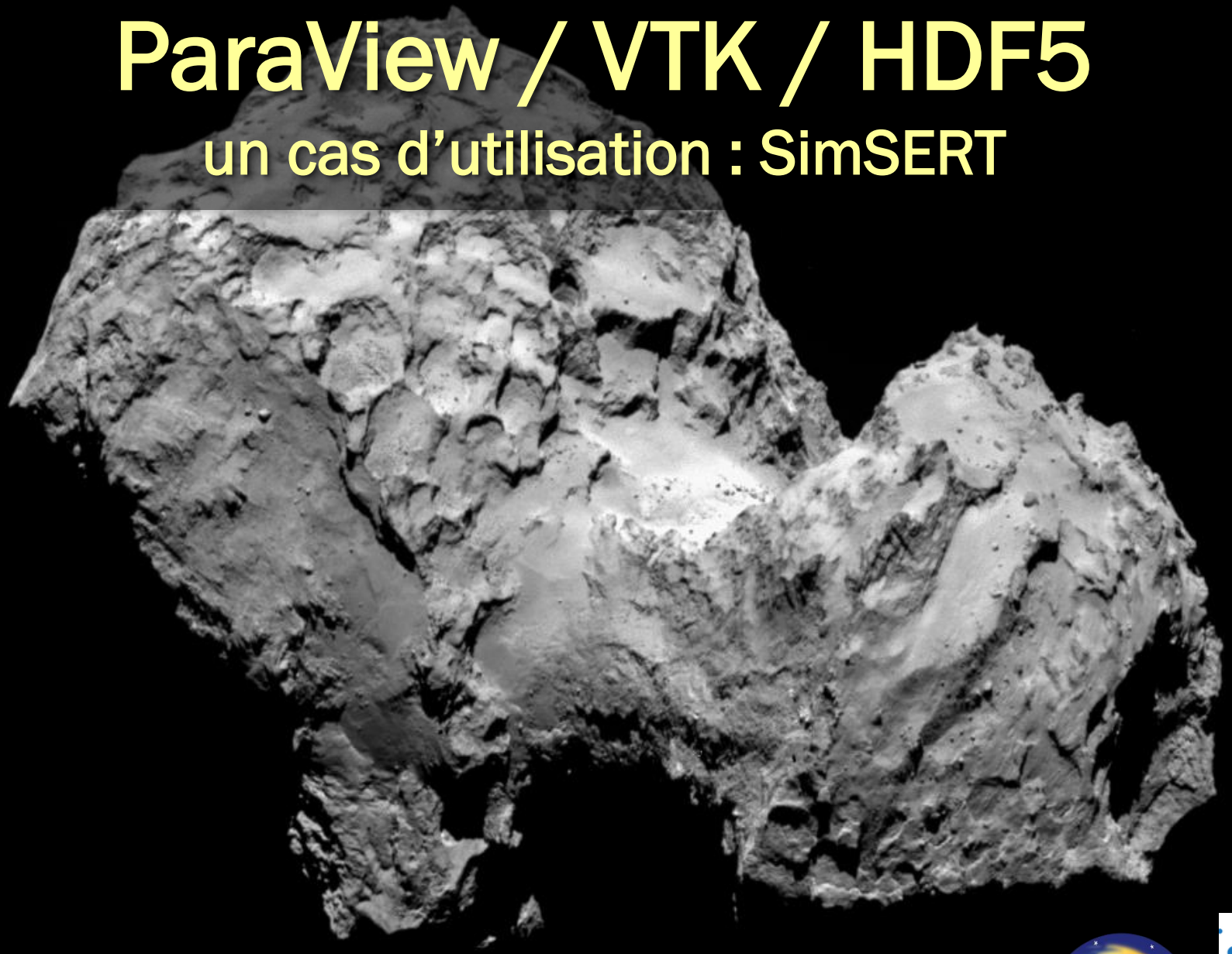
ParaView / VTK / HDF5

un cas d'utilisation : SimSERT



ParaView / VTK / HDF5

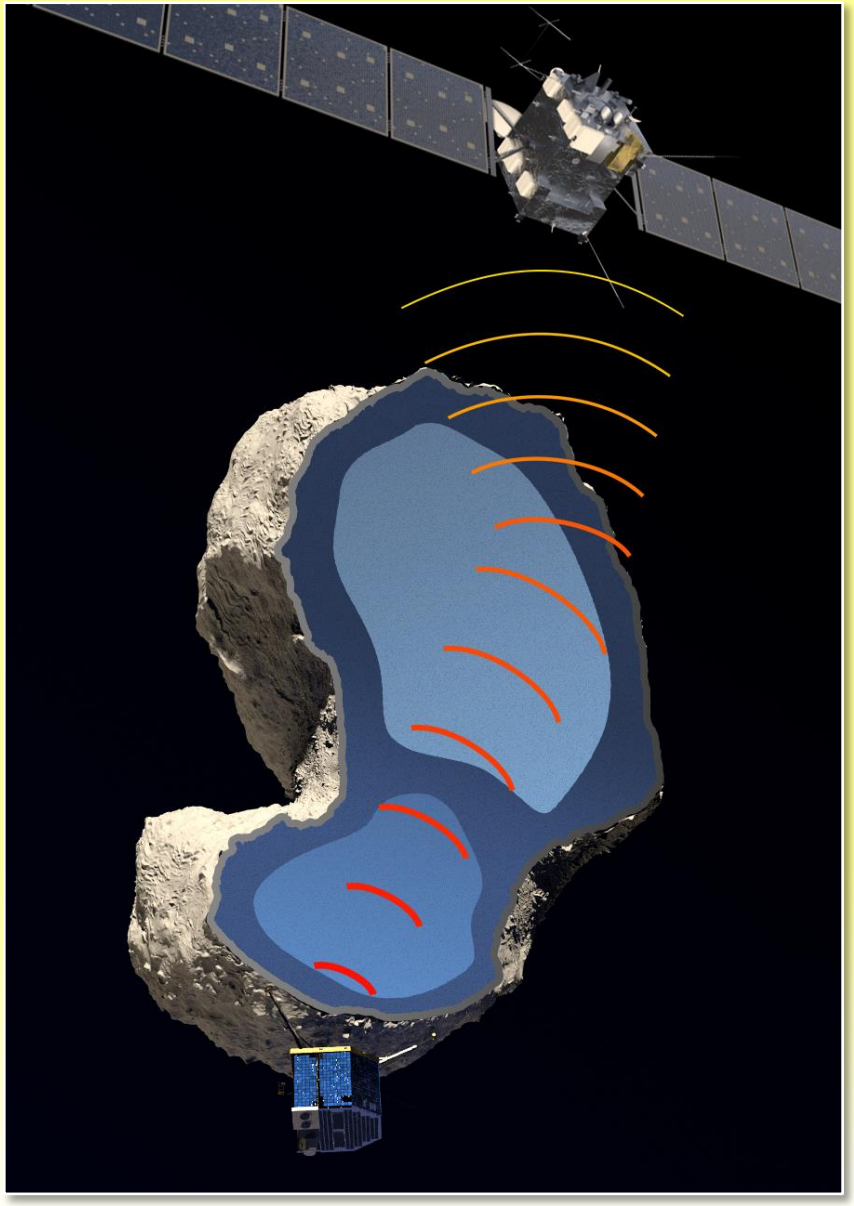
un cas d'utilisation : SimSERT

1. Contexte : CONSERT / ROSETTA

2. ParaView :
   Fonctionnalités, pipeline VTK
   Traitement et visualisation sur cluster
   Retour d'expérience

3. Structures de données :
   VTK et HDF5

CONSERT / ROSETTA - SimSERT



- COmet Nucleus Sounding by Radiowave Transmission
- SimSERT : Propagation radar en transmission sur un grand domaine
  => **ray-tracing**
- Besoin pour **l'analyse scientifique** et la préparation des **opérations**
- Visualisation et exploitation
  HDF5/VTK/ParaView/Qt

ParaView

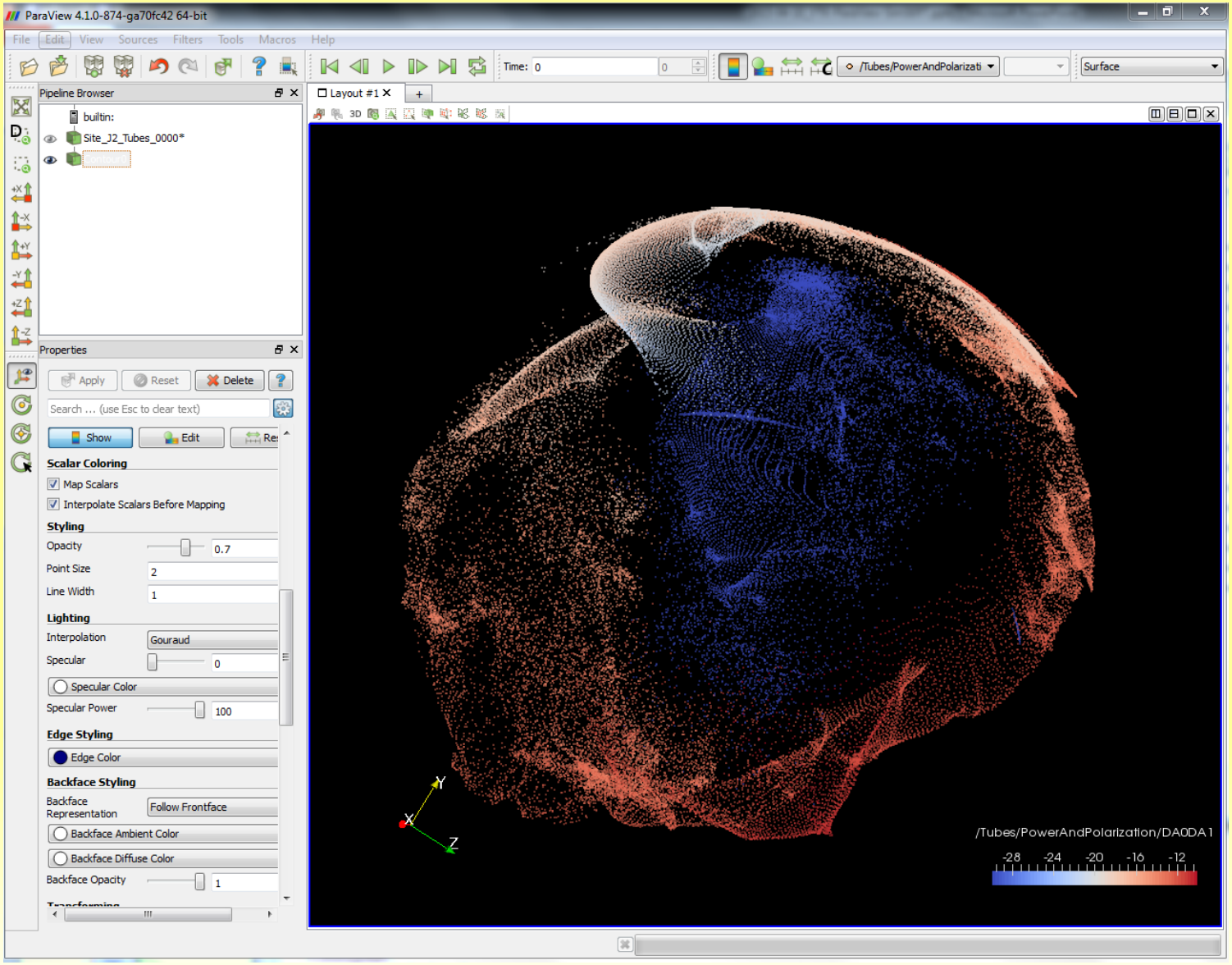
- Visualisation de données 3D + temps : points, lignes, surfaces, volumes
- Spreadsheets et graphs 2D en complément

- Stéréoscopie
- Gestion d'IHM avancées (VRPN)

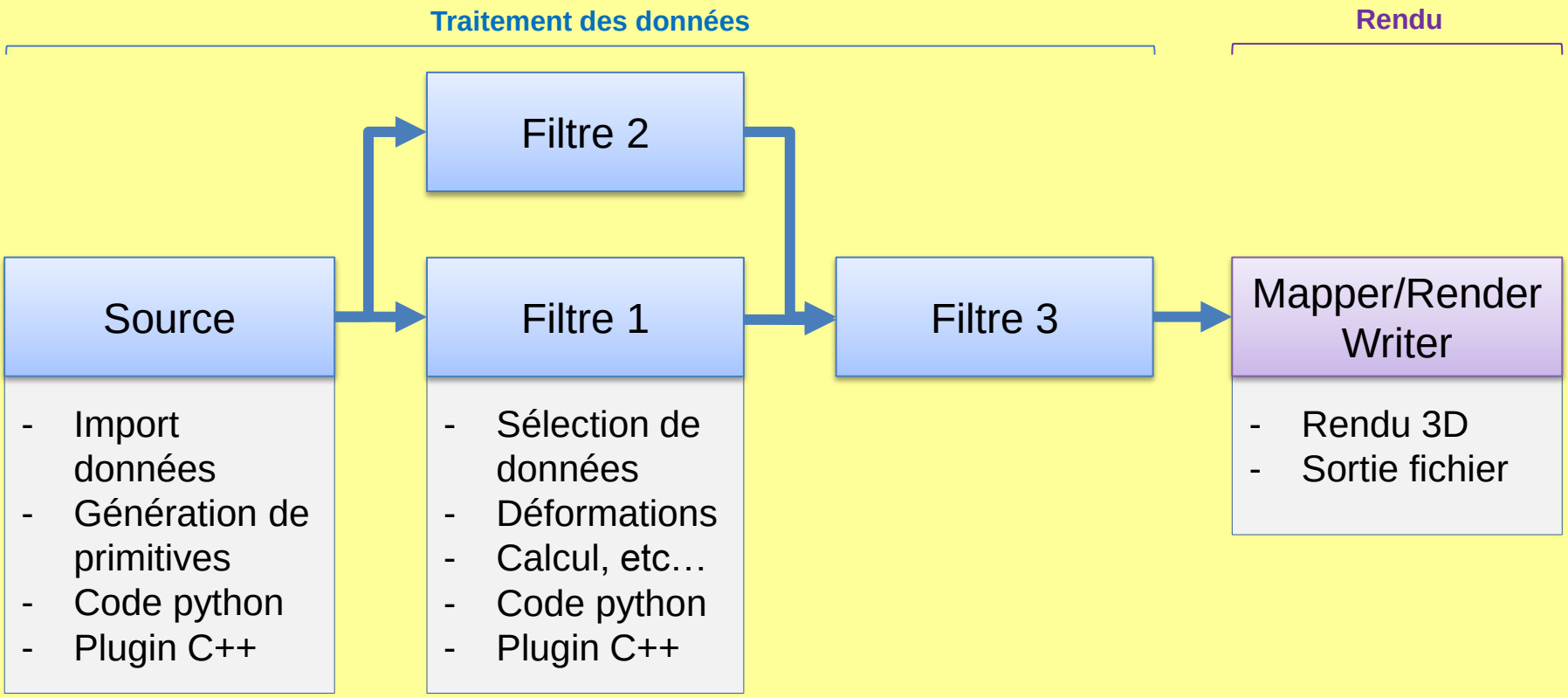
- Outil de prototypage rapide
  - Import de données
  - Fonctions de traitement évoluées
  - Construction d'un pipeline intéractif
  - Support de python (numpy & matplotlib)

- Vidéos, screenshots

ParaView

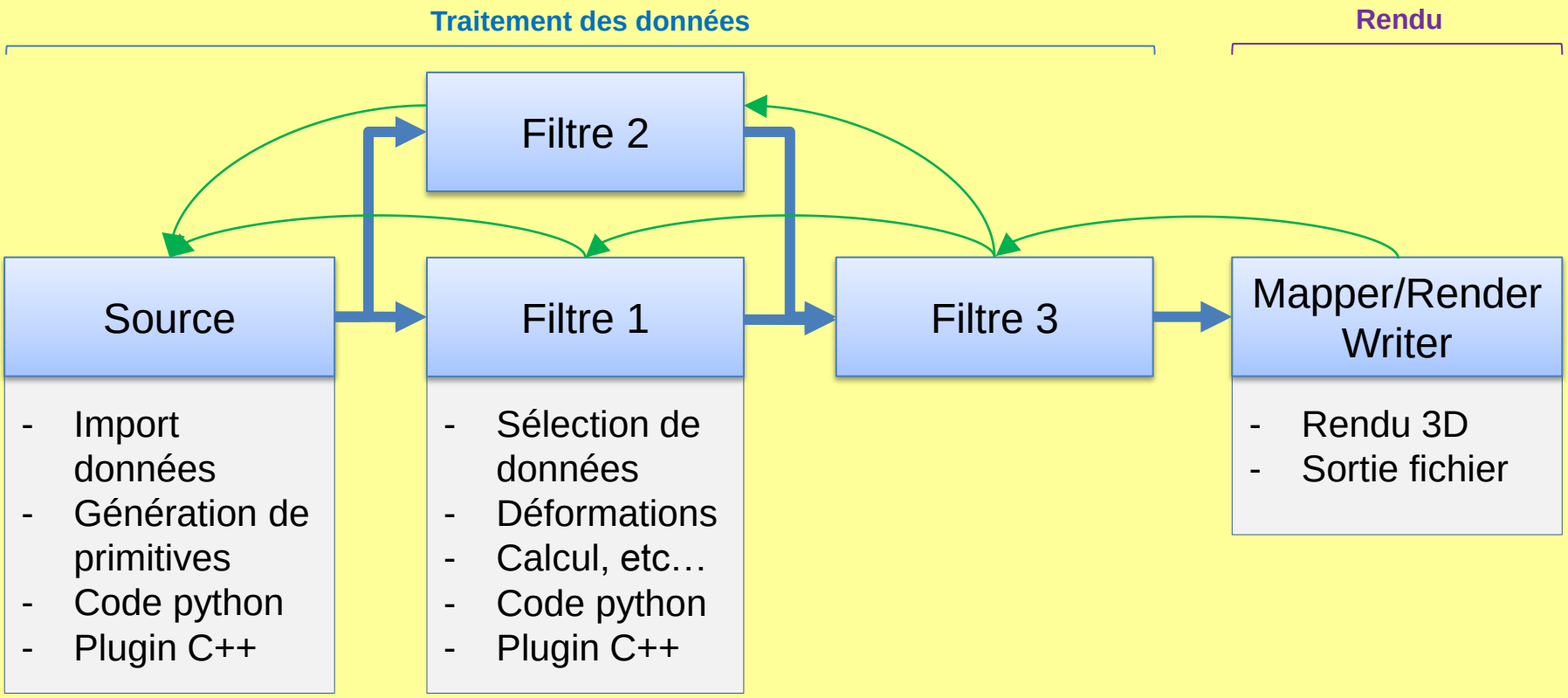

ParaView 4.1.0-874-ga70fc42 64-bit
File Edit View Sources Filters Tools Macros Help
Time: 0
/Tubes/PowerAndPolarizati
Surface
Pipeline Browser
builtin:
Site_J2_Tubes_0000*
Contour1
Layout #1
Properties
Apply Reset Delete
Search ... (use Esc to clear text)
Show Edit
Scalar Coloring
Map Scalars
Interpolate Scalars Before Mapping
Styling
Opacity 0.7
Point Size 2
Line Width 1
Lighting
Interpolation Gouraud
Specular 0
Specular Color
Specular Power 100
Edge Styling
Edge Color
Backface Styling
Backface Representation Follow Frontface
Backface Ambient Color
Backface Diffuse Color
Backface Opacity 1
/Tubes/PowerAndPolarization/DA0DA1
-28 -24 -20 -16 -12

VTK: le pipeline



Flux des données : “downstream”

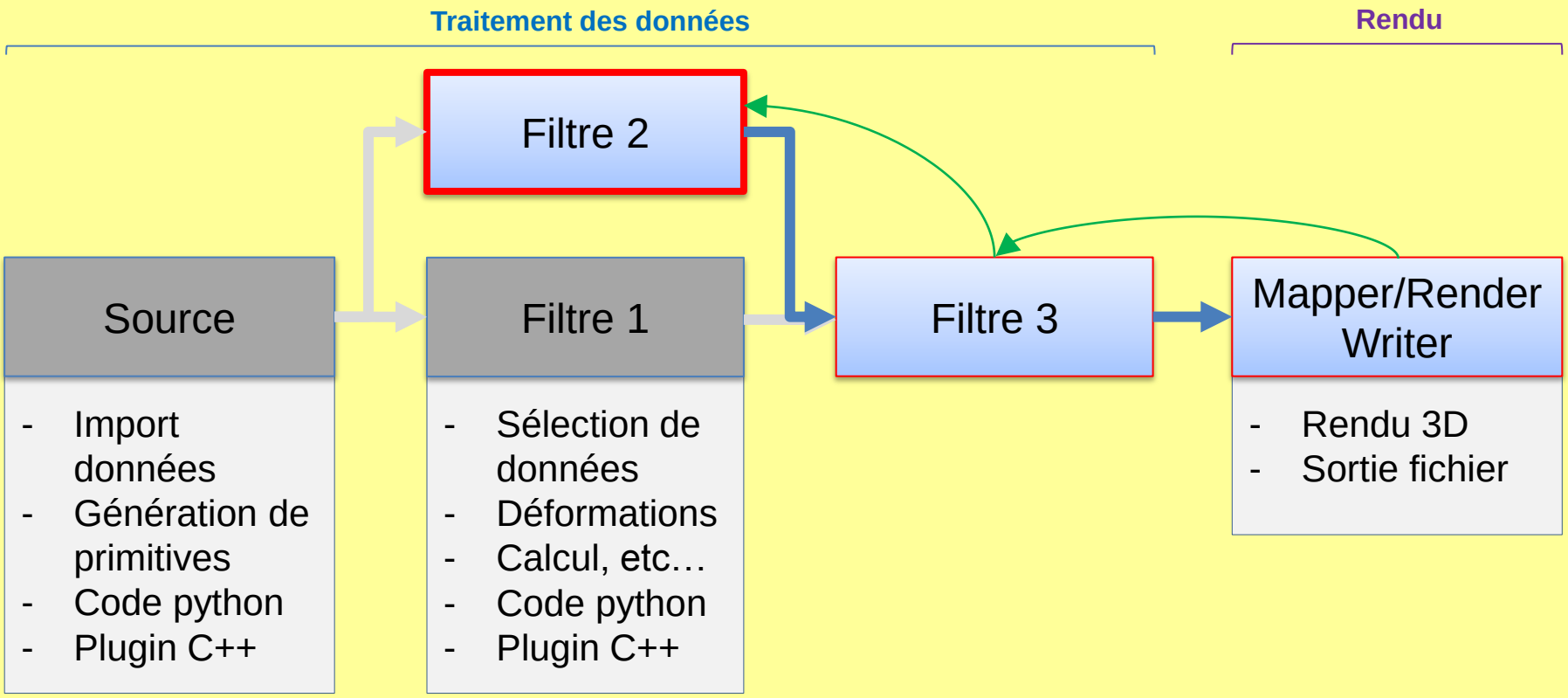
VTK: le pipeline



Flux des données : “downstream”

Flux des demandes de mise à jour : “upstream”

VTK: le pipeline



Propriétés inchangées

Flux des données : “downstream”

Flux des demandes de mise à jour : “upstream”

=> **seuls les filtres nécessaires sont mis à jour**

VTK: le pipeline

- Contrôle des flux de données et d'informations
- Modules de traitements indépendants, réutilisables et paramétrables interactivement

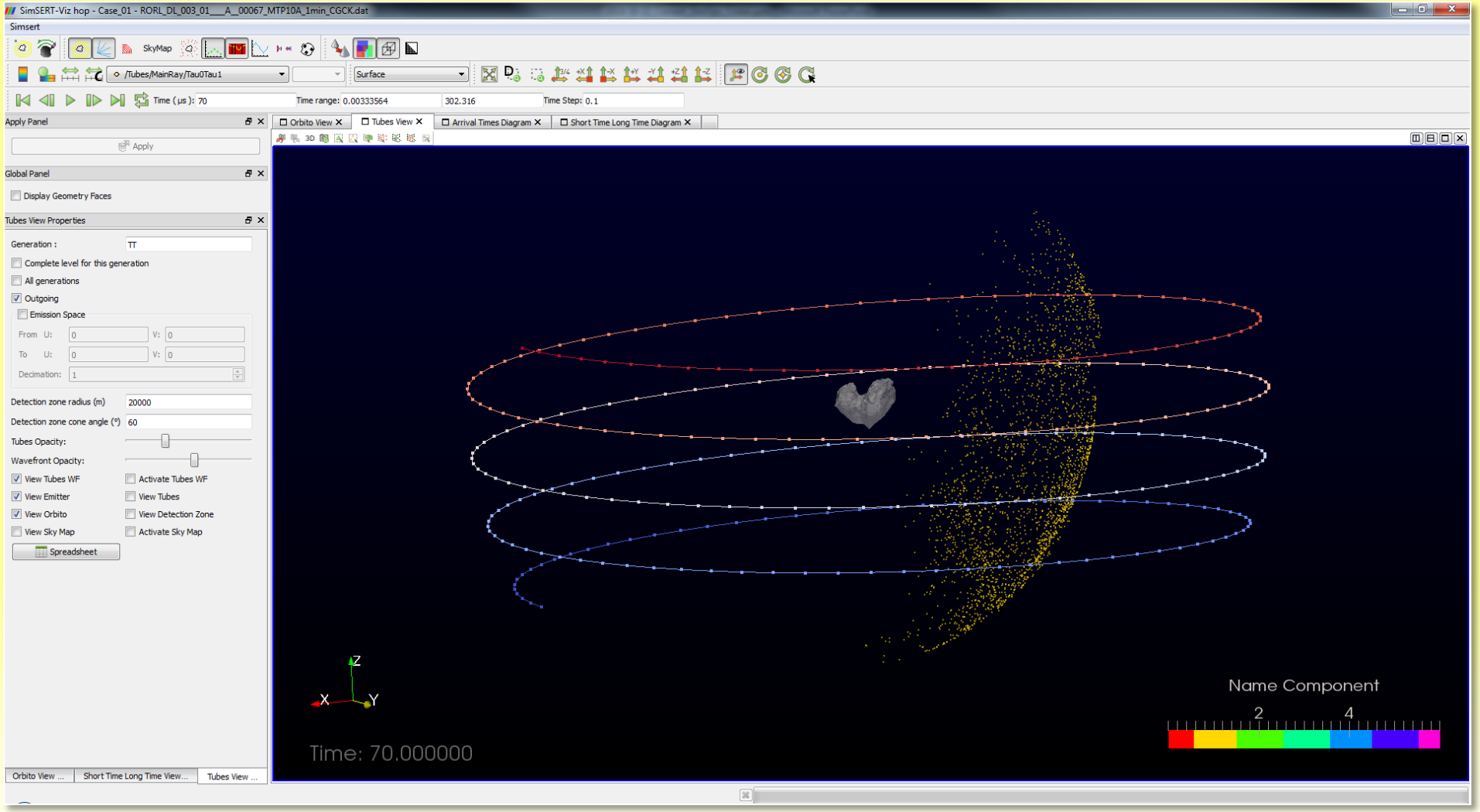
- "Lazy update"

- "Streaming" (découpage des données en pièces ou blocs structurés) si la capacité mémoire n'est pas suffisante

- Aisément extensible en C++ (écriture de nouveaux filtres)

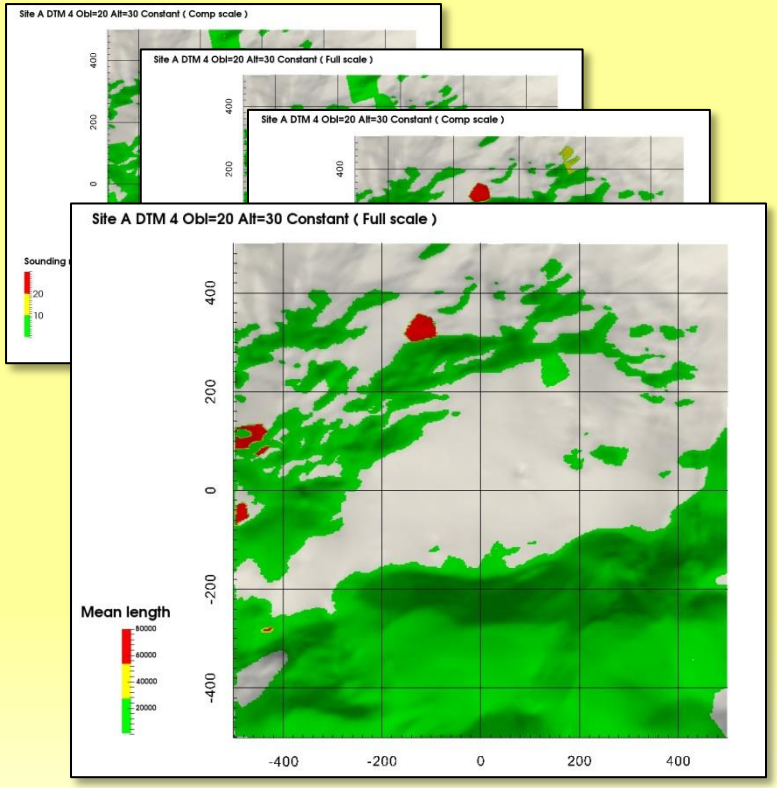
SimSERT

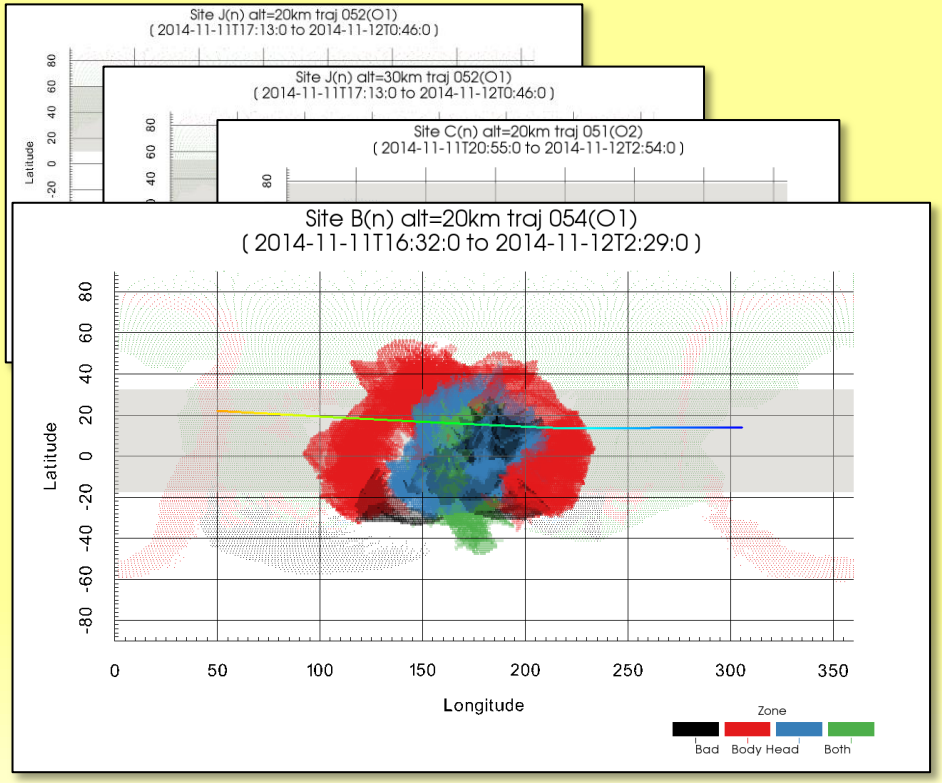
- Extension de la GUI de ParaView (Qt)



ParaView

- Process complètement automatisables via python pour générer des visualisations, des statistiques, etc…





CIMENT et Parallélisation

- Distribution de job : “embarrassingly parallel”
  - Aucune communication entre les process
  - On ne parle pas ici de parallélisation
  - Typique de l’utilisation **grille** (CiGri)
- Parallélisation en mémoire distribuée
  - Communication minimisée entre les process (MPI)
  - Typique de l’utilisation **cluster** (OAR) ou noeud
- Parallélisation en mémoire partagée
  - La RAM est complètement partagée entre les threads
  - Typique de l’utilisation **noeud** ou CPU (OpenMP, pthreads)

CIMENT et Parallélisation

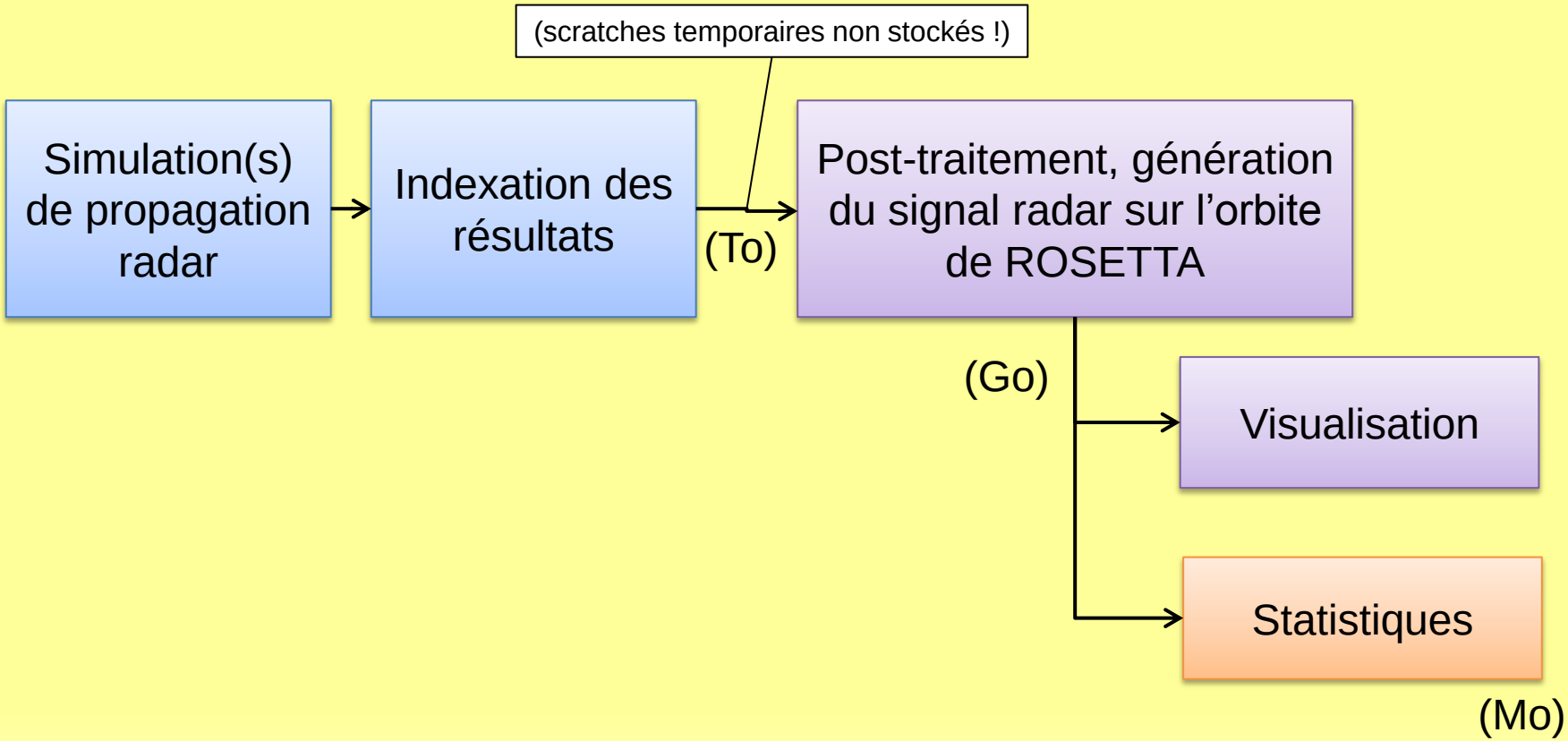
- Distribution de job : “embarrassingly parallel”
  - Aucune communication entre les process
  - On ne parle pas ici de parallélisation
  - Typique de l’utilisation **grille** (CiGri)

- Parallélisation en mémoire distribuée
  - Communication minimisée entre les process (MPI)
  - Typique de l’utilisation **cluster** (OAR) ou noeud

ParaView se situe à ce niveau

- Parallélisation en mémoire partagée
  - La RAM est complètement partagée entre les threads
  - Typique de l’utilisation **noeud** ou CPU (OpenMP, pthreads)

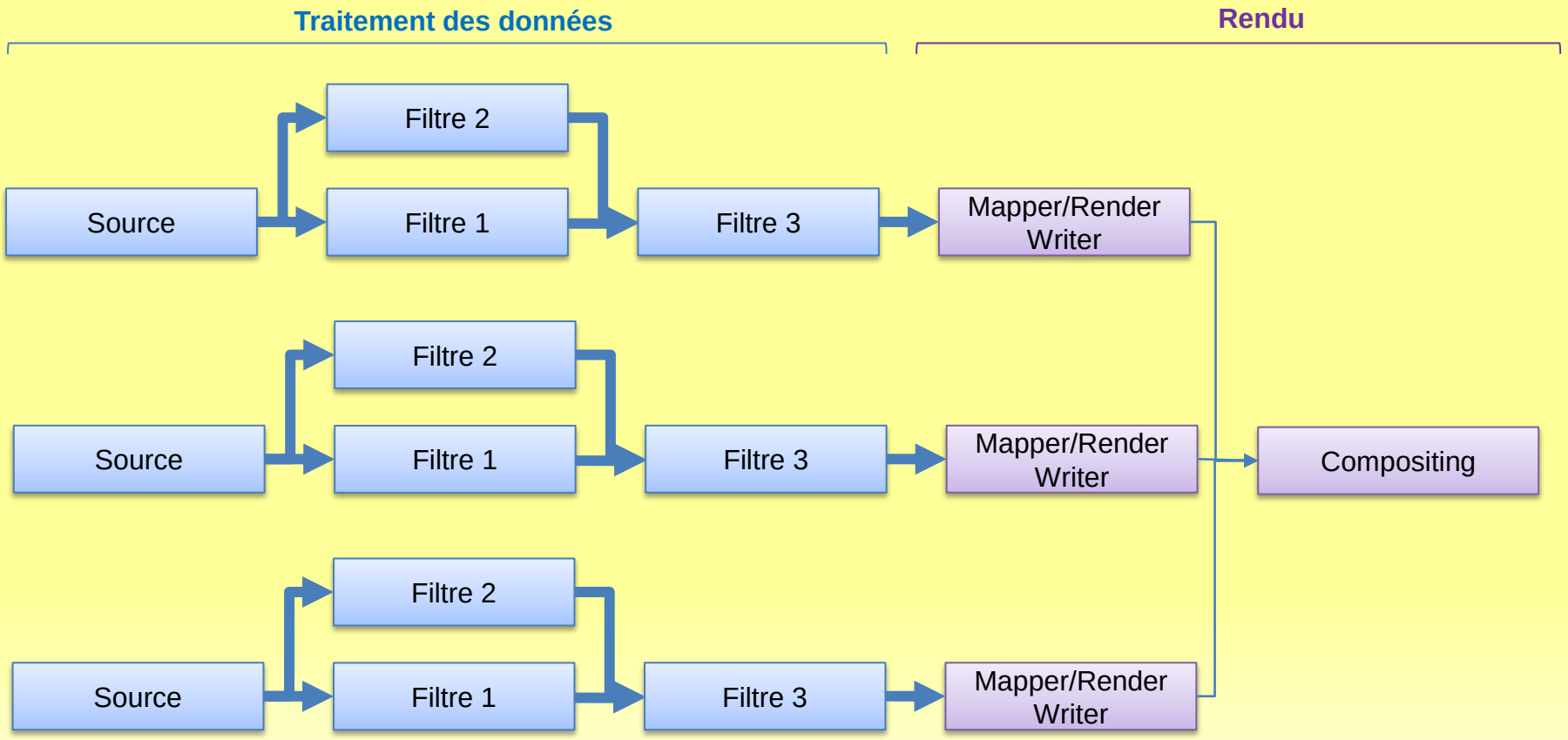
SimSERT et Parallélisation



Grille	Distribution de jobs indépendants (pthread) sur la grille en mode best-effort
Cluster	Exécution sur poste local (SimSERT-Viz), ou cluster (MPI) + OpenMP
Noeud	Noeud simple, serveur de calcul dédié (pvpython + numpy)

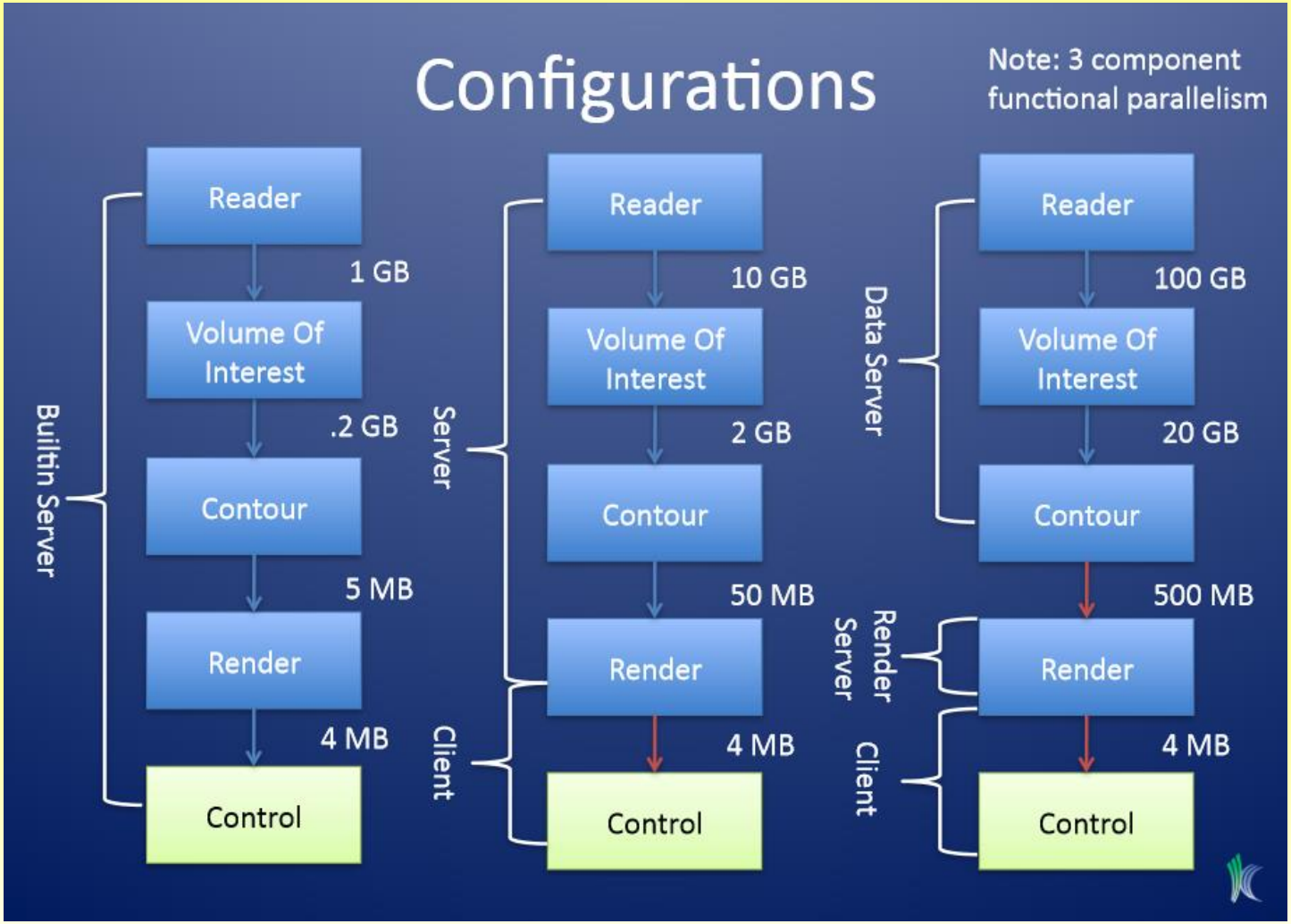
ParaView : le “Para”

- Exécution parralèle : pipeline dupliqué, données découpées



ParaView : le “Para”

- Répartition data / render / control



ParaView : le “Para”

- Architecture de parallélisation modulable

Architecture	Taille des données sources	Taille des données visualisées	Résolution d’affichage	Contraintes matérielles
(1) pvserver mode data + client paraview*	++ (le traitement doit réduire les données)	=	=	La machine client paraview doit avoir un GPU** puissant et une très bonne connexion réseau : Les données arrivent toutes sur un même noeud
(2) pvdataserver + pvrenderserver + client paraview*	++	+	=	Serveurs de data et render exactement couplés (même nombre), sinon on se retrouve dans la situation (1)
(3) client paraview* seul	=	=	=	Une machine avec CPU + GPU** puissants
(4) pvserver + client paraview*	+	+	=	Tous les noeuds doivent être équipés de GPU** mais aussi de CPU puissants si les données modifiables sont importantes
(5) mode “tiling” + Client paraview*			+	Plus de noeud avec GPU** sont nécessaires (jamais testé en fait, vraiment utile pour les très hautes résolutions, multi-surfaces)

* Le client paraview peut être la GUI de ParaView ou pvpython

** GPU = carte graphique, ne s’applique pas pour les GPGPU dédiés (type Tesla)

ParaView : le “Para”

- Architecture de parallélisation modulable

Architecture	Taille des données sources	Taille des données visualisées	Résolution d’affichage	Contraintes matérielles
(1) pvserver mode data + client paraview*	++ (le traitement doit réduire les données)	=	=	La machine client paraview doit avoir un GPU** puissant et une très bonne connexion réseau : Les données arrivent toutes sur un même noeud
(2) pvdataserver + pvrenderserver + client paraview*	++	+	=	Serveurs de data et render exactement couplés (même nombre), sinon on se retrouve dans la situation (1)
(3) client paraview* seul	=	=	=	Une machine avec CPU + GPU** puissants
(4) pvserver + client paraview*	+	+	=	Tous les noeuds doivent être équipés de GPU** mais aussi de CPU puissants si les données modifiables sont importantes
(5) mode “tiling” + Client paraview*			+	Plus de noeud avec GPU** sont nécessaires (jamais testé en fait, vraiment utile pour les très hautes résolutions, multi-surfaces)

* Le client paraview peut être la GUI de ParaView ou pvpython

** GPU = carte graphique, ne s’applique pas pour les GPGPU dédiés (type Tesla)

ParaView : retour d'expérience

- Les plus
  - Parallélisation sans connaître MPI
  - Flexiblité / évolutivité, bien adapté au projet scientifique
  - Multi-plateforme
  - Communauté très (ré)active
  - VTK repose sur une architecture mature et efficace

- Les moins
  - Instabilité (très peu de contrôle d'erreur dans ParaView)
  - Quelques points de détail pénibles dans l'ergonomie
  - Architecture logicielle de ParaView complexe (problèmes rencontrés en dérivant la GUI)
  - Difficultés à la compilation (dépendances)
  - Parfois très gourmand en RAM, avec les filtres de base

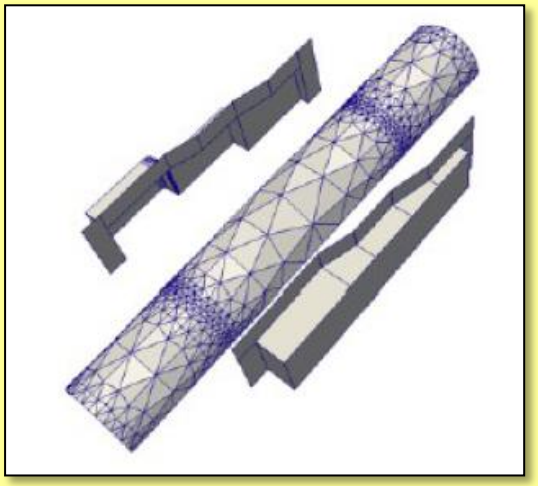
Structures de données : VTK

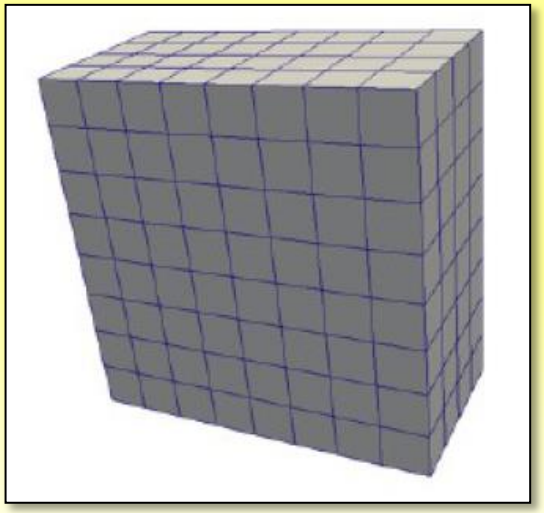
- Structures de données dédiées à la visualisation
- S'avère très efficace en traitement également

- Structures
  - Géométrie : points
  - Topologie : cellules
  - Données structurées / non structurées
  - Série temporelles

- Données attachées
  - "vtkArrays" assurent la coalescence des données
  - Lecture à la demande

- Points faibles
  - matrices creuses
  - accès aléatoires partiels





HDF5

- Un format de stockage pour des données binaires structurées, notamment pour les matrices à plusieurs dimensions

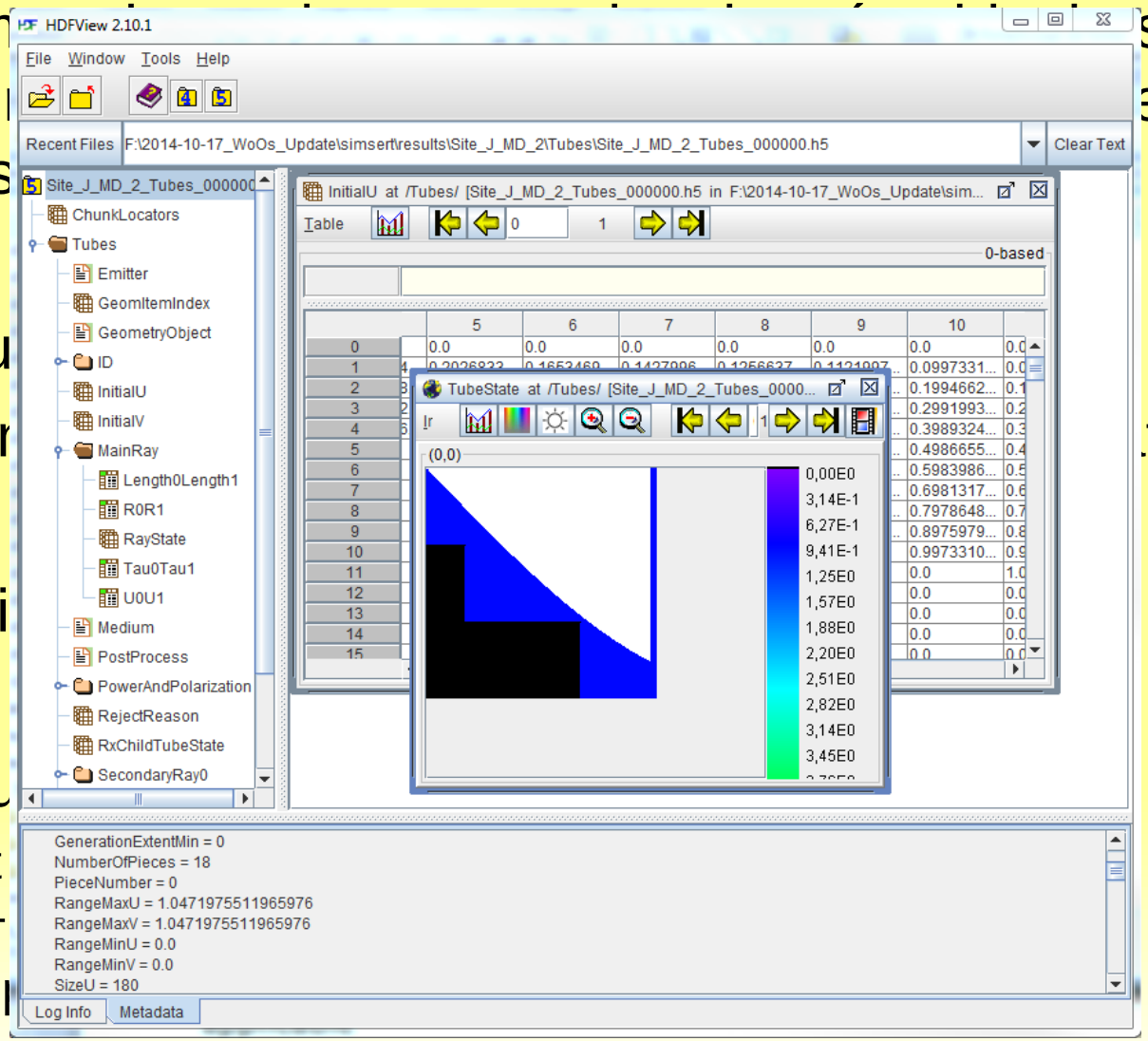
- Structuration arborescente auto-décrite (group, dataset)
- Gestion native d'un système de méta-données (attributs)

- Outil bien pratique : HDFView

- API multiplateforme à plusieurs niveaux
  - Haut niveau : portée dans de nombreux langages (C++, Java, python, IDL, matlab(?)
  - Complète : C uniquement

HDF5

- Un form… …s structu… …eurs dimens…
- Structu…
- Gestion… …ttributs)
- Outil bi…
- API mu…
  - Haut… (C++…
  - Comp…



HDF5

- Stockage et mapping mémoire (dataspaces)
- Accès aléatoires optimisés
- Stockage de matrices creuses
  - Découpage des datasets en chunks
  - Compression par chunk
- Gestion de “familles” de fichiers (découpage)
- API parallèle (MPI)

- Avec SimSERT
  - Besoin de nombreux modes de lectures partielles et aléatoires
  - Optimisation des paramètres de chunk :
    compromis taille des paquets / taille des fichiers

HDF5 & VTK

- Remarque : même approche de la structuration “array” plutôt que “objet”
- On peut donc directement lire un fichier HDF5 dans une structure VTK
- ➔ On optimise l’utilisation et les allocations mémoire jusqu’au rendu 3D

Quelques liens pour aller plus loin...

- **Présentations CED 2015**
  (Caroline Bligny, Bruno Bzeznik, Franck Pérignon - LJK/CIMENT)
- **Site Kitware VTK + ParaView** :
  http://www.kitware.com/opensource/paraview.html
  http://www.kitware.com/opensource/vtk.html
- **Mise en place cluster ParaView**:
  http://www.vtk.org/Wiki/images/a/a1/Cluster09_PV_Tut_Setup.pdf
- **Site HDF5**: http://www.hdfgroup.org/HDF5/
- **Site CIMENT** : https://ciment.ujf-grenoble.fr/wiki-pub/index.php
- **Site Visit** : https://wci.llnl.gov/simulation/computer-codes/visit/
- **G-SCOP (INPG) – Projet Visionnair** : http://www.infra-visionair.eu./
- **Blog rosetta ☺** : http://blogs.esa.int/rosetta/