Hands on exercises with ORCHIDEE OFF-LINE

Revised for training session 2016-11-24 - 2016-11-25

Josefine Ghattas, IPSL

The goal of this exercise is to learn how to install, compile and launch a basic test case with ORCHIDEE in off-line mode. Exercice 3-4 are using libIGCM. All exercises can be done at curie/TGCC, Ada/IDRIS or obelix/LSCE. All commands needed for the basic exercises are listed in the text.

Today at Ada: use training account

You can change the keyboard language on the training computer by touching **shift + alt** at the same time.

During the training session everybody works on temporary training accounts at IDRIS. These acounts have specific priority in the queue to avoid to much waiting time. First connect to ipcours then open a terminal and connect to ada via ssh:

```
ssh -X ada337
```

Install the IPSL environnement at your account during the first connexion to ada and add export LL_RES_ID. The LL_RES_ID is a specific reservation with computing resources available only during the training session. It can not be used later. Do the following using X different each day, see the whiteboard:

```
cd $HOME
cp ~rpsl035/.bash_login .
rm .bash_profile
vi .bash_login
=> add in the end of the file
export LL_RES_ID=ada336.X.r

source $HOME/.bash_login
```

The training accounts do not have acces to ergon which is the archive machine at IDRIS. Therefore for the last exercises with libIGCM, specify in each config.card to use a directory in the workdir at Ada by setting in the UserChoicies section:

```
ARCHIVE=$WORKDIR/ERGON
```

1 Install and compile

1.1 Install ORCHIDEE trunk for offline use

Download first **modipsl** and explore what is inside. modipsl contains some tools in the directory **util**. In util, scripts are found for : extraction (*model, mod.def*), creation of makefiles (*ins_make, AA_make.gdef*), creation of job (*ins_job*) and some more. modipsl is also a empty file tree that will receive the models and tools.

Start this exercise by extracting modipsl in a new directory :

```
cd $WORKDIR
mkdir TESTOFFLINE ; cd TESTOFFLINE
svn co http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl
cd modipsl/util
ls
```

The script **model** is used to download a specific predefined configuration with the model sources and tools needed. The script uses the file **mod.def** that contains specifications for each configuration predefined. Use **./model -h** to see all existing configurations and **./model -h** *config_name* for information about a specific configuration. Same information can be found in the file mod.def. Download a configuration using **./model** *config_name*.

For these exercises you will use the configuration ORCHIDEE_trunk which is an offline set up using the latest version of the trunk ORCHIDEE. Open mod.def and look at lines begging with ORCHIDEE_trunk and then extract as follow :

```
vi mod.def                # Explore lines beggining with ORCHIDEE_trunk
./model ORCHIDEE_trunk
```

Now explore the directories in modipsl. You will find all source code for ORCHIDEE in directory modipsl/modeles. You also find the directory IOIPSL and XIOS which are fortran and C libraries linked to ORCHIDEE for input/output issues. In directory modipsl/config/ORCHIDEE_OL you find scripts to run ORCHIDEE using libIGCM. libIGCM is a tool developed at IPSL to run coupled and off-line simulations. Specific training session about libIGCM are given by the Platforme groupe at IPSL.
Go into each dircetory and check which versions have been extracted. You're supposed to find same information as you can see in mod.def. Use svn to know version and revision number.

```
cd ../modeles/ORCHIDEE
svn info
cd ../IOIPSL/src
svn info
cd ../../../libIGCM
svn info
```

The makefile was created automatically in the end of the script model, done by the script **ins_make**. ins_make will detect at which machine you are working on and create adapted makefiles. By default ins_make recognize the following machines : curie at TGCC, ada at IDRIS, obelix at LSCE and ciclad and climserv at IPSL. ins_make can also be re-launched manually. For example this is needed if you move the modipsl directory or if you want to create makefiles for another target machine. The main makefile is found in modipsl/config/ORCHIDEE_OL directory.

Now compile the model :

```
cd ../config/ORCHIDEE_OL
gmake
```

When the compiling is finished you will find the executables orchidee_ol and xios_server.exe in modipsl/bin.

1.2 Install a branch of ORCHIDEE

Do as follow if you will like to install a specific branch instead of the trunk of ORCHIDEE. Install first modipsl as before. Then open the file mod.def and look for the secion ORCHIDEE_trunk. Change the line containing ORCHIDEE/trunk into the name for the branch you will use.
For exemple, to extract ORCHIDEE-MICT, do the following :

```
cd $WORKDIR; mkdir TESTMICT ; cd TESTMICT
svn co http://forge.ipsl.jussieu.fr/igcmg/svn/modipsl/trunk modipsl
cd modipsl/util
```

In mod.def, change the line

```
#-C- ORCHIDEE_trunk  trunk/ORCHIDEE                    HEAD 14 ORCHIDEE   modeles
```

into

```
#-C- ORCHIDEE_trunk  branches/ORCHIDEE-MICT/ORCHIDEE   HEAD 14 ORCHIDEE   modeles
```

Finaly extract as before:

```
./model ORCHIDEE_trunk
```

Note that *branches/ORCHIDEE-MICT/ORCHIDEE* can be changed to another path on the svn repository, for example *branches/ORCHIDEE-CN-CAN/ORCHIDEE* or *branches/ORCHIDEE-SOM/ORCHIDEE*. It can also be the path to a personal version.
You can also set a specific revision number by changing HEAD into a revision number.

2 Test simulations

We will now do some test simulations using the ORCHIDEE trunk offline installation. This will be done interactively to easier understand what is happening. To run the model you need at least the following files in the run directory :

- orchidee_ol : ORCHIDEE executable
- run.def : parameter text file
- forcing_file.nc : climate forcing variables (this file can have another name, in that case it should be indicated in run.def file)
- PFTmap.nc : vegetation map
- soils_param.nc : initialization of soil parameters
- iodef.xml, context_orchidee.xml, field_def_orchidee.xml, file_def_orchidee.xml : parameters files for output settings using XIOS

2.1 First regional run

Create a new directory outside modipsl to run the model and copy or link the ORCHIDEE executable :

```
cd TESTOFFLINE; mkdir RUN1; cd RUN1
ln -s ../modipsl/bin/orchidee_ol .
```

Create the parameter file by saving the following lines into a file named run.def :

```
TIME_LENGTH=31D
STOMATE_OK_STOMATE= y
LIMIT_WEST = -10.
LIMIT_EAST =  30.
LIMIT_NORTH = 70.
LIMIT_SOUTH = 30.
```

Copy xml files(iodef.xml, context_orchidee.xml, field_def_orchidee.xml, file_def_orchidee.xml) from models/ORCHIDEE/src_xml directory into the run directory.

```
cp ../modipsl/modeles/ORCHIDEE/src_xml/* .
```

file_def_orchidee.xml describes the output files, frequencies and the variable content. Change all occurences of _AUTO_ according to the comments in the beggining of the file. We suggest for these exercices to set daily output frequency. You can activate all files. Read more about the xml files in the appendix.

Copy or link the netcdf files from the shared repository IGCM into your run directory. The location of the shared repository IGCM depends on the machine but the content is synchronized between the different repositories. You can use export if your shell is bash (for tcsh shell: replace export by set):

```
# At obelix :
export R_IN=/home/orchideeshare/igcmg/IGCM/
# At Ada:
export R_IN=/workgpfs/rech/psl/rpsl035/IGCM
# At curie:
export R_IN=/ccc/work/cont003/igcmg/igcmg/IGCM
```

```
ln -s $R_IN/SRF/METEO/CRU-NCEP/v5.3.2/twodeg/cruncep_twodeg_1901.nc forcing_file.nc
ln -s $R_IN/SRF/PFTMAPS/CMIP5/PFTmap_1850to2005_AR5_LUHa.rc2/PFTmap_IPCC_1901.nc PFTmap.nc
ln -s $R_IN/SRF/soils_param.nc .
```

You can use ncdump to see what is in the netcdf files. For example :

```
ncdump -h forcing_file.nc
```

Now launch the model :

```
./orchidee_ol      # or ./orchidee_ol > out_exec
```

When the model finished correctly, following log message is found in the output text file out_orchidee_0000:

```
 END of dim2_driver
```

2.2 Relaunch in the same directory

If you want to relaunch the model in the same directory, then you need to delete restart and output files previously created by the model. Do this now and re-run the model :

```
rm driver_rest_out.nc sechiba_rest_out.nc stomate_rest_out.nc
rm sechiba_history.nc sechiba_out_2.nc
rm stomate_history.nc stomate_ipcc_history.nc
rm out_*
./orchidee_ol
```

2.3 Continue a simulation using restart files

The model writes restart files in the end of each execution period, after a TIME_LENGTH. These files contain all state variables needed to continue a simulation without loosing information. To continue the simulation you need to rename the restart files produced in previous run and activate reading of these files in run.def with parameters SECHIBA_restart_in, STOMATE_RESTART_FILEIN and RESTART_FILEIN. Save the output files sechiba_history.nc and stomate_history.nc for later analyses.

Add in run.def :

```
SECHIBA_restart_in=sechiba_rest_in.nc
STOMATE_RESTART_FILEIN=stomate_rest_in.nc
RESTART_FILEIN=driver_rest_in.nc
```

Rename restart files :

```
mv driver_rest_out.nc  driver_rest_in.nc
mv sechiba_rest_out.nc sechiba_rest_in.nc
mv stomate_rest_out.nc stomate_rest_in.nc
```

Save output :

```
mv sechiba_history.nc sechiba_history_month01.nc
mv stomate_history.nc stomate_history_month01.nc
```

Relaunch the model :

```
./orchidee_ol > out_exec
```

Note that for longer simulations **libIGCM** is used to chain the executions without manually copy or move of files. libIGCM is a powerful tool but it is important to know how the model works.

2.4 Visualization with ferret

Here is a small example to visualize sechiba_history.nc using ferret.

```
> ferret
use sechiba_history.nc # read file
sh d                        # list content in file
shade CONTFRAC              # 2D plot of a variable
go land                     # add contour of continents
shade TEMP_SOL[l=1]         # 2D plot of TEMP_SOL for first time step
shade TEMP_SOL[l=@ave]      # 2D plot of TEMP_SOL average over all time steps
shade SWDOWN[i=@ave]        # zonal plot
plot SWDOWN[i=@ave,j=@ave]  # plot mean value over time
quit
```

2.5 Change output levels

Do different small runs where you change output frequency, number of variables in the files, activate or deactivate files, change name of the variables in in the output file, etc. Do these kind of changes in file_def_orchidee.xml. Read more about xml files in the appendix.

2.6 Add a new output variable in ORCHIDEE

Create a diagnostic output variable for the variable resp_hetero_litter calculated in stomate_litter.f90 using XIOS. To do this, add in src_stomate/stomate_litter.f90:

```
! Load XIOS in the beginning of the module
use xios_orchidee
...
! Add send to XIOS in the end of the subroutine littercalc
CALL xios_orchidee_send_field("resp_hetero_litter",resp_hetero_litter)
```

Compile in modipsl/config/ORCHIDEE_OL:

```
cd config/ORCHIDEE_OL
gmake
```

Create a new run directory as before. The variable has dimension DIMENSION(npts,nvm) in the subroutine. The corresponding axis for nvm has id=nvm. Add in field_def_orchidee.xml the definition of the new variable:

```
<field id="resp_hetero_litter" name="RESP_H_LITT" long_name="..." unit="?" axis_ref="nvm"/>
```

Add in file_def_orchidee.xml in the section for the file where you want to add the variable:

```
<field field_ref="resp_hetero_litter" level="1"/>
```

Use **svn diff** in modeles/ORCHIDEE to see your modifications in the code. Launch as before. Verify that the variable is in the output file.

3 Test in parallel run mode

The default paraleliation mode for ORCHIDEE offline configuration is MPI. The model can then be run on 1 or several cores MPI. Coupled to LMDZ, the defaut parallelization mode is hybrid mode with mixt MPI and OpenMP.

For writing ouput files, since 1 year, the default is using XIOS. This makes it possible to run either in attached mode or in server mode. When running in attached mode, only the executable orchidee_ol is used. XIOS is used as an library in the model. When running in server mode, the executable xios_server.exe is launched together with the executable orchidee_ol. This is a more complex way to launch the model but it is more efficient when simulating on a big region or with high resolution.

3.1 Running ORCHIDEE with XIOS in attached mode

You will now launch a global test run on 32 MPI processes (only use 8 at obelix) in attached mode. When running in attached mode it is possible to set XIOS to write one global output file(mode one_file) or to write partial output files, one per local MPI domain(mode multiple_file). When running in multiple_mode, the output files need to be post-processed to contain the full domain. This is done using the tool **rebuild**. We will here use the one_file mode.

Prepare a new run directory as in the first exercise but do not put any regional limits in run.def (remove the LIMIT_ parameters). Open the xml files and see that attached mode is activated in iodef.xml(using_server=false) and that one_file mode is activated in file_def_orchidee.xml.

```
# in iodef.xml
<variable id="using_server" type="boolean">false</variable>

# in file_def_orchidee.xml
<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">
```

Write a file Job_orchidee as in the appendix to launch the model in the batch system instead of interactivly. The job file will be different for different machines, choose the job for ada as in the appendix.

3.2 Check reproductibility of results

When running on 1 or several processes, the results should be the same. Create a new directory and make the same test but on half of the processes. Check that restart files and output files are the same. You can use cdo to check that netcdf files are identical.

```
> cdo -diffv file1.nc file2.nc
```

Check also the difference in time due to the change of number of processors. In the ideal case, if ORCHIDEE would be perfectly scalable the job should run 4 times faster on 4 processes than on 1. This is not the case but at least you should notice a significant gain in time while increasing the number of processes.

3.3 Running ORCHIDEE with XIOS in server mode

A more efficient way to run ORCHIDEE with XIOS is to activate server mode. The ORCHIDEE executable and the XIOS server executable are launched in MPMD mode (Multiple Program Multiple Data). In the exercise before in attached mode, ORCHIDEE was launched in SPMD mode (Single Program Multiple Data).

Prepare a new run directory as the exercise before with XIOS. Add this time also the xios server executable. Change to true in iodef.xml to run in server mode:

```
ln -s ../modipsl/bin/xios_server.exe .
vi iodef.xml
  # Change to : <variable id="using_server" type="boolean">true</variable>
```

Create a new job to launch orchidee_ol executable on 31 MPI processes and xios_server.exe executable on 1 MPI process. Note that when using 2 servers(by launching xios_server.exe on 2 MPI) and option multiple_file, then each server will write a partial domaine and reconstruction is needed after the run (by using rebuild, see further below). When having 1 server it does not matter if the option one_file or multiple_file is used. The server will write to the full domaine. See job example in the appendix. Create the Job_orchidee file and the run file. Launch the job as before using llsubmit/ccc_msub/qsub depending on the machine.

3.4 Optional: How to rebuild output if running multiple_file mode

Parallel jobs will write to several text files, on per process, out_orchidee_0000, out_orchidee_0001... and the output netcdf files will be written in local domain sechiba_history_000.nc, sechiba_history_0001.nc etc. A reconstruction of the output netcdf files to the total domain is necessary. This is done with the *rebuild* tool developped at IPSL as an extension of IOIPSL.

rebuild is installed at the different machines here:

```
# at obelix:
/home/users/igcmg/rebuild/bin/
# at curie:
/ccc/cont003/home/dsm/p86ipsl/rebuild/src_X64_CURIE/modipsl_v2_2_2_netcdf4.2/bin
# at Ada:
/linkhome/rech/psl/rpsl035/bin
```

If you've installed the IPSL environnment (.bash_login at ada) you already have rebuild in your path. Try using it which rebuild.
Do the rebuild as follow :

```
rebuild -o sechiba_history.nc sechiba_history_00*
```

4 Simulations using libIGCM

The following exercises will now use the ORCHIDEE_trunk configuration with libIGCM.

There are some differences between ORCHIDEE_trunk configuration and the coupled configurations such as LMDZOR_v6 par example. In the configuration ORCHIDEE_trunk it is not needed to create the submit directory. Instead different predefined experiment directories already exist. They can be copied and used directly. These directories are OOL_SEC_STO, OOL_SEC and SPINUP_ANALYTIC. They follow the standard rules described in the training and documentation for libIGCM. The DRIVER directory do not exist but the "comp.driver" files are found in the COMP directory. The directory FORCESOIL and TESTSTOMATE also follow the same structure but these experiments are not maintained any more as the corresponding propgrams are not maintained in the source code of ORCHIDEE trunk. SPINUP and ENSEMBLE directories contain experiences that are more complicated and are not taught in the course.

We will here work with the OOL_SEC_STO and SPINUP_ANALYTIC experiments.

Specific system option during the training course

To help the post-treatment job to start running easier, you can set the time limit in the beginning of the job to a lower value for short exercices like we do here. Change in libIGCM/AA_create_ts in the section ada to have :

```
#-Q- ada # @ wall_clock_limit = 1:00:00
```

If the create_ts.job alredy exists, then remove it. It will be recreated when you run ins_job.

4.1 SPINUP_ANALYTIC experiment

You'll now set up a spinup simulation using libIGCM. Start by coping the SPINUP_ANALYTIC directory:

```
cd modipsl/config/ORCHIDEE_OL
cp -r SPINUP_ANALYTIC MyTestSpinup
cd MyTestSpinup
```

Look into the config.card. The variables CyclicBegin and CyclicEnd describe the years to loop over. Setting these 2 variables in config.card makes the variable CyclicYear available. CyclicYear is used in the orchidee_ol.card to copy the forcing file. For this exercise loop over years 1901-1910. Check CyclicBegin and CyclicEnd and modify if necessary. Limit the region to a grid-cell. Set in run.def:

```
LIMIT_WEST = -60.
LIMIT_EAST = -58.
LIMIT_NORTH = -8.
LIMIT_SOUTH = -10.
```

Set up the simulation to run over 4 forcing periodes (40 years in total). In config.card, set DateEnd=1940-12-31. Set SpaceName=TEST to deactivate pack post treatement if running at curie/ada. Set TimeSeriesFrequency=20Y and SeasonalFrequency=NONE.
It is important to adjust the numer of cores used (number of MPI and OMP) to the domain which is used. Here we run on 1 grid-cell and therefor set 1 MPI on the line for the executable to run in sequential mode. You also need to deactivate XIOS server by removing the lines for IOS in ListOfComponents and Executable. Increase PeriodNb before launching the job.

- Why should you deactivate XIOS in server mode in this example?

- How do you calculate PeriodNb?
- Which forcing file is used and where is it stored? Where is the share repository IGCM?
- Where is the output stored? How can you change the place for the ARCHIVE directory? Note that this is recommended only at obelix.
- In orchidee_ol.card you can use the variable year or CyclicYear. Which are the differences?
- The variable SPINUP_PERIOD is calculated by the stomate.driver and set in run.def. Where can you see the run.def file that was used during simulation? What is the SPINUP_PERIOD?

When the time-series have been done, you can have a look at the evolution of the carbon pools. Go to the output directory

```
cd .../IGCM_OUT/..../JobName/SBG/Analyse/TS_YE
ferret
use JobName_19010101_19401231_1Y_CARBON_ACTIVE.nc
use JobName_19010101_19401231_1Y_CARBON_SLOW.nc
use JobName_19010101_19401231_1Y_CARBON_PASSIVE.nc

set v ul; plot CARBON_ACTIVE[k=@ave,i=@ave,j=@ave,d=1]
set v ur; plot CARBON_SLOW[k=@ave,i=@ave,j=@ave,d=2]
set v ll; plot CARBON_PASSIVE[k=@ave,i=@ave,j=@ave,d=3]
```

4.2 OOL_SEC_STO experiment

Set up an experiment with sechiba and stomate using the experiment directory OOL_SEC_STO.

- Set up the simulation by period of 1 month for a total simulation lenght of 3 months.
- Activate the option for floodplains in run.def. Look in the file orchidee.default in the source directory to have the exact name for the parameter activating this option.
- Add the input file floodplains.nc. This file is only needed for the first period. The information will be stored in the restart file. Search in the shared repository IGCM (directory R_IN) to find the input file.
- Output the variable floodplains in 2 files by frequence 1 month and 1 day.

Prepare the rest of the job as usual. Launch the test and analyse the results.

5 Appendix

5.1 Description of some parameters in run.def

The file run.def contains parameters to run the model. A line beginning with a # is a comment. Default values will be used for all parameters not set in run.def. You can find the list of all parameters and their default values in modipsl/modeles/ORCHIDEE/orchidee.default .

- **TIME_LENGTH** gives the simulation lenght for each execution. In this test case TIME_LENGTH is 31 days. It is possible to run one year by putting **TIME_LENGTH=1Y**. It is not possible to run less than one day. In global or regionl simulations, we do not advice to run more than 1 year per execution but for site simulations it is recommanded to run the full forcing file length in one execution.

- **LIMIT_EAST, LIMIT_WEST, LIMIT_NORTH** and **LIMIT_SOUTH** are borders(in degrees) for the horizontal domain to be modelize. The default values correspond to the domain of the forcing file. The difference between EAST-WEST and NORTH-SOUTH must be at least one degree. The model will stop if the domain does not cover any land points with error message :

  ```
  FATAL ERROR FROM ROUTINE dim2_driver
   --> number of land points error.
   -->  is zero !
   --> stop driver
  ```

  Note that the current version of ORCHIDEE have problems to detect these kind of errors when running in parallel mode.

- **STOMATE_OK_STOMATE** parameter activates coupling to the stomate in ORCHIDEE.

- **XIOS_ORCHIDEE_OK** parameter activates use of XIOS, the default is yes. Setting XIOS_ORCHIDEE_OK=n activates use of IOIPSL for output writing instead of IOIPSL.

5.2 Description of xml files

The xml files are used to configure the output files when using XIOS(default mode). The xml files are stored in ORCHIDEE/src_xml directory. When running the model using libIGCM, the file_def_orchidee.xml is changed where it says _AUTO_. The following 4 files are needed for ORCHIDEE:

- iodef.xml : this file is the first file read by XIOS. The variable using_server specifies if the executable xios_server.exe should be launched at the same time as orchidee_ol.

  ```
  <variable id="using_server"              type="bool">false</variable>
  or
  <variable id="using_server"              type="bool">true</variable>
  ```

- context_orchidee.def : containing axis and grid information

- field_def_orchidee.xml : contains one line per output variable sent from the model. This file is only changed if new output are added in the model. A variable is output from the model with a call to subroutine xios_orchidee_send_field.

- file_def_orchidee.xml : contains specifications about the output files and contents. This is the file to be changed for all modifications in the output settings. This file is modified by orchidee_ol.driver when running with libIGCM. It is only modified where the keyword _AUTO_ is set. You can change the _AUTO_ as you wish and make other changes according to your needs, they will never be overwritten. When running without libIGCM you must change all _AUTO_.

5.3 Example of a parallel job files

5.3.1 Job using XIOS in attached mode at ada

This is a job file example for running orchidee with XIOS in attached mode at ada. Create Job_orchidee with following lines :

```
#!/bin/ksh
# @ job_name = test
# @ job_type = parallel
# @ output = Script_Output
# @ error = Script_Output
# @ total_tasks = 32
# @ wall_clock_limit = 1:00:00
# @ queue
/usr/bin/time poe ./orchidee_ol
```

Submit the job to the queue with the command llsubmit. Check its running status with the llq or qq. Do following:

```
> llsubmit Job_orchidee       # submit the job
> llq -u login                # check the job's running status
```

5.3.2 Job using XIOS in attached mode at obelix

This is a job file example for running orchidee with XIOS in attached mode at obelix. Create Job_orchidee with following lines :

```
#PBS -N test
#PBS -m a
#PBS -j oe
#PBS -o Script_Output
#PBS -S /bin/ksh
#PBS -l nodes=2:ppn=4
cd $PBS_O_WORKDIR
time mpirun ./orchidee_ol
```

Submit the job to the queue with the command qsub. Check with the qstat and use the command qcat to see the progress of the job. Do following

```
> qsub Job_orchidee       # submit the job
> qstat -u login          # check the job's running status
> qcat job_id |more       # check the progress of the job. job_id is given by qstat
```

5.3.3 Job using XIOS in attached mode at curie

This is a job file example for running orchidee with XIOS in attached mode at curie. Create Job_orchidee with following lines :

```
#!/bin/ksh
#MSUB -r test              # name of the job
#MSUB -o Script_Output     # name of output file for standard messages
#MSUB -e Script_Output     # name of output file for error messages
#MSUB -eo
#MSUB -n 32                # Request numbre of cores
#MSUB -T 1800              # Time limit in seconds
#MSUB -Q test              # Queue test, priority acces
#MSUB -q standard
#MSUB -A genXXX            # Set your project id
BATCH_NUM_PROC_TOT=$BRIDGE_MSUB_NPROC
set +x
cd ${BRIDGE_MSUB_PWD}
/usr/bin/time ccc_mprun -n 32 ./orchidee_ol
```

Submit the job to the queue with the command ccc_msub. Check with the ccc_mstat. Do following

```
> ccc_msub Job_orchidee       # submit the job
> ccc_mstat -u login          # check the job's running status
```

5.3.4 Job using XIOS in server mode at ada

Create Job_orchidee_server with following lines(the headings are the same as before) :

```
#!/bin/ksh
# @ job_name = test
# @ job_type = parallel
# @ output = Script_Output
# @ error = Script_Output
# @ total_tasks = 32
# @ wall_clock_limit = 1:00:00
# @ class = cours
# @ queue
/usr/bin/time poe -pgmmodel mpmd -cmdfile ./run_file
```

Create also the run_file with the following lines (31 lines with orchidee_ol):

```
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
```

```
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./orchidee_ol
./xios_server.exe
```

Now launch the job:

```
llsubmit Job_orchidee
```