XIOS - Practical session

The aim of such a practical session is to make you discover some of XIOS2 functionalities by using test cases. Parts 1, 2, 3 and 4 are independant : sources and xml files needed for each part are in separate directories.

One advice for this practical session : use the online user's guide :
http://forge.ipsl.jussieu.fr/ioserver/attachment/wiki/WikiStart/XIOS_user_guide.pdf

0) Installation of XIOS and practical test cases

Connexion on the computing machine.

Extraction of XIOS in working directory. XIOS is handled through IPSL forge subversion server.

```
mkdir WORK_XIOS ; cd WORK_XIOS  ;
svn co http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/trunk XIOS ;
cd XIOS ;
```

Compilation of XIOS (--help to have available options, --avail to have available architectures).

```
# Available compilation options/mode
./make_xios --help (or ./make_xios -h) ;
# Available architectures/machines
 ./make_xios –avail ;
# What does an arch configuration file look  like ?
cat arch/arch_X64_CURIE.*
# Compilation (for example Curie)
./make_xios --prod --arch X64_CURIE --job 4
# Check that libxios.a and executable file exists.
ls bin/xios_server.exe lib/libxios.a
```

Download and install the following tar file http://dods.extra.cea.fr/work/p86caub/XIOS_TAR.tar on your computing machine. Untar the file and tar subfiles to obtain 4 sub-directories (i.e 4 practical session parts) : **TP_intro, TP_comp, TP_perfs, TP_grilles**. Each sub_directory is independnat and contains a simple *Makefile* that you have to adapt regarding your computing machine (especially compiler name and environment variables XIOS_DIR, NETCDF_INC_DIR, NETCDF_LIB_DIR).

1) XIOS – Writing/reading of data

Go into TP_intro directory. The main program TP_intro.f90 is the XIOS library test case that you will use during this practical session : this program is equivalent to your model you are used to run. Edit the program and have a look on different parts. Then, compile the program.

```
cd TP_intro
# Edit the program TP_intro.f90 (emacs/vi/gedit)
vi TP_intro.f90
gmake
```

a. Include file

« iodef.xml » is the main configuration file of XIOS.

```
# Edit the xml file (emacs/vi/nedit)
vi iodef.xml
```

In « iodef.xml » file, activate « atmosphere » context by uncommenting the associated line.

The « src » functionality is very useful to split xml files into many files depending on context definition, domain definition, field definition,…

In this example, « src="./context_atmosphere.xml" » means that information related to atmosphere context are specified in « context_atmosphere.xml » file. Edit the file « context_atmosphere.xml » and have a look on its contents :

- Parts related to « field », « file », « domain » and « grid ».
- Attributes of « field » and « file ».

```
# Run the program (by default, the test runs on 8 computing cores in attached mode for XIOS)
llsubmit Job
```

b. Mode multiple/one file

Have a look on what you obtain : number of files, number of records into files, …

```
ls output_atmosphere_2D* ; ncdump –h output_atmosphere_2D_0.nc
```

How many « output_atmosphere_2D* » files do you have ? Switch to « one_file » mode by modifying « context_atmosphere .xml » file and relaunch the Job. How many files do you obtain ? Check that the field is correct: you are able to visualize one of output fields (for example, « tsol » variable) via the use of “ferret” :

```
ferret
yes?use output_atmosphere_2D.nc
yes?shade TSOL[l=1]
```

Beware, “one_file” functionality works only with **parallel** NetCDF/HDF5 library.

c. Creation of a new grid

Variables « tsol », « flat » et « slp » (written in « output_atmosphere_2D.nc » file) are 2D fields defined on « grid_atm_2D » grid which is defined in « context_atmosphere.xml » file and componed of 2D domain « domain_atm ». Now, we want to write out 3D fields on a grid componed of 2D domain « domain_atm » (already existing) and vertical axis « axis_atm ».

To do that, you have to :

- create this new vertical axis « axis_atm » in « context_atmosphere.xml » file in « axis_definition » section.
- add (by uncommenting) definition of the axis in the main program test_TP.f90.
- create a new grid based on « domain_atm » domain and « axis_atm » vertical axis in « context_atmosphere.xml » file. The name of this new grid will be: «grid_atm_3D »

d. Use of a new grid

We want now to write out new fields « temp » and « rhum » on the new grid. To do that, you have to :

- define fields in xml file. Information about these new fields are :
  - id="temp" name="temp" long_name="Air temperature" unit="K" grid_ref="grid_atm_3D"
  - id="rhum" name="rhum" long_name="Relative humidity" grid_ref="grid_atm_3D"
- activate the sending of these fields in the model test_TP.f90
- activate the writing of these fields in a file defined as follows :
  - id="output_atmosphere_3D" name="output_atmosphere_3D"

```
# Compile
gmake
# Run the program
llsubmit Job
```

Check with « ferret » or « ncdump » that new fields have been written out successfully in « output_atmosphere_3D.nc » file.

e. Temporal operations

By default, fields which are written are “daily average” values. Modify the xml file (especially « operation » attribute and « output_freq » attribute)to write out field values as follows :

- hourly averaged
- timestep instantaneous
- daily maximum

Check the number of record in the output files, for example :

```
ncdump –h output_atmosphere_2D.nc
```

f. Arithmetic operations

Use arithmetic operations to write out temperature fields in degC (instead of default degK). To do that, you have to add in « *output_atmosphere_3D.nc »* file a new field with attributes inheritated from field « temp » and following specifications :

- name="tempC" long_name="Air temperature en degC" unit="degC"

Check output files by plotting « tempC » variable (by using « ferret »).

Have a look on the User’s Guide, part « How to use arithmetic operations ».

g. How to chain temporal operations

Use temporal operations to write out (in « *output_atmosphere_3D.nc* » file) the average of daily maximum over 5 days of « temp » field. This new field inherites « temp » field attributes with following specifications :

- field_ref="temp" name="temp_ave_max"

To do that, use an intermediate field defined as follows :

- field id="temp_max" field_ref="temp" operation="maximum" grid_ref="grid_atm_3D"

Check output files by plotting « temp_ave_max » variable (by using « ferret »).

Have a look on the User's Guide, part « How to chain multiple temporal operations ».

h. Times series

Write out fields « flat », « tsol » et « slp » in "time series" files (i.e a file which contains a single variable over a period). These fields will have following specifications:

- both in « Time Series » files AND in classic output files « output_atmosphere_2D.nc »
- name of « time series » files starts with « TS »
- 1D split for « time series » files

To do that, you have to use following file attributes :

- timeseries="both"
- ts_prefix="TS"
- ts_split_freq="1d"

and following field attribute :

- ts_enabled=".true."

Available options for « timeseries » file attribute are :

- none : no TS file.
- both : TS file for fields with attribute « ts_enabled=.true. » and output of all of fields (ts_enabled = .true. et ts_enabled = .false.) in classic output file.
- only : only TS file for fields with attribute «ts_enabled=".true." ».
- exclusive : TS file for fields with attribute «ts_enabled=".true." » and output of fields without attribute «ts_enabled=".true." » in classic output file.

A file « xios_registry.bin » allows to handle timesseries in case of end/start of a run. This file has to be considered as a restart file (output at the end of a run, input at the beginning of the following one). It is created during the run and allows XIOS to know the name of "timeseries" files to complete during the run.

In order to have « times series » append in the same file over the simulation, you have to activate « append » mode by using file attribute « append= true ».

i. Reading of the data

Reading from a file is a new XIOS2 functionality. Here, you have to add in temporal loop the reading of the variable « tsol » from the TS file « TS_sol.nc », as written previously (first copy the file « TS_sol.nc » into « TS_sol_saved.nc » in order to use it). The XIOS API function to call in the model is « xios_recv_field ». Write the variable that is read to a new file « TS_sol_read.nc ». With "ncdiff" command, compare the output file with the initial file « TS_sol_saved.nc ».

```
TP_intro.f90 : CALL xios_recv_field("tsol_recv",src_field_recv_tsol(:,:))
<file id="tsol_recv" name="TS_tsol_saved" output_freq="1ts" mode="read" enabled=".true.">
    <field id="tsol_recv" freq_offset="1ts" name="tsol" grid_ref="grid_atm_2D" operation="instant" enabled=".true." />
  </file>
```

By default, the first reading is done at timestep 0 : to start the reading at timestep 1 (1st index of temporal loop) , you have to use field attribute «freq_offset="1ts" ».

2) Compression

cd ~/TP_comp
 # Edit TP_comp.f90 (emacs/vi/gedit)
vi TP_comp.f90
gmake

a. HDF5 compression

« gzip » compression is available in HDF5 and allows to reduce the size of output files. Field attribute « compression_level » allows to activate this compression in XIOS. Compression level must be between 0 and 9 (0 = no compression). Here, you have to use this attribute to write out your file « output_atmosphere_2D.nc » in compressed mode of level 2. Compare the size of output files with and without compression. You can increase the output frequency of the outputs to see a greater impact of the compression.

HDF5 compression is not available in parallel mode : that means you have to run in « multiple_file » mode or in "server" mode with only one server.

b. Compressed grids

« Compression with Gathering »(Section 8.2. Compression by Gathering, CF convention)functionality allows to store in NetCDF file only non masked values by removing variables missing points (missing values). The use of this functionality is possible with XIOS2 and allows to :

- reduce the size of the file by removing masked points
- write out several grids points in one file (stations,…)

Use the array « src_mask » (in TP_comp.f90) to define a mask related to the domain « domain_atm » : you have to use the function « xios_set_domain_attr » and the argument « mask_1D ». Write out the variable “tsol” (using this compression) in a new output file « output_atmosphere_2D_compressed.nc » : to do that, you have to use the field attribute « indexed_output=".true."» for the field « tsol » in xml file.

gmake ; llsubmit Job

Have a look on the output file, especially on the NetCDF attribute « compress = "lat lon" ».

3) Performances - Memory

Now, let’s have a look on writing performances of XIOS library.
In order to have as significative as possible results, we recommend you to perform runs on tmp/scratch filesystems, i.e fastest filesystems of your computing machine (see variable RUNDIR in the Job).

cd ~/TP_perfs ; gmake

1) By default, the **« attached »** (no server) mode and **« multiple_file »** mode are activated : writing are performed in synchroneous mode by 8 client process. Every client process writes to its own file in sequential mode.

llsubmit Job

Have a look on reports XIOS writes to « xios_client_0.out » file and elapsed time of the run.

2) Switch to the **« one_file »** mode (still in « **attached** » mode) as file attribute of all of files: writings are performed in synchroneous mode by 8 clients process. All of client process write to the same file in parallel mode.

llsubmit Job

Have a look on reports XIOS writes to « xios_client_0.out » file and elapsed time of the run. What are the differences compared to the “multiple_file” mode ?

3) Continue with **« one_file »** mode AND switch to **server** mode. To do that, you have to launch executable file of the model AND executable file of the server (mode MPMD) : uncomment lines in the Job, adjust the number of cores for the server (NCPUS_SERVER variable) and the number of cores in the header of the job. Writings are performed in asynchroneous mode by the server.

llsubmit Job

Have a look on reports on time and ratios XIOS writes to « xios_client_0.out » and « xios_server_0.out » files and elapsed time of the run. What are the differences compared to the "attached" mode ?

4) Continue with **« one_file »** mode AND switch to « **server** » mode on **2, 4, 6, 8,…servers**. To do that you have to adjust NCPUS_SERVER variable in the Job file and the number of cores in the header of the Job. Writings are performed in asynchroneous mode by servers : all of servers write to the same file in parallel mode.

llsubmit Job

Have a look on reports on time and ratios XIOS writes to « xios_client_0.out » and « xios_server_0.out » files and elapsed time of the run. What is the optimal configuration in terms of numbers of servers ?

5) By default, the mode **« performance »** is activated. Switch to the **« memory »** mode by modifying iodef.xml, relaunch the Job and have a look on reports in "server" mode. What are the differences in terms of buffer memory between "memory" and "performance" mode ? What are the differences in terms of time between "memory" and "performance" mode ?

« memory » : memory optimization : buffers are able to treat one field at once.
« performance » : performance optimization (reduce elapsed time of the run) : buffers are able to treat all of fields at a writing timestep.

4) XIOS - Grids - Transformations

Let's have a look on grid transformations.

cd ~/TP_transf ; gmake

a. Zooms

Here, we want to write out « tsol » variable on a part of global grid « grid_atm_2D ». This part is a square of 10 points over 10 points, starting at first indexes i and j. The variable is written out in a new file « output_atmosphere_2D_zoom.nc ». To do that, you have to create a new grid « grid_atm_2D_zoom », composed of a new domain « domain_atm_zoom » : this new domain refers to global domain "domain_atm" and a transformation "zoom_domain" has to be defined as follows :

```
<grid id="grid_atm_2D_zoom">
  <domain domain_ref="domain_atm" >
    <zoom_domain id="domain_atm_zoom" ibegin="0" jbegin="0" ni="10" nj="10" />
  </domain>
</grid>
```

Indexes of arrays used by XIOS start from 0.

Zoom functionality may be useful to write out in a file the value on one specific point of the global grid.

llsubmit Job

Have a look on the variable « tsol » in the file « output_atmosphere_2D_zoom.nc ».

b. How to generate grids - Interpolations

Interpolation is a new XIOS2 functionality. With no change in the code, we want to :

- write out « tsol » variable in a file « output_atmosphere_2D.nc » on the native grid.
- write out « tsol » variable on a **360 x 180 resolution regular grid** in a file « output_atmosphere_2D_HR.nc ». To do that, you have to define a new grid « grid_atm_2D_HR » composed of « domain_atm_HR » domain which is the result of interpolation of « domain_atm » (native domain) on the « domain_atm_HR » (new HR domain). Two steps/filters are needed to define this interpolation :
    - generate the new HR regular grid
    - interpolation on the new HR grid

  These two filters are defined as follows :

```
<domain id="domain_atm_HR" ni_glo="360" nj_glo="180" type="rectilinear">
    <generate_rectilinear_domain />
    <interpolate_domain/>
 </domain>

<grid id="grid_atm_2D_HR">
    <domain domain_ref="domain_atm_HR" />
</grid>
```

Then, you have to define a new file with the new field defined on the new grid :

```
<field field_ref="tsol" grid_ref="grid_atm_2D_HR" />
```

llsubmit Job

Have a look (visualization with ferret) on « tsol » variable on the two different grids (native grid and regular HR grid). Try again defining a new grid with new resolution and visualize the result.