XIOS training

How to extract XIOS:

```
svn checkout http://forge.ipsl.jussieu.fr/ioserver/svn/XIOS/trunk
```

How to compile XIOS:

./make_xios

Option	Value	Default	Description
- - arch	arch_name		Mandatory. Define target architecture
- - avail			Show available target architectures
- - prod			Compilation in production mode (default)
- - debug			Compilation in debug mode
- - full			Generate dependencies and recompile from scratch
- - build_dir	build_directory		Name of the build directory
- - job	ntasks	1	To use parallel compilation with ntasks processus
- - netcdf_lib	netcdf_par netcdf_seq netcdf_internal	netcdf_par	Choice of netcdf library
- - help			Show all available options and descriptions

You can create your own architecture files by adapting :

arch.fcm	Mandatory. Define compile options.
arch.env	Define environment variables, load modules.
arch.path	Define paths to libraries.

You should see "Build command finished …" to confirme a successful compilation.

How to include XIOS in your fortran program

```
SUBROUTINE Hello_world

  USE XIOS
  TYPE(xios_duration) :: dtime
  INTEGER :: comm

  CALL MPI_Init()

  CALL xios_initialize("client", return_comm=comm)
  CALL xios_context_initialize("Hello_world", comm)

  CALL xios_define_calendar(type="Gregorian")
  dtime%second=3600
  CALL xios_set_timestep(dtime)

  CALL xios_close_context_definition()

  CALL xios_context_finalize()
  CALL xios_finalize()

  CALL MPI_Finalize()

END SUBROUTINE Hello_world
```

How to link XIOS to your program

export XIOS_DIR=path_to_yout_build_dir
-I$(XIOS_DIR)/inc
-L$(XIOS)/lib -lxios

Hands-on 0 (optional)

Compile XIOS

1	cd XIOS_TRAINING
2	XIOS_TO_COMPILE contains one copy of the XIOS trunk source
3	bash ./compile_xios.sh **OR** cd XIOS_TO_COMPILE ; ./make_xios …
4	Define the proper compile options if you use ./make_xios (arch, debug, build_dir, …)

Before doing the hands-on exercises :

Please go to the root fold of the training **"XIOS_TRAINING"**.
Type **"source ./hands-on.env"** to initialize the correct computing environment.
In each hands-on folder, you should firstly run the command **"./init-hands-on.sh"** to initiate the codes. You can re-run this command at any time you want to restart the exercise.
To compile the program, you can use **"make"** command.
To launch the executable, you can use **"./run_ubuntu"** which will use 4 MPI processes.
Step by step solutions can be found in the "answer" folder.

Hands-on 1

We will begin with a "Hello XIOS" fortran program and make it XIOS-compliant.

The "iodef.xml" is a mandatory input file for XIOS. Do not modify it for the moment. We will work on it later.

1	Initialize XIOS with id "client" Assign the return communicator to an integer "comm"	xios_initialize return_comm = comm
2	Initialize a context "test"	xios_context_initialize
3	Define the calendar (mandatory)	xios_define_calendar(type="Gregorian")
4	Set the time step to 1 hour	dtime%second=3600 xios_set_timestep
5	Close the context definition	xios_close_context_definition
6	Close the context	xios_context_finalize
7	Free the XIOS communicator and end XIOS	xios_finalize
8	Run the program and you should have several XIOS report files	xios_client_*.out xios_client_*.err

Hands-on 2

Define a well-formed configuration file (iodef.xml) in XML step by step, for the following case :

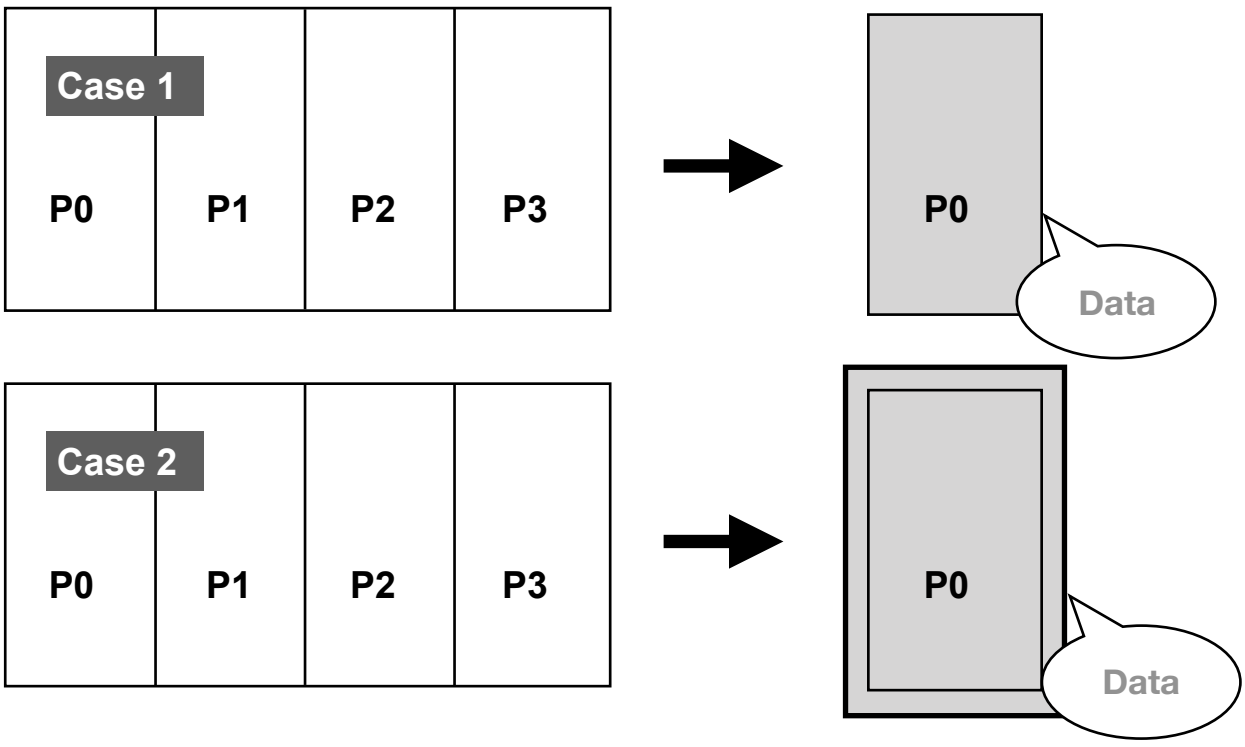
- Gregorian calendar (set you favorite date as the start date)
- Rectilinear discretization composed of an 2D domain and a vertical axis
- We want to output a 3D field after each time step
- We want to have a single output file

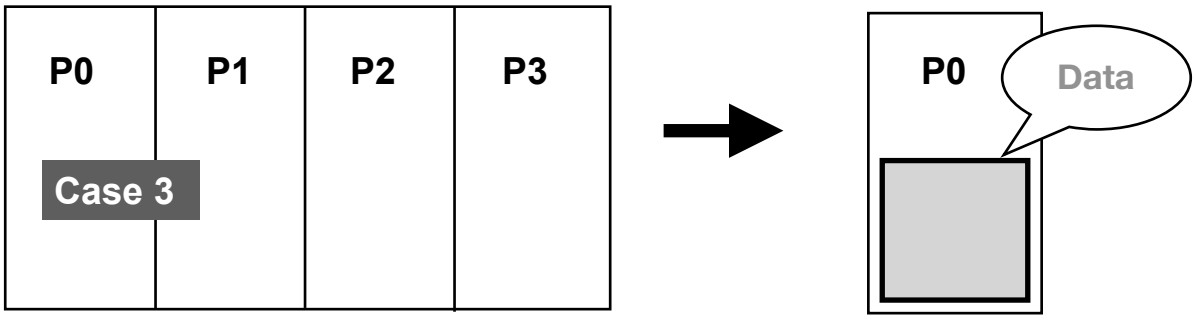
1	- Define a context element "test" - Instead of defining the calendar in fortran interface, define the calendar element in xml - Print out the defined value of "start_date" and "time_origin"	id type, start_date, time_origin xios_date_convert_to_string
2	- Define an axis element "axis_A" of size 10 - Define the coordinate value [100 200 … 1000] of the axis in xml and print out the value from fortran	id, n_glo, value xios_get_axis_attr
3	- Define a domain element "domain_A" of size 60x20 - Set the longitude value and latitude values for domain - longitude ranging from -180° to 180° - latitude ranging from -90° to 90°	id, type, ni_glo, nj_glo lonvalue, latvalue xios_get/set_domain_attr
4	- Define a grid element "grid_A" composed of "domain_A" and "axis_A"	id, domain_ref, axis_ref
5	- Define a field element "field_A" associated with "grid_A" - Send the field to XIOS in a temporal loop	id, operation, grid_ref xios_update_calendar xios_send_field
6	- Define the file element and output "field_A"	id, output_freq, type

Hands-on 3

Test different configurations of data size with respect to the domain size.

Case 1 : data size = domain size
Case 2 : data size > domain size : ghost layers (data offset negative)
Case 3 : data size < domain size : domain partially masked (data offset positive)





1	- Get the size of the 2D grid - Evenly distribute the domain along i (or longitude) onto the processes - You need to resize the lonvalue and latvalue to fit the size of subdomain - Set field_A the exact size of the subdomain and output to file "output.nc"	xios_get/set_domain_attr ni, nj, ibegin, jbegin
2	- Set field_A to have one layer of ghost cells and output to file	data_dim data_ni, data_ibegin data_nj, data_jbegin defalut_value
3	- Set field_A of size smaller than the subdomain and output to file - Don't forget to set "default_value" attribute to properly distinguish masked portion of the domain	

Hands-on 4

Test the file splitting method and the time_series functionality.

1	- Define two more fields "field_B" and "field_C" on grid "grid_A"	test_tp4.f90 iodef.xml
2	- Define in iodef.xml another output file "output_BC.nc" including instant outputs of "field_B" and "field_C"	output_freq
3	- Split "output.nc" into files, each of them containing output over 2 hours.	split_freq
4	- Enable the time series attribut for "field_B" - Set the splitting frequency to "2h" (or "2ts") - You should have no difference in the output files	ts_enabled ts_split_freq
5	- Test different mode for the time series output	timeseries

Hands-on 5

Read the netcdf file "input.nc" and output the read field.

1	- Check the dimensions of the input file using ncdump for example - The input grid is curvilinear and the field's name is "field_A_recv"	input.nc
2	- Construct a grid "grid_read" corresponding to the input field's grid - Define "field_read" on "grid_read" - Set the field's name to "field_A_recv"	iodef.xml

3	- Read the field content at each time step using "xios_recv_field" (freq_offset="1ts") - Copy the read value to another field "field_copy" - Output "field_copy" to "output_copy.nc" - We should have "output_copy.nc" same to "input.nc"	iodef.xml test_tp5.f90
4	- Read the file before the beginning of temporal loop (freq_offset="0ts")	

Hands-on 6

In the program, the hourly temperature is calculated. The grid is 2D for the sake of simplicity.

1	Define the weekly maximum temperature at noon, and the weekly minimum temperature at midnight
2	Compute the time variance of the temperature and output it daily

Hands-on 7

Using generic testcase program to get familiar with different transformations.
The generic_testcase program is designed to test all XIOS functionalities. Comes with the source, you have already many grids and fields predefined. You can customize the program by defining in iodef.xml, the type of the domain (lmdz, nemo, gaussian, dynamico, orchidee). You can change the size of the domain (ni, nj) and the size of axis (nlev). You can also decide if you want to put mask on the axis (axis_mask) or on the domain (domain_mask).

Test the following spatial transformation filters (refer to XIOS_reference_guide for full description of the filters) :

1	- Extract a subdomain of size 10x5 from "domain" starting at cell (1,2) - Construct grid "grid2D_extract_2dom" from the subdomain - Interpolate "field2D" on this grid and output the field to file "output_extract_2dom.nc" - (Optional) Output "field2D" to file "output.nc" and compare the result with "output_extract_2dom.nc"	extract_domain ni, nj, ibegin, jbegin
2	- Extract the horizontal axis from "domain" with vertical position = 2 - Construct grid "grid2D_extract_2ax" with the axis element - Interpolate "field2D" on this grid and output the field to file "output_extract_2ax.nc"	extract_domain direction, position
3	- Create an 0D grid "grid1D_reduce_axis" by summing all elements of "axis" - Interpolate "field_Z" on this grid and output to file "output_reduce_axis.nc" - (Optional) Output "field_Z" to file "output.nc" and compare the numerical result	reduce_axis operation

4	- Create an axis element "axis_reduce" of size "18" by reducing "domain" along the i-direction set the reduce operation to "sum" - Define a grid "grid2D_reduce_domain" with only the axis element - Interpolate "field2D" on this grid and output the field to file "output_reduce_dom.nc"	reduce_domain operation, direction, local
5	- Define a 1D grid "grid1D_inverse" by inverting "axis" - Interpolate "field_Z" on "grid1D_inverse" and output to file "output_inverse.nc"	inverse_axis
6	- Generate a rectilinear domain "recti_domain" of size 20x20 - Set the longitude bounds to -90° and 90° - Set the latitude bounds to -30° and 30° - Define grid "grid2D_recti" with the newly generated domain - interpolate "field2D" on this grid and output to file "output_interpolate_domain.nc"	generate_rectilinear_domain interpolate_domain type, ni_glo, nj_glo bounds_lon_start bounds_lon_end bounds_lat_start bounds_lat_end
7	- We want to obtain the values of "field3D" on different pressure levels : 70000, 50000, 30000, 10000 - (Optional) Output "field3D" and "pressure" to file "output.nc" - Define an axis element "dst_axis" of size 4 and set its value to 70000, 50000, 30000, 10000 (representing different levels of pressure) - Interpolate "field3D" by setting the coordinate to pressure and output the result to file "output_interpolate_axis.nc"	interpolate_axis order, type, coordinate
8	- Chain some transformations - Example : extract an axis of size 36 from "domain" and then extract a subaxis of size 10 from it. Interpolate "field2D" on the grid and output to file "output_chained.nc"	extract_domain extract_axis

Hands-on 8

Using transformations to solve a more realistic problem :
The temperature of a two dimensional region is recorded at an hourly frequency. What to define in the xml file in order to output the average temperature between 1 p.m. and 4 p.m. at a daily frequency ?

Please note that, we do not ask for the spatial average over the region. We are expecting, for every location in the region, the average temperature between 1 and 4 p.m.

1	- Create a grid "grid_1" using "domain" and a scalar element. - Reshape "temperature" on "grid_1" using a arithmetic operation and output the obtained field "temp_1" to file "output.nc" at an hourly frequency
2	- Create a grid "grid_2" using "domain" and an axis of size 24 - Use the temporal splitting filter to transform the 2D field "temp_1" to a 3D field "temp_2" - Output "temp_2" to file "output_ts.nc" with a daily frequency
3	- Create a grid "grid_3" with "domain" and an axis of size 4 - Use the extract_axis transformation to extract the desired values of "temp_2" (values at 1, 2, 3, and 4 p.m. each day) - Output the field "temp_3" to file "output_ts.nc"

4	- Create a grid "grid_4" with "domain and a scalar element - Use the reduce_axis transformation to calculate the average value of "temp_3" - Output the field "temp_4" to file "output_ts.nc"

Hands-on 9

This exercise uses the same code as in hands-on 7. The only difference is that here we reduced the file output frequency to 1 hour and only one time step is performed. It is for reducing the volume of the output files and the graph file.

1	Activate the graph functionality be setting the "build_workflow_graph" attribute to "true" for fields that you want to focus.
2	For example, enabled the graph output for "field2D" (line 230 of context_atm.xml)
3	You should obtain a JSON file "graph_data_atm.json"
4	Open the graph.html in web navigator and load the JSON graph file
5	Voila !

Hands-on 10

We continue with the same source.
Run the program with different mode : attached mode, one-level server mode, two-level server mode. Vary the number of servers and see the difference in performance.
Due to the limitation of computing capacity of the virtual machine, your may encounter difficulties in testing all configurations. We did some tests beforehand using Irène and you can find performance reports in the answer repository.

Hands-on 11

This program contains several errors. Run the program and look at the error output.

1	Error is occuring when parsing XML flux	XML syntax error
2	Some mandatory attributes are not defined	
3	Enable the xios_stack variable in iodef.xml This gives the full trace back of the error in the XIOS environment	xios_stack
4	Program runs OK but no output	use graph to check the workflow status