DE LA RECHERCHE À L'INDUSTRIE

XIOS

OUTPUT WHOLE CMIP6 DATA THROUGH THE NEW XIOS PARALLEL WORKFLOW FUNCTIONALITIES

M.H. Nguyen, R. Lacroix, O. Abramkina,Y.,Wang, A. Caubel, Y. Meurdesoif (IPSL/LSCE), S. Denvil (IPSL)

S. Senesi, D. Saint Martin (CNRM), M.P. Moine, S. Valcke (CERFACS)

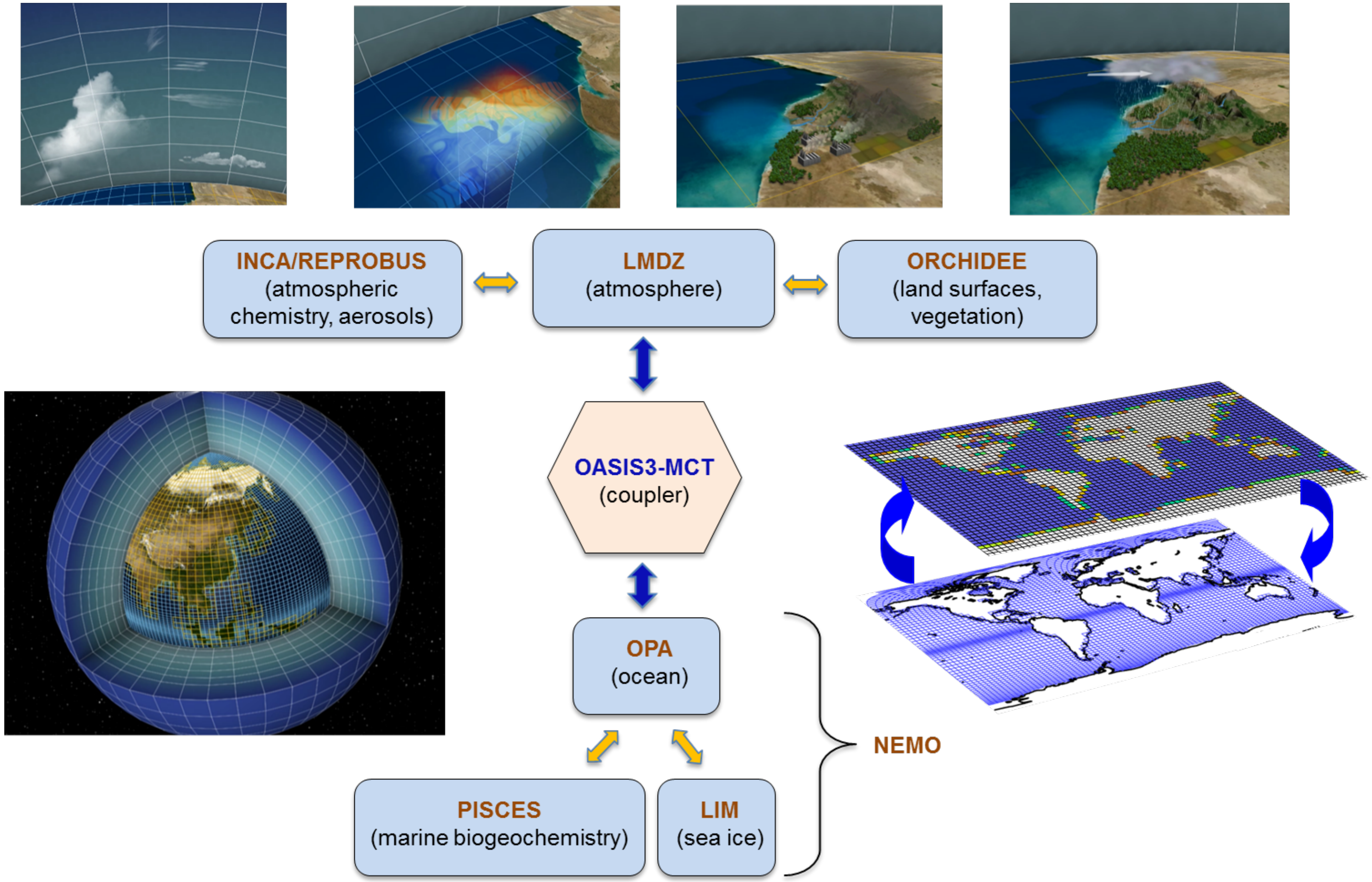
5th ENES HPC Workshop on "HPC for high resolution climate and weather modelling"

May 17th - 18th, 2018, LECCE

Yann Meurdesoif – CEA/DRF/LSCE & IPSL

www.cea.fr

THE IPSL EARTH SYSTEM MODEL



- Typical run : model IPSLCM6-LR
  - Resolutions
    - Atmosphere : 144x143x79 (2 °, 79 vertical levels)
    - Ocean : ORCA1, L75 (1° , 75 vertical levels)
- Performances : 16 SYPD on 930 cores on Curie

IPSL MODEL AND CMIP6 PROTOCOL

CMIP6 : 28 MIPs, 228 experiments, 2280 CMOR variables, 49 tables,…

- 40 000 years of simulation to perform

- **Data Request (XML file)**
  - Specifying the variables which are needed for each experiments.
  - High variability in the DRQ: from one experiment to the other, from one simulated year to the next one, from a modelling group to an other depending on the MIPs it is engaged in,…

Computing and storage resources

- 300 millions computing hours
  - 200 millions for development
  - 100 millions for production
- Production of 14 PB of data
- Distribution of 2 PB of data (ESGF)

Version	Atm resolution	Ocean resolution	SYPD
IPSL-CM6-VLR (atm chem, paleo)	*3°, L39*	*2°, L31*	*75*
IPSL-CM6-LR	**2°, L79**	**1°, L75**	**16**
IPSL-CM6-MR	1°, L79	0.25° or 1° L75	??
IPSL-CM6-HR (DYNAMICO)	0.6°, L79	0.25°, L75	??

FROM CMIP5 TO CMIP6

CMIP5 workflow (very traumatic memories…)

Run the model

Computing

(Tim…

…andard)

…ulation
…« failed »/simulatio…
…elaunch
- sequential tasks
- big amount of intermediate data

More time spent to post-treat than to produce

Publication

Analysis

ESGF/Web access

CMIP6 data workflow

Run the model & Output data (XIOS2)

Computing

Publication

Analysis

ESGF/Web access

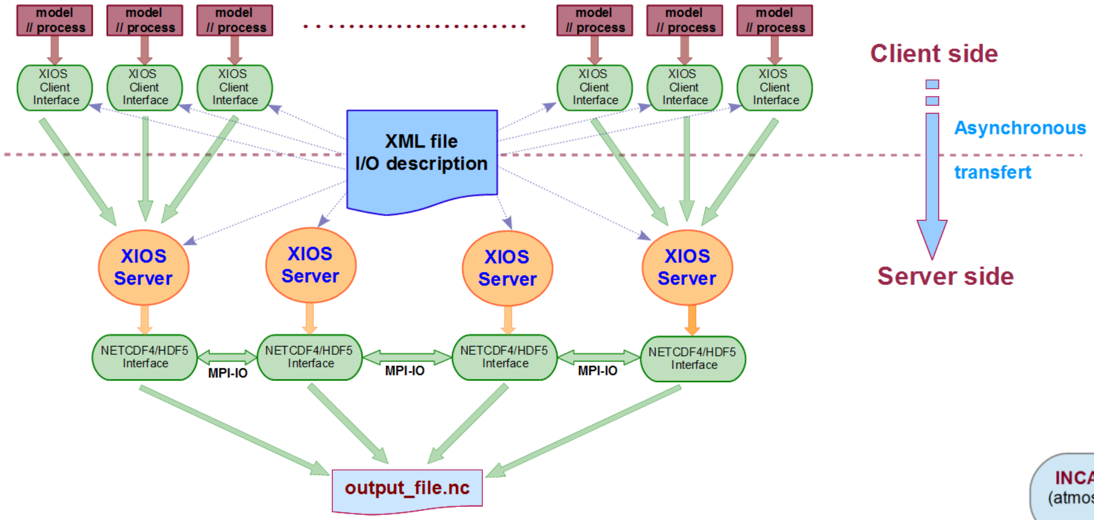
For CMIP6 : we have a dream…

- ❑ Output directly data from the model to avoid the nightmare of post-treatment and cmorization
- ❑ Ouput CMIP6-publication-ready data files

XIOS : "XML-IO-SERVER"

Flexible data output description through an external XML file.

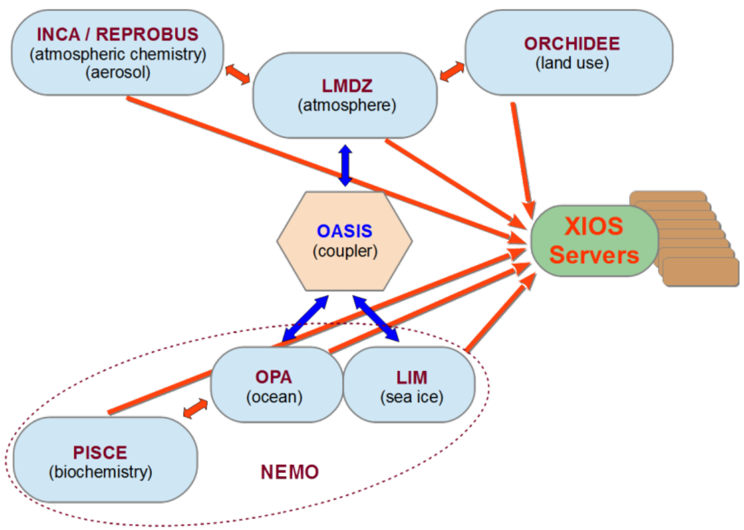
XIOS servers : asynchronous processes exclusively dedicated to output



- Overlap computation and I/O
- Rearrange data for better output efficiency
- Use parallel I/O for better efficiency
  - Aggregate I/O bandwidth of parallel file system
  - One piece files, no need to rebuild

Manage coupled models climate simulation

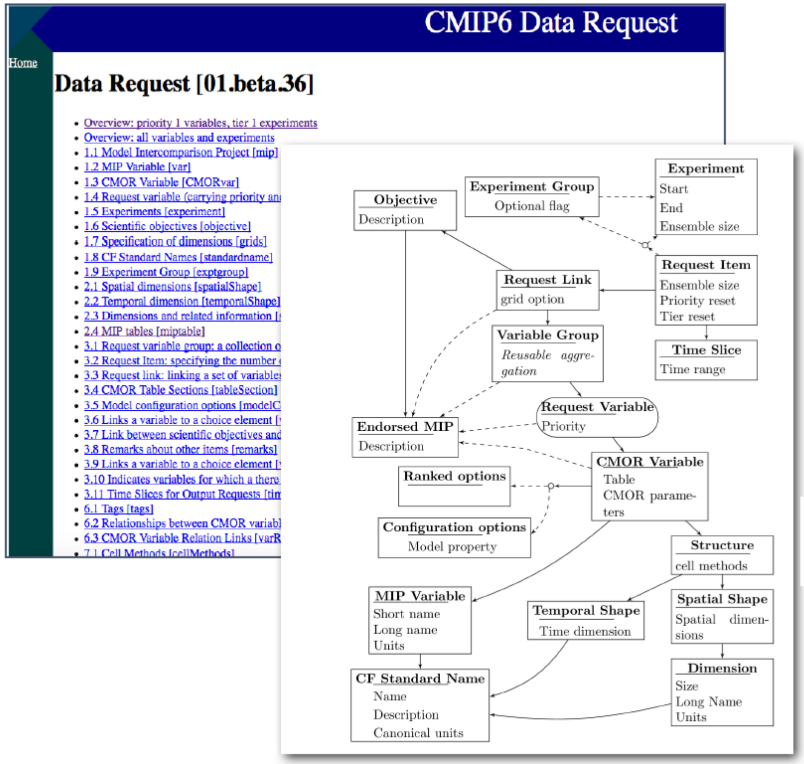
- Large numbers of cores (10 000+)
- One pool of I/O servers for all models



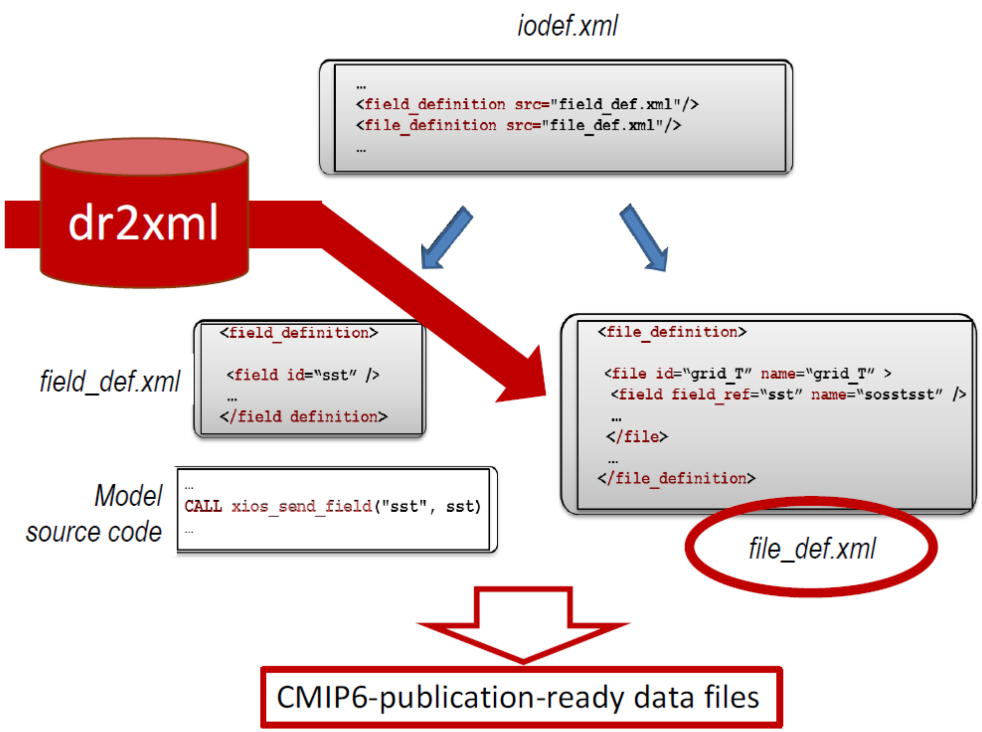
CNRM and IPSL are sharing the same CMIP6 workflow based on XIOS-2

- DR2XML, developed by CNRM (S. Sénesi)
  - Translates CMIP6 Data Request to XIOS configuration files (Python script)
- IPSL implement the missing functionalities into XIOS-2
- Data Request ➔ DR2XML ➔ XIOS (XML) input Files ➔ Data production

Data Request python API + XML files
(Martin Juckes)



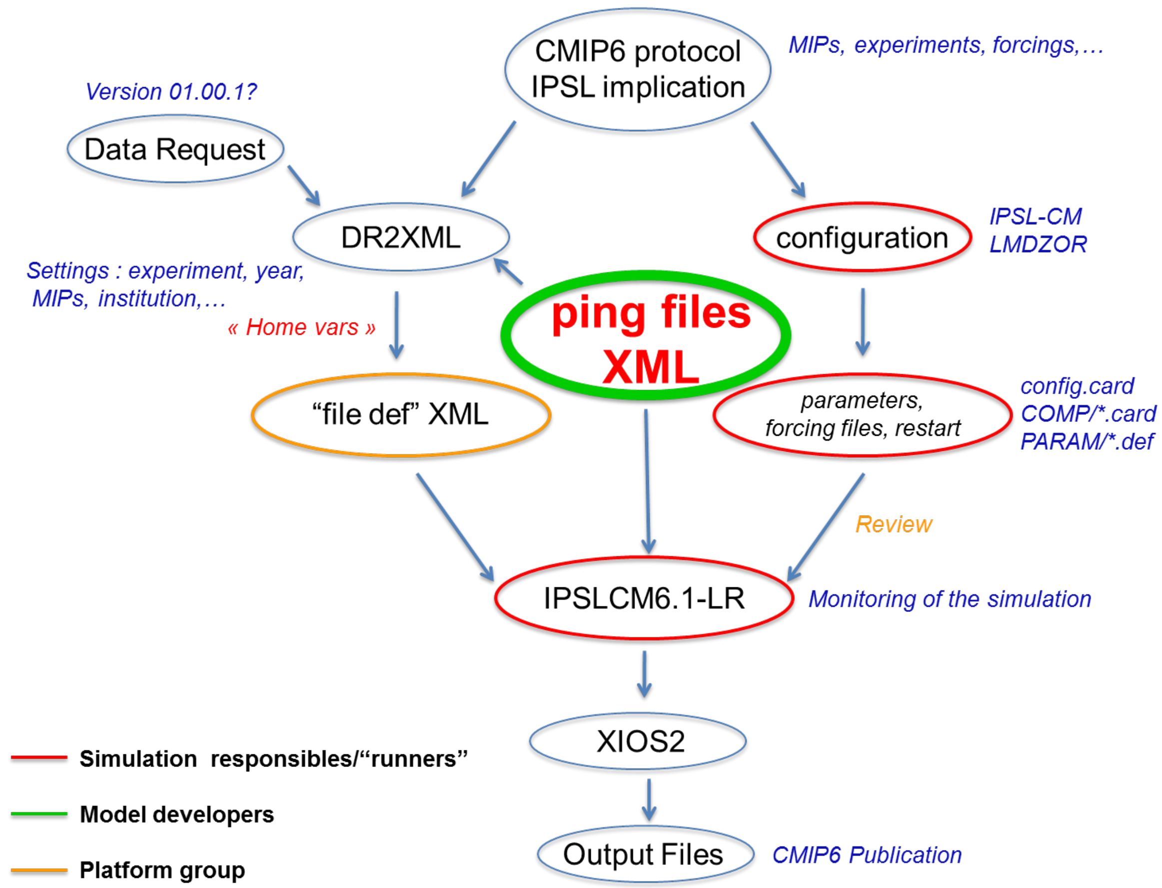
XIOS-enable model
(e.g. NEMO)



Ping file : variables Id from model <-> Data request Id (via DR2XML)

```
<field id="CMIP6_ps"  field_ref="psol"/>
```

- CMIP6 workflow is applied only on "ping" fields
- "Separation of concern" between standard output and CMIP6 output



AUTOMATIC CF / CMOR CONFORMANCE

- XIOS output fully CF 1.7 compliant
  - Axis & coordinates
  - Variables and associated metadata
  - Time axis management

- DR2XML generate automatically CMOR compliant XML input files for XIOS
  - CMOR specific global file attributes
  - CMOR specific associated metadata of variables
  - Renaming of axis & coordinates as required by Data Request

- Automatic time series management
  - One file by variable
  - Automatic generation of UUID (tracking_id)
  - Automatic chunk splitting at a given frequency specifically to an output file
    - Constant size for chunk of file variable
    - Automatic file name suffix corresponding to the period of chunk
  - An output file can be reopen and appended by the next run

DR2XML generate ~90 000 XML code line by experiment

HOW THAT IT LOOKS ?


```
 <file append="true" compression_level="4" convention_str="CF-1.7 CMIP-6.2" id="ps_3hr_gr" name="ps_3hr_IPSL-CM6A-LR_historical_r3i1p1f1_gr_%start_date%-%end_date
  <variable name="activity_id" type="string"> CMIP   </variable>
  <variable name="contact" type="string"> ipsl-cmip6@listes.ipsl.fr   </variable>
  <variable name="data_specs_version" type="string"> 01.00.21   </variable>
  <variable name="dr2xml_version" type="string"> 1.3   </variable>
  <variable name="experiment_id" type="string"> historical   </variable>
  <variable name="description" type="string"> CMIP6 historical   </variable>
  <variable name="title" type="string"> CMIP6 historical   </variable>
  <variable name="experiment" type="string"> all-forcing simulation of the recent past   </variable>
  <variable name="external_variables" type="string"> areacella   </variable>
  <variable name="forcing_index" type="int"> 1   </variable>
  <variable name="frequency" type="string"> 3hrPt   </variable>
  <variable name="further_info_url" type="string"> https://furtherinfo.es-doc.org/CMIP6.IPSL.IPSL-CM6A-LR.historical.none.r3i1p1f1   </variable>
  <variable name="grid" type="string"> LMDZ grid   </variable>
  <variable name="grid_label" type="string"> gr   </variable>
  <variable name="nominal_resolution" type="string"> 250 km   </variable>
  <variable name="history" type="string"> none   </variable>
  <variable name="initialization_index" type="int"> 1   </variable>
  <variable name="institution_id" type="string"> IPSL   </variable>
  <variable name="institution" type="string"> Institut Pierre Simon Laplace, Paris 75252, France   </variable>
  <variable name="license" type="string"> CMIP6 model data produced by IPSL is licensed under a Creative Commons Attribution-NonCommercial-ShareAlike 4.0 Internat
  <variable name="mip_era" type="string"> CMIP6   </variable>
  <variable name="parent_experiment_id" type="string"> piControl   </variable>
  <variable name="parent_mip_era" type="string"> CMIP6   </variable>
  <variable name="parent_activity_id" type="string"> CMIP   </variable>
  <variable name="parent_source_id" type="string"> IPSL-CM6A-LR   </variable>
  <variable name="parent_time_units" type="string"> days since 1850-01-01 00:00:00   </variable>
  <variable name="parent_variant_label" type="string"> r3i1p1f1   </variable>
  <variable name="branch_method" type="string"> standard   </variable>
  <variable name="branch_time_in_parent" type="double"> 21914.0D   </variable>
  <variable name="branch_time_in_child" type="double"> 0.0D   </variable>
  <variable name="physics_index" type="int"> 1   </variable>
  <variable name="product" type="string"> model-output   </variable>
  <variable name="realization_index" type="int"> 3   </variable>
  <variable name="realm" type="string"> atmos   </variable>
  <variable name="source" type="string"> IPSL-CM6A-LR (2017):
atmos: LMDZ (NPv6, N96; 144 x 143 longitude/latitude; 79 levels; top level 40000 m)
land: ORCHIDEE (v2.0, Water/Carbon/Energy mode)
ocean: NEMO-OPA (eORCA1.3, tripolar primarily 1deg; 362 x 332 longitude/latitude; 75 levels; top grid cell 0-2 m)
ocnBgchem: NEMO-PISCES
seaIce: NEMO-LIM3   </variable>
  <variable name="source_id" type="string"> IPSL-CM6A-LR   </variable>
  <variable name="source_type" type="string"> AOGCM   </variable>
  <variable name="sub_experiment_id" type="string"> none   </variable>
  <variable name="sub_experiment" type="string"> none   </variable>
  <variable name="table_id" type="string"> 3hr   </variable>
  <variable name="title" type="string"> IPSL-CM6A-LR model output prepared for CMIP6 / CMIP historical   </variable>
  <variable name="variable_id" type="string"> ps   </variable>
  <variable name="variant_info" type="string"> Restart from another point in piControl.. Information provided by this attribute may in some cases be flawed. Users
  <variable name="variant_label" type="string"> r3i1p1f1   </variable>
  <variable name="EXPID" type="string"> historical   </variable>
  <variable name="CMIP6_CV_version" type="string"> cv=6.2.3.5-2-g63b123e   </variable>
  <variable name="dr2xml_md5sum" type="string"> 00e1a4f623b35a33620b9828c66bd1c8   </variable>
  <variable name="model_version" type="string"> 6.1.2   </variable>
  <field field_ref="CMIP6_ps_instant" name="ps"  freq_op="3h" grid_ref="grid_glo_greordered" cell_methods="area: mean time: point" cell_methods_mode="overwrite" c
     <variable name="standard_name" type="string"> surface_air_pressure </variable>
     <variable name="description" type="string"> sampled synoptically to diagnose atmospheric tides, this is better than mean sea level pressure. </variable>
     <variable name="long_name" type="string"> Surface Air Pressure </variable>
     <variable name="history" type="string"> none </variable>
     <variable name="units" type="string"> Pa </variable>
     <variable name="cell_methods" type="string"> area: mean time: point </variable>
     <variable name="cell_measures" type="string"> area: areacella </variable>
     <variable name="interval_operation" type="string"> 900 s </variable>
     </field>
</file>
```

CMIP6 output require a lot a diagnostics

- Unit rescaling
- Normalization by area or level height
- Time integration (averaging, minimum, maximum)
- Vertical interpolation in pressure levels
- Extraction on specific pressure levels
- Vertical or global summation
- Horizontal remapping
- Zonal mean
- Diurnal cycle, seasonal means
- Cfsites (points station extraction)
- Transects (flux across ocean straight)
- Many more complex diagnostics (ex : Eliassen Palm flux)
- ……

Why do not taking advantage of thousands of computing cores allocated, to make the post-treatment in parallel, all along the simulation ?

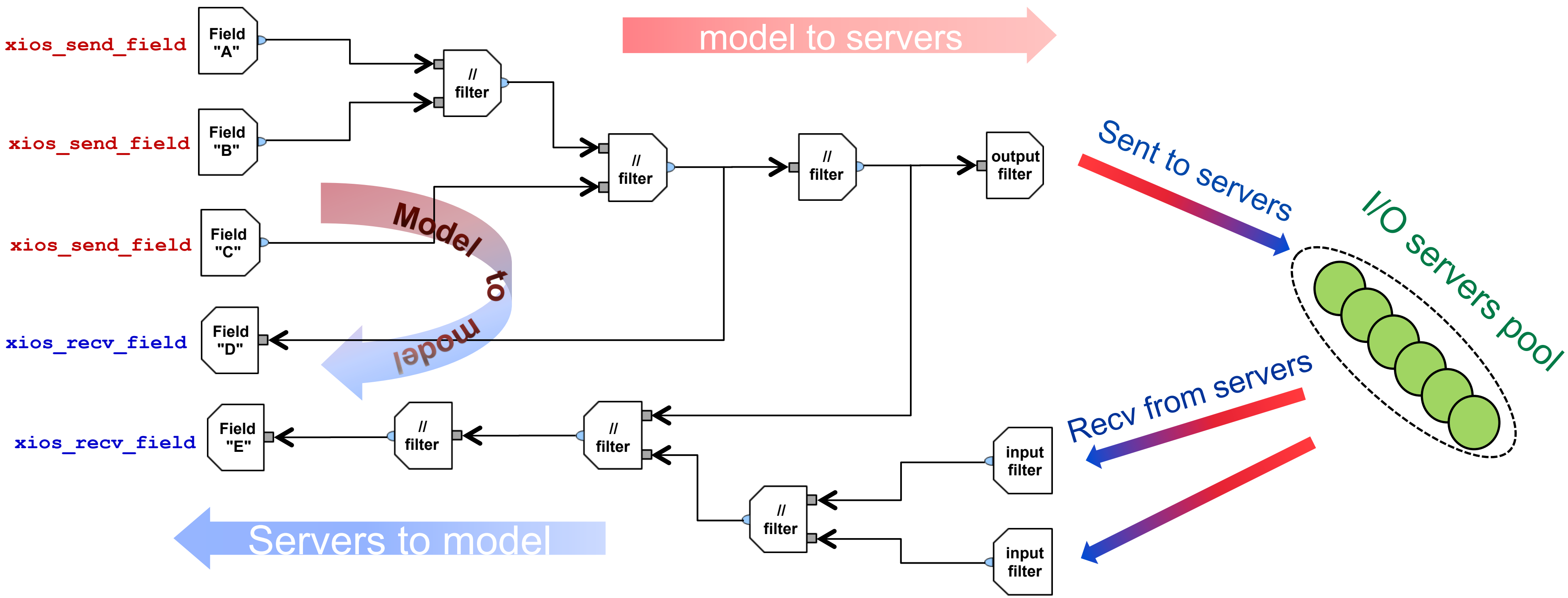
- Avoid to read, write, re-read, re-write, re-re-read, etc..., uselessly…

All these diagnostics can be computed online described using new XIOS-2 workflow functionalities

- Diagnostics are described externally by XML files

XIOS-2 embed an internal parallel workflow/dataflow
The XML files describe a parallel task graph

- Incoming data are representing data flux, assigned to a timestamp
  - Each flux can be connected to one or more filters
- Filters are connected to one or more input flux and generate a new flux on output
  - All Filters can be chained to achieve complex treatment
  - All filters are **parallel and scalable**



FILTERS

3 families of computing filters

- **Temporal filters : perform time integration**
  - Integrate input flux from a series of timestamp
  - Output flux with a new timestamp
  - Ex : instant, average, maximum, minimum, accumulate
  - Soon : time interpolation filter

```
<field id="temp" unit="K" operation="average"/>
```

- **Arithmetic filters**
  - Combine different flux from a same timestamp
  - Perform arithmetic operations for each grid point
  - ex: $C=(A+B)/(A*B)$; $D=e^{-C*D}/3$

```
<field id="A" />
<field id="B" />
<field id="C"  > (A + B) / (A*B)    </field>
<field id="D"  >  exp(-C*this) / 3  </field>
```

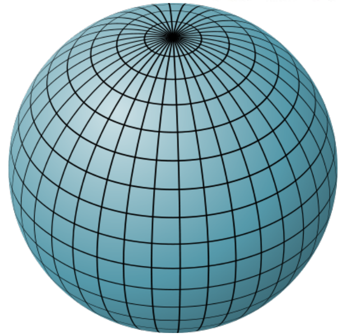
- Can be chained with temporal filter to achieve more complex treatment
  - Ex : Compute the time standard deviation of a temperature field $\sigma\approx\sqrt{\langle T^2\rangle-\langle T\rangle^2}$ every month

```
<field id="T" operation="average"/>
<field id="T2" operation="average"/> T*T </field>
<field id="sigma" freq_op="1mo"/> sqrt(@T2-@T*@T) </field>
```

  - Ex: Compute the monthly averaging of the daily maximum of temperature

```
<field id="T"            operation="maximum"/>
<field id="daily_Tmax" operation="average" freq_op="1d"> @T </field>
<field id="ave_daily_Tmax"                 freq_op="1mo"> @daily_Tmax </field>
```

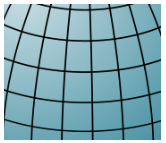
FILTERS



○ Spatial Filters

- Geometrical shape of the input flux is modified
- Defined by a grid transformation from a source grid to a target grid

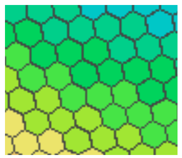
- Ex: field remapping over horizontal layer



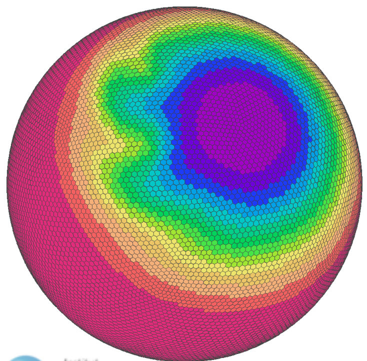
```
<grid id="grid_src" />
  <domain id="domain_src" type="rectilinear"/>
  <axis   axis_ref="an_axis"  />
</grid >
```

```
<field id="field_src" grid_ref="grid_src" />
```

```
<field id="field_dest" field_ref="field_src"
                       grid_ref="grid_dest" />
```

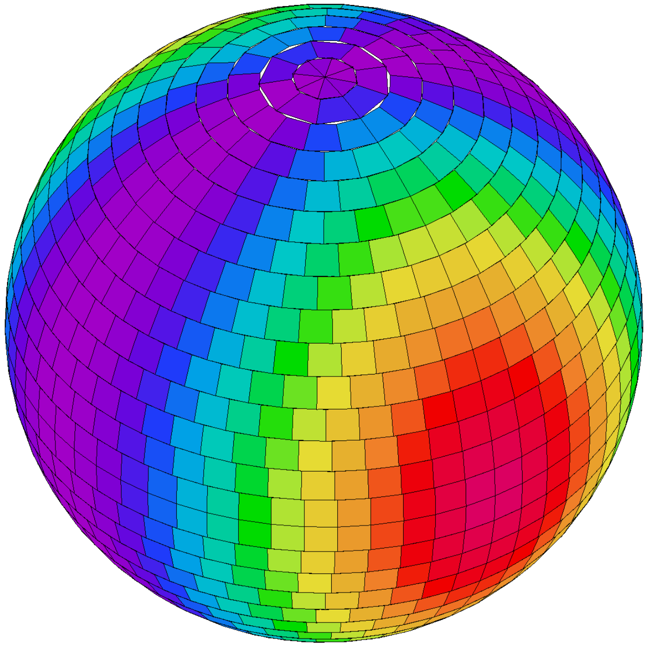


```
<grid id="grid_dest" />
  <domain id="domain_dest" type="unstructured">
    <domain_interpolate order="2"/>
  </domain>
  <axis   axis_ref="an_axis"  />
</grid >
```

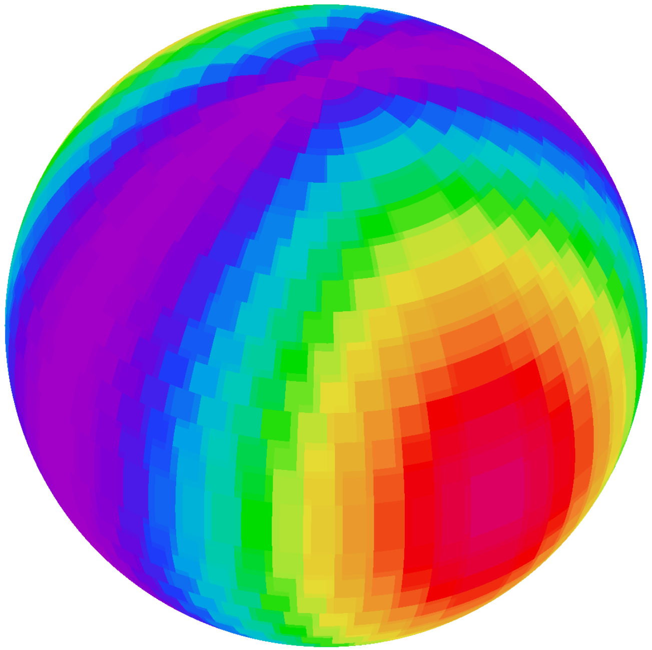


2ND ORDER CONSERVATIVE REMAPPING ON THE SPHERE (ICOMEX : E.KRITSIKIS, T. DUBOS, Y. MEURDESOIF)

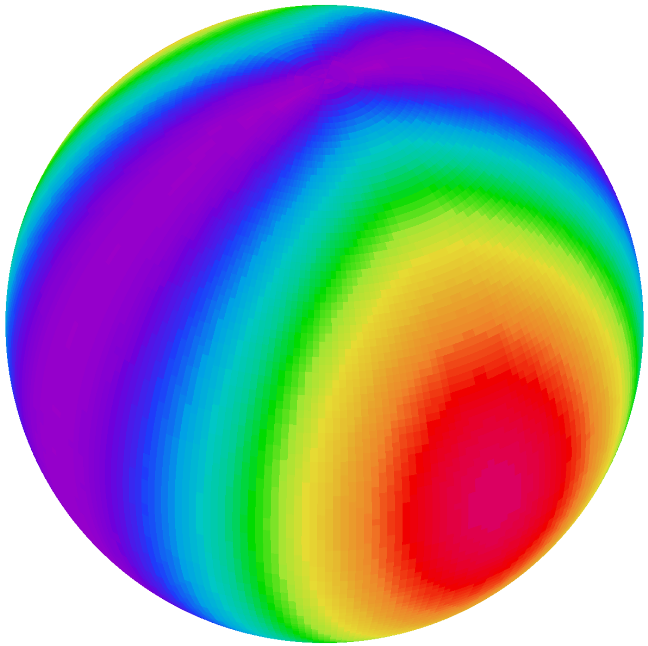
- Parallel weight computation on "the fly"
- Parallel remapping, management of masked cells…
- Handle geodesic unstructured mesh (great circle) and rectilinear lon-lat or gaussian mesh (great and small circle)
- Ex : Remapping Gaussian reduced 60x30 -> regular lon-lat 2°



Source mesh



Remapping order 1



Remapping order 2

OTHER AVAILABLE PARALLEL FILTERS….

- (domain -> domain) : **<zoom_domain />** : extract area of interest
- (axis -> axis) : **<zoom_axis/>** : extract part of an axis
- (axis->scalar) : **<extract_axis_to_scalar/>** : axis slice extraction
- (domain->axis): **<extract_domain_to_axis/>** : latitude or longitude extraction
- (axis->axis) : **<inverse axis/>** : invert axis
- (axis->axis) : **<interpolate_axis>** : axis interpolation, possibly on pressure level
- (domain->domain) : **<interpolate_domain/>** : horizontal remapping
- (domain) : **<generate_rectilinear_domain/>** : create a rectilinear mesh
- (domain->scalar) : **<reduce_domain_to_scalar/>** : global domain reduction (sum, average, max, min,…)
- (domain->axis): **<reduce_domain_to_axis/>** : partial domain reduction along i or j direction
- (axis->scalar) : **<reduce_axis_to_scalar/>** : axis reduction (sum, average, max, min, …)
- (scalar->axis) : **<temporal_splitting/>** : diurnal cycle
- (scalar->axis) : **<duplicate_scalar_to_axis>** : duplicate data along a new axis
- (domain->domain) : **<reorder_domain/>** : reorder indexes of horizontal domain
- (domain->domain): **<expand_domain/>** : expand local domain at first neighbor and transfer ghost cells
- (domain) : **<compute_connectivity/>** : find the connectivity of an unstructured domain

And many others in future…

CHAINING FILTERS…

Filters can be chained to compute more complex diagnostics

- Zonal mean of monthly average of temperature at 850, 500 and 350 hPa

Temp -> horizontal remapping on 1° regular mesh -> vertical interpolation on pressure levels -> reduction (average) over the longitude -> time averaging -> output

```xml
<grid id="src">
  <domain id="hlayer">
  <axis id="height">
</grid>

<grid id="grid_interp">
  <domain id="reg_lon_lat" type="rectilinear" ni_glo="360" nj_glo="180"/>
    <generate_rectilinear_domain/>
    <interpolate_domain order="2"/>
  </domain>
  <axis id="pressure_level" n_glo="3" value="(0,3)[850 500 350]" />
    <axis_interpolate coordinates="pressure" />
  </axis>
</grid>

<grid id="grid_zonal">
  <axis id="lat" n_glo="180">
    <reduce_domain_to_axis direction="jDir" operation="average"/>
  </axis>
  <axis id="pressure_level" />
</grid>


<field id="Temp" grid_ref="grid_src">
<field id="pressure" grid_ref="grid_src">

<file id="output" output_freq="1mo" />
  <field field_ref="Temp" operation="average" grid_ref="grid_zonal" grid_path="grid_interp"/>
</file>
```

For some complex diagnostics need to be done into model, XIOS reentrance can be used (ex: barotropic stream function in Nemo ocean model)

- Ex : compute a diagnostic with the monthly average of temp and return it to be output
  - Send temperature to XIOS at every time step
  - After every month XIOS compute the average which can be retrieve from the model
  - Compute the diagnostic into the model side
  - Send the computed diagnostic to XIOS to be written

- Fortran model side

```fortran
CALL xios_send_field("temp",temp)                   ! send temperature value each time step

IF (xios_field_is_active("diag",at_current_timestep=true) THEN  ! end of month ?
   CALL xios_recv_field("temp_ave",temp_ave)  ! get the monthly average of temp
   CALL make_complex_diag(temp_ave,diag)      ! compute the diagnostic
   CALL xios_send_field("diag",diag)          ! send the diagnostic to write
ENDIF
```

- XIOS side (XML)

```xml
<field id="temp" grid_ref="grid_model" operation="average"/>  <!--temp sent every timestep-->
<field id="temp_ave" freq_op="1mo" grid_ref="grid_model" read_acess="true">@temp</field> <!--compute average-->

<file id="output" freq_op="1mo"/>
  <field id="diag" grid_ref="grid_model" operation="instant"> <!--monthly output of diag-->
</file>
```

CMIP6 is running now !!!

- Configuration : model IPSLCM6-LR
  - Atmosphere : 144x143x79 (2 °, 79 vertical levels)
  - Ocean : ORCA1, L75 (1° , 75 vertical levels)
  - Performances : 16 SYPD on 930 cores on Curie (Bull, intel Sandy-Bridge)
- CMIP6 light I/O throughput (piControl, large part of the CMIP6 runs)
  - Config : 1 years (1850) piControl : 4 XIOS servers
  - No I/O **: 4980 s**
  - Only IO Standard (monthly output) : **5460s** (+10%)
  - Only CMIP6 I/O : **5460 s** (+10%, 0% compared to standard I/O)
  - CMIP6 + standard : **5820 s** (+16%, +6% compared to only standard I/O)
- CMIP6 medium I/O throughput : 1 year historical 1850, CMIP6 I/O + standard
  - 927 files / variables, 158 Gb (compressed)
  - 12 XIOS servers
  - CMIP6 + standard : **6454 s** (+18 % compared to only standard I/O)

IPSL CMIP6 WORKFLOW PERFORMANCE

- CMIP6 High I/O throughput : one year Full CMIP6 I/O output + standard I/O
  - 1173 files/variables, 1.5 Tb (compressed)

config	time	% Vs standard I/O
4 XIOS -2 NODES	16440 s	+201 %
8 XIOS - 4 NODES	13020 s	+138 %
16 XIOS - 2 NODES	9300 s	+70 %
16 XIOS - 4 NODES	9600 s	+75 %
16 XIOS - 8 NODES	9360 s	+71 %
24 XIOS - 2 NODES	8460 s	+54 %
24 XIOS - 8 NODES	8040 s	+47 %
24 XIOS - 12 NODES	**7860 s**	**+44 %**
32 XIOS - 2 NODES	8460 s	+55 %

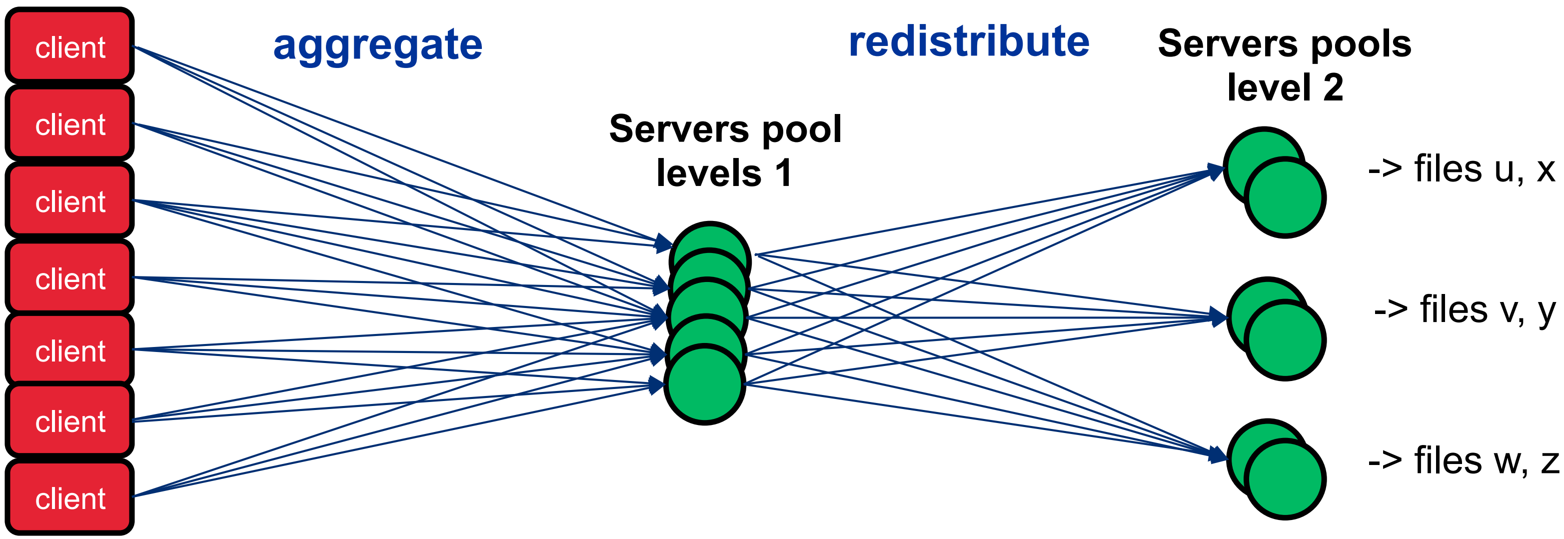
  - Non negligible impact on computing time : +44 %
  - Impact come from workflow cost, not I/O
  - But for a low number of runs (<5%), so it remains acceptable

IMPROVING PERFORMANCES : 2ND LEVEL OF SERVERS

- Parallel I/O with Netcdf / HDF5 does not scale well, especially for small mesh
- High cost to open and close a file in parallel mode
  - With 1000+ files the impact is strong
- Unable to compress data on the "fly" using Netcdf / HDF5 parallel I/O
  - Limitation will be removed in future HDF5 1.10 versions

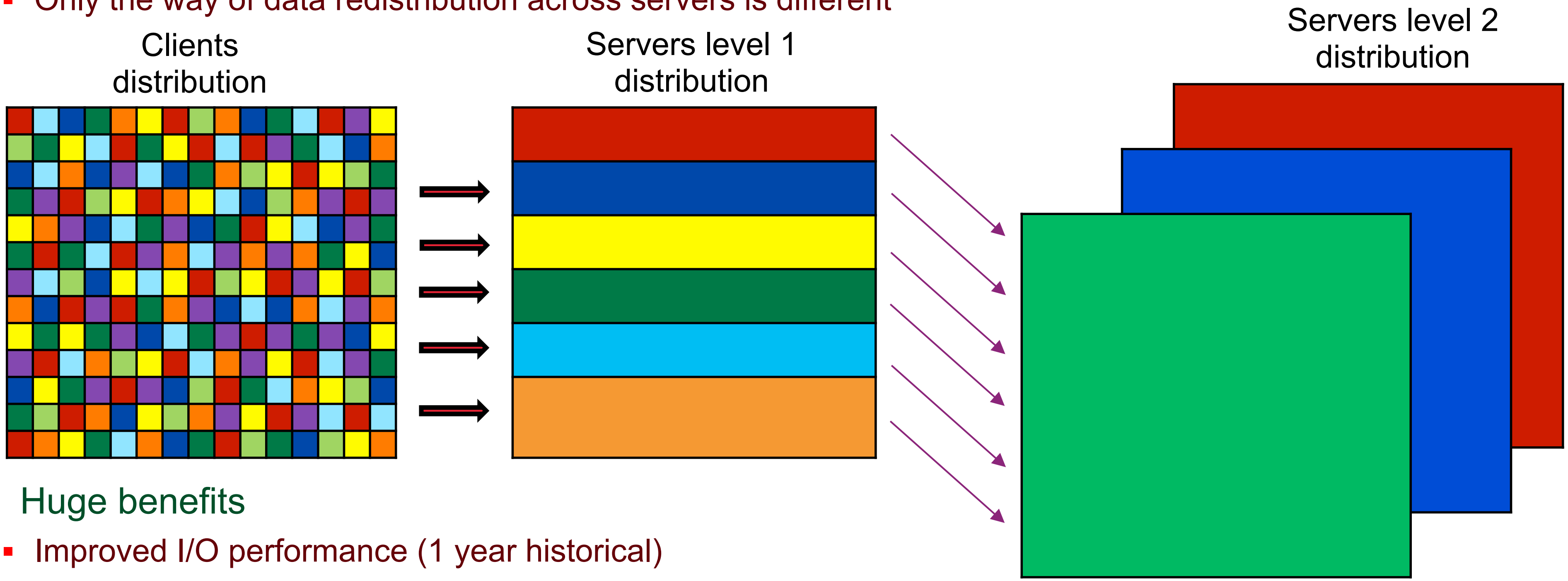
Add a second level of servers to write files in sequential mode

- First level will aggregate fields from client and redistribute its to second level
- Second level received field on global mesh and make sequential write
- I/O parallelism is achieved by write sequential files concurrently



2ND LEVEL OF SERVERS

- Internally, the first levels of server acts like a client pool for the second level of servers
  - Only the way of data redistribution across servers is different



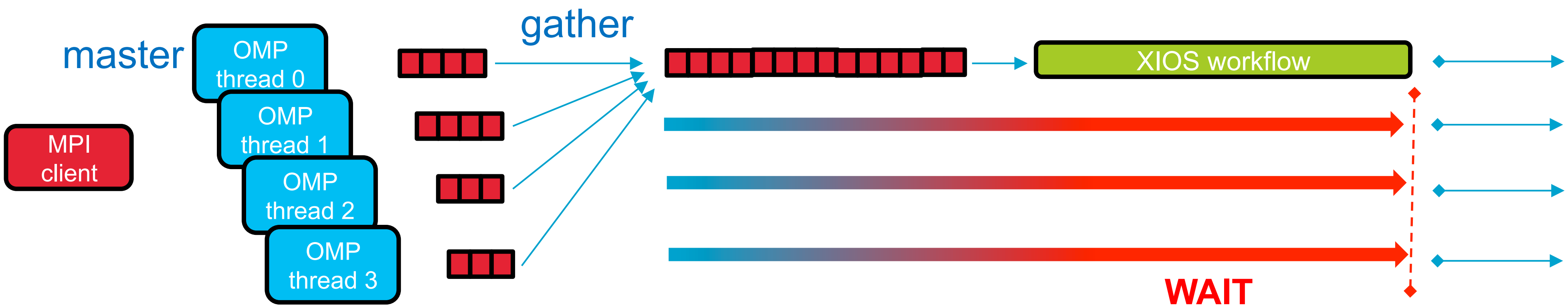
- Huge benefits
  - Improved I/O performance (1 year historical)
    - 12 XIOS servers level 1 : **23 000 s**
    - 12 XIOS servers : 6 level 1 + 6 levels 2 : **6454 s**
  - Activate compression (1 year historical)
    - Without compression : **327 Gb**
    - With compression : **158 Gb** : storage divided by more of 2
  - Easy to use : 2 parameters in XIOS namelist
    - use_server2="true"
    - ratio_server2="% of server 2" (default 50%)

AN OTHER WAY OF IMPROVEMENT : MULTITHREADISM ?

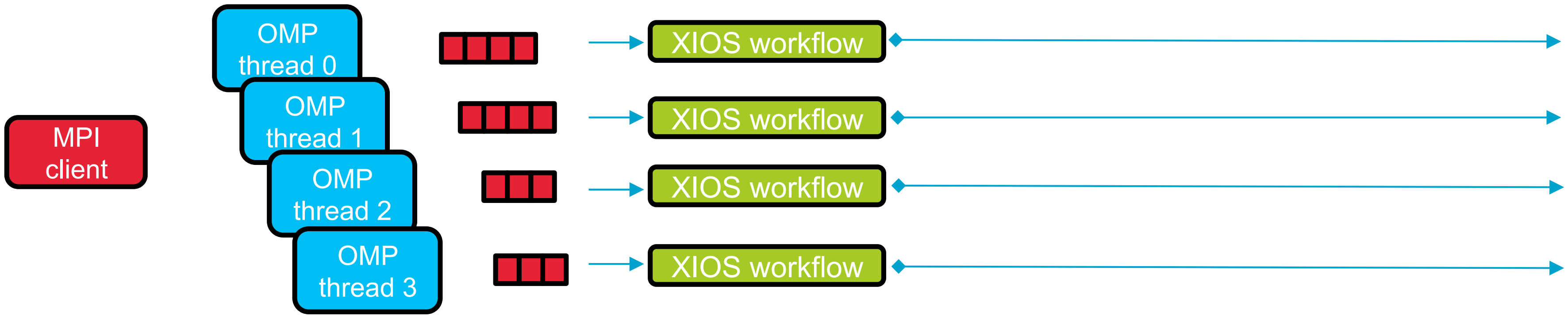
Problem :

LMDZ (atmosphere) : MPI + OpenMP ⟷ XIOS : pure MPI

- Currently : OpenMP models gather their data on master thread before calling XIOS
  - Big bottleneck : master thread do the job for all others waiting threads



- Future : all threads will participate to the XIOS call
  - All threads will do their job part



HOW IT CAN BE DONE ?

XIOS parallel protocol is complex :

We don't want to add a new level of parallelism in the code management

=> Adapt the actual implementation of parallelism to multithread

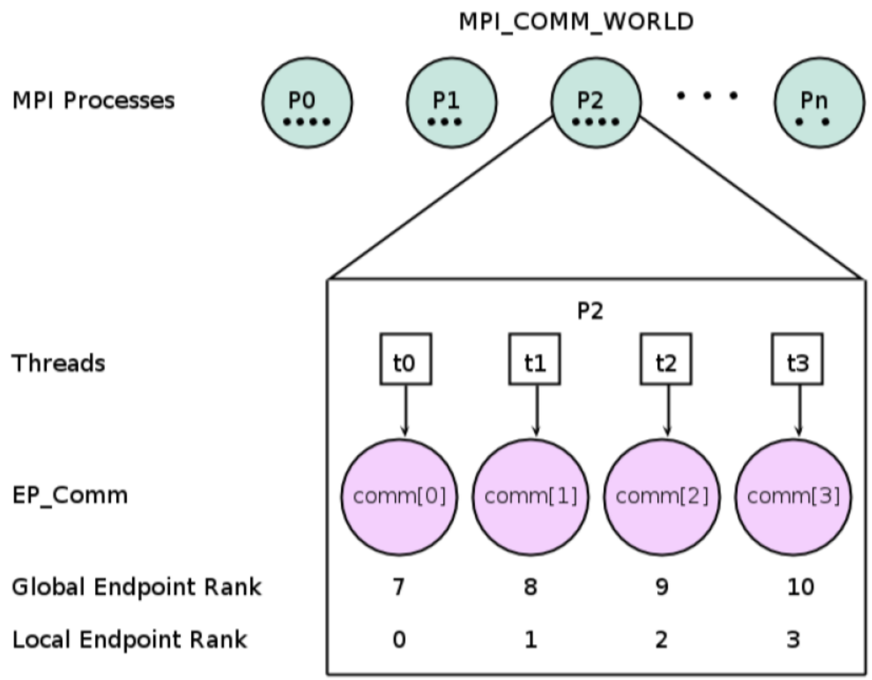
Idea : consider each OpenMP threads like a process that can communicate through the MPI library with other threads of other MPI process.

Solution : MPI ENDPOINTS

- Proposal under discussion on MPI-4 forum, waiting for adoption or not….

```
int MPI_Comm_create_endpoints(MPI_Comm parent_comm, int num_ep,
                              MPI_Info info, MPI_Comm out_comm_hdls[])
```

- Create a new communicator of size of total number of threads
- Each thread receive a rank and can make transfer to other threads using standard MPI call



EP-LIB (Y. WANG, ESIWACE)

EP-LIB : XIOS team has developed a new MPI wrapper which implement MPI endpoints functionalities on top of standard MPI library

- Subset of MPI1 and MPI2 : P2P, collectives, one sided communication…
- Embedded into XIOS but can be used for other purpose
- MPI model can be used in MPI+OpenMP without major change
  - Privatize shared variable (static for C, SAVE for Fortran)
  - Fortran : !$OMP THREADPRIVATE (var)
  - C/C++ : *#pragma omp threadprivate*(var)

On going work….

- Implemented into XIOS (client part)
- Atmospheric model adapted (LMDZ)
- First test…

EP-LIB FIRST RESULTS

Very preliminary results

- Test : LMDZ 144x142x79, heavy daily output (level 10)

			job time (s)	XIOS time (s)	job speedup	xios speedup
10 days	12MPIx8OMP	no EP	699	75,10	1,15	3,73
		EP	609	20,12		
	12MPIx4OMP	no EP	1017	74,97	1,07	2,83
		EP	949	26,51		
	12MPIx2OMP	no EP	1756	74,98	1,03	1,69
		EP	1702	44,37		
	12MPIx1OMP	no EP	3289	75,55	1,01	0,99
		EP	3263	76,26		
	6MPIx8OMP	MPI	1337	143,51	1,40	5,12
		EP	953	28,00		
	6MPIx4OMP	MPI	1978	143,62	1,16	3,31
		EP	1702	43,37		

- Expected speed-up: > 6 on 8 threads
- Need some improvement…

CONCLUSION

- Hard work to develop workflow functionalities that satisfy CMIP6 requirement

- Hard work to develop DR2XML which make the translation between Data Request and XIOS

But CMIP6 workflow is fully functional now !!

- Performances are quiet satisfying on the LR model

But a lot of room for improvement

- Improve the transfer protocol and I/O for large number of servers
- Improve the scalability of some workflow filters
- More generally improve the workflow performance

Next step : HighRes MIP

- 50km ~ 25 km runs