesm.exe  >> cesm.log.$LID 2>&1
```

Before you run

– checking your case

xmlchange and check_input_data

xmlchange

use to change variables in xml files

```
> cd $CASEROOT
> ./xmlchange STOP_OPTION=nmonths
> ./xmlchange STOP_N=6
```

which will change the variables STOP_OPTION and STOP_N in the file env_run.xml

check_input_data

- Checks if the required data files for the case exist on local disk in the appropriate subdirectory of $DIN_LOC_ROOT.
- If any of the required datasets do not exist locally, then **check_input_data** will download them to the $DIN_LOC_ROOT directory hierarchy via interaction with the input data server.

Some steps to take BEFORE starting a long run

1. first build the case (case.build)
2. verify that input data is present – and download if not (check_inputdata)
3. perform a debug test (xmlchange DEBUG=TRUE) if you have changed to code or namelist
4. run a performance test and check timing file

Determining best processor layout can be a huge benefit for performance!

Why a debug test?

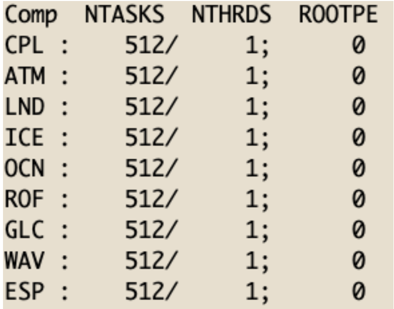
- Normally if the xml variable DEBUG is not enabled the compiler flags to check for floating point exceptions or array bound problems are not turned on. Its important to do this before you run a long run. You can do a 1 day test with debug on out of your $CASEROOT.

- Simply do the following:
  - ./case.build – clean-all
  - ./xmlchange DEBUG=TRUE
  - ./xmlchange STOP_OPTION=ndays
  - ./xmlchange STOP_N=1
  - ./case.build
  - ./case.submit
- If this works – the simply reset the default values and build again
  - ./case.build –clean-all
  - ./xmlchange DEBUG=FALSE
  - ./case.build
- If it crashes – need to resolve the problem!

How to validate performance

- Start with a simple case - clm, datm, rof, cpl

- ./create_newcase \
  --case test_perf \
  --compset 2000_**DATM%GSWP3v1_CLM50%BGC-CROP**_SICE_SOCN_**MOSART**_SGLC_SWAV \
  --res f19_f19_mtn14
- cd test_perf
- ./xmlchange NTASKS=512
- ./xmlchange DIN_LOC_ROOT_CLMFORC=/cluster/shared/noresm/inputdata/atm/datm7
- ./pelayout
- ./case.setup
- ./case.build
- ./case.submit



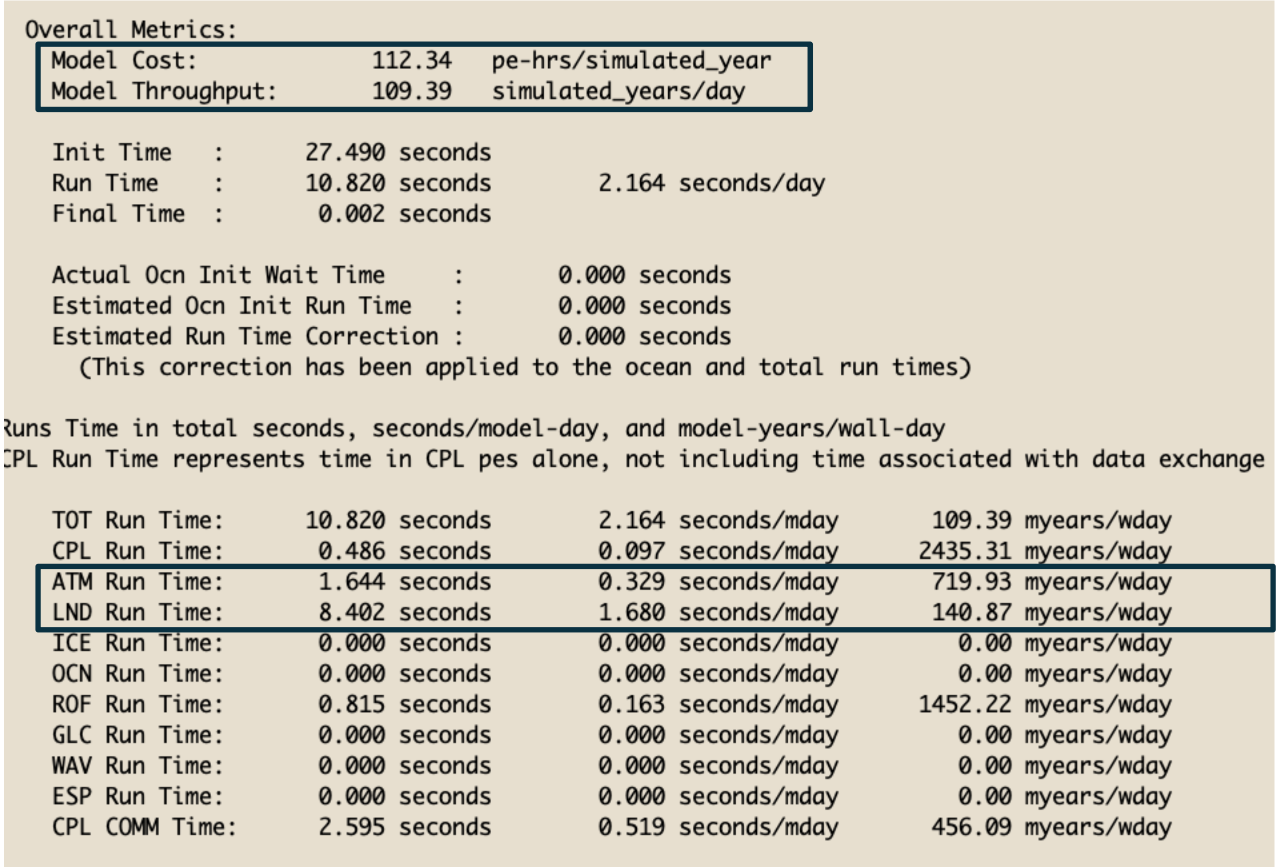
```
Comp  NTASKS  NTHRDS  ROOTPE
CPL :    512/     1;      0
ATM :    512/     1;      0
LND :    512/     1;      0
ICE :    512/     1;      0
OCN :    512/     1;      0
ROF :    512/     1;      0
GLC :    512/     1;      0
WAV :    512/     1;      0
ESP :    512/     1;      0
```

However, we know that DATM and CLM run concurrently in time – so can place DATM on a separate processor set
Let's start with a 5 day test and see the timing file

How can we get better performance?

> cd timing/test_perf/

> look at cesm_timing.test_perf.$id

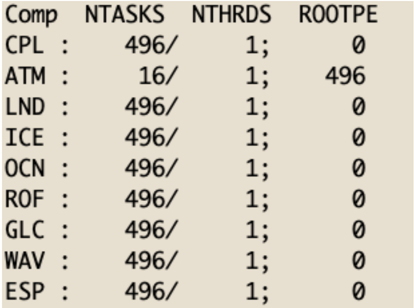


```
 Overall Metrics:
   Model Cost:             112.34   pe-hrs/simulated_year
   Model Throughput:       109.39   simulated_years/day

   Init Time   :      27.490 seconds
   Run Time    :      10.820 seconds        2.164 seconds/day
   Final Time  :       0.002 seconds

   Actual Ocn Init Wait Time     :        0.000 seconds
   Estimated Ocn Init Run Time   :        0.000 seconds
   Estimated Run Time Correction :        0.000 seconds
     (This correction has been applied to the ocean and total run times)

Runs Time in total seconds, seconds/model-day, and model-years/wall-day
CPL Run Time represents time in CPL pes alone, not including time associated with data exchange

   TOT Run Time:      10.820 seconds        2.164 seconds/mday       109.39 myears/wday
   CPL Run Time:       0.486 seconds        0.097 seconds/mday      2435.31 myears/wday
   ATM Run Time:       1.644 seconds        0.329 seconds/mday       719.93 myears/wday
   LND Run Time:       8.402 seconds        1.680 seconds/mday       140.87 myears/wday
   ICE Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   OCN Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   ROF Run Time:       0.815 seconds        0.163 seconds/mday      1452.22 myears/wday
   GLC Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   WAV Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   ESP Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   CPL COMM Time:      2.595 seconds        0.519 seconds/mday       456.09 myears/wday
```

Let's reset the PE-layout and
Put the atm on its own pes –
but with much fewer tasks

```
./xmlchange NTASKS=496
./xmlchange NTASKS_ATM=16
./xmlchange ROOTPE_ATM=496
```

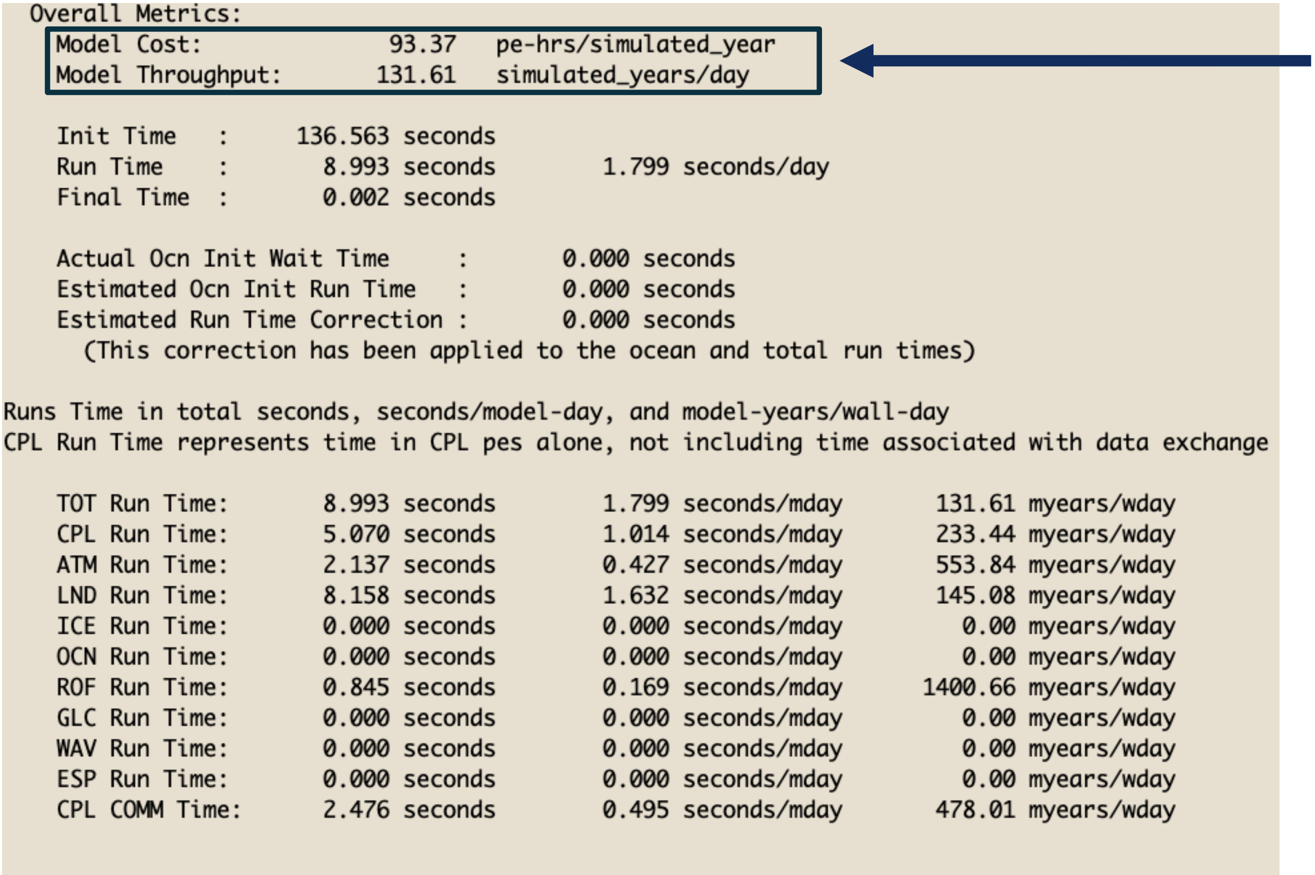
./pelayout now gives



```
Comp  NTASKS  NTHRDS  ROOTPE
CPL :    496/      1;      0
ATM :     16/      1;    496
LND :    496/      1;      0
ICE :    496/      1;      0
OCN :    496/      1;      0
ROF :    496/      1;      0
GLC :    496/      1;      0
WAV :    496/      1;      0
ESP :    496/      1;      0
```

How can we get better performance? (cont)

Now let’s look at the new timing file



```
 Overall Metrics:
   Model Cost:              93.37   pe-hrs/simulated_year
   Model Throughput:       131.61   simulated_years/day

   Init Time   :     136.563 seconds
   Run Time    :       8.993 seconds        1.799 seconds/day
   Final Time  :       0.002 seconds

   Actual Ocn Init Wait Time     :       0.000 seconds
   Estimated Ocn Init Run Time   :       0.000 seconds
   Estimated Run Time Correction :       0.000 seconds
     (This correction has been applied to the ocean and total run times)

Runs Time in total seconds, seconds/model-day, and model-years/wall-day
CPL Run Time represents time in CPL pes alone, not including time associated with data exchange

   TOT Run Time:       8.993 seconds        1.799 seconds/mday       131.61 myears/wday
   CPL Run Time:       5.070 seconds        1.014 seconds/mday       233.44 myears/wday
   ATM Run Time:       2.137 seconds        0.427 seconds/mday       553.84 myears/wday
   LND Run Time:       8.158 seconds        1.632 seconds/mday       145.08 myears/wday
   ICE Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   OCN Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   ROF Run Time:       0.845 seconds        0.169 seconds/mday      1400.66 myears/wday
   GLC Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   WAV Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   ESP Run Time:       0.000 seconds        0.000 seconds/mday         0.00 myears/wday
   CPL COMM Time:      2.476 seconds        0.495 seconds/mday       478.01 myears/wday
```

Throughput is increased by ~30% with the same number of total PES

Model cost is less

HUGE performance saving
If you will run a long time!