High Resolution EC Earth Porting, Benchmarking on Linux Cluster

15th April, 2011

Chandan Basu, NSC, LiU, Sweden
with
Uwe Fladrich, SMHI, Sweden
Eric Maisonnave, CERFACS, France

Introduction

The EC-EARTH[1] is an earth system modeling application. The three main components of EC-EARTH are IFS (for atomospher), NEMO (for ocean) and OASIS (for coupling). We have tested a high resolution version of EC-EARTH configuration for scaling. The ocean component used for this configuration is ORCA025 which is a 1/4° global configuration. The atmosphere component uses T799 resolution. The source code of NEMO is v-3.2, IFS is Cycle 36r1 and the coupler is OASIS3. The source code of EC-EARTH with necessary modifications, input files and run scripts and are obtained from SMHI. The system used for our scaling studies is EKMAN[2] which is a large linux - X86-64 - infiniband cluster. As many of the modern day clusters fall in this category our benchmarking will be indicative of performance of high resolution EC-EARTH on these type of clusters. The performance of any MPI program may depend on various components, e.g., compilers, MPI libraries, network, filesystem, processor technologies etc. As EC-EARTH is a large and complex program it's performance needs to be tested against these variables. In the first phase of our optimization work we are looking into these factors for a optimized run. The goal is to figure out the bottlenecks and to look at suitable strategies to improve the scaling.

Test system : The details of our test system, EKMAN, with details of hardware & software is given in Table 1.

Table 1

System	Ekman	
Processor	Quad-Core AMD Opteron 2374 HE @ 2.19 GHz	
Interconnect	DDR Infiniband, full bisection bandwidth	
Node	2 processor, 8 core	
No. of nodes	1268	
MPI processes / node	8	
global filesystem	Lustre over infiniband	
Compiler	Intel compiler v-10.1, v-11.1 & v-12.0.1.069 (eval version)	
MPI	Scali MPI v-5.6.6-59413, OpenMPI v-1.4.3	

Compiling the source code

The compilation procedure of EC-EARTH code is described in detail in "compiling-and-running-short-guide.pdf" file which is present in doc/ folder under EC-EARTH root folder. The important thing is to set up an appropriate xml file with description of system specific settings for package paths, compilers, flags, libaries etc. Some template xml files are already present in build-config/ folder under EC-EARTH root folder. The settings used by us is ekman.xml and is given in Appendix A. As the

1 http://ecearth.knmi.nl/
2 http://www.pdc.kth.se/resources/computers/ekman/ekman-hardware

individual packages used in EC-EARTH have quite different build environment we use the semi-automatic utility eexcon found under utility/ folder in EC-EARTH root folder. This utility reads the specific xml file under build-config folder and sets up appropriate Makefiles and other settings for each component package. We have set up a script called ecearth.sh for compilation. The script is placed under EC-EARTH root folder. The script ecearth.sh is given in Appendix B.

Running high resolution EC-EARTH

EC-EARTH build stage produces multiple binaries, one each for NEMO, OASIS and IFS. The MPI implementation used for EC_EARTH should be capable of running multi-binary executable. For our experiments we have used Scali MPI and OpenMPI both of which are capable of handling this. For NEMO package the number of processors are specified at the compile time and should be run with same number of processors. For IFS and OASIS number of processors can be chosen at run time.

EC-EARTH run requires a large number of input files. The input files are namelist files or data files. The namelist type files are input fiels describing different parameter values which control the flow of run. The data files generally contain initial / boundary / restart values obtained from some previous runs or from some experiments. The data files are stored in a common place and accessed from there.

The EC-EARTH run is started by a run script. The run script does three things – 1) copies / symlinks relevant data files in the run directory, 2) creates namelist files based on parameters provided in the run script, 3) launches the mpi job. As EC-EARTH is a multi binary run the step 3 requires exact specification of binary – rank – node map. A typical run script is given in Appendix-C. The run script is given to the job scheduler which starts the job script when the requested resources are allocated for the run.

For optimized run each component runs on sepecific group of nodes. Also due to the specific requirement mpirun is setup such that OASIS gets the 0 - (oas_numproc -1) ranks where oas_numproc is the # of OASIS procs. For OpenMPI runs this is achieved by creating an appfile. A typical appfile looks like :

```
-hostfile oas_nodes -np 10 -x LD_LIBRARY_PATH -x OASIS3 -x OASIS3DEBUGLEVEL  -x
LOCAL_DEFINITION_TEMPLATES  oasis3.MPI1.x

-hostfile ifs_nodes -np 512 -x LD_LIBRARY_PATH -x DR_HOOK_IGNORE_SIGNALS -x OASIS3 -x
OASIS3DEBUGLEVEL  -x LOCAL_DEFINITION_TEMPLATES  ifsmaster-eexcon -v ecmwf -e HRES

-hostfile nem_nodes -np 320 -x LD_LIBRARY_PATH -x   opa-eexcon-16x20
```

where oas_nodes, ifs_nodes and nem_nodes are the node files for OASIS, IFS and NEMO respectively. This appfile is given to mpirun :
mpirun -f appfile.

Scaling of High resolution EC-EARTH

The scaling of EC-EARTH run depends on choice of input parameters especially the coupling

frequency, i/o frequency etc. We have chosen following parameters for our runs :

cpl freq = 3 hrs
run length = 48 hrs
ifs time step = 720 s
Frequency of history write ups = 30 ifs time steps
Frequency of spectral diagnostics = 5 time steps

nemo time step=1200 s
lim time step=3600 s
nemo frequency of write in the output file = 72 nemo time step

In a 48 hour run we have all the significant steps repeated a number of times. So although the total runtime is small it is indicative of scaling behaviour. The results of our runs are shown in Table 1 and Figure 1 and Figure 2. In figures 1 & 2 we have shown the normalized run times.

Table 1

# MPI processes				# step	average time/step (s)					
					only comp steps		only io steps		comp+io steps	
Total	NEMO	IFS	OASIS		Scali MPI	Open MPI	Scali MPI	Open MPI	Scali MPI	Open MPI
394	256	128	10	240	7.19	7.27	17.75	17.80	9.19	9.90
522	256	256	10	240	4.26	4.35	10.34	10.35	4.89	5.66
778	256	512	10	240	2.38	2.45	6.17	6.18	3.10	3.13
1034	256	768	10	240	1.80	1.71	4.89	4.88	2.73	2.32
1290	256	1024	10	240	XX[1]	1.37	XX[1]	5.58	XX[1]	2.17
1466	256	1200	10	240	XX[1]	1.90	XX[1]	6.89	XX[1]	2.85
458	320	128	10	240	7.28	7.60	18.89	19.02	9.81	9.84
586	320	256	10	240	4.35	4.41	10.50	10.74	6.60	5.65
842	320	512	10	240	2.43	2.52	5.84	5.85	2.86	2.87
1098	320	768	10	240	1.65	1.52	4.23	4.22	2.15	2.01
1354	320	1024	10	240	XX[1]	1.14	XX[1]	3.57	XX[1]	1.61
1530	320	1200	10	240	XX[1]	1.08	XX[1]	3.39	XX[1]	1.51

1 Scali MPI jobs can run for maximum ~ 1000 MPI processes

Figure 1 : Nemo # of ranks is 256

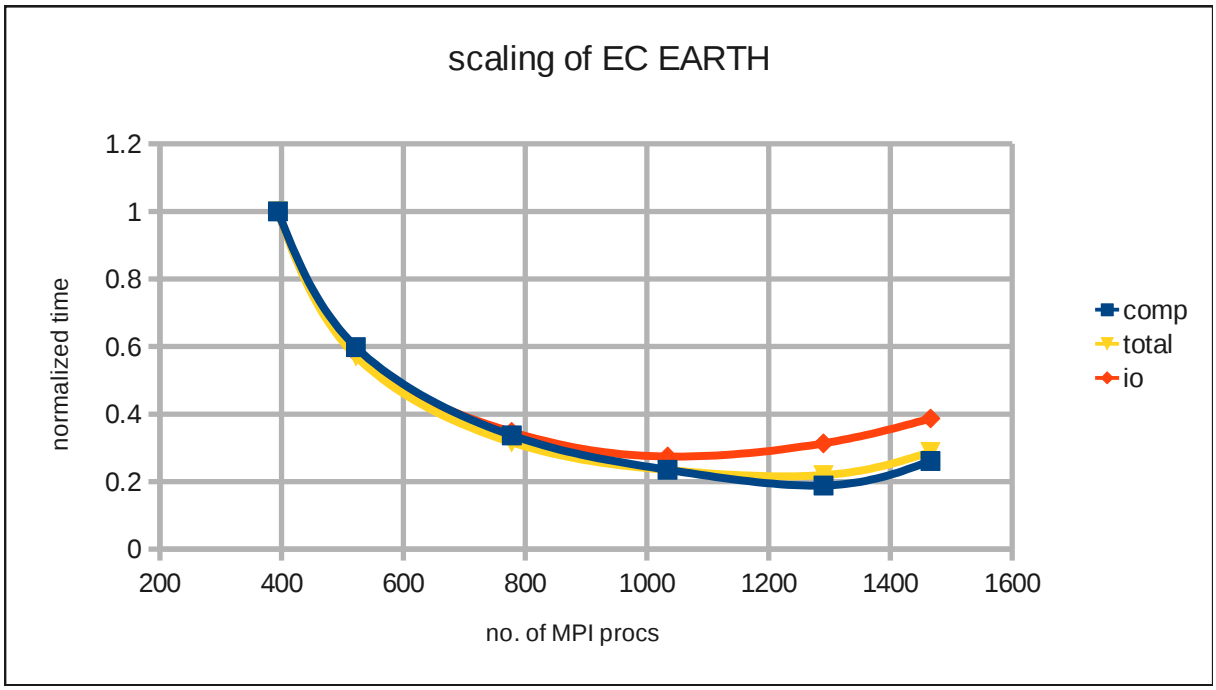
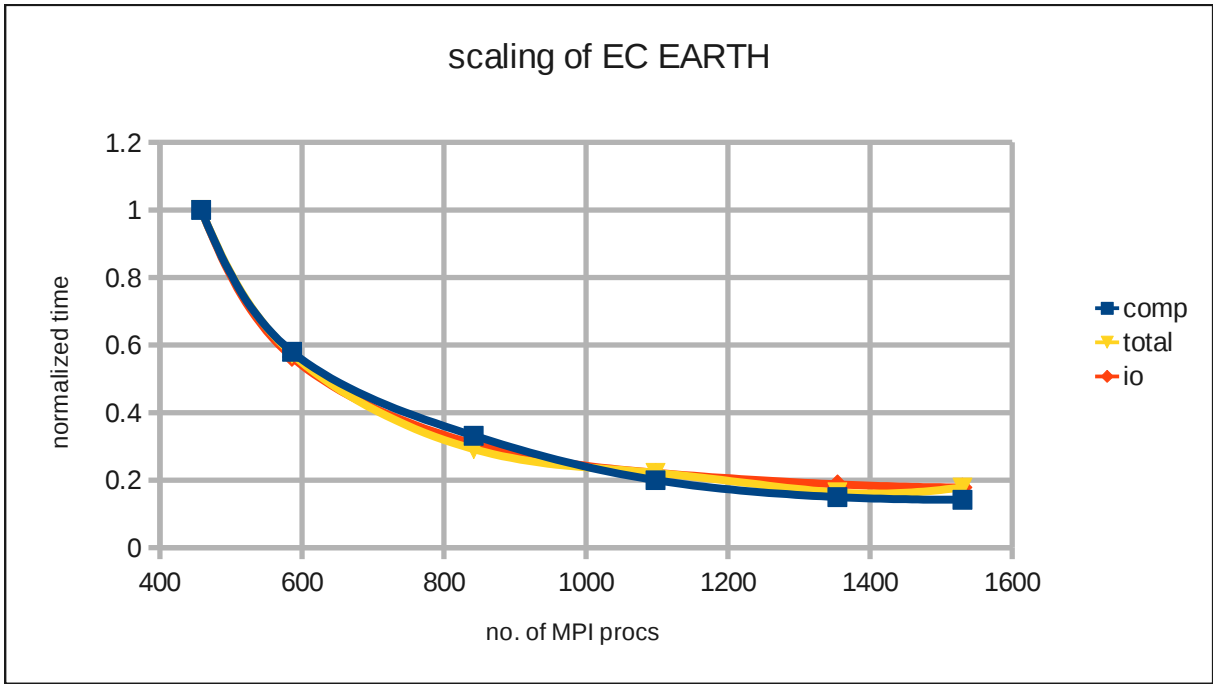


Figure 2 : Nemo # of ranks is 320



Discussion

The high resolution EC-EARTH scales well upto 1500 MPI processes. If NEMO # of ranks is 256 then scaling is achieved by increasing the IFS # of ranks. Upto IFS # of ranks 1024 IFS is slower than NEMO. So we do not see any speedup by increasing NEMO # of ranks. But at NEMO # of ranks 256 and IFS # of ranks 1200 NEMO is slower. So by setting NEMO # of ranks 320 and IFS # of ranks 1200 further speedup can be seen.

The time steps in EC-EARTH which has i/o are much slower. We have separately shown scaling of i/o steps, computational steps and i/o+computational steps. This gives us an idea about which steps are more bottleneck in the overall scaling. The i/o steps are 2 – 3 times more expensive than computational steps. In our experiment every 30^{th} time step (every 6^{th} hour) is i/o step. Actual impact of the i/o steps will depend on the i/o frequency for a particular experiment. Also the i/o file-system used for our experiment is lustre parallel file-system mounted through infiniband. Other slower file-systems, e.g., luster on ethernet can make i/o even slower. Another aspect related to i/o is some i/o steps are much more slower than other i/o steps. This is probably because of load on file-system from other programs. This may be increasingly more problem for bigger clusters. One of the tuning target for EC-EARTH will be more efficient i/o. One idea that we would like to try on peta-flop systems is using node local disks for intermediate storage. This way the i/o will be parallel and independent of global file system.

Appendix A

file : build-config/ekman.xml

```
<experiment>
   <!-- FILESET "default" <<< -->
   <fileset longname="Default fileset" name="default">
      <translation longname="Nemo makefile configuration" name="nemo_makefile_config">
         <template filename="../nemo3-orca025/util/Makefile.d/Makefile.config.eexcon.tmpl"/>
         <target filename="../nemo3-orca025/util/Makefile.d/Makefile.config.eexcon"/>
      </translation>

      <translation longname="Nemo global definition file" name="nemo_aa_make_gdef">
         <template filename="../nemo3-orca025/util/AA_make.gdef.tmpl"/>
         <target filename="../nemo3-orca025/util/AA_make.gdef"/>
      </translation>

      <translation longname="Nemo ... file" name="nemo_par_oce">
         <template filename="../nemo3-orca025/modeles/NEMO/OPA_SRC/par_oce.F90.tmpl"/>
         <target filename="../nemo3-orca025/modeles/NEMO/OPA_SRC/par_oce.F90"/>
      </translation>

      <translation longname="IFS/36 makefile configuration" name="ifs36_makefile_config">
         <template filename="../ifs-36r1/Makefile.d/Makefile.config.eexcon.tmpl"/>
         <target filename="../ifs-36r1/Makefile.d/Makefile.config.eexcon"/>
      </translation>

      <translation longname="OASIS makefile configuration" name="oasis_makefile_config">
         <template filename="../oasis3/util/make_dir/Makefile.d/Makefile.config.eexcon.tmpl"/>
         <target filename="../oasis3/util/make_dir/Makefile.d/Makefile.config.eexcon"/>
      </translation>
   </fileset>

   <!-- PLATFORM "ekman-intel-scampi" <<< -->
   <!--
      HOST:       ekman.pdc.kth.se
      ARCH:       linux_x86_64
      CPU MODEL:   Quad-Core AMD Opteron Processor 2374 HE
      USER:       cbasu
      COMPILER:    i-compilers
      MPI:        ScaMPI
      BLAS/LAPACK: Intel MKL
   -->
   <platform name="ekman-intel-scampi">
      <param name="ECEARTH_BASE_DIR"
           longname="EC-Earth base directory"
           type="PATH">
        /cfs/ekman/nobackup/c/cbasu/install/ecearth/ecearth3-r263

      </param>

      <!-- MPI -->
      <param name="MPI_BASE_DIR"
           longname="MPI base directory"
           type="PATH">
        /pdc/vol/scampi/5.6.6-59413
      </param>

      <param name="MPI_INC_SUBDIR"
           longname="MPI include directory relative to base dir"
           type="PATH">
        include64
      </param>
```

```
<param name="MPI_LIB_SUBDIR"
      longname="MPI lib directory relative to base dir"
      type="PATH">
   lib64
</param>

<param name="MPI_LIBS_WITHOUT_L"
      longname="MPI libraries (without -l prefix)"
      type="STRING">
   fmpi mpi
</param>

<!-- LAPACK -->
<param name="LAPACK_BASE_DIR"
      longname="LAPACK base directory"
      type="PATH">
   /pdc/vol/mkl/10.0.3.020
</param>

<param name="LAPACK_LIB_SUBDIR"
      longname="LAPACK lib directory relative to base dir"
      type="PATH">
   lib/em64t
</param>

<param name="LAPACK_LIBS_WITHOUT_L"
      longname="LAPACK libraries (without -l prefix)"
      type="STRING">
   mkl_lapack mkl guide pthread
</param>

<!-- NetCDF -->
<param name="NETCDF_BASE_DIR"

      longname="NetCDF base directory"
      type="PATH">
   /afs/pdc.kth.se/home/u/ufla/Public/Projects/netcdf/intel-mcmodel-medium
</param>

<param name="NETCDF_INC_SUBDIR"
      longname="NetCDF include directory relative to base dir"
      type="PATH">
   include
</param>

<param name="NETCDF_LIB_SUBDIR"
      longname="NetCDF lib directory relative to base dir"
      type="PATH">
   lib
</param>

<param name="NETCDF_LIBS_WITHOUT_L"
      longname="NetCDF libraries (without -l prefix)"
      type="STRING">
   netcdf
</param>

<!-- MAKE -->
<param name="MAKE"
      longname="Make command (GNU make >3.81 needed!)"
      type="STRING">
   make
</param>

<!-- F90 Compiler -->
```

```
<param name="F90"
      longname="F90 Compiler"
      type="STRING">
   ifort
</param>

<param name="F90FLAGS"
      longname="General F90 flags for compiling"
      type="STRING">
   -O2 -g -traceback -vec-report0 -r8
</param>

<!-- C Compiler -->
<param name="CC"
      longname="C Compiler"
      type="STRING">
   icc

</param>

<param name="CFLAGS"
      longname="General C flags for compiling"
      type="STRING">
   -O2 -g -traceback
</param>

<!-- Linker -->
<param name="LD"
      longname="Linker"
      type="STRING">
   ifort
</param>

<param name="LDFLAGS"
      longname="General flags for linking"
      type="STRING">
   -O2 -g -traceback
</param>

<!-- Library builder -->
<param name="AR"
      longname="Command for building libraries from object files (usually ar)"
      type="STRING">
   ar
</param>

<param name="ARFLAGS"
      longname="Flags for library building command (When using ar: include u)"
      type="STRING">
   curv
</param>

<!-- C Preprocessor -->
<param name="CPP"
      longname="C preprocessor command"
      type="STRING">
   icc
</param>

<param name="CPPFLAGS"
      longname="C preprocessor flags"
      type="STRING">
   -P -C
</param>
```

```
<!-- OASIS specific stuff -->

<param name="OASIS_ADD_F90FLAGS"
       longname="More F90 flags for Oasis"
       type="STRING">
   -mcmodel=medium -132 -check pointers -check uninit
</param>

<param name="OASIS_ADD_CPPDEFS"
       longname="More CPP/FPP macros for Oasis"
       type="STRING">
   use_oasis_para
</param>

<param name="OASIS_ADD_LDFLAGS"
       longname="More LD flags for Oasis"
       type="STRING">
   -mcmodel=medium
</param>

<!-- NEMO specific stuff -->
<param name="NEMO_ADD_F90FLAGS"
       longname="More F90 flags for Nemo"
       type="STRING">
   -module $(MODDIR) -mcmodel=medium -check pointers -check uninit
</param>

<param name="NEMO_ADD_LDFLAGS"
       longname="More LD flags for Nemo"
       type="STRING">
   -shared-intel -mcmodel=medium
</param>

<!-- IFS specific stuff -->
<param name="IFS_FPPDEFS"
       longname="Preprocessor defs for IFS sources"
       type="STRING">
   linux LINUX LITTLE LITTLE_ENDIAN POINTER_64 BLAS
</param>

<param name="IFSAUX_ADD_F90FLAGS"
       longname="More F90 flags for ifs/ifsaux"
       type="STRING">
</param>

<!-- EMOS specific stuff -->
<param name="EMOS_ADD_FFLAGS"
       longname="Additional Fortran compiler flags for EMOSLIB"
       type="STRING">
   -i4
</param>
<param name="EMOS_FPPDEFS"
       longname="Fortran preprocessor defs for EMOSLIB"
       type="STRING">
   linux LITTLE_ENDIAN POINTER_64 INTEGER_IS_INT REAL_8 REAL_BIGGER_THAN_INTEGER

</param>
<param name="EMOS_CPPDEFS"
       longname="C preprocessor defs for EMOSLIB"
       type="STRING">
   $(EMOS_FPPDEFS) FOPEN64
</param>

<!-- F90 Dependency generator -->
<param name="MAKEDEPF90"
```

```
        longname="F90 dependency generator"
        type="STRING">
      $(ECEARTH_BASE_DIR)/util/makedepf90/bin/makedepf90
    </param>
  </platform>

  <!-- END of PLATFORM "ekman-intel-scampi" >>> -->
  <!-- PLATFORM "ekman-intel-openmpi" <<< -->

  <!--
    HOST:       ekman.pdc.kth.se
    ARCH:       linux_x86_64
    CPU MODEL:   Quad-Core AMD Opteron Processor 2374 HE
    USER:       cbasu
    COMPILER:    Intel compiler (icc+ifort)
    MPI:        Open MPI
    BLAS/LAPACK: Intel MKL
  -->
  <platform name="ekman-intel-openmpi">
    <param name="ECEARTH_BASE_DIR"
        longname="EC-Earth base directory"
        type="PATH">
      /cfs/ekman/nobackup/c/cbasu/install/ecearth/ecearth3-r263
    </param>
          <!-- MPI -->
          <param name="MPI_BASE_DIR"
              longname="MPI base directory"
              type="PATH">
          </param>

    <param name="MPI_INC_SUBDIR"
            longname="MPI include directory relative to base dir"
            type="PATH">
      </param>
    <param name="MPI_LIB_SUBDIR"
              longname="MPI lib directory relative to base dir"
                type="PATH">
          </param>

    <param name="MPI_LIBS_WITHOUT_L"
              longname="MPI libraries (without -l prefix)"
              type="STRING">
          </param>

    <!-- LAPACK -->
    <param name="LAPACK_BASE_DIR"
        longname="LAPACK base directory"
        type="PATH">
      /pdc/vol/i-compilers/11.1/icc/mkl
    </param>

    <param name="LAPACK_LIB_SUBDIR"
        longname="LAPACK lib directory relative to base dir"
        type="PATH">
      lib/em64t
    </param>

    <param name="LAPACK_LIBS_WITHOUT_L"
        longname="LAPACK libraries (without -l prefix)"
        type="STRING">
      mkl_lapack95_lp64 mkl_intel_lp64 mkl_sequential mkl_core pthread
    </param>

    <!-- NetCDF -->
    <param name="NETCDF_BASE_DIR"
```

```
      longname="NetCDF base directory"
      type="PATH">
  /afs/pdc.kth.se/home/u/ufla/Public/Projects/netcdf/intel-mcmodel-medium
</param>

<param name="NETCDF_INC_SUBDIR"
      longname="NetCDF include directory relative to base dir"
      type="PATH">
   include
</param>

<param name="NETCDF_LIB_SUBDIR"
      longname="NetCDF lib directory relative to base dir"
      type="PATH">
   lib
</param>


<param name="NETCDF_LIBS_WITHOUT_L"
      longname="NetCDF libraries (without -l prefix)"
      type="STRING">
   netcdf
</param>

<!-- MAKE -->
<param name="MAKE"
      longname="Make command (GNU make >3.81 needed!)"
      type="STRING">
   make

</param>
<!-- F90 Compiler -->
<param name="F90"
      longname="F90 Compiler"
      type="STRING">
          mpif90
</param>

<param name="F90FLAGS"
      longname="General F90 flags for compiling"
     type="STRING">
   -O2 -g -traceback -vec-report0 -r8
</param>

<!-- C Compiler -->
<param name="CC"
      longname="C Compiler"
      type="STRING">
             mpicc
</param>

<param name="CFLAGS"
      longname="General C flags for compiling"
      type="STRING">
   -O2 -g -traceback
</param>

<!-- Linker -->
<param name="LD"
      longname="Linker"
      type="STRING">
   mpif90
</param>

<param name="LDFLAGS"
```

```
      longname="General flags for linking"
      type="STRING">
   -O2 -g -traceback
</param>

<!-- Library builder -->
<param name="AR"
      longname="Command for building libraries from object files (usually ar)"
      type="STRING">
   ar
</param>

<param name="ARFLAGS"
      longname="Flags for library building command (When using ar: include u)"
      type="STRING">
   curv
</param>

<!-- C Preprocessor -->
<param name="CPP"
      longname="C preprocessor command"
      type="STRING">
   mpicc_my
</param>

<param name="CPPFLAGS"
      longname="C preprocessor flags"
      type="STRING">
   -P -C
</param>

<!-- OASIS specific stuff -->
<param name="OASIS_ADD_F90FLAGS"
      longname="More F90 flags for Oasis"
      type="STRING">
   -mcmodel=medium  -132 -check pointers -check uninit
</param>

<param name="OASIS_ADD_CPPDEFS"
      longname="More CPP/FPP macros for Oasis"
      type="STRING">
   use_oasis_para
</param>

<param name="OASIS_ADD_LDFLAGS"
      longname="More LD flags for Oasis"
      type="STRING">
   -mcmodel=medium
</param>


<!-- NEMO specific stuff -->
<param name="NEMO_ADD_F90FLAGS"
      longname="More F90 flags for Nemo"
      type="STRING">
            -module $(MODDIR) -mcmodel=medium -check pointers -check uninit
</param>

<param name="NEMO_ADD_LDFLAGS"
      longname="More LD flags for Nemo"
      type="STRING">
   -shared-intel -mcmodel=medium
</param>

<!-- IFS specific stuff -->
```

```
    <param name="IFS_FPPDEFS"
         longname="Preprocessor defs for IFS sources"
         type="STRING">
      linux LINUX LITTLE LITTLE_ENDIAN POINTER_64 BLAS
    </param>

    <param name="IFSAUX_ADD_F90FLAGS"
         longname="More F90 flags for ifs/ifsaux"
         type="STRING">
    </param>

    <!-- EMOS specific stuff -->
    <param name="EMOS_ADD_FFLAGS"
         longname="Additional Fortran compiler flags for EMOSLIB"
         type="STRING">
      -i4
    </param>

    <param name="EMOS_FPPDEFS"
         longname="Fortran preprocessor defs for EMOSLIB"
         type="STRING">
      linux LITTLE_ENDIAN POINTER_64 INTEGER_IS_INT REAL_8 REAL_BIGGER_THAN_INTEGER
    </param>

    <param name="EMOS_CPPDEFS"
         longname="C preprocessor defs for EMOSLIB"
         type="STRING">
      $(EMOS_FPPDEFS) FOPEN64
    </param>

    <!-- F90 Dependency generator -->
    <param name="MAKEDEPF90"
         longname="F90 dependency generator"
         type="STRING">
      $(ECEARTH_BASE_DIR)/util/makedepf90/bin/makedepf90
    </param>

  </platform>
  <!-- END of PLATFORM "ekman-intel-openmpi" >>> -->
  <!-- MODEL "nemo" <<< -->
  <model longname="Nemo ocean model" name="nemo">

    <param name="numproc_i" longname="Number of procs along i direction" type="INTEGER">
      16
    </param>

    <param name="numproc_j" longname="Number of procs along j direction" type="INTEGER">
      16
    </param>

  </model>
  <!-- END of MODEL "nemo" >>> -->

</experiment>
```

Appendix B

file: ecearth.sh

```
#!/bin/sh
here=`pwd`
cd ifs-36r1

#../util/eexcon/eexcon -p ekman-intel-scampi -f default ../build-config/ekman.xml
../util/eexcon/eexcon -p ekman-intel-openmpi -f default ../build-config/ekman.xml

if [ "$?" = "1" ]; then
  echo "exit 1"
  exit
fi

cd ${here}/oasis3/util/make_dir
make BUILD_ARCH=eexcon -f TopMakefileOasis3 realclean
make -f TopMakefileOasis3 BUILD_ARCH=eexcon

cd ${here}/nemo3-orca025/modeles/UTIL/
./fait_config ORCA025_LIM2
cd ../../util/
./ins_make -t ecearth3 -m MPI1
cd ../config/ORCA025_LIM2/

make BUILD_ARCH=eexcon clean
make BUILD_ARCH=eexcon

cd ${here}/ifs-36r1
make BUILD_ARCH=eexcon realclean
make -j 6 BUILD_ARCH=eexcon lib
make BUILD_ARCH=eexcon master
```

Appendix C

Run script for launching EC-EARTH with OpenMPI. The run script calls some scripts internally. Those are not given here.

File : run_ompi.sh

```
#!/bin/sh


# On ekman, run with:
#   esubmit -m -n <nodes> -t <time> './<script> 1> job.${SP_JID}.out 2>&1'
#   esubmit -m -n 50 -t 50 './run.sh 1> out/job.${SP_JID}.out 2>&1'


set -ue


# ============================================================================
# *** BEGIN Configuration
# ============================================================================


# ----------------------------------------------------------------------------
# *** General configuration
# ----------------------------------------------------------------------------


run_arch=intel-openmpi


# Experiment name (exactly 4 letters!)
exp_name=HRES
ifs_grid=T799
nem_grid=ORCA025


# Directories
ecearth_base_dir=/cfs/ekman/nobackup/c/cbasu/install/ecearth/ecearth3-r263
start_dir=${PWD}
ctrl_file_dir=${start_dir}/ctrl
run_dir=/cfs/ekman/nobackup/c/cbasu/data/${exp_name}
```

```
ini_data_dir=/cfs/ekman/nobackup/e/emaison/ecearth3/setup/${ifs_grid}-${nem_grid}
#ini_data_dir=/cfs/ekman/nobackup/u/ufla/ecearth3/setup/${ifs_grid}-${nem_grid}

# ...
run_length_hrs=48
cpl_freq_hrs=3
run_length_prev_hrs=0

# ...
proc_per_node=8
build_arch=eexcon

# -----------------------------------------------------------------------------
# *** IFS configuration
# -----------------------------------------------------------------------------


ifs_numproc_v=16
ifs_numproc_w=8
(( ifs_numproc = $ifs_numproc_v * $ifs_numproc_w ))

ifs_exe_file=${ecearth_base_dir}/ifs-36r1/bin/ifsmaster-${build_arch}

#emos_gribtemplates=${ecearth_base_dir}/ifs-36r1/src/emos-370/gribtemplates
emos_gribtemplates=gribtemplates

ifs_lastout=true

ifs_tstep=720

(( ifs_run_length = $run_length_hrs * 3600 / $ifs_tstep ))

(( ifs_output_freq = 6 * 3600 / $ifs_tstep ))
```

```
(( ifs_di_freq = 1 * 3600 / $ifs_tstep ))

(( ifs_cpl_freq = $cpl_freq_hrs * 3600 / $ifs_tstep ))

# ------------------------------------------------------------------------------
# *** Nemo/LIM configuration
# ------------------------------------------------------------------------------


# 1 NEMO not compiled with key_iomput (classic IOIPSL outputs)
#               nem_ios=-1
# 2 NEMO compiled with key_iomput (IOM outputs without separate ioserver)
#               nem_ios=0
# 3 NEMO compiled with key_iomput (IOM outputs with separate ioserver)
#               nem_ios=1
nem_ios=-1

[ $nem_ios -gt 0 ] && useserv=.true. || useserv=.false.

nem_numproc_i=16
nem_numproc_j=16

nem_exe_dir=${ecearth_base_dir}/nemo3-orca025/bin
nem_exe_file=${nem_exe_dir}/opa-${build_arch}-${nem_numproc_i}x${nem_numproc_j}
[ $nem_ios -gt -1 ] && nem_exe_file=${nem_exe_file}-iomod
ios_exe_file=${nem_exe_dir}/ioserver-${build_arch}

nem_numproc=$(( nem_numproc_i * nem_numproc_j ))

nem_tstep=1200
lim_tstep=3600
```

```
(( nem_run_length = $run_length_hrs * 3600 / $nem_tstep ))

# daily output
(( nem_output_freq = 86400 / $nem_tstep ))

(( nem_bc_compute = $lim_tstep / $nem_tstep ))

# ------------------------------------------------------------------------
# *** Oasis configuration
# ------------------------------------------------------------------------

oas_namcut_script=${start_dir}/namsplit.pl

oas_numproc=10

oas_exe_file=${ecearth_base_dir}/oasis3/${build_arch}/bin/oasis3.MPI1.x

# ========================================================================
# *** END of Configuration
# ========================================================================

function cleanup()
{
   set +u

   if [ -r ${tempfile} ]
   then
      rm -f ${tempfile}
   fi

   stdout_file=${start_dir}/out/job.${SP_JID}.out

   if [ -r ${stdout_file} -a -d ${run_dir} ]
```

```
   then
      cp ${stdout_file} ${run_dir}
   fi
}

trap 'cleanup' 0 1 2 15

function info()
{
   echo "*II* $@"
}

function error()
{
   echo "*EE* $@"
   exit 1
}

# ------------------------------------------------------------------------
# *** This is where the code begins ...
# ------------------------------------------------------------------------

#modules="i-compilers/10.1.2008-02-07 openmpi/1.3-intel"
modules="i-compilers/10.1.2008-02-07 beta-modules openmpi/1.4.3-intel_10.1"
# not working properly: changes initial date of restart file ...
#modules="i-compilers/11.1 beta-modules scampi"

set +u
source /pdc/modules/etc/init/sh
module purge
module add easy/1.8.ekman
for m in "${modules} grib"
```

```
do
   module add ${m}
done
set -u

if [ -z "${LD_LIBRARY_PATH}" ]
then
   export LD_LIBRARY_PATH="/pdc/vol/i-compilers/11.1/icc/mkl/lib/em64t"
else
   export LD_LIBRARY_PATH="${LD_LIBRARY_PATH}:/pdc/vol/i-compilers/11.1/icc/mkl/lib/em64t"
fi

# Create/clean run dir
if [ ! -d ${run_dir} ]
then
   mkdir -p ${run_dir}
else
   rm -fr ${run_dir}/*
fi

# Go to run_dir. Everything is done from here
cd ${run_dir}

# Link executables
ln -s ${ifs_exe_file}
ln -s ${oas_exe_file}
ln -s ${nem_exe_file}
[ $nem_ios -gt 0 ] && ln -s ${ios_exe_file}

# Create IFS and Nemo/LIM namelists
if (( $ifs_numproc > 512 ))
then
  . ${ctrl_file_dir}/namelist.ifs36.sh > fort.4
```

```
else
   . ${ctrl_file_dir}/namelist.ifs36_test.sh > fort.4
fi
. ${ctrl_file_dir}/namelist.nemo.sh  > namelist
. ${ctrl_file_dir}/namelist.lim2.sh  > namelist_ice
[ $nem_ios -gt -1 ] && . ${ctrl_file_dir}/namelist.ios.sh  > xmlio_server.def
. ${ctrl_file_dir}/namcouple.sh.seq      > namcouple_tmp
if (( $nem_ios < 1 ))
then
  sed -e /IOS_PROCS/d -e s/'.*'ionemo/"2 ifsmod oceanx"/ namcouple_tmp > namcouple
else
  sed -e s/IOS_PROCS/"1 0"/ namcouple_tmp > namcouple
fi

# Split Oasis namcuple file
${oas_namcut_script} ${oas_numproc} namcouple

# IFS data files
ln -s ${ini_data_dir}/ifs/ICMGGa07lINIUA ICMGG${exp_name}INIUA
ln -s ${ini_data_dir}/ifs/ICMGGa07lINIT  ICMGG${exp_name}INIT
ln -s ${ini_data_dir}/ifs/ICMSHa07lINIT  ICMSH${exp_name}INIT
ln -s ${ini_data_dir}/ifs/postins
ln -s ${ini_data_dir}/ifs/dirlist
ln -s ${ini_data_dir}/ifs/rtables/* .

mkdir -p gribtemplates
ln -s ${ecearth_base_dir}/ifs-36r1/src/emos-370/gribtemplates/* gribtemplates/

# ...
tempfile=tmp.$$
grib_set -s dataDate=19951215               \
        ${ini_data_dir}/ifs/ICMCL-split_12 \
```

```
        ICMCL${exp_name}INIT
for m in 01 02 03 04 05 06 07 08 09 10 11 12
do
   grib_set -s dataDate=1996${m}15              \
            ${ini_data_dir}/ifs/ICMCL-split_${m} \
            ${tempfile}
   cat ${tempfile} >> ICMCL${exp_name}INIT
done
grib_set -s dataDate=19970115               \
        ${ini_data_dir}/ifs/ICMCL-split_01 \
        ${tempfile}
cat ${tempfile} >> ICMCL${exp_name}INIT
rm -f ${tempfile}

# Nemo data files
ln -s ${ini_data_dir}/nemo/bathy_meter.nc
ln -s ${ini_data_dir}/nemo/coordinates.nc

ln -s ${ini_data_dir}/nemo/data_1m_potential_temperature_nomask_01.nc data_1m_potential_temperature_nomask_1.nc
ln -s ${ini_data_dir}/nemo/data_1m_potential_temperature_nomask_02.nc data_1m_potential_temperature_nomask_2.nc
ln -s ${ini_data_dir}/nemo/data_1m_potential_temperature_nomask_03.nc data_1m_potential_temperature_nomask_3.nc

ln -s ${ini_data_dir}/nemo/data_1m_salinity_nomask_01.nc data_1m_salinity_nomask_1.nc
ln -s ${ini_data_dir}/nemo/data_1m_salinity_nomask_02.nc data_1m_salinity_nomask_2.nc
ln -s ${ini_data_dir}/nemo/data_1m_salinity_nomask_03.nc data_1m_salinity_nomask_3.nc

ln -s ${ini_data_dir}/nemo/ahmcoef

# IOM data file
[ $nem_ios -gt -1 ] && ln -s ${ini_data_dir}/nemo/iodef.orca025.xml iodef.xml

# Oasis name_table file
ln -s ${ini_data_dir}/oasis/cf_name_table.txt
```

```
# Oasis weight dirs
ln -s ${ini_data_dir}/oasis/areas.nc
ln -s ${ini_data_dir}/oasis/grids.nc
ln -s ${ini_data_dir}/oasis/masks.nc

#note: some rmp_$n.nc files are useless
for (( n=0; n<oas_numproc; n++ ))
  do
     ln -s ${ini_data_dir}/oasis/wgt/rmp_A400_to_Ot25_GAUSWGT.nc rmp_A400_to_Ot25_GAUSWGT_$n.nc
     ln -s ${ini_data_dir}/oasis/wgt/rmp_Ot25_to_A400_GAUSWGT.nc rmp_Ot25_to_A400_GAUSWGT_$n.nc
  done
unset n

# Oasis restart files
cpl_fields="TAUX_OCE TAUY_OCE TAUX_ICE TAUY_ICE \
          QS___OCE QS___ICE QNS__OCE QNS__ICE DQDT_ICE \
          PRCP_LIQ PRCP_SOL EVAP_TOT EVAP_ICE RNF__OCE \
          O_SSTSST O_TepIce O_AlbIce OIceFrac O_IceTck O_SnwTck"

for f in ${cpl_fields}
do
    cp ${ini_data_dir}/oasis/rst/$f .
done

export DR_HOOK_IGNORE_SIGNALS='-1'
export OASIS3=yes
export OASIS3DEBUGLEVEL=2
export LOCAL_DEFINITION_TEMPLATES=${emos_gribtemplates}

ulimit -s unlimited
str_ios=""
```

```
# Compute node distribution

set +u

VARS="-x LD_LIBRARY_PATH -x DR_HOOK_IGNORE_SIGNALS -x OASIS3 -x OASIS3DEBUGLEVEL -x LOCAL_DEFINITION_TEMPLATES "

oas_numproc_orig=$oas_numproc

ifs_numproc_orig=$ifs_numproc

nem_numproc_orig=$nem_numproc

rm -rf oas_nodes ifs_nodes nem_nodes

for node in `cat ${start_dir}/nodes`

#for node in `cat ${SP_HOSTFILE}`

do

  if (( oas_numproc > 0 ))

  then

   (( n = proc_per_node<oas_numproc?proc_per_node:oas_numproc ))

   str_oas="${str_oas}${node} ${n} "

   (( oas_numproc -= n ))

      echo "${node}" >> oas_nodes

   continue

  fi

  if (( ifs_numproc > 0 ))

  then

   (( n = proc_per_node<ifs_numproc?proc_per_node:ifs_numproc ))

   str_ifs="${str_ifs}${node} ${n} "

   (( ifs_numproc -= n ))

      echo "${node}" >> ifs_nodes

   continue

  fi

  if (( nem_numproc > 0 ))

  then

   (( n = proc_per_node<nem_numproc?proc_per_node:nem_numproc ))

    str_nem="${str_nem}${node} ${n} "

    (( nem_numproc -= n ))
```

```
        echo "${node}" >> nem_nodes
      continue
    fi
done
echo "-hostfile oas_nodes -np $oas_numproc_orig $VARS oasis3.MPI1.x" > appfile
echo "-hostfile ifs_nodes -np $ifs_numproc_orig $VARS ifsmaster-${build_arch} -v ecmwf -e ${exp_name} " >> appfile
echo "-hostfile nem_nodes -np $nem_numproc_orig $VARS opa-${build_arch}-${nem_numproc_i}x${nem_numproc_j}" >> appfile

set -u
if (( oas_numproc > 0 ||  ifs_numproc > 0 || nem_numproc > 0 ))
then
  info  "Oasis (missing ${oas_numproc} procs): ${str_oas}"
  info  "IFS   (missing ${ifs_numproc} procs): ${str_ifs}"
  info  "Nemo  (missing ${nem_numproc} procs): ${str_nem}"
  error "Not enough nodes available!"
else
  info "========================"
  info "Node/proc distribution:"
  info "------------------------"
  info "Oasis: ${str_oas}"
  info "IFS:   ${str_ifs}"
  info "Nemo:  ${str_nem}"
  info "========================"
fi

## bind the procs to cpu cores ##
#for node in `cat ${SP_HOSTFILE}`
#do
#  ssh $node "sh ${start_dir}/bind.sh oasis3 ifsmaster opa" & ## run in the background on every node
#done
################################

t_start=`date +"%s"`
```

```
mpirun -app appfile

t_stop=`date +"%s"`

t_run=$((t_stop-t_start))

t_hrs=$((t_run/3600))

t_min=$(((t_run-(t_hrs*3600))/60))

t_sec=$((t_run-(t_hrs*3600)-(t_min*60)))


info "==> Finished run after ${t_hrs}hrs ${t_min}min ${t_sec}sec"
```