How to configure output files in ORCHIDEE and some about XIOS

Presentation by Josefine Ghattas, IPSL
ORCHIDEE Training course 16-17/01/2020

Files created by ORCHIDEE

Restart files

- Containing all state variables in ORCHIDEE at the last time step of the execution
- These files are needed as input to start next iteration
- driver_rest_out.nc, sechiba_rest_out.nc, stomate_rest_out.nc
- Read and written by IOIPSL

Diagnostic output files

- Optional files containing variables from ORCHIDEE
- One file per frequency, different operations possible
- As many files as wanted, as many variables as wanted
- For example sechiba_history.nc, stomate_history.nc,...
- Produced by XIOS or by IOIPSL(not maintained)

Output diagnostic files with XIOS

Thanks to
- Yann Meurdesoif, LSCE, the main developer of XIOS
- Arnaud Cauel, LSCE who did the first implementation of XIOS in ORCHIDEE

XIOS is a tool developped for the IPSL modèles to optain better performances and more flexible mangagement of output files. In a near future, XIOS will also handle reading of input files in ORCHIDEE.

In ORCHIDEE:

- src_parallel/**xios_orchidee.f90** : all interfacing to XIOS
- **src_xml** : directory with all xml files for running with XIOS
- Flag **XIOS_ORCHIDEE_OK** to activate running with XIOS
- Preprocessing key **XIOS** to enabel linking to XIOS

Read and interpolate files with XIOS

XIOS can also be used to **read and interpolate** input netcdf files in ORCHIDEE. This is necessary when using a non regular grid such as when coupling to DYNAMICO (core dynamic atomspheric on icosahedral grid).

In ORCHIDEE:

- Activate by setting **XIOS_INTERPOLATION=y** in run.def
- src_xml/context_input_orchidee.xml specification file
- Still under test

Read and write with IOIPSL

IOIPSL is a fortran library developped at IPSL, used since long time in ORCHIDEE to

- **read input files** (call flinopen, call flinget)
- **read and write restart files** (call restget, restput)
- **read parameter files run.def** (call getin)
- **write output files** with diagnostic variables (call histdef, call histwrite)

(*) The default method for writing output files is now changed to use XIOS. The method using IOIPSL is still left in ORCHIDEE but new variables are not added. This possibility will be removed in coming version (date not yet decided).

(**) The method for reading and interpolating input files will be change to use XIOS in a near future.

xios_orchidee_send_field

Example from thermosoil_main:

```
USE xios_orchidee

REAL(r_std),DIMENSION (kjpindex)         :: soilflx
REAL(r_std),DIMENSION (kjpindex)         :: surfheat_incr
REAL(r_std),DIMENSION (kjpindex, ngrnd) :: ptn
...

CALL xios_orchidee_send_field("ptn",ptn)
CALL xios_orchidee_send_field("Qg",soilflx)
CALL xios_orchidee_send_field("DelSurfHeat",surfheat_incr)
```

Syntax: **CALL xios_orchidee_send_field(field_id, field)**

field_id: a unique identifier, the same id is set in the field definition in parameter file field_def_orchidee.xml which must be present at run time
CHARACTER(len=*)

field: the variable to send to XIOS. The variable is on landpoint grid, it can have 1 or 2 supplementary axis:
REAL(r_std), DIMENSION(kjpindex) or
REAL(r_std), DIMENSION(kjpindex,:)

Convention in ORCHIDEE : use the same name for the id as the variable name

xml parameter files

To run ORCHIDEE with XIOS all diagnostic output files are configured thourgh xml files. Follwing 5 files needs to be present at each execution :

- **iodef.xml** => Main input file for XIOS. This file includes the 2 context files below
- **context_orchidee.xml** => Grid and axis information, include field and file def
- **context_input_orchidee.xml** => Specify all reading of input files. Reading with XIOS is optional but this file is needed for all cases
- **field_def_orchidee.xml** **=> Definition for each variable send in ORCHIDEE**
  => Only change if added new variable in ORCHIDEE
- **file_def_orchidee.xml** **=> Specify all output files and there variables**
  => Change to set your output level
  => Remove variables, change levels, change freq.

The above xml file are stored in ORCHIDEE/src_xml directory.
The file context_input_orchidee.xml was introduced in the ORCHIDEE trunk in revision 5565. It was not included in tag ORCHIDEE_2_0 but exists in tag ORCHIDEE_2_1.

file_def_orchidee.xml

```
<!-- ============================================================================================== -->
<!-- file_def_orchidee.xml : Definition of output files                                             -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Sechiba file 2 -->
  <file id="sechiba2" name="sechiba_out_2" output_level="2" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="mrsos" level="1"/>
    <field field_ref="mrro" level="2"/>
    ...
  </file>

  <!-- Stomate file 1 -->
  <file id="stomate1" name="stomate_history" output_level="10" output_freq="86400s">
    <field field_ref="RESOLUTION_X" level="1"/>
    <field field_ref="RESOLUTION_Y" level="1"/>
    <field field_ref="CONTFRAC_STOMATE" level="1"/>
  </file>
</file_definition>
```

file_def_orchidee.xml

```
<!-- ============================================================================================== -->
<!-- file_def_orchidee.xml : Definition of output files                                             -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_r
    <field field_r
    <field field_r
    <field field_r
    <field field_r
    ...
  </file>

  <!-- Sechiba file 2 -->
  <file id="sechiba2" name="sechiba_out_2" output_level="2" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="mrsos" level="1"/>
    <field field_ref="mrro" level="2"/>
    ...
  </file>

  <!-- Stomate file 1 -->
  <file id="stomate1" name="stomate_history" output_level="10" output_freq="86400s">
    <field field_ref="RESOLUTION_X" level="1"/>
    <field field_ref="RESOLUTION_Y" level="1"/>
    <field field_ref="CONTFRAC_STOMATE" level="1"/>
  </file>
</file_definition>
```

Information about all files written by ORCHIDEE

type “one_file” or “multiple_file” : XIOS will gather information from all processes on a single output file or not

file_def_orchidee.xml

```
<!-- ================================================================================================== -->
<!-- file_def_orchidee.xml : Definition of output files                                                 -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Sechiba file 2 -->
  <file id="sechiba2" name="sechiba_out_2" output_level="2" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="mrsos" level="1"/>
    <field field_ref="mrro" level="2"/>
    ...
  </file>

  <!-- Stomate file 1 -->
  <file id="stomate1" name="stomate_history" output_level="10" output_freq="86400s">
    <field field_ref="RESOLUTION_X" level="1"/>
    <field field_ref="RESOLUTION_Y" level="1"/>
    <field field_ref="CONTFRAC_STOMATE" level="1"/>
  </file>
</file_definition>
```

file_def_orchidee.xml

```
<!-- ================================================================================================== -->
<!-- file_def_orchidee.xml : Definition of output files                                                 -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Sechi
  <file id="
    <field f
    <field f
    <field f
    <field f
    <field f
    ...
  </file>

  <!-- Stoma
  <file id="
    <field f
    <field f
    <field f
  </file>
</file_defin
```

Information line about one file

name	filename, suffix .nc will be added to the filename
output_level	**“x”** : all variables listed below with level less or equal to x will be written
output_freq	”1d”, “1800s”, “1ts”, “1mo”, “3h”, “1y” : frequency for the file
enabled	“.TRUE.” / “.FALSE.” : create the file, true is default

file_def_orchidee.xml

```
<!-- ============================================================================================== -->
<!-- file_def_orchidee.xml : Definition of output files                                             -->

<file_definition type="one_file" par_access="collective" enabled=".TRUE." min_digits="4">

  <!-- Sechiba file 1 -->
  <file id="sechiba1" name="sechiba_history" output_level="11" output_freq="1d" enabled=".TRUE.">
    <field field_ref="Areas" level="1"/>
    <field field_ref="LandPoints" level="1"/>
    <field field_ref="Contfrac" level="1"/>
    <field field_ref="evap" level="1"/>
    <field field_ref="coastalflow" level="1"/>
    <field field_ref="riverflow" level="2"/>
    <field field_ref="temp_sol_C" level="2"/>
    ...
  </file>

  <!-- Sechiba
  <file id="se
    <field fie
    <field fie
    <field fie
    <field fie
    <field fie
    ...
  </file>

  <!-- Stomate
  <file id="st
    <field fie
    <field fie
    <field fie
  </file>
</file_definit
```

A line per variable added in the file

field_ref	reference id as set in field_def_orchidee.xml file
level	**“x”** : the variable is only written if this level is less or equal of output_level set at the file description line above.
name / long_name	”new_name” : name of the variable in the output file. If it is not set, the name set in field_def_orchidee.xml will be used.
enabled	“.TRUE.” / “.FALSE.” : write the variable, true is the default.
operation	can be added, overwrites settings in field_def “average”, “min”, “max”, “instant”

Add a new variable in ORCHIDEE

1) Add in the ORCHIDEE module where the variable is calculated:

CALL xios_orchidee_send_field("new_var",new_var)

2) In field_def_orchidee.xml : add declaration of the variable

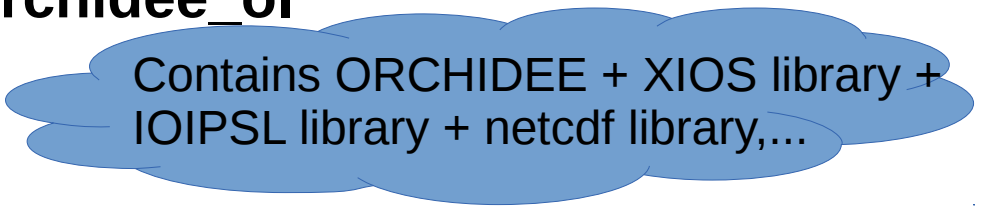
3) In file_def_orchidee.xml : add the variable in all file sections where you want to write it

*) If the variable is only calculated for a specific option, add an exception in xios_orchidee_init. This avoid that the variable will be initialized in the output file without being written if using the same .xml files.

Running ORCHIDEE with XIOS

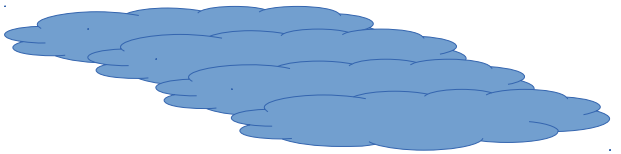
After compilation 2 executables are found in modipsl/bin

orchidee_ol

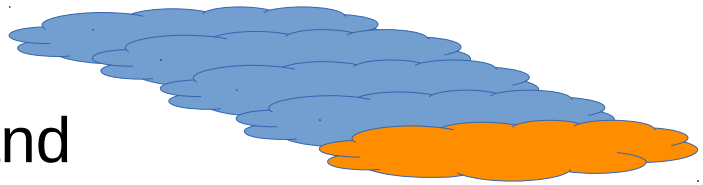


xios_server.exe

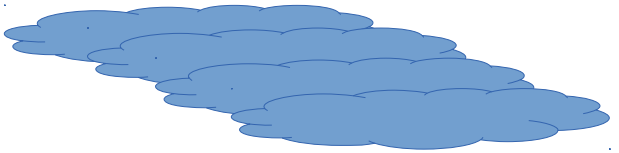
Launch only orchidee_ol on one or several core, this is called to use
XIOS in attached mode



Launch orchidee_ol on one or several core and the executable XIOS, this is called to use
XIOS in server mode



Running in attached mode



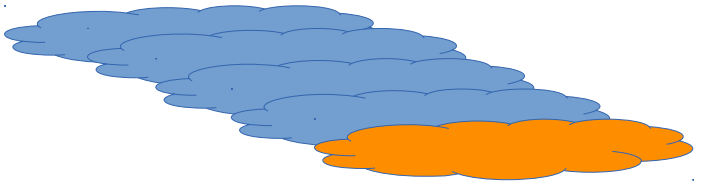
Requirements for running ORCHIDEE with XIOS in attached mode:

- 1 executable: orchidee_ol
- 5 xml files : iodef.xml, context_orchidee.xml,
  field_def_orchidee.xml,
  file_def_orchidee.xml, file_def_input_orchidee.xml
- Parameter file: run.def
- Input netcdf files: forcing_file.nc, PFTmap.nc, ...

It it possible to run on 1 or several core (processor)

Note: xml files are stored in ORCHIDEE/src_xml

Running with server



Requirements for running ORCHIDEE with XIOS using server:

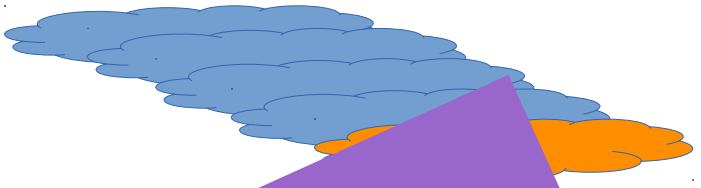
- **2 executables:** **orchidee_ol** and **xios_server.exe**
- And all other input files as before

Note : the 2 executables must be launched together in Multiple Program Multiple Data (MPMD) mode. MPI is used for the communications between the executables.

For example:
For ORCHIDEE in offline mode, 2-degree global resolution, we advice
- 31 core(processors) for orchidee_ol at irene or 19 core at jean-zay
- 1 core for xios_server.exe

Use at least 2 core for orchidee_ol and 1 for xios_server.exe

Running with server



Requirements for running ORCHIDEE with

- **2 executables : orchidee_ol**
- And all other input files

Note : the 2 ex
Program
com

F
ad
- 31
- 1 c

Use at

- More efficient => we want this method
- Not so easy to write the run script
- Different on each platform

=> USE LIBIGCM

on, we

or 19 core at jean-zay

dee_ol and 1 for xios_server.exe

Using libIGCM configurations
server mode by default

Default mode is using 1 server XIOS in libIGCM configurations (ORCHIDEE_trunk, LMDZOR_v6, IPSLCM6)

config.card

- Component IOS represents XIOS
- Set number of cores MPI for each executable with 1MPI for the xios server.

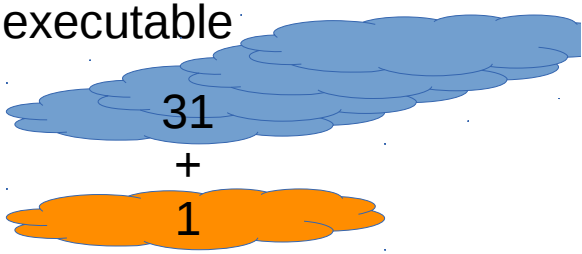
```
#============================================================
#D-- ListOfComponents -
[ListOfComponents]
#D- For each component, Name of component, Tag of component
SRF= (sechiba, orchidee_trunk)
SBG= (stomate, orchidee_trunk)
OOL= (orchidee_ol, orchidee_trunk)
IOS= (xios, XIOS)

#D-- Executable -
[Executable]
#D- For each component, Real name of executable
SRF= ("", "")
SBG= ("", "")
OOL= (orchidee_ol, orchidee_ol, 31MPI)
IOS= (xios_server.exe, xios.x, 1MPI)

...

#D-- IOS -
[IOS]
WriteFrequency=""
Restart= n
RestartDate=
RestartJobName=
RestartPath=
```

Number of core MPI per executable



Using libIGCM configurations
control of output

- All xml files except iodef.xml are copied from source directory ORCHIDEE/src_xml

- **Use output_level and output_freq variables in sechiba.card/orchidee.card and stomate.card**

  or

- **Change directly in ORCHIDEE/src_xml/file_def_orchidee.xml**
  Keyword AUTO is changed by libIGCM but can be changed manually as well.

Questions?

Learn better by do it yourself during the hands on session!