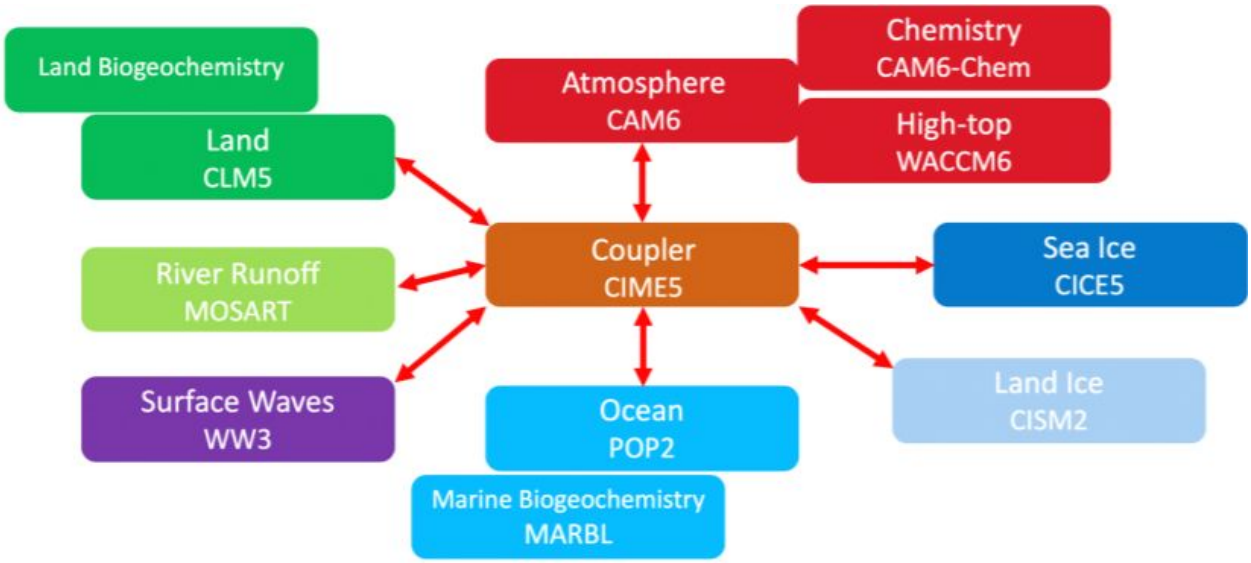
Abstract

Community Earth System Model (CESM; Gent et al., 2011; Lindsay et al., 2014) of the University Corporation for Atmospheric Research (UCAR) and National Center for Atmospheric Research (NCAR), Boulder, USA is a fully-coupled, **community**, global climate **model** that provides state-of-the-art computer simulations of the **Earth's** past, present, and future climate. NorESM (Norwegian Earth System Model) builds on the CESM but with specific additions including:

- Chemistry-aerosol-cloud module of the atmosphere component developed by MET and UiO.
- An ocean component that originates from the Miami Isopycnic Coordinate Ocean Model (MICOM) but extensively modified at NERSC and UNI.
- Hamburg Model of Ocean Carbon Cycle adopted for use with an isopycnic ocean model by UiB and UNI.
- New options for computing turbulent atmosphere-ocean fluxes and maintaining atmospheric energy and momentum consistency implemented by UNI.

Application overview

CESM/NorESM is composed of separate models simultaneously simulating the Earth's atmosphere, ocean, land, river run-off, land-ice, and sea-ice, plus one central coupler/moderator component. CESM/NorESM allows researchers to conduct fundamental research into the Earth's past, present, and future climate states.



CESM can be run on a number of different hardware platforms, and has a relatively flexible design with respect to processor layout of components. However, it has not been built for running within containers.

The following are the external system and software requirements for installing and running CESM2.

- UNIX style operating system such as CNL, AIX or Linux
- python >= 2.7
- perl 5 and perl(XML::LibXML)
- subversion client (version 1.8 or greater but less than v1.11) for downloading CAM, POP, and WW3
- git client (1.8 or greater)
- Fortran compiler with support for Fortran 2003
- C compiler
- MPI
- NetCDF 4.3 or newer (C and Fortran)
- pnetcdf 1.7.0 is required and 1.8.1 is optional but recommended
- LAPACK and BLAS (we use mkl when using Intel compilers)
- CMake 2.8.6 or newer

Parallel-netCDF (pnetcdf) version 1.7.0 or later should be used with CESM. It is a library that is file-format compatible with netCDF, and provides higher performance by using MPI-IO. Pnetcdf is enabled by setting the `$PNETCDF_PATH` Makefile variable in the `Macros.make` file.

Containerization approach

Anne already did quite some work before the hackathon. At the beginning of the hackathon we already had containers with five small models. This is a list of the models (from smallest to largest):
- FKESSLER (Moist baroclinic wave with Kessler microphysics)
- F1850 (Pre-industrial cam/clm with prescribed ice/ocn)

- CESM historic with CAM6 and CLM5 (no ocean)
- VR-CESM historic with CAM6 and CLM5 (no ocean) and variable resolution
- NorESM-CAM

Anne has also been working on two more containers with bigger configuration (fully coupled configurations), but in this case there were still build-time issues to solve:
- B1850
- NorESM fully coupled N1850 compset and f19_tn14 resolution (the ultimate goal)

More information and links to github repositories (for each configuration) can be found at https://nordicesmhub.github.io/containers/

We decided to go through the models from smallest to largest and make sure that they can all run on Piz Daint.

Setup for running on Piz Daint

Use sarus for running containers:

module load sarus

FKESSLER

We needed to rebuild the container, in order to have MPICH 3.1.4 (compatible with Cray). After that, we could successfully run on Dom:

sarus pull nordicesmhub/fkessler:1.0

mkdir -p /scratch/snx3000/hck07/container-hackaton-for-modelers/run/fkessler

sarus run -t
--mount=src=/scratch/snx3000/hck07/container-hackaton-for-modelers/run/fkessler,dst=/run,type=bind
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/root/inputdata,type=bind nordicesmhub/fkessler:1.0 bash -c ‘rm $(which mpiexec); cd /root/cases/fkessler; ./case.submit; cp -r /root/work/fkessler/run/* /run/’

srun -C gpu --reservation=esiwace_1 --pty -n 4 -N 1 sarus run -t --mpi
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata,dst=/root/in

putdata,type=bind nordicesmhub/fkessler:1.0 bash -c "cd /root/work/fkessler/run && /root/work/fkessler/bld/cesm.exe"

Docker container is available on docker hub (`docker pull nordicesmhub/fkessler:1.0`) and source code is freely available at https://github.com/NordicESMhub/fkessler_docker

F1850

This is still a simple configuration with pre-industrial (1850) atmosphere and land components and prescribed ice and ocean. See https://github.com/NordicESMhub/F1850_docker for dockerfile and associated scripts. The docker image is saved in quay https://quay.io/repository/nordicesmhub/cesm_f1850

mkdir -p /scratch/snx3000tds/mkean/container-hackaton-for-modelers/run/F1850

sarus run -t --mount=src=/scratch/snx3000tds/mkean/container-hackaton-for-modelers/run/F1850,dst=/run,type=bind --mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata,dst=/root/inputdata,type=bind nordicesmhub/cesm_f1850:1.0 bash -c ‘rm $(which mpiexec); cd /root/cases/F1850; ./case.submit; cp -r /root/work/F1850/run /run/’

srun -C mc -n 8 -N 1 sarus run --mpi --mount=src=/scratch/snx3000tds/mkean/container-hackaton-for-modelers/run/F1850,dst=/root/work/F1850/run,type=bind --mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata,dst=/root/inputdata,type=bind nordicesmhub/cesm_f1850:1.0 bash -c 'cd /root/work/F1850/run && /root/work/F1850/bld/cesm.exe'

VR-CESM

Variable resolution. Interesting as it is a variable grid centered over Europe and suitable for climate Science at high latitudes. See https://github.com/NordicESMhub/VR-CESM_docker with docker containers on docker hub https://hub.docker.com/repository/docker/nordicesmhub/cesm_vr

srun -C gpu --reservation=esiwace_1 sarus load /project/csstaff/mkean/container-hackaton-for-modelers/images/vr-cesm.tar nordicesmhub/vr_cesm:1.0

mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesm

sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesm,dst=/run,type=bind
--mount=src=/project/csstaff/$USER/container-hackaton-for-modelers/data/inputdata_vr,dst=/root/inputdata,type=bind load/nordicesmhub/vr_cesm:1.0 bash -c ‘rm $(which mpiexec); cd /root/cases/vr-cesm; ./case.submit; cp -r /root/work/vr-cesm/run/* /run/’

srun -C gpu --reservation=esiwace_1 -N 1 -n 12 sarus run --mpi
--mount=src=/scratch/snx3000/mkean/container-hackaton-for-modelers/run/vr_cesm,dst=/root/work/vr-cesm/run,type=bind
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/root/inputdata,type=bind load/nordicesmhub/vr_cesm:1.0 bash -c 'cd /root/work/vr-cesm/run && /root/work/vr-cesm/bld/cesm.exe'

We built and ran several cases to check results and attempt to get information on the scalability.

VR-CESM case 2

```
CASE INFO:
  nodes: 4
  total tasks: 48
  tasks per node: 12
  thread count: 1

BATCH INFO:
  FOR JOB: case.run
    ENV:
      Setting Environment KMP_STACKSIZE=64M
      Setting Environment OMP_NUM_THREADS=1
    SUBMIT CMD:
      None

  FOR JOB: case.st_archive
    ENV:
      Setting Environment KMP_STACKSIZE=64M
      Setting Environment OMP_NUM_THREADS=1
    SUBMIT CMD:
      None
```

mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase2

```
 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase2,dst=
/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'rm $(which mpiexec); cd /root/cases/vr-cesmcase2; ./preview_run; ./case.submit; cp -r
/root/work/vr-cesmcase2/run/* /run/' srun -C gpu \
 --reservation=esiwace_1 -N 3 -n 12 sarus run --mpi
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase2,dst=
/root/work/vr-cesmcase2/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash -c 'cd /root/work/vr-cesmcase2/run
&& /root/work/vr-cesmcase2/bld/cesm.exe'


 srun -C gpu --reservation=esiwace_1 -N 4 -n 48 -t 360 sarus run --mpi \
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase2,dst=
/root/work/vr-cesmcase2/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'cd /root/work/vr-cesmcase2/run && /root/work/vr-cesmcase2/bld/cesm.exe'
```

VR-CESM case 3

```
mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase3

 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase3,dst=
/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'rm $(which mpiexec); cd /root/cases/vr-cesmcase3; ./preview_run; ./case.submit; cp -r
/root/work/vr-cesmcase3/run/* /run/' srun -C gpu \
 --reservation=esiwace_1 -N 3 -n 12 sarus run --mpi
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase3,dst=
/root/work/vr-cesmcase3/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash -c 'cd /root/work/vr-cesmcase3/run
&& /root/work/vr-cesmcase3/bld/cesm.exe'


 srun -C gpu --reservation=esiwace_1 -N 8 -n 96 -t 360 sarus run --mpi \
```

--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase3,dst=/root/work/vr-cesmcase3/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/root/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'cd /root/work/vr-cesmcase3/run && /root/work/vr-cesmcase3/bld/cesm.exe'

VR-CESM case 4

CASE INFO:
  nodes: 16
  total tasks: 192
  tasks per node: 12
  thread count: 1

mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase4

 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase4,dst=/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/root/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'rm $(which mpiexec); cd /root/cases/vr-cesmcase4; ./preview_run; ./case.submit; cp -r /root/work/vr-cesmcase4/run/* /run/' srun -C gpu \
 --reservation=esiwace_1 -N 3 -n 12 sarus run --mpi
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase4,dst=/root/work/vr-cesmcase4/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/root/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash -c 'cd /root/work/vr-cesmcase4/run && /root/work/vr-cesmcase4/bld/cesm.exe'

 srun -C gpu --reservation=esiwace_1 -N 16 -n 192 -t 360 sarus run --mpi \
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase4,dst=/root/work/vr-cesmcase4/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/root/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'cd /root/work/vr-cesmcase4/run && /root/work/vr-cesmcase4/bld/cesm.exe'

Case 5

nodes: 42
total tasks: 504
tasks per node: 12
thread count: 1

```
mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase5

 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase5,dst=
/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'rm $(which mpiexec); cd /root/cases/vr-cesmcase5; ./preview_run; ./case.submit; cp -r
/root/work/vr-cesmcase5/run/* /run/' srun -C gpu \
 --reservation=esiwace_1 -N 3 -n 12 sarus run --mpi
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase5,dst=
/root/work/vr-cesmcase5/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash -c 'cd /root/work/vr-cesmcase5/run
&& /root/work/vr-cesmcase5/bld/cesm.exe'
```

```
 srun -C gpu --reservation=esiwace_2 -N 42 -n 504 -t 360 sarus run --mpi \
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/vr_cesmcase5,dst=
/root/work/vr-cesmcase5/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_vr,dst=/roo
t/inputdata,type=bind load/nordicesmhub/vr_cesm:2.0 bash \
 -c 'cd /root/work/vr-cesmcase5/run && /root/work/vr-cesmcase5/bld/cesm.exe; if [
${SLURM_PROCID} = 0 ]; then cp -R /root/cases/vr-cesmcase5 /root/work/vr-cesmcase5/run/.;
fi'
```

B1850

Initially we couldn’t build the code because of linker errors such as:

/usr/bin/ld: cime_comp_mod.o: in function `__cime_comp_mod_MOD_cime_final':
109 cime_comp_mod.F90:(.text+0x8bd): relocation truncated to fit: R_X86_64_PC32 against symbol `__seq_comm_mct_MOD_cpl_inst_tag' defined in .bss section in ../../gnu/mpich/nodebug/nothreads/mct/noesmf/c1a1l1i1o1r1g1w1e1/lib/libcsm_share.a(seq_comm_mct.o)

The linker error was due to the fact that the .bss section of the program (static variables) is very big (over 7 GB) and the machine instructions generated with the default code model are not able to reference symbols with an offset greater than 32 bits. To solve the issue we recompiled the application with the “medium” memory model. To do so we modified ~/.cime/config_compilers.xml in the container:

```
# cat ~/.cime/config_compilers.xml
<?xml version="1.0"?>

<config_compilers version="2.0">

  <compiler COMPILER="gnu" MACH="espresso">
        <LD>mpifort</LD>
        <AR>ar</AR>
        <SFC>gfortran</SFC>
        <SCC>cc</SCC>
        <SCXX>c++</SCXX>
        <MPIFC>mpifort</MPIFC>
        <MPICC>mpicc</MPICC>
        <MPICXX>mpicxx</MPICXX>
        <NETCDF_PATH>/usr</NETCDF_PATH>
        <CFLAGS>
        <append DEBUG="FALSE"> -mcmodel=large -O2</append>
        </CFLAGS>
        <FFLAGS>
        <append DEBUG="FALSE"> -mcmodel=medium -O2</append>
        <append MODEL="micom"> -fdefault-real-8 </append>
        <append MODEL="cam"> -finit-local-zero </append>
        </FFLAGS>
        <SLIBS>
        <append> -L$(NETCDF_PATH)/lib -lnetcdff -lnetcdf -ldl -mcmodel=large
</append>
        <append> -lblas -llapack -lgomp -lpthread -lm -fopenmp </append>
        <append> -Wl,--verbose -Wl,-Map=/root/inputdata/output.map_test_mdmodel
</append>
        </SLIBS>
  </compiler>
```

</config_compilers>

Test

```
sarus load b1850_gnu_108.tar nordicesmhub_b1850:1.0

mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/B1850

 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/B1850,dst=/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst=/root/inputdata,type=bind load/library/nordicesmhub_b1850:1.0 bash \
 -c 'rm $(which mpiexec); cd /root/cases/B1850; ./preview_run; ./case.submit; cp -r /root/work/B1850/run/* /run/'

 srun -C gpu --reservation=esiwace_2 -N 9 -n 108 -t 360 sarus run --mpi \
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/B1850,dst=/root/work/B1850/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst=/root/inputdata,type=bind     load/library/nordicesmhub_b1850:1.0    bash \
 -c 'cd /root/work/B1850/run && /root/work/B1850/bld/cesm.exe; if [ ${SLURM_PROCID} = 0 ]; then cp -R /root/cases/B1850 /root/work/B1850/run/.; fi'
```

Day-3

Run B1850 with different number of processors and both intel and gnu compilers.

Step-1: Preparation for runs

```
sarus load nordicesmhub_b1850_gnu.tar nordicesmhub_b1850:gnu
sarus load nordicesmhub_b1850_intel.tar nordicesmhub_b1850:intel

for i in 1 2 3 4 5; do
   mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/intel/B1850case$i
   mkdir -p /scratch/snx3000/$USER/container-hackaton-for-modelers/run/gnu/B1850case$i
done
```

Prepare gnu

```
for i in 1 2 3 4 5; do
 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/gnu/B1850case$i,d
st=/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst
=/root/inputdata,type=bind load/library/nordicesmhub_b1850:gnu bash \
 -c “rm \$(which mpiexec); cd /root/cases/B1850case${i}; ./preview_run; ./case.submit; cp -r
/root/work/B1850case${i}/run/* /run/”
done
```

Prepare intel

```
for i in 1 2 3 4 5; do
 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/intel/B1850case$i,
dst=/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst
=/root/inputdata,type=bind load/library/nordicesmhub_b1850:intel  bash \
 -c ". /opt/intel/parallel_studio_xe_2018.1.038/psxevars.sh; rm \$(which mpiexec); cd
/root/cases/B1850case${i}; ./preview_run; ./case.submit; cp -r /root/work/B1850case${i}/run/*
/run/"
done
```

```
for i in 1 2 3 4 5; do
 sarus run -t
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/intel/B1850case$i,
dst=/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst
=/root/inputdata,type=bind load/library/nordicesmhub_b1850:intel  bash \
 -c ". /opt/intel/parallel_studio_xe_2018.1.038/psxevars.sh; rm \$(which mpiexec); cd
/root/cases/B1850case${i}; ./preview_run; ./case.submit; cp -r /root/work/B1850case${i}/run/*
/run/"
done
```

Test

```
 srun -C gpu --reservation=esiwace_2 -N 4 -n 48 -t 360 sarus run --mpi \
```

```
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/intel/B1850case1,dst=/root/work/B1850case1/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst=/root/inputdata,type=bind  load/library/nordicesmhub_b1850:intel       bash \
 -c 'cd /root/work/B1850case1/run && /root/work/B1850case1/bld/cesm.exe; if [ ${SLURM_PROCID} = 0 ]; then cp -R /root/cases/B1850ase1 /root/work/B1850case1/run/.; fi'
```

Test config 1 with gnu

```
 srun -C gpu --reservation=esiwace_2 -N 4 -n 48 -t 360 sarus run --mpi \
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/gnu/B1850case1,dst=/root/work/B1850case1/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst=/root/inputdata,type=bind  load/library/nordicesmhub_b1850:intel       bash \
 -c 'cd /root/work/B1850case1/run && /root/work/B1850case1/bld/cesm.exe; if [ ${SLURM_PROCID} = 0 ]; then cp -R /root/cases/B1850ase1 /root/work/B1850case1/run/.; fi'
```

Test config 1 with intel

```
srun -C gpu --reservation=esiwace_2 -N 4 -n 48 -t 360 sarus run --mpi \
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/intel/B1850case1,dst=/root/work/B1850case1/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst=/root/inputdata,type=bind  load/library/nordicesmhub_b1850:intel       bash \
 -c 'cd /root/work/B1850case1/run && /root/work/B1850case1/bld/cesm.exe; if [ ${SLURM_PROCID} = 0 ]; then cp -R /root/cases/B1850case1 /root/work/B1850case1/run/.; fi'
```

“Production runs”

```
 srun -C gpu --reservation=esiwace_2 -N 9 -n 108 -t 360 sarus run --mpi \
--mount=src=/scratch/snx3000/$USER/container-hackaton-for-modelers/run/B1850,dst=/root/work/B1850/run,type=bind \
--mount=src=/project/csstaff/mkean/container-hackaton-for-modelers/data/inputdata_B1850,dst=/root/inputdata,type=bind     load/library/nordicesmhub_b1850:1.0     bash \
 -c 'cd /root/work/B1850/run && /root/work/B1850/bld/cesm.exe; if [ ${SLURM_PROCID} = 0 ]; then cp -R /root/cases/B1850 /root/work/B1850/run/.; fi'
```

Running with intel

ldconfig was not correct so we had to create a new file intel.mpi.conf (in the container) in /etc/ld.so.conf.d with

```
 /opt/intel/compilers_and_libraries_2018.1.163/linux/mpi/intel64/lib
```

Because sarus needs to know where the MPI libraries are and it uses ldconfig to find out.

docker export 28c7d8c6ca16 -o nordicesmhub_b1850_intel_active.tar

Results with gnu compilers for the 4 cases:

```
# Executed B1850 (gnu case #4) in 1370 seconds
# Executed B1850 (gnu case #2) in 2614 seconds
# Executed B1850 (gnu case #3) in 2071 seconds
# Executed B1850 (gnu case #1) in 3677 seconds
# Executed B1850 (gnu case #4) in 1338 seconds
```

It exhibits similar scaling (poor…) than what we have on our HPC with a fully coupled configuration.

Final conclusions

We managed to run all our cases (from the smallest/simplest to the most complex with a fully coupled version of CESM/NorESM).

Sarus documentation is very good; having a short tutorial would be great too (especially to show how to deal with batch scheduler; for instance with slurm). The hackathon was very successful thanks to Kean who obviously knew how to solve every single problem we had.

Problems we had when using intel compilers with sarus (“lost layer”) were encountered by other teams too. I learned a lot while trying to fix it; in particular how sarus works (“behind the scene”).

We now have a clear roadmap for our model and will continue to do more tests. Sarus has been installed on a small virtual machine (16 processors and 128GB memory) and hopefully will be installed on the Norwegian national infrastructure very soon.