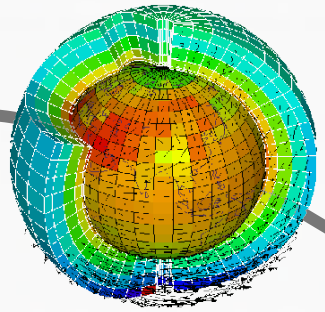
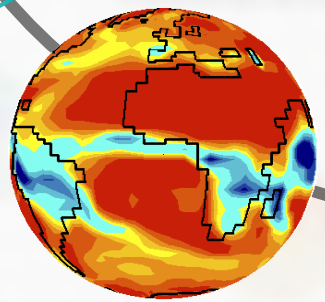
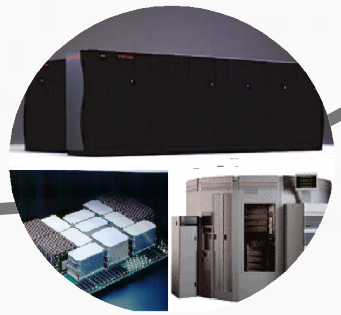
Production et dissémination de grandes masses de données à l'IPSL

$$\frac{\partial \mathbf{U}}{\partial t} + 2\Omega \times \mathbf{U} = .$$

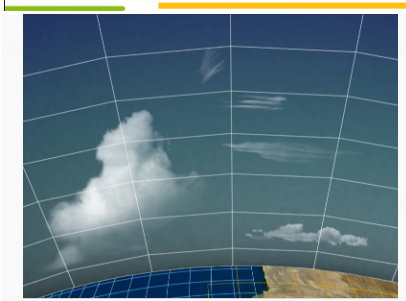
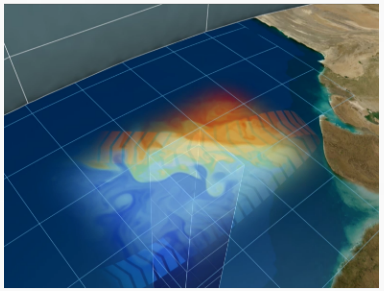
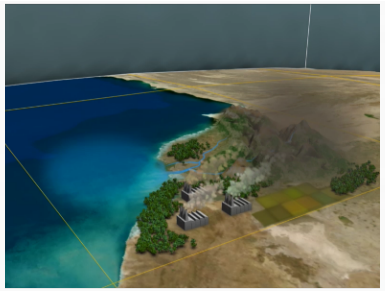
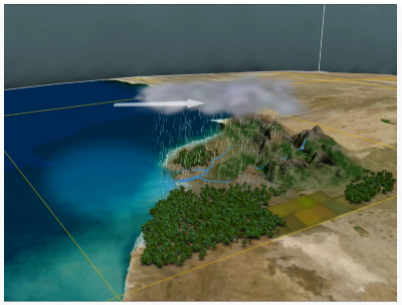


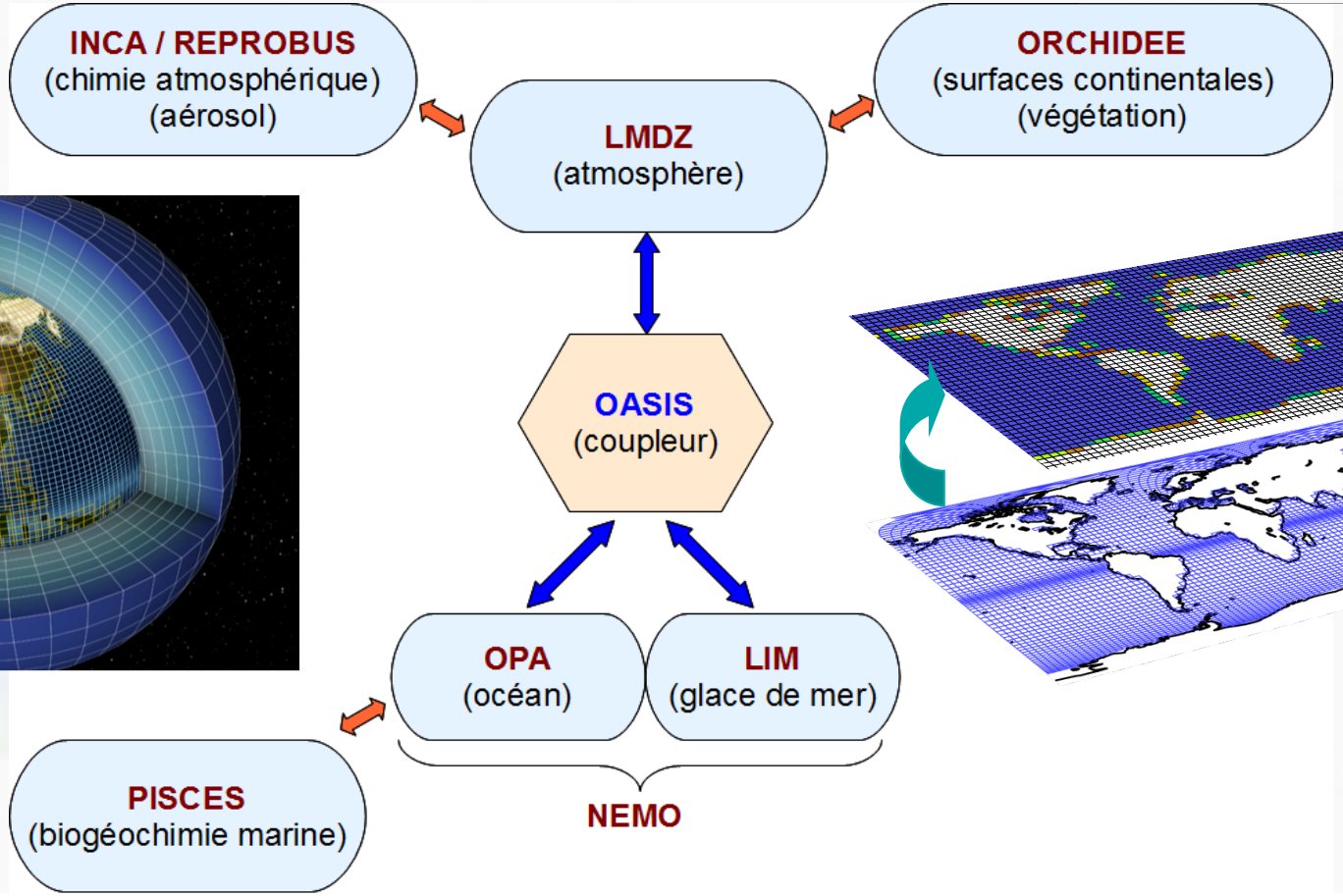


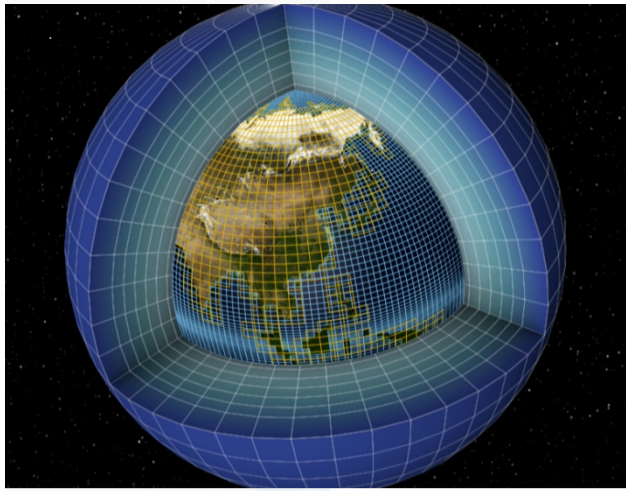


```
CTER*(*) pfilename
EGER,INTENT(IN) :: pim,pjm
EAL,DIMENSION(pim,pjm),INTENT(IN) :
INTEGER,INTENT(IN):: par_orix, par_sz
INTEGER,INTENT(IN) :: pitau0
REAL,INTENT(IN) :: pdate0, pdeltat
INTEGER,INTENT(OUT) :: pfileid, phorii

INCLUDE "netcdf.inc"

NTEGER :: ncid, iret
TEGER :: leng, lengf, lenga
CTER(LEN=120) :: file, tfil
ER*30 :: timenow
```

Le modèle couplé "Système Terre" de l'IPSL





L'exercice CMIP 5 – IPCC AR5

- **CMIP5** (Coupled Model Intercomparison Projet phase 5)
  IPCC AR5 (Intergovernmental Panel Climate Change, Assesment Report 5)
  - Des simulations « centennales » variées:
    - 20 et 21è siècles (historiques + scénarios futurs)
    - paléoclimat, dernier millénaire…
  - Des modèles de complexités différentes:
    - Modèle climatique "physique" (AOGCM)
    - Modèles avec cycle biogéochimique (modèle système Terre)
    - Configurations idéalisées (aqua-planète, …)
  - Des simulations décennales à hautes résolutions…
  - Fichiers journaliers et mensuels
    - plus de 800 variables différentes
- Produit un énorme flots de données
  => 2 Po de données produites pour l'IPSL dont 0.5 Po distribuées
  (le climat représente 80% du stockage de donnée au CCRT)
  - Production efficace des données
  - Post-traitement
  - Stockage
  - Distribution

Caractéristiques des sorties "histoires" des modèles de l'IPSL

Fichiers

- Chaque modèle produit ces propres fichiers « histoires » au format netcdf.
- Les fichiers sont composés de plusieurs dizaines de variables sur la grille du modèle, intégrées sur plusieurs centaines d'années.
- Les fréquences de sortie des fichiers peuvent être horaires (3h, 6h), journalières et/ou mensuelles.

Variables

- Les champs peuvent être 2D (champ de surface) ou 3D (grille globale).
- Les champs sont intégrés temporellement suivant la fréquence de sortie des fichiers : valeurs instantanées, moyenne temporelle sur la période, valeurs minimum ou maximum...
- A chaque champ sont associées de nombreuses meta-données permettant sa description (titre, description, unité, axes associés,etc....).
- Un même champ peut apparaître dans plusieurs fichiers (horaire, journalier, mensuel)

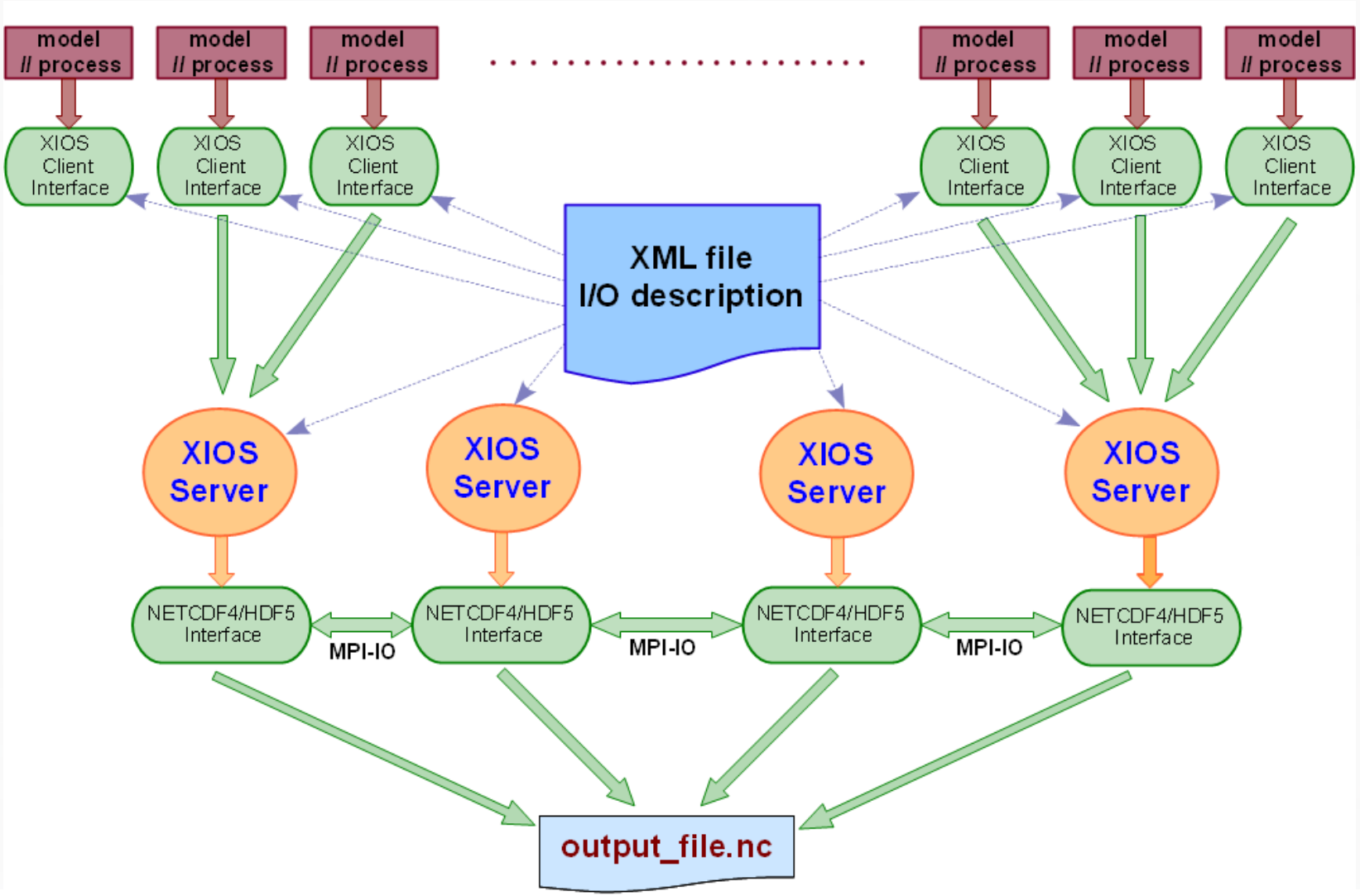
Approche actuelle : la bibliothèque IOIPSL

- Sortie des fichiers au format netcdf, écrite en fortan.
- Gestion des calendriers, des fichiers de redémarrage et des sorties histoires.
- Gestion des opérations de moyennage temporel, minimum/maximum.

Très bon outils, mais souffre de quelques inconvénients :

- Manque de souplesse :
  - De (trop) nombreux paramètres lors des phases de définition et d'écriture due à la gestion des méta-données.
  - Nécessité de conserver de nombreux indices (handle) liés aux fichiers ou aux variables => concentration des appels I/O.
  - Répétitions non nécessaires de nombreux paramètres.
  - Nécessité de recompiler lors de chaque modification de paramètres I/O.
- Problèmes de performance
  - Aucune gestion du parallélisme ou du multi-threadisme.
  - 1 fichier par processus MPI, les fichiers doivent être reconstruits en post-traitement.
  - De gros problèmes de performances lors du passage à l'échelle pour les sorties et les reconstructions des fichiers.

- La nouvelle approche : XIOS (XML-IO-SERVER)



2 principaux objectifs

Flexibilité et souplesse

- Externalisation de la description des I/O dans un fichier XML.
  - Gestion hiérarchique avec concept d' héritage.
  - Définitions plus simples et plus compactes.
  - Évite les répétitions inutiles.
- Simplification de la gestion des I/O au niveau du code.
  - Minimisation des appels liés aux définitions des I/O.
  - Minimisation du nombre d' arguments des appels.
- Écriture d' un champ : un identifiant et la donnée
  - `CALL xios_send_field("field_id", field)`
- Permet de modifier la définition des I/O sans recompiler.
  - Tout est dynamique, le fichier XML est analysé à l' exécution.

XIOS

- Performance
  - L'écriture des données ne doit pas impacter l'exécution du code.
  - Utilisation d'un ou plusieurs serveurs exclusivement dédiés au I/O.
    - Transfert asynchrone des données des clients vers les serveurs.
    - Écriture indépendante et asynchrone des données par chaque serveur.
  - Utilisation des systèmes de fichiers parallèles via netcdf4/hdf5 => MPI_IO.
    - Écritures simultanées de plusieurs processus dans un même fichier.
    - Plus de phase de reconstruction en post-traitement.

- Historique du développement
  - Fin 2009 : Démonstration de faisabilité : XMLIO/SERVER
    - Totalement écrit en fortran 90.
    - Implémente les fonctionnalités XML et client/serveur.
  - Mi-2010 – fin 2011 : réécriture complète en C++ => XIOS
    - Projet Européen IS-ENES.
    - Programmation orientée objet.
      - interopérabilité C++/C/Fortran à travers la norme Fortran 2003.
    - 25000 lignes de codes sous SVN.

Description XML - « Hello Word »

```
<simulation>
  <context id="hello_word" calendar_type="Gregorian" start_date="2012-02-27 15:00:00">

    <axis_definition>
      <axis id="axis_A" value="1.0" size="1" />
    </axis_definition>

    <domain_definition>
      <domain id="domain_A" />
    </domain_definition>

    <grid_definition>
      <grid id="grid_A" domain_ref="domain_A"  axis_ref="axis_A" />
    </grid_definition>

    <field_definition >
      <field id="field_A"  operation="average" freq_op="1h" grid_ref="grid_A" />
    </field_definition>

    <file_definition type="one_file" output_freq="1d" enabled=".TRUE.">
      <file id="output" name="output_file">
        <field field_ref="field_A" />
      </file>
    </file_definition>

  </context>
</simulation>
```

Interfaçage fortran

- L'arborescence XML peut être crée ou complétée grâce à l'API fortran.
  - ex: ajout d'attributs au champ « toce »

```
CALL xios_set_field_attribut(id="toce",long_name="Temperature", unit="deg C", enabled=".TRUE.")
```

  - Les champs sont envoyés à chaque pas de temps

```
CALL xios_send_field(id="field_A",field_A)
```

Héritage

Principe d' héritage

- Héritage parent->enfant via les groupes

```
<field_definition level="1" prec="4" operation="average" enabled=".TRUE.">
  <field_group id="grid_W" domain_ref="grid_W">

    <field_group axis_ref="depthw">
      <field id="woce"       long_name="ocean vertical velocity"            unit="m/s" />
      <field id="woce_eff"   long_name="effective ocean vertical velocity"  unit="m/s" />
    </field_group>

    <field id="aht2d"        long_name="lateral eddy diffusivity"           unit="m2/s" />

  </field_group>
</field_definition>
```

- Héritage par référence

```
<field id="toce" long_name="temperature (Celcius)" unit="degC" grid_ref="Grid_T" />
<field id="toce_K"  field_ref="toce" long_name="temperature (Kelvin)"  unit="degK" />
```

Fonctionnalités Client/Serveur

- "Serveur" XIOS :
  - Pool de processus MPI dédiés aux I/O.
  - Les "clients" sont les processus MPI des codes de calcul.

- Chaque code client communique avec les serveurs via la notion de "context"
  - Chaque pool client dispose de son propre communicateur MPI.
  - Un inter-communicateur MPI est crée entre le pool de clients et le pool de serveurs.
  - A chaque inter-communicateur est associé un "context"

- Véritable notion de "service" MPI
  - L'enregistrement est dynamique.

```
CALL xios_context_initialize("context_id", comm)
```

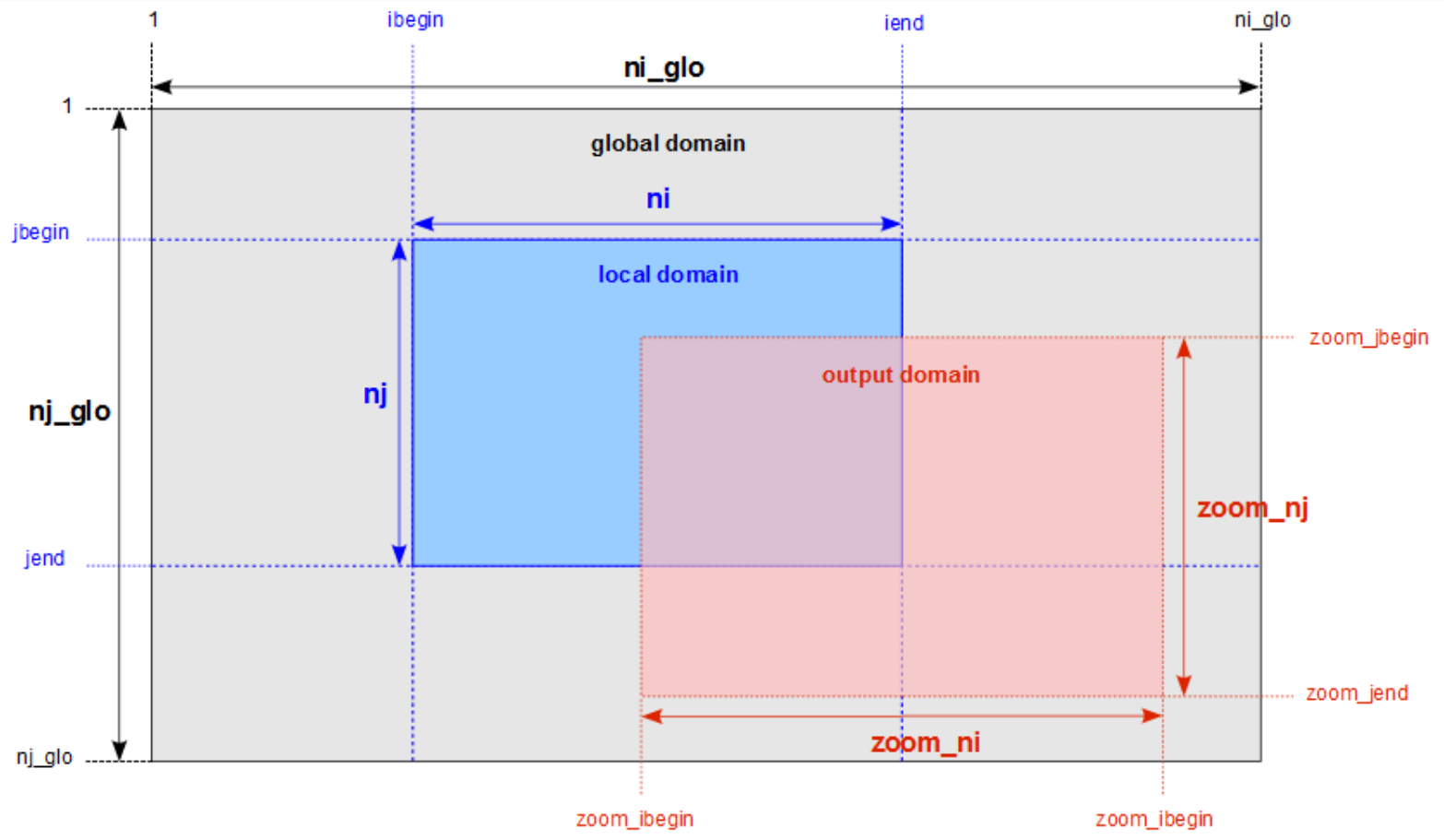
  - Un même pool de serveur peut gérer plusieurs pools de clients.

- Idée maîtresse : les codes de calculs ne doivent pas être impactés par les IOs
  - Communications point à point synchrones non bloquantes entre clients et serveurs.
  - MPI_ISSend, MPI_IRecv, MPI_Test, MPI_IProbe...
  - Permet le recouvrement Calcul/Communication/Écritures.
  - Utilisation de buffers pour lisser les pics d'I/O.

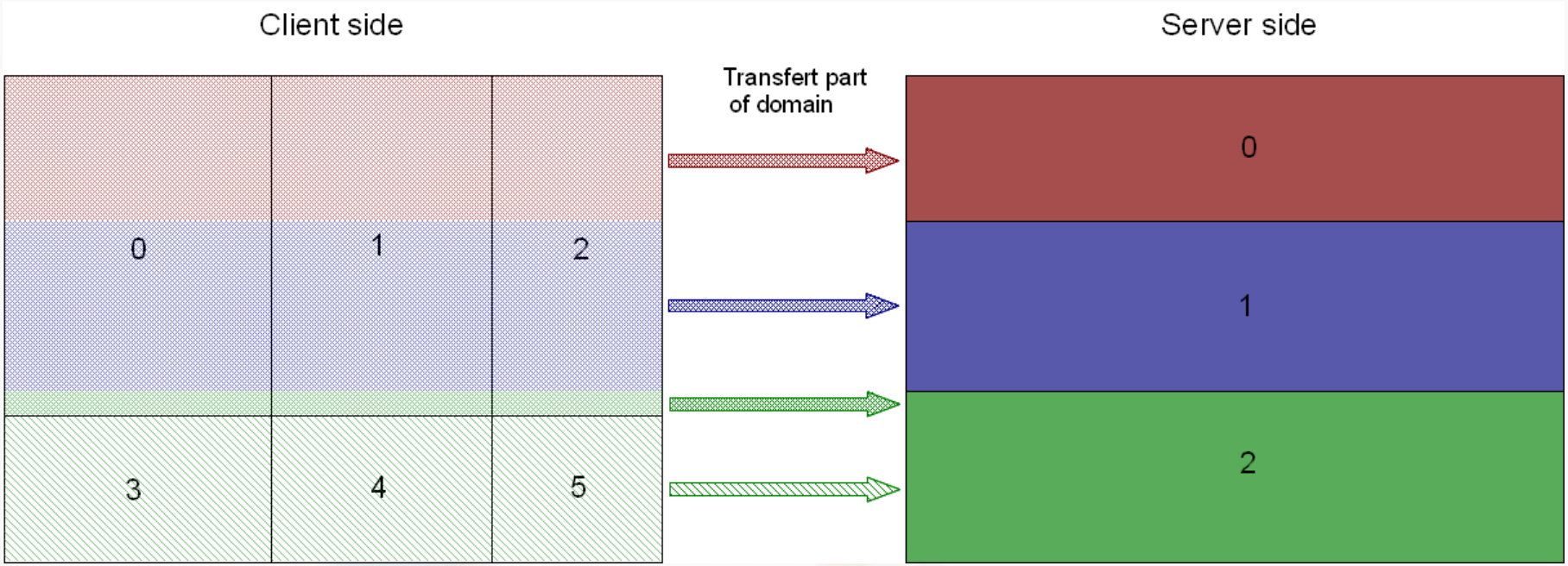
- Les communications entre clients et serveurs utilise des principes de programmation RPC (Remote Proceduring CALL) à travers MPI
  - Un message est auto-descriptif.
  - Un message est rempli coté client en empaquetant les arguments et les données.
  - Quand le message est reçu coté serveur, l'en-tête est analysée et le message est acheminé à destination.
  - Le message est ensuite dépaqueté puis la méthode correspondante est appelée.

Coté client : Distribution des données

- Domaine global : ni_glo x nj_glo.
- Domaine local : ni x nj.



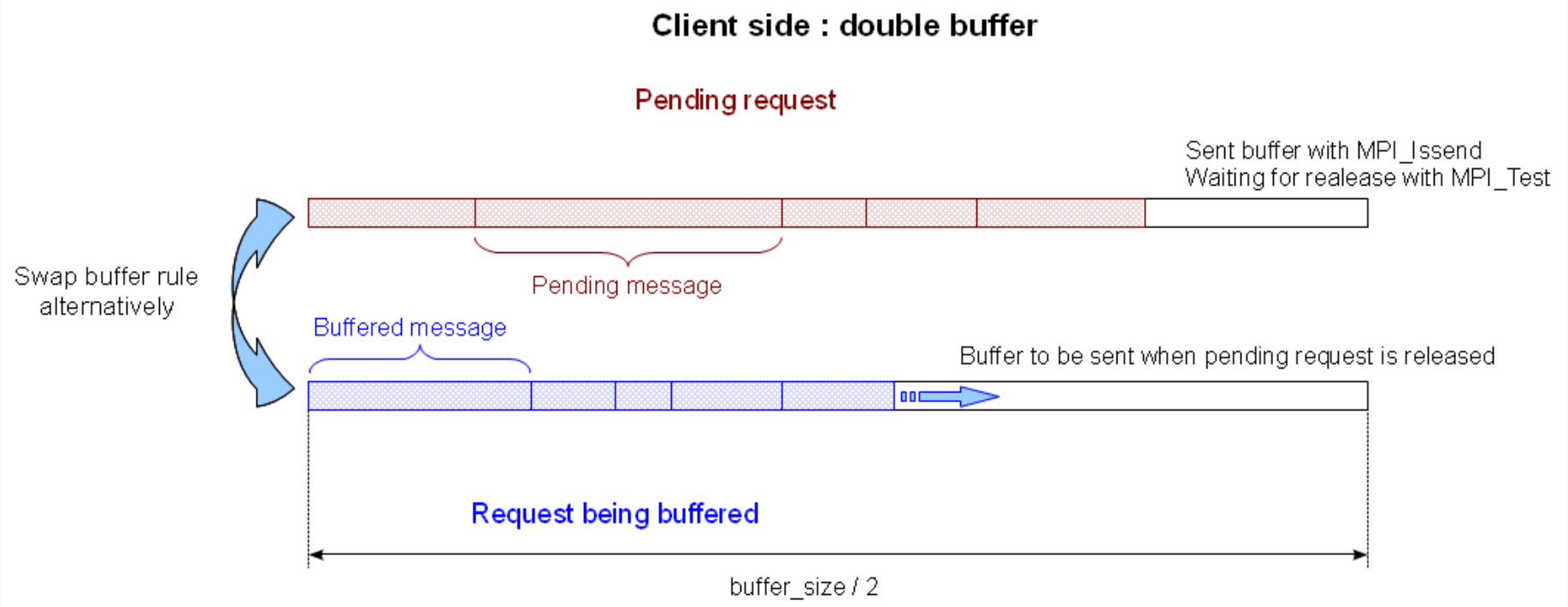
Coté serveur



- Les clients {0, 1, 2} envoient leur part de domaine aux serveurs {0, 1, 2}.
- Les clients {3, 4, 5} envoient leur part de domaine au serveur 2.

- La distribution des données coté serveur est équi-répartie suivant la seconde dimension.
- Un client peut communiquer avec plusieurs serveurs.
- Un serveur peut recevoir des données de plusieurs clients.

Coté client : double buffers



- Les messages sont concaténés puis envoyer en une seule requête MPI lorsque la précédente requête est reçue.
- Communications non bloquantes : MPI_Issend/MPI_Test.
- Lorsque les buffers sont pleins => on rentre en mode bloquant.

Protocole de communication

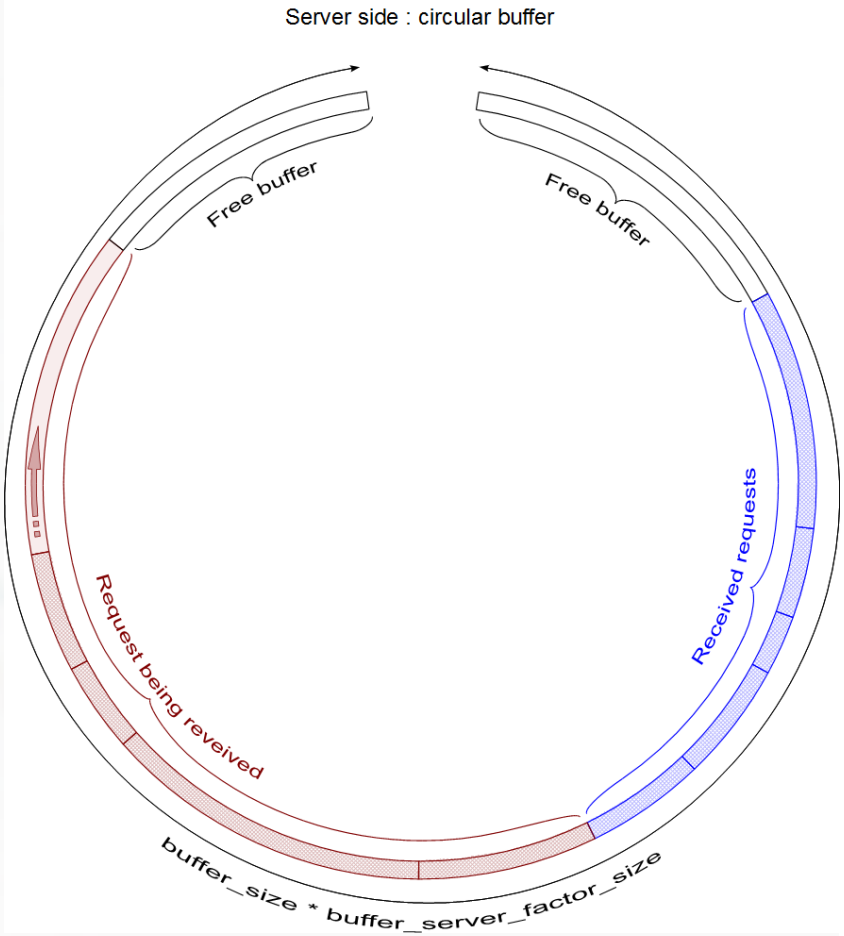
Coté serveur : buffer circulaire

- Arrivé d'un message => MPI_Iprobe.
- Réception : MPI_Irecv / MPI_Test.

- On reçoit en priorité les messages afin de libérer les buffers coté client.

- On traite les messages et effectue les écritures lorsque qu'il n'y a plus de requête en attente de réception.

- Lorsque les buffers sont pleins, on passe en mode bloquant :
  - Traitement des messages pour libérer les buffers.
  - Ajout de serveur XIOS pour alléger la charge.



Sortie au format NETCDF4/HDF5

- Utilisation des écritures parallèle via HDF5//
  - MPI/IO en bout de chaîne.

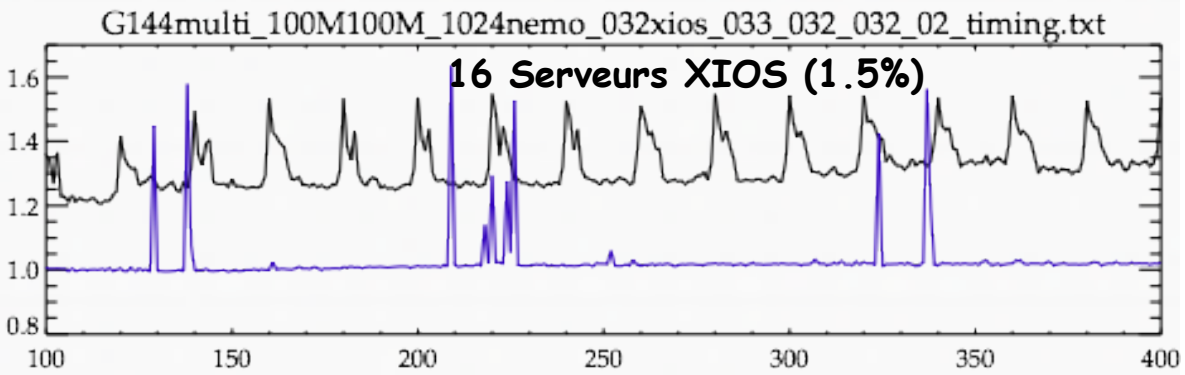
2 options possibles : "multiple_file" ou "one_file"

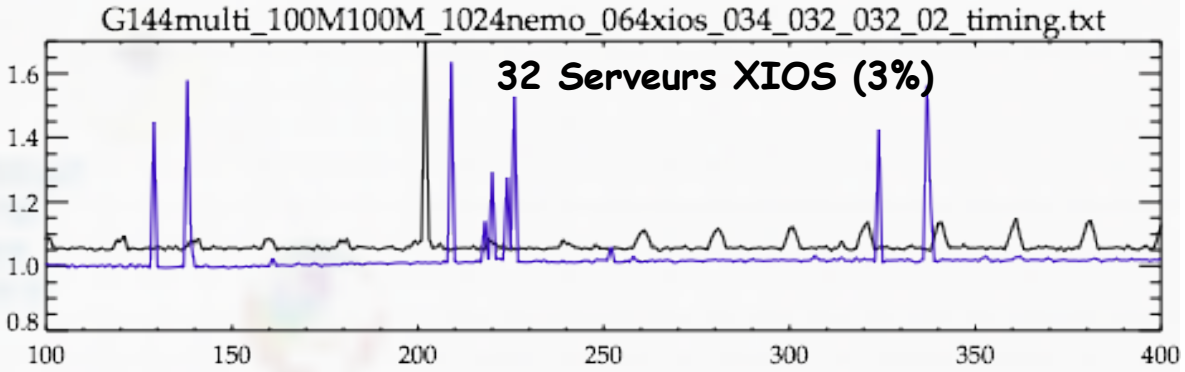
- multiple_file : un fichier par serveur.
  - Pas d'écriture parallèle.
  - Phase de reconstruction nécessaire en post-traitement.
- one_file : un fichier unique.
  - Utilisation des écritures parallèles, accès indépendants ou collectifs.

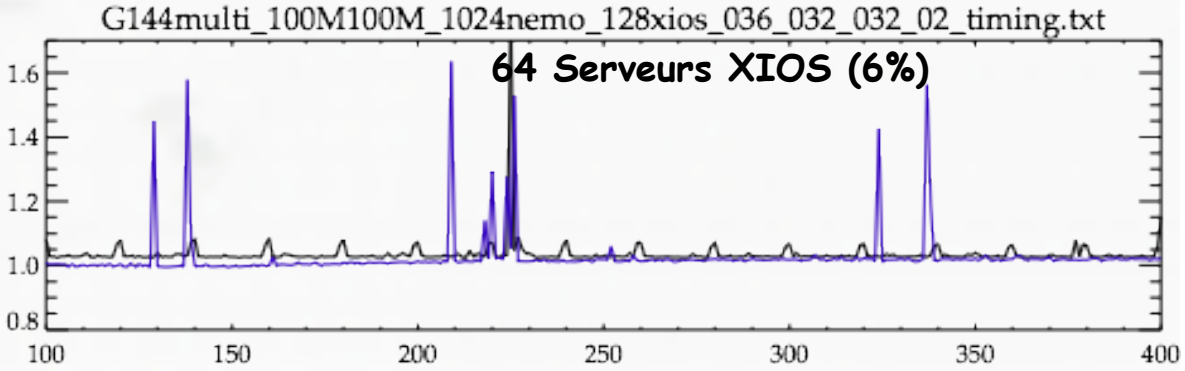
Cas test : XIOS/NEMO

- Configuration GYRE 144 (4322x2882)
  - pdt=180s, 720 pdt (36 h)
  - NEMO : 1024 proc. MPI
- Temps par itération
  - **Sortie horaire**
  - **Sans IO**
  - 246 Go écrit en 4 fichiers

```
62G Feb 25 04:36 BIG1h_st32_1h_grid_T.nc
28G Feb 25 04:37 BIG1h_st32_1h_grid_U.nc
26G Feb 25 04:31 BIG1h_st32_1h_grid_V.nc
130G Feb 25 04:35 BIG1h_st32_1h_grid_W.nc
```

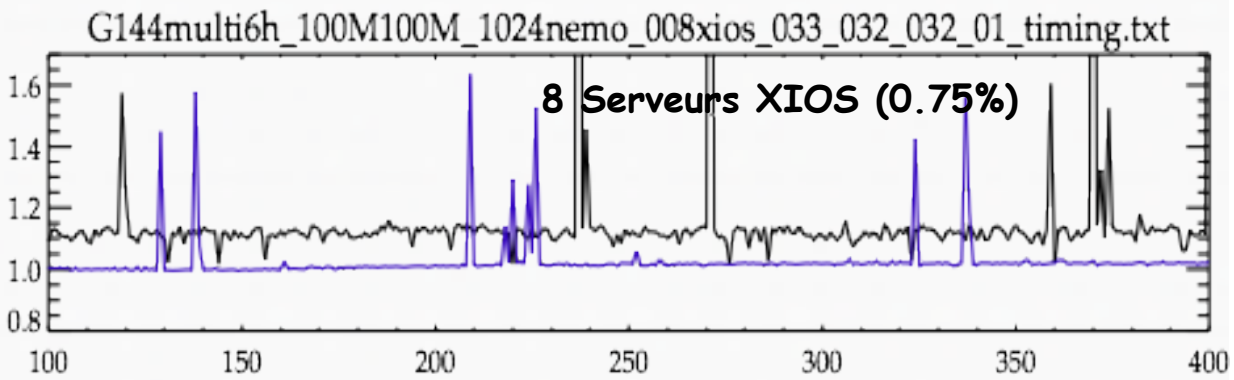


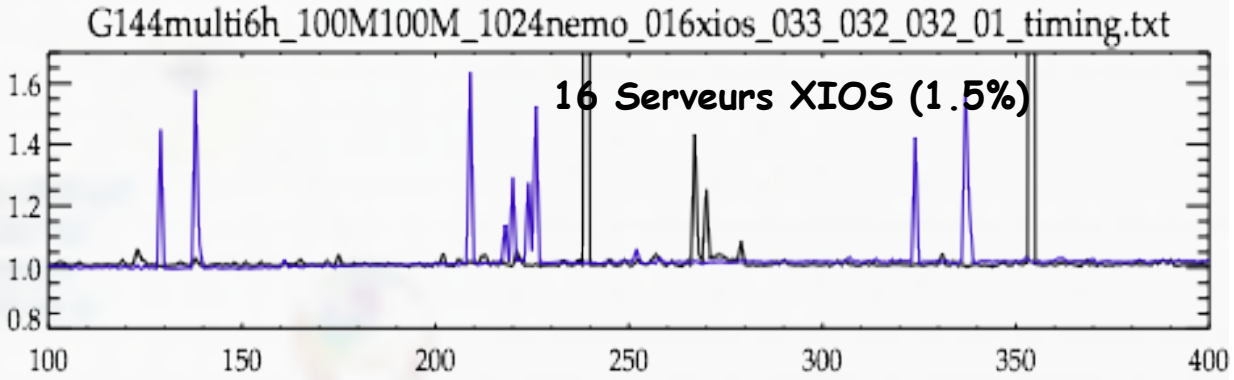


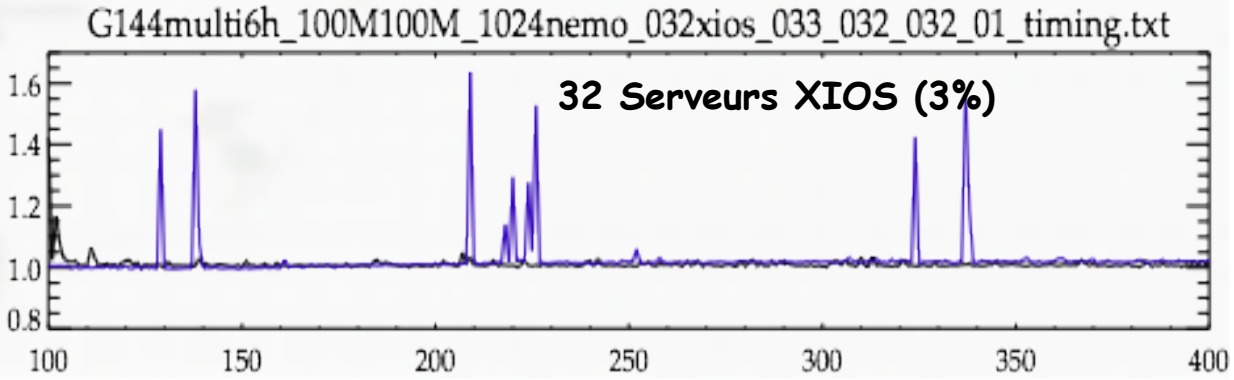


Cas test : XIOS/NEMO

- Cas test plus léger
  - Sortie à 6h
  - Sans IO

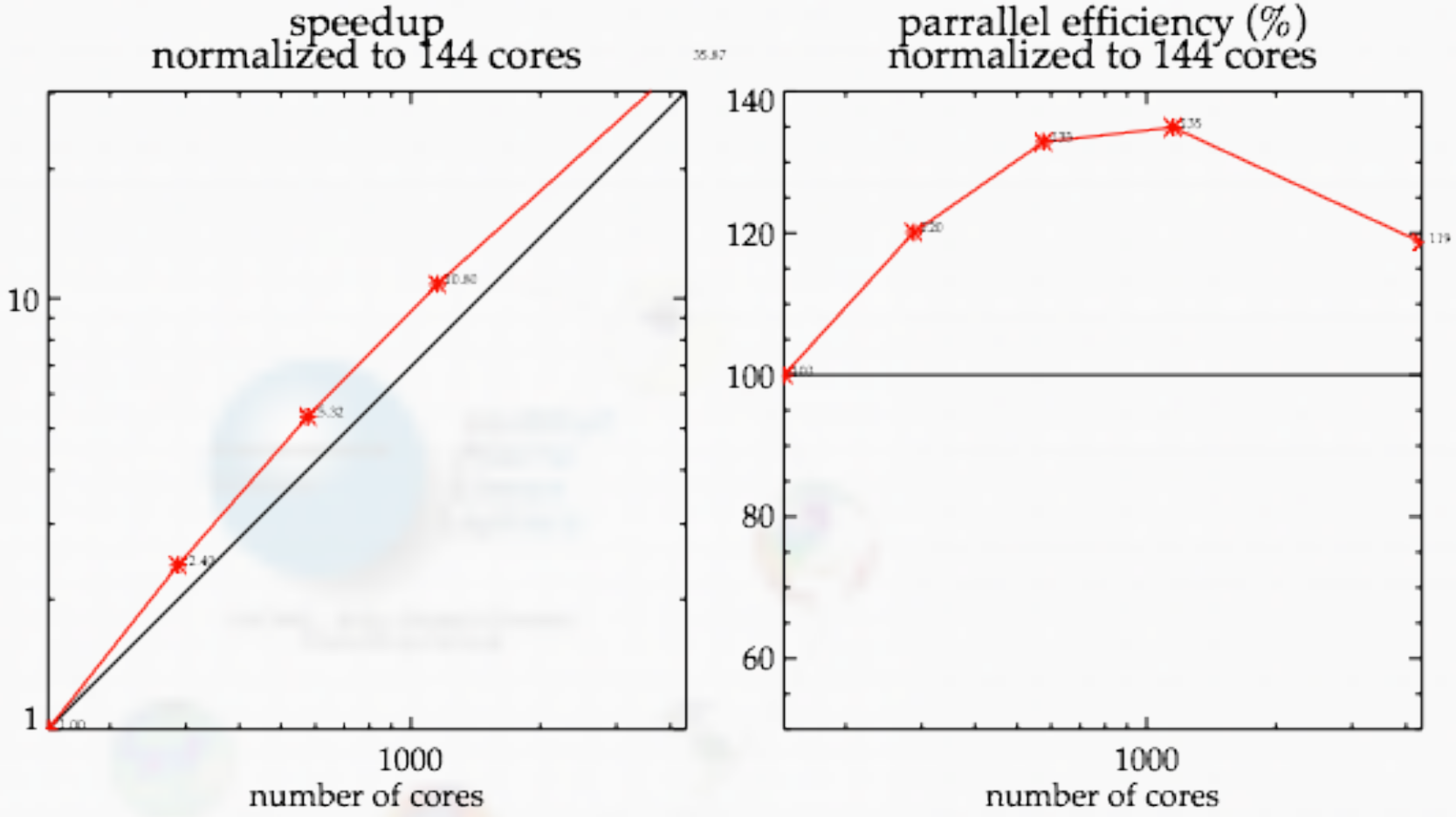




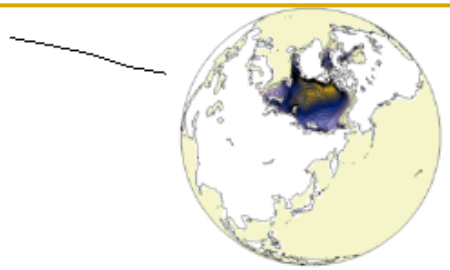


Cas test : XIOS/NEMO : speed-up

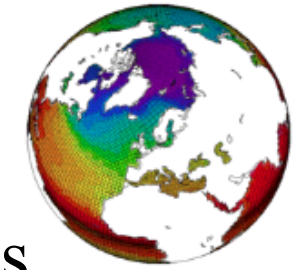
CURIE Fat Nodes: NEMO 3_4_b GYRE Big IO multi_file, jp_cfg = 144



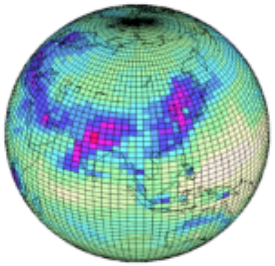
ncores:	144	288	576	1152	4352
timing:	8000	3330	1505	741	223

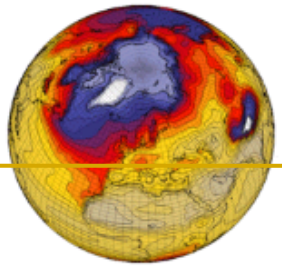


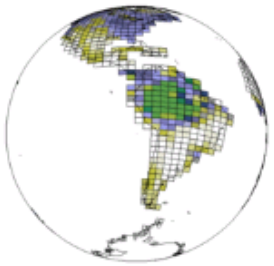
S.Denvil, A.Bhardwaj, F.Corsini, M.Morgan, H.Moulron, J.Raciazek, P.Weill, JL.Dufresne, K.Ramage, G.Wiber (CCRT), D.Gorecki (CCRT), F.Valley (CCRT), A.Ansari (IDRIS), PF.Lavallée (IDRIS), ESGF et al.



Dissémination de grandes masses de données (IPSL Climate Modeling Center)







Environmental Sciences
Institut Pierre Simon Laplace
IGCMG - IPSL Global Climate Modelling Group

Climate modeling data management requirements

- Move the data a minimum, keep them close to supercomputing centres if possible
- When data needs to be moved do it quickly and with a minimum amount of human intervention
- Keep a track of what has been produced, particularly what is on deep storage
- Exploiting a federation of sites (HPC centers, clusters, …)

CMIP5/AR5

Coupled Model Intercomparison Project (5)

- Coordination de l'aspect données
  - PCMDI: Communauté CMIP5
  - BADC et WDCC: Communauté Climat Européenne et IPCC working group 2 et 3 ("mandat" IPCC-DATA.ORG)
- 3 Décembre 2008: MoU Tripartite (PCMDI, BADC, WDCC)
- > 20 groupes de modélisation du climat
- > 45 modèles du système climatique.
- > 50 expériences numériques
- > 86 simulations/modèle pour satisfaire aux expériences
- > 6500 ans de simulation
- > 1 Po de données actuellement (130 To de l'IPSL)
- 1.5 Po (3 Po) envisagés d'ici 1 an (2 ans)

The Earth System Grid Federation (ESGF) Collaboration

- United States:

  ANL (Argonne National Laboratory)

  LBNL (Lawrence Berkeley National Laboratory)

  LLNL/PCMDI (Lawrence Livermore National Laboratory)

  NASA/JPL (NASA / Jet Propulsion Laboratory)

  NCAR (National Center for Atmospheric Research)

  ORNL (Oak Ridge National Laboratory)

- Europe:

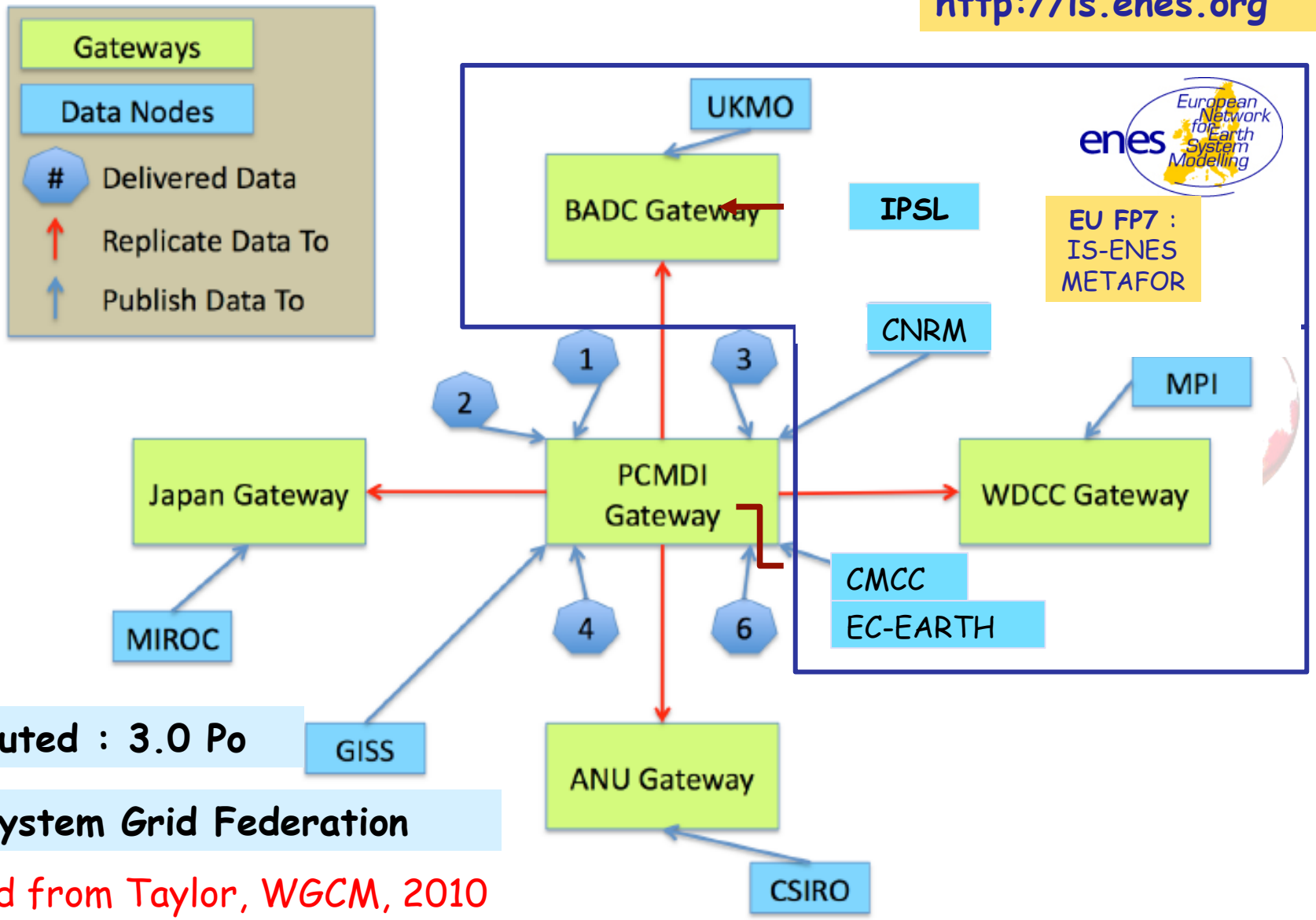
  BADC (British Atmospheric Data Center)

  CMCC (Centro Euro-Mediterraneo per i Cambiamenti Climatici)

  DKRZ (Deutsches KlimaRechenZentrum)

  IPSL (Institut Pierre Simon Laplace)

http://is.enes.org

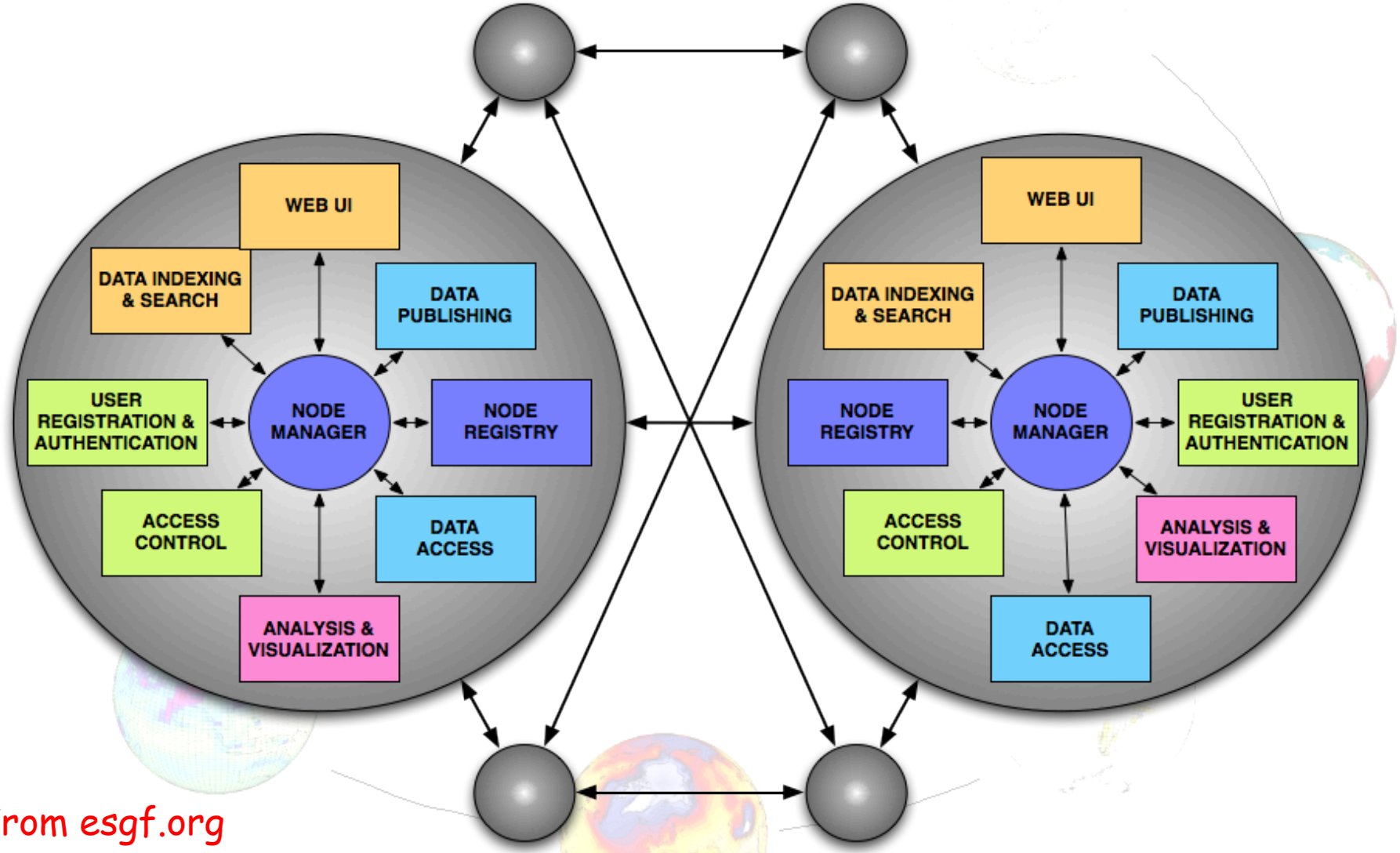


Distributed : 3.0 Po

Earth System Grid Federation

Adapted from Taylor, WGCM, 2010

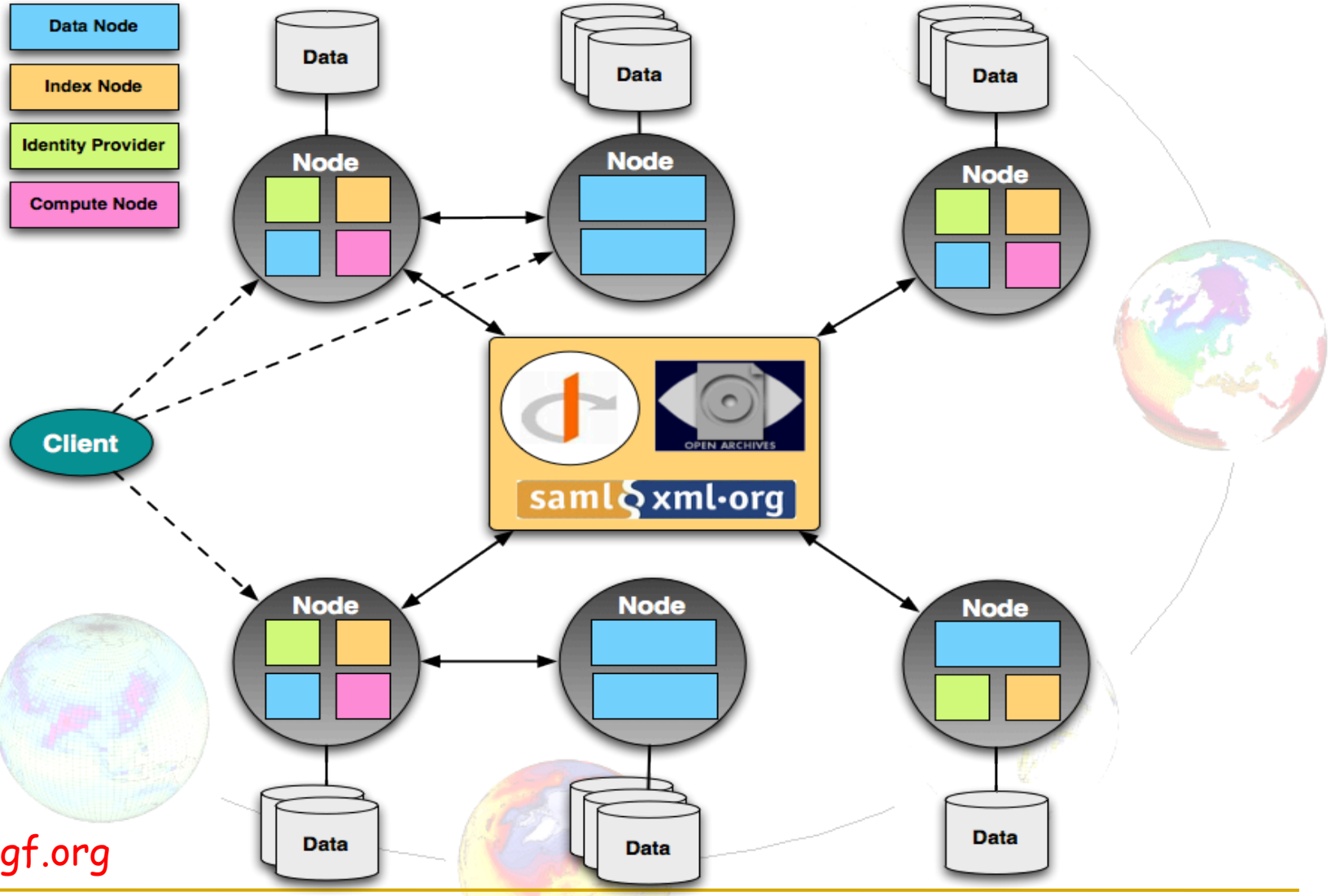
Software infrastructure (p2p)



From esgf.org

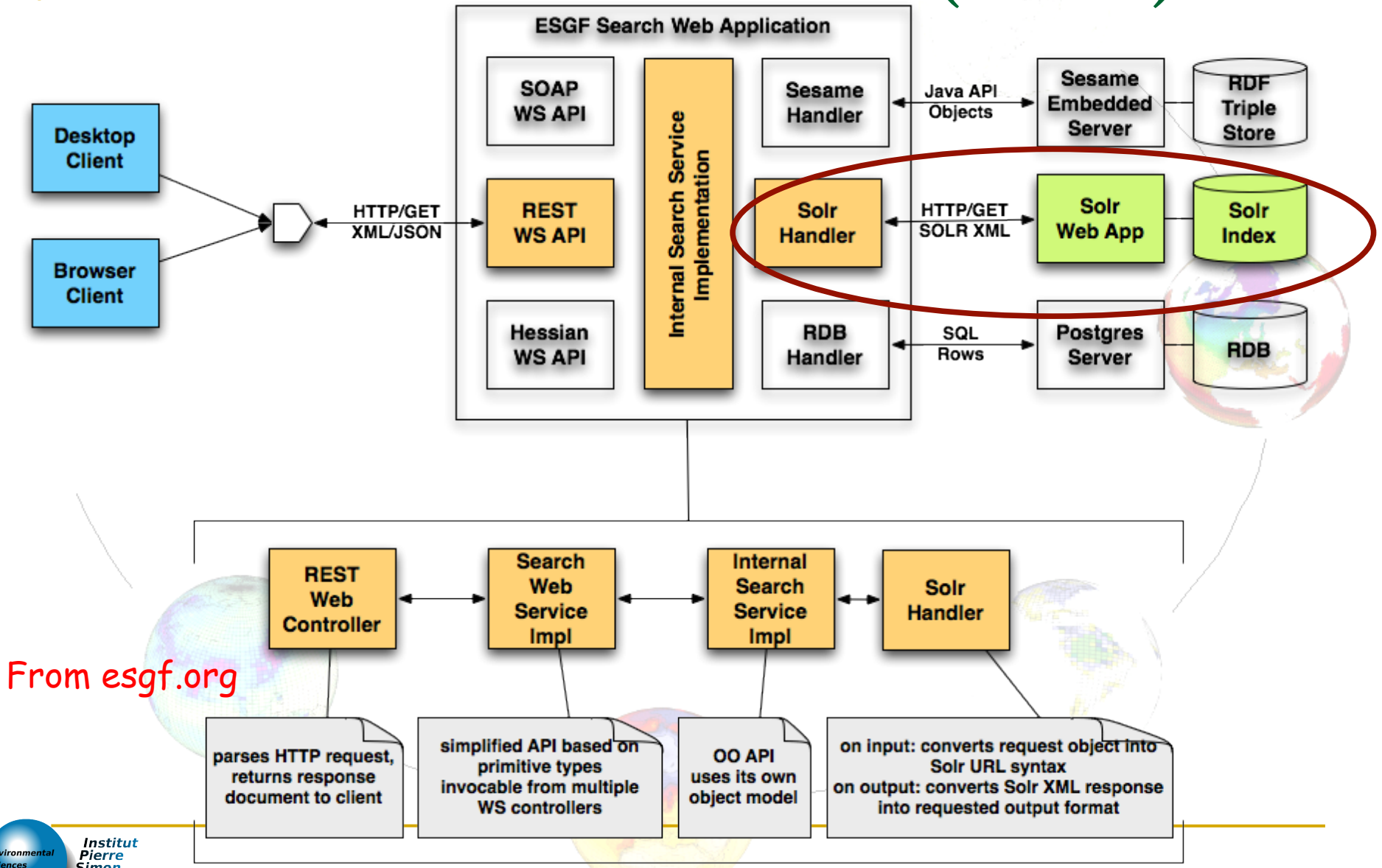
Environmental Sciences
Institut Pierre Simon Laplace
IGCMG - IPSL Global Climate Modelling Group

Software infrastructure (sso)



From esgf.org

Software infrastructure (search)



From esgf.org

État des lieux CMIP5 : IPSL

- **Volume (To) / Nombre de fichiers téléchargés**

Avril 2011	: 1.26 To / 777 fichiers
Mai 2011	: 2.96 To / 1855 fichiers
Juin 2011	: 16.2 To / 7274 fichiers
Juillet 2011	: 1.51 To / 2028 fichiers
Août 2011	: 13.9 To / 8627 fichiers
Septembre 2011	: 5.9 To / 1946 fichiers
Octobre 2011	: 11.1 To / 8758 fichiers
Novembre 2011	: 22.5 To / 17238 fichiers
Décembre 2011	: 36.7 To / 15402 fichiers
Janvier 2012	: 41.2 To / 23 589 fichiers

Total : 154.7 To / 89302 fichiers

Nombre d'utilisateurs ayant demandé un téléchargement : **1135**

Synchro-Data

CMIP5/ESGF data access tool client

IPSL - ESGF - EGI

J.Raciazek, S.Denvil

Contact

Jripsl (AT) ipsl.jussieu.fr

Overview

- Search for CMIP5 data in ESGF

  File selection using DRS facets (realm, freq, experiment, ensemble, variable). So called template.

  Incremental search (keep track of what have already been downloaded)

- Metadata analysis

  Compute total files size

  Check if all variables get a match in ESGF

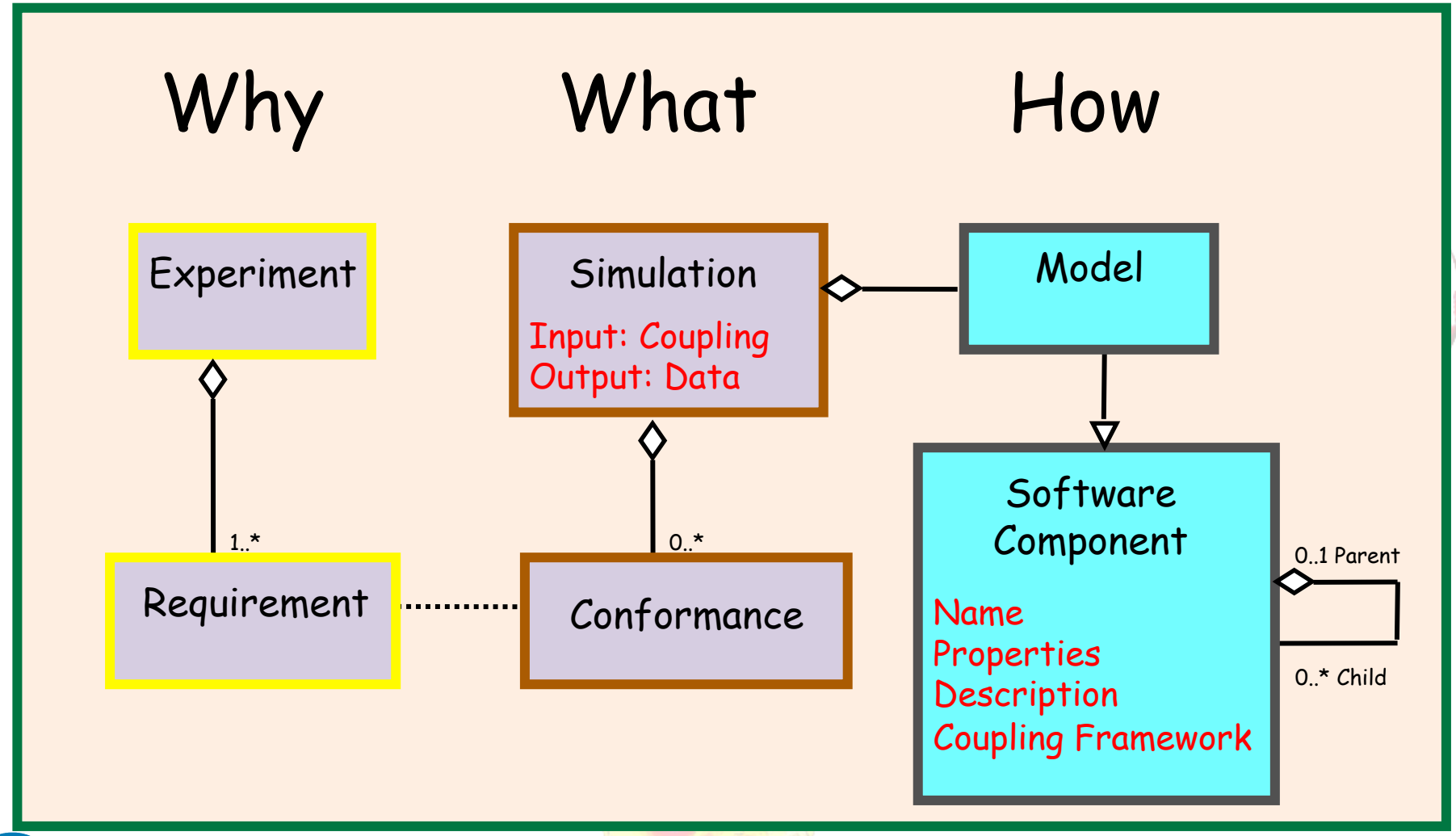
- Transfer files from ESGF to local filesystem

  Transparent handling of x509 Certificate based security

  HTTP Parallel download

Une simulation climatique

http://metaforclimate.eu/trac/browser/CIM/tags/version-1.5



ES-DOC

Earth **S**cience **Doc**umentation

OBJECTIF	Diffusion internationale de métadonnée **CIM**
HISTOIRE	Extension du projet Européen Metafor (WP4)
COMMENT	Développement d'outils et de services “open source”
LICENSE	CeCILL-GPL

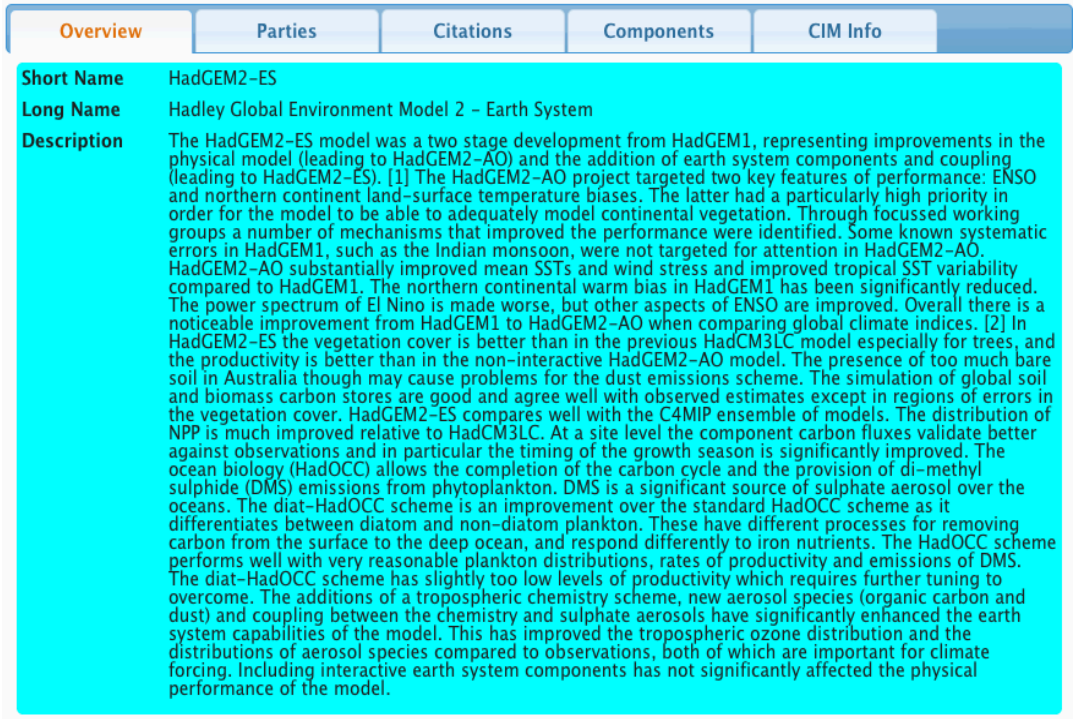
Environmental Sciences — Institut Pierre Simon Laplace

IGCMG - IPSL Global Climate Modelling Group

metafor

ES-DOC

CIM HTML Viewer



Projects @ 01/03/2012
ESGF-P2P Node Front End
KNMI Impacts Portal
IPSL Prodiguer Portal

Technology
Javascript / HTML
Web Services (CIM Portal)
JSON

Environmental Sciences — Institut Pierre Simon Laplace

IGCMG - IPSL Global Climate Modelling Group

metafor

Perspectives CMIP à 10 ans

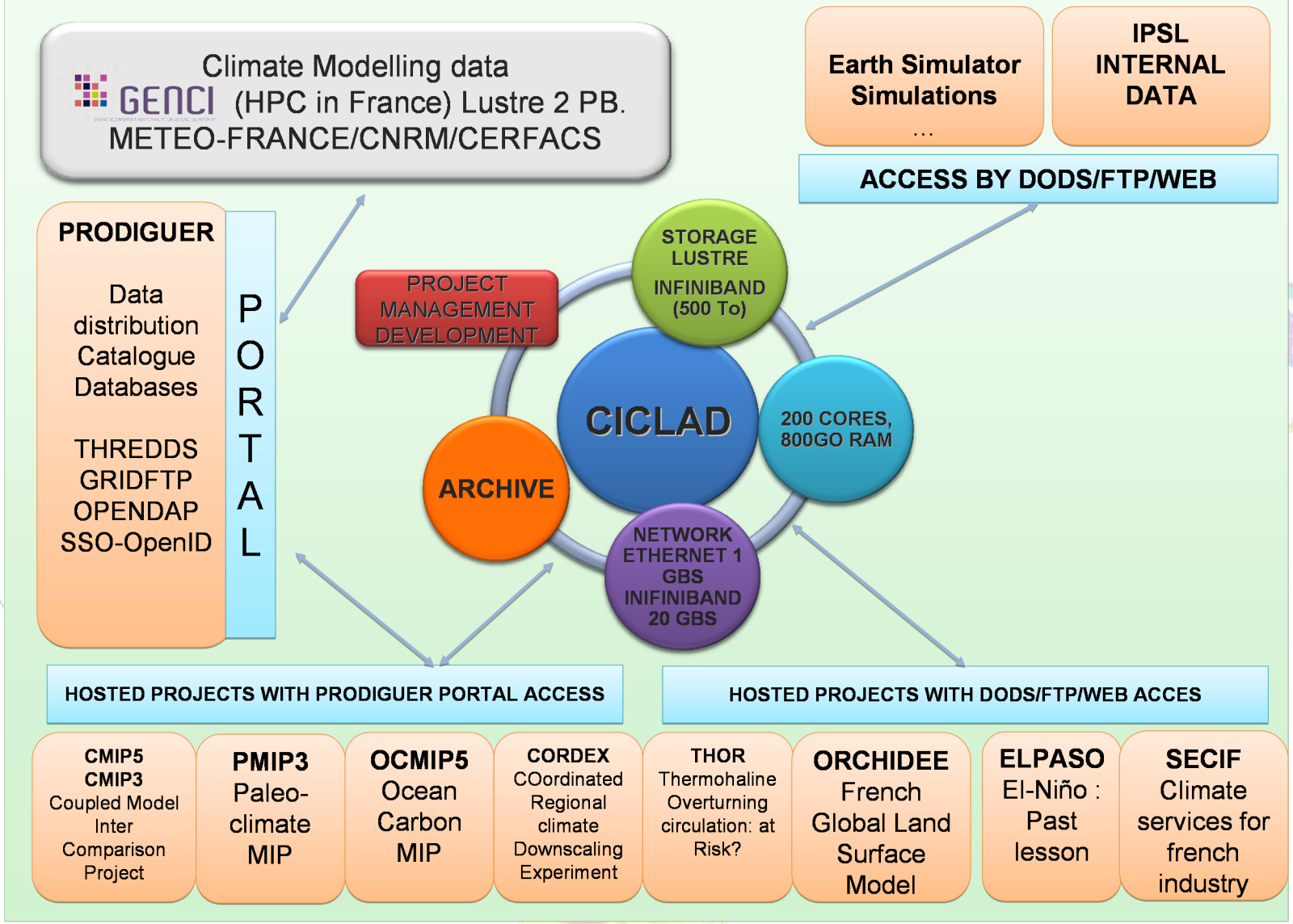
	CMIP5	CMIP6	CMIP7
Year	2012	2017	2022
Power factor	1	30	1000
Npp	200	357	647
Resolution [km]	100	56	31
Number of mesh points [millions]	3,2	18,1	108,4
Ensemble size	120	214	388
Number of variables	800	1068	1439
Interval of 3-dimensional output (hours)	6	4	3
Years simulated	90000	120170	161898
Storage density	0,00002	0,00002	0,00002
Distributed Archive Size (Pb)	**3,19**	**86,05**	**2260,20**

Résumé

- Le processus de génération de données en Modélisation du Climat a besoin de

- Centre de calculs (Peta**flops**)
- Centre de traitements (Peta**scale**)
- Centre de données (Peta**scale**)
- Interconnections Réseaux (Peta**scale**)
- Déployer/maintenir des **couches logicielles**
- Distribuer les résultats à une **large communauté**
- Suivre des **normes/standards** internationaux

IPSL e-infrastructure to sustain climate analysis



Calcul/Stockage/Analyse

- **Calcul : ciclad.ipsl.jussieu.fr**

  Cluster de calcul (20 nœuds dual quad-core AMD)

  « Climate analysis enabled » netcdf, cdo, ferret, matlab…

- **Stockage : /prodigfs/esg/CMIP5/merge**

  Sous ensemble CMIP5 (320 To → 440 To avril 2012)

- **Support à l'analyse : Prodiguer**

  Un outil puissant de réplication CMIP5

  Un data node «maison» (accès opendap sur agrégation) (année 1→10 + 11→20 = 1→20)