Documentation - Scheduled runs of the ICON code at ECMWF

Document version: Phase 0 (Q2/2012)

General Notes

Case Setup

Objectives

- run time: 30 days
- time step size: 60 sec, 30*24*3600 / 60 = 43200 steps
- output every 6h = 21600 sec

Restrictions

- Results are stored in a dedicated directory
- IFS2ICON based on Ruby scripting language
- (Experimental) GRIB2 output available for plot suite

TODO

- Write platform independent scripts (using includes and conditionals).
- (With full access to IFS MARS data:) Enable retrieval for SMSDATE in task `get_data.sms`
- increase no. of SMSTRIES for operational use
- set wall clock time limits and node numbers
- check error code of the model binary
- runs build and dumpstate computation with a larger frequency.
- ~~Show a current computation date in XCDP~~
- Use multi-threaded version of CDOs
- Use `/home/ms/de/dw7/bin/grib_def` for GRIB definition.
- perform IFS2ICON in a temporary directory (otherwise this might exceed the quota on `$PERM`).

SMS suite ICON_R2B06_30d

The ICON runs employ ECMWF's scheduler “SMS”, cf. http://www.ecmwf.int/publications/manuals/sms/.

- Model is run by a dedicated user.
- SMS files are stored in the Redmine `svn` version control system

The described scheduler setup for the ICON model is partly based on a similar script suite for BCeps (H. Frank, User `zde`), see

```
/home/ms/de/zde/BCeps/bceps.def
```

Directory layout

- The actual case setup for ICON is stored in the file `def/case_setup`.

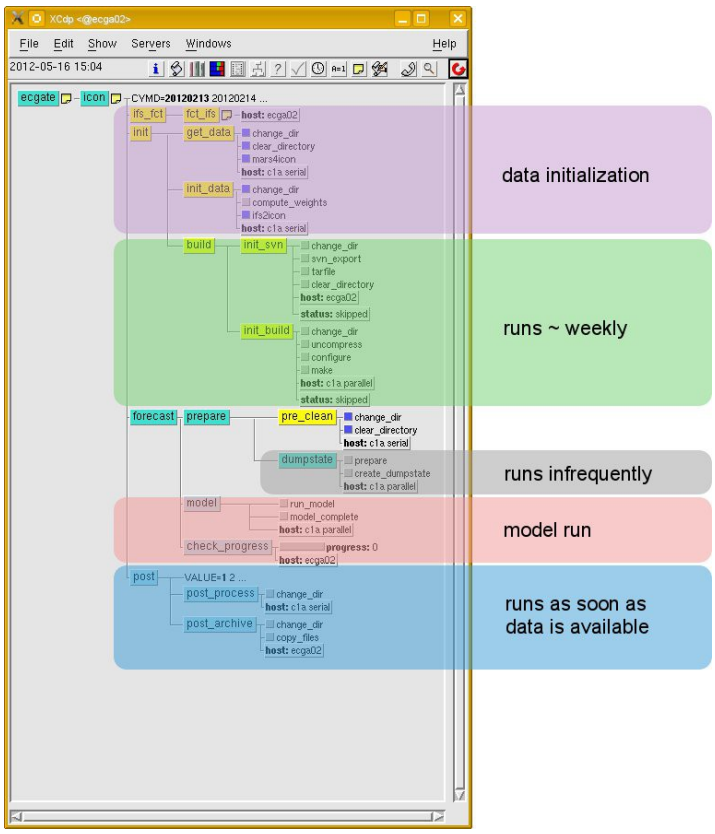
- The SMS suite is defined in `$HOME/ICON_R2B06_30d/def/smsfiles/icon.def`
- All temporary SMS files are stored in `$HOME/sms_server` and `$HOME/sms`.
- Temporary results are stored in `$TMPDIR` on ecgate and `$TEMP/sms` on the cluster file system.

All script files and the model setup are located in a directory `$HOME/ICON_R2B06_30d` of the following layout:

```
$HOME/ICON_R2B06_30d
    def                     ! scripts for operating the suite
        smsfiles
        include
        icon_scripts        ! checkout from ICON SVN: trunk/icon-dev/scripts
        bin                 ! additional script files
    output
        model               ! current model output
        log                 ! SMS suite logfiles
    doc                     ! this documentation
    cluster                 ! [only on compute cluster]
        icon-dev            ! latest build of SVN trunk [only on compute
cluster]
        input               ! static input (time-independent)
            extpar
            grids
            radiation
            ifs2icon        ! model input from IFS
               cellweights
               config
        dumpstate           ! interpolation coefficients
```

Note: An identical directory structure is necessary on the compute cluster file system,

`/perm/ms/de/dfi0`



Suite definition

Diagram of the ICON SMS suite:

- Dependencies between tasks are denoted with lines.
- Less frequently processed tasks are denoted by brackets `[]`.

```
|--> ICON_R2B06_30d
|       |get_data|     [init_svn]
|           |              |
|      |init_data|    [init_build]
|           |--------------|
|      [pre_clean]
|           |
|     [dumpstate]
|           |
|        |model|------------------|
|           |              |------------|
|           |        |post_archive|  |post_process|
|           |              |            |
|-----------|--------------|------------|
```

List of SMS tasks

task	description	depends on event
init		
get_data	retrieve IFS data from MARS	
init_data	process IFS data for ICON	`get_data==complete`
init_svn	update ICON sources	NOT `$skip_build_process`
init_build	build ICON binary	NOT `$skip_build_process` AND `init_svn==complete`
forecast		`init===complete`
pre_clean	initial clean-up	
dumpstate	create/check for dump state	
model	model run	`fct_prepare==complete`
postprocess		`fct_model:run_model`
post_archive	copy output/store in database/trigger transfer to DWD	
post_process	extract output, generate plots	

- The task `init_data` comprises the computation of cell_weights – but only if they have not been previously computed. This is indicated by the existence of a file

  `cluster/input/ifs2icon/cellweights/initialized.flag`

- The task `init_data` loads the Climate data Operators `cdo` version 1.5.0. Note that the default version 1.4.6 installed on `c1a` cannot be used with `ifs2icon` while for the newer 1.5.3 NetCDF support has not been compiled in.
- The task `init_data`(`ifs2icon` contains a call to `cdo merge` which consumes a considerable amount of memory (for R2B06, the default of ~780MB is not sufficient).

Trigger mechanism

The whole SMS suite is restarted (`repeat date` keyword) on a regular basis. The suite contains an artificial dummy task, `fct_ifs`, whose completion is the starting condition for the whole suite.

There exists a small shell script `def/smsfiles/job.trigger` which is triggered by the IFS framework itself, i.e. via

`ecaccess-job-submit`

This shell script calls a CDP command:

`force complete fct_ifs`

and thus activates the whole ICON SMS suite.

Setting up the SMS

sms_start

command

`sms_start`

gives

```
User "3455" attempting to start sms server on "ecgate" using PROGNUM = "903455"
and with SMSHOME of "/home/ms/de/dfi0/sms_server"

Checking if the SMS is already running on ecgate
...
```

Then write host and number to `.cdprc`:

`alias myalias set SMS_PROG 903455 \; login ecgate UID 1`

Running the suite

In `ICON_r2B06_30d/def/smsfiles/` call

`./start_sms.sh`

Run the suite with

`cdp`

```
 CDP> myalias
 CDP> play icon.def
 CDP> begin icon
```

Check the status of the suite with

`cdp`

```
 CDP> myalias
 CDP> status -f
      /{sub}   icon{sub}   t1[sub]
```

Restarting the suite

Add the following to your `$HOME/.cdprc` file:

```
  define icon_restart {
     cancel -y icon
     play icon.def
     begin icon
  }
```

Then restart the ICON suite by typing

`cdp`

```
 CDP> myalias
 CDP> icon_restart

 cancel:/icon by UID@220
 unknown:/
```

```
# MSG:play:Sending icon to ecgate
# MSG:play:Suite icon defined.
begin:/icon started by UID@220
```

Logserver

On the compute cluster `c1a` a special process is in charge of copying log files back to `ecgate` s.t. they can be viewed in `xcdp`:

```
cd $HOME/ICON_r2B06_30d/def/bin
./logserver
```

Now a log server daemon must be running:

```
ps -ef | grep $USER
```

yield something like

```
...  418144       1   0 10:34:29 pts/172  0:00 /usr/bin/perl
/usr/local/bin/logsvr.pl
```

Technical infrastructure

.rhosts

Enable `rcp` and `rsh` between the compute cluster and `ecgate` by setting the `$HOME/.rhosts` file:

```
echo "ecga02 $USER" >> ~/.rhosts
```

For copying files between ecgate and compute cluster we use the commands `rcp` or `ecrcp`.

SVN access

Automatic builds perform a complete `svn export`.

Please note: For direct access to ZMAW's subversion repository, changes in `$HOME/.subversion/servers` are necessary:

```
http-proxy-host = ******
http-proxy-port = ******
```

(Ask L. Kornblueh, F. Prill for details).

Local Ruby installation

Script language `ruby` has to be installed locally (user account).

Prerequisites:

1. Ruby source code downloaded from http://www.ruby-lang.org/de/downloads/ to `$PERM/software`.
2. Package “extcsv” required; download from http://rubygems.org/gems/extcsv to `$PERM/software/packages`.
3. Package “gnuplot” required; download from http://rubygems.org/gems/gnuplot to `$PERM/software/packages`.

Installation process:

```
tar xvf ruby-1.9.3-p125.tar
cd ruby-1.9.3-p125
./configure --prefix=$PERM/software/ruby-1.9.3-p125_build
gmake -j6
gmake install
cd ../packages
$PERM/software/ruby-1.9.3-p125_build/bin/gem install extcsv
$PERM/software/ruby-1.9.3-p125_build/bin/gem install gnuplot
```

Patch required: In `$PERM/software/ruby-1.9.3-p125_build/lib/ruby/gems/1.9.1/gems/extcsv-0.12.0/lib/extcsv_diagram.rb` comment out require statement:

```
#require 'extcsv_units'
```

Local CDO installation

It is necessary to compile a local version of the Climate Data Operators, which is newer than v1.5.0, because the system installation of the CDOs on `c1a` has not been compiled with NetCDF support. The configure script must be provided with the correct system paths for NetCDF, Jasper and GRIB_API:

```
ls -rlt
gunzip cdo-1.5.4.tar.gz
tar xvf cdo-1.5.4.tar
cd cdo-1.5.4
./configure -with-netcdf=/usr/local/apps/netcdf/3.6.3/LP64
--prefix=/perm/ms/de/dfi0/software/cdo-1.5.4_build --with-
grib_api=/usr/local/lib/metaps/lib/grib_api/1.9.9 --with-
jasper=/usr/local/apps/jasper/1.900.1/LP64
gmake -j2
gmake install
```

After successful compilation, all IFS2ICON processes must run with

`$PERM/software/cdo-1.5.4_build/bin/cdo`

To this end, the task `init_data.sms` exports an environment variable `CDOBIN` which is then used by the Ruby script `ifs2icon.rb`.

GRIB_API settings

We want to use the same IFS2ICON configuration settings for ECMWF as well as for DWD, therefore the GRIB2 short names must be taken from DWD's definition files:

```
export GRIB_DEFINITION_PATH=%SCPERM%/software/usr/local/grib_api/release/share-
1.9.9/definitions.edzw:/usr/local/lib/metaps/lib/grib_api/1.9.9/share/definition
s
```

The directory ”%SCPERM%/software/usr/local/grib_api/release/share-1.9.9/definitions.edzw” can be tar'ed and copied from DWD's HPC file system.

Alternative approach: DWD GRIB definitions are already installed under

`~dwd/grib_api/definitions.edzw-${my_api_version}`

One can set them using the script `/home/ms/de/dw7/bin/grib_def`.

Local settings

MARS4ICON script must be executable:

```
chmod +x ~/ICON_r2B06_30d/icon-dev/scripts/preprocessing/mars4icon_smi_32r3+
```

Date conversion script `datconv` must be in PERL's search path:

```
cd $HOME/bin
ln -s ~dfr/routfox/bin/datconv .
ln -s ~dfr/routfox $HOME/
export PERL5LIB="/home/ms/de/dfi0/routfox/perl"
```