Le Projet XMLIO-SERVER

Description du projet

Le projet

- Évolution de la bibliothèque d'Entrée/Sortie de l'IPSL : IOIPSL

- XMLIO-SERVER est dédié à la gestion des fichiers d'historiques :
  - Champs instantanés, moyennes journalières, mensuelles …

- Soumission dans le cadre d'IS-ENES
  - Maquette fonctionnelle : V1 (Y. Meurdesoif, A. Caubel)
  - Développement de la bibliothèque et recodage : V2
    - 18 mois de CDD : H. Ozdoba.

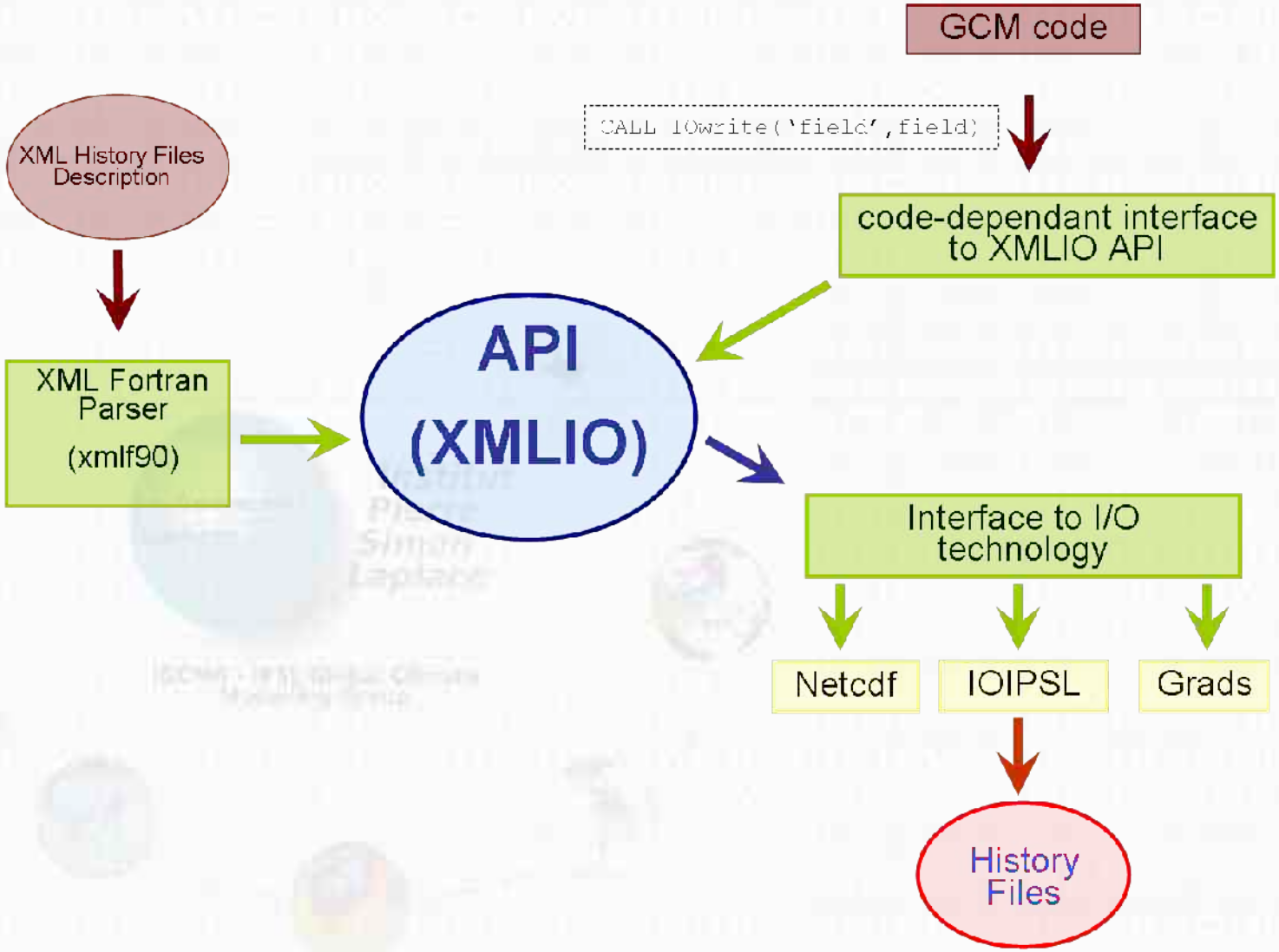
- Modèle impliqué : NEMO
  - Utilisé en standard sous SVN
  - Exercice AR5 utilisant XMLIO-SERVER

Description du projet

- Objectifs :
  - Flexibilité des I/Os : XMLIO
    - Interface minimaliste au niveau des codes
      - Minimiser les appels dans le code.
      - Minimiser le passage d'arguments.
      - Appelable à n'importe quel niveau du code sans avoir à stocker d'index (type netcdf)

        ex : `CALL iowrite('tsol', tsol)`
      - Interface pérenne, non sujette aux modifications/améliorations de la bibliothèque d'I/O.
      - Branchement transparent vers de nouvelles technologies/formats d'I/O.
    - Description externe de l'information sur les I/Os
      - Déclaration des champs à sortir (unité, nom, description, fréquence, fichier de sortie
      - Format facilement extensible pour les ajouts futurs (=> XML)
      - Format facilement éditable, vision organisée, structurée et compacte des I/Os.
    - Potentiellement : Possibilité de construire une interface graphique utilisateur (GUI)
    - Possibilité de branchement vers différents formats/technologies I/O : NetCDF, NetCDF_par, HDF, Grads, bibliothèques I/O « maison » (ex IOIPSL),…

Fonctionnement de XMLIO



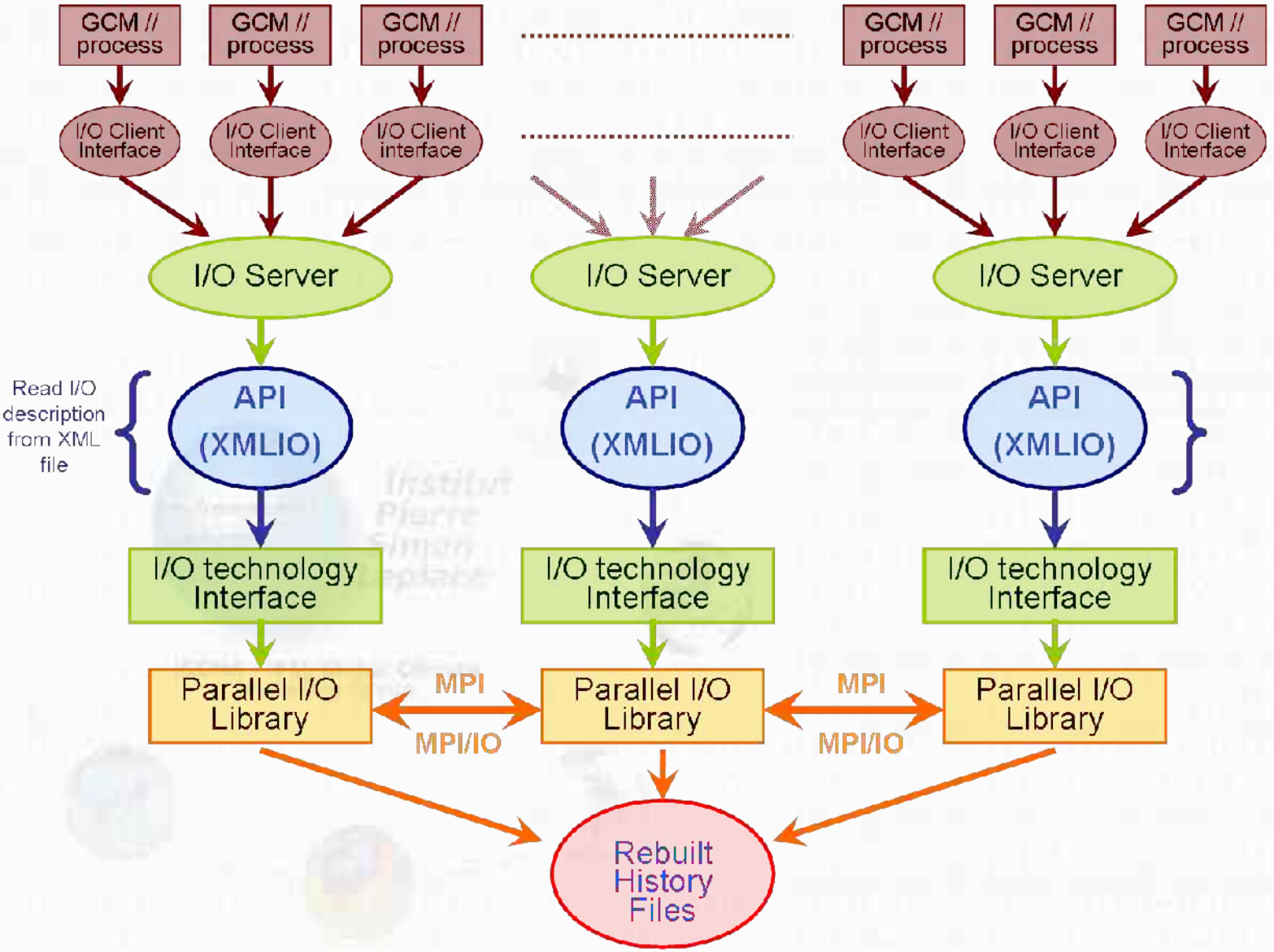
Fonctionnement en mode serveur

Performance et gestion du parallélisme : IOSERVER

- Aujourd'hui : les I/Os coûtent ~15% à 20% du temps de calcul, chaque proc. sort un fichier, reconstruction des fichiers en post-traitement.
- Sur un grand nombre de proc., on passe autant de temps à calculer qu'à reconstruire.

- I/O déportées/asynchrones de type « client/serveur »
  - Asynchronisme : pas de surcoût en temps de restitution.
  - Un serveur gère plusieurs (codes) client.
  - Pour les jobs massivement parallèles, plusieurs serveurs gèrent les IOs en fonction de l'équilibrage du ratio calcul/IO.
- Les serveurs gèrent les fichiers unifiés à l'aide de netcdf 4 (ou netcdf_par) en utilisant MPI/IO sur les systèmes de fichier parallèle.
  - la reconstruction des fichiers en post-traitement n'est plus utile.

Fonctionnement en mode serveur



Exemple de fichier de configuration XML

Définition des axes verticaux

```
<axis_definition>
    <axis id="deptht" description="Vertical T levels" unit="m" />
    <axis id="depthu" description="Vertical U levels" unit="m" />
</axis_definition>
```

Définition des grilles horizontales et des zooms

```
<grid_definition>
    <grid id="grid_T" description="grid T" />
    <grid id="grid_U" description="grid U" />
        <zoom id="a_zoom" ibegin="100" jbegin="100" ni="10" nj="10" />
    </grid>
</grid_definition>
```

Exemple de fichier de configuration XML

Définition des champs

```
<field_definition operation="ave(X)" freq_op="7200" >
    <group id="grid_T" axis_ref="none" grid_ref="grid_T">
        <field id="sosstsst" description="Sea Surface temperature"       unit="C"    />
        <field id="sosaline" description="Sea Surface Salinity"          unit="PSU"  />
        <field id="votemper" description="Temperature" axis_ref="deptht" unit="C"    />
    </group>

    <group id="grid_U"  axis_ref="depthu" grid_ref="grid_U">
        <field id="vozocrtx" description="Zonal Current"      unit="m/s"   />
        <field id="vozoeivu" description="Zonal EIV Current"  unit="m/s"   />
    </group>
</field_definition>
```

- Principaux attributs
  - name , description, unit
  - operation, freq_op
  - level, enabled
  - axis_ref, grid_ref …

Exemple de fichier de configuration XML

Définition des fichiers de sortie

```
<file_definition enabled=".TRUE.">
  <file id="GYRE_grid_T" description="sortie journalière" output_freq="86400" enabled=".FALSE."/>
      <field ref="votemper"  />
      <field ref="sosaline"  />
  </file>
  <file id="GYRE_grid_U" name="daily_U" description="sortie journalière" output_freq="86400"/>
      <group zoom_ref="a_zoom"
          <field ref="votemper"  enabled=".FALSE." />
          <field ref="vozoeivu"  />
      </group>
  </file>
</file_definition>
```

- Principaux attributs
  - name , description
  - output_freq
  - output_level, enabled

Au niveau du code …

Interface fortran

- Phase d'initialisation
  - Initialiser la bibliothèque (calendrier) puis lancer le parsing XML
  - Compléter le fichier XML en Fixant les attributs manquants.
  - Fin de la phase de définition : CALL close_io_definition
- Envoie des champs
  - passage à un nouveau pas de temps : CALL update_timestep
  - envoie des champs : CALL iowrite('field_id',field)
- Fin
  - CALL stop_ioserver
- Potentiellement : Possibilité de construire une interface graphique utilisateur (GUI)
- Possibilité de branchement vers différents formats/technologies I/O : NetCDF, NetCDF_par, HDF, Grads, bibliothèques I/O « maison » (ex IOIPSL),…