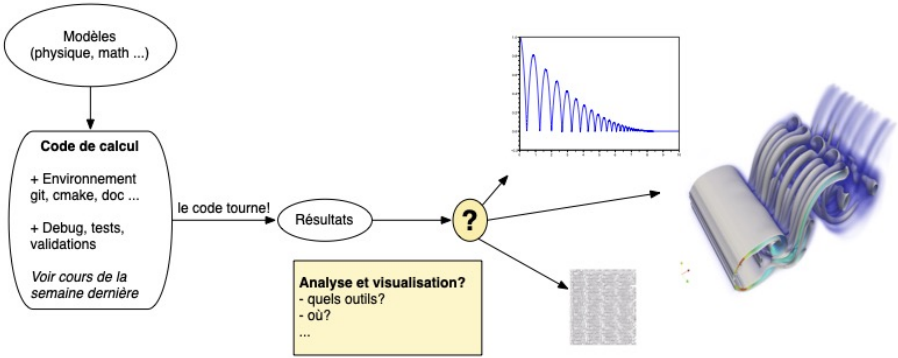
Une introduction aux formats de données et à la visualisation en calcul scientifique - Entrée-sorties parallèles

Caroline Bligny, Mondher Chekki, Albanne Lecointre, Franck Pérignon

mai 2023

1 Contexte et présentation du module
2 En amont, le Plan de Gestion des Données
3 Système de fichiers
4 Anticiper le post-traitement et la visualisation
5 Visualisation en calcul scientifique, quelques généralités
6 Structure et écriture des données
7 Outils de visualisation
8 Introduction aux Entrées-Sorties parallèles avec MPIIO
9 NetCDF et NCtools
10 HDF5 et H5tools
11 MPI-IO et ParallelHDF5
12 TPs

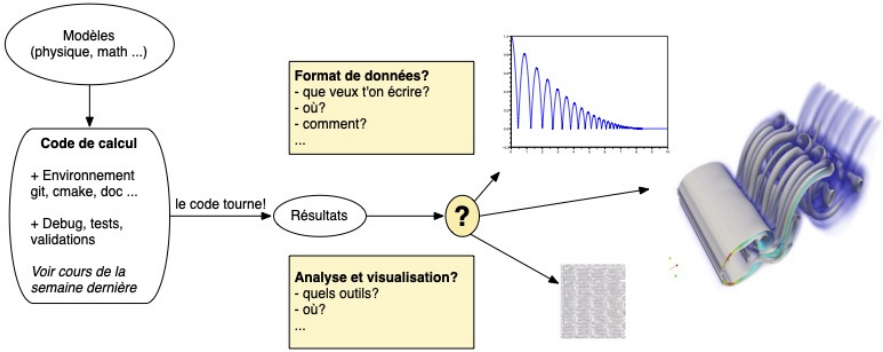
Contexte : (post-)traitement des résultats d'une simulation



(Post-)traitement : quand et comment exploiter, traiter et mettre en valeur les résultats ? Quels sont les outils disponibles/adaptés ?

- visualisation
- traitement des données (étude statistique, diagnostiques ...)

Contexte : (post-)traitement des résultats d'une simulation



Formats de données : comment (sous quelle forme) écrire les données ?
Formats de fichiers : comment "organiser" les fichiers et les données qu'ils contiennent ?
Structure de données : comment "organiser" les données, pour les visualiser ?

Contexte, pourquoi ce cours?

Ce module **Introduction au calcul parallèle** fait partie d'un ensemble de formations transverses proposées par le collège doctoral, en partenariat avec GRICAD et MaiMoSiNE, intitulé **Outils pour le traitement de données, le développement logiciel et le calcul scientifique**.
Pour commencer, nous allons donc :

- présenter le contexte de ces cours,
- présenter brièvement GRICAD et MaiMoSiNE.

Formations transverses CED - Outils pour le traitement de données, le développement logiciel et le calcul scientifique

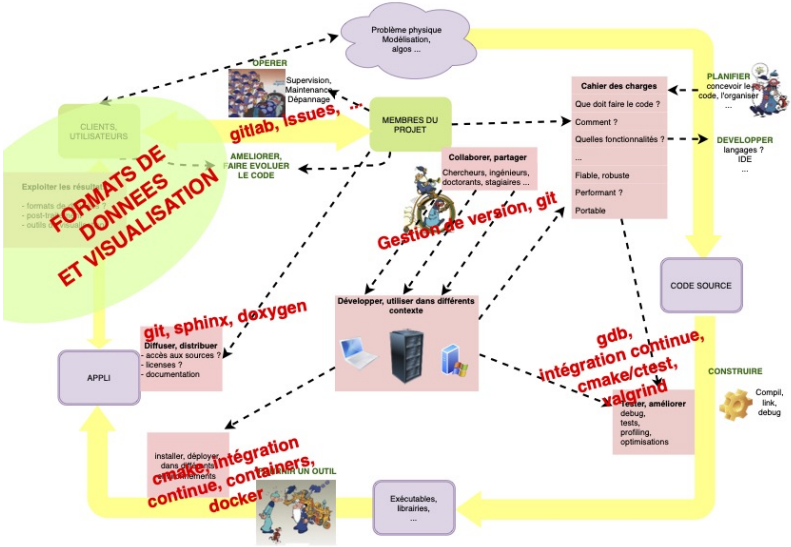
Accès aux documents (ce cours et les autres):

https://pole-calcul-formation.gricad-pages.univ-grenoble-alpes.fr/ced/

Les formations :

- Les bases du systeme linux pour le calcul scientifique
- Gestion de projets et developpement collaboratifs - utilisation de la plateforme gricad-gitlab
- Des sources a l'executable : la chaine de compilation
- Introduction au calcul parallele
- Introduction aux formats de donnees et a la visualisation en calcul scientifique - entrees-sorties paralleles

Produire et exploiter des résultats



L'UAR GRICAD

Unité multi-tutelles (CNRS, UGA, GINP, INRIA) d'Appui à la Recherche
GRenoble Infra CAlcul et Données
Missions:

- **Accompagnement et conseils** aux chercheurs sur leurs besoins liés **au calcul et aux données**
- **Mise à disposition à l'ensemble des chercheurs et personnels en support de la recherche** d'infrastructures avancées et mutualisées pour le calcul intensif et l'exploitation des données de la recherche
- Participation aux infrastructures de site en terme d'hébergement, de stockage et de virtualisation

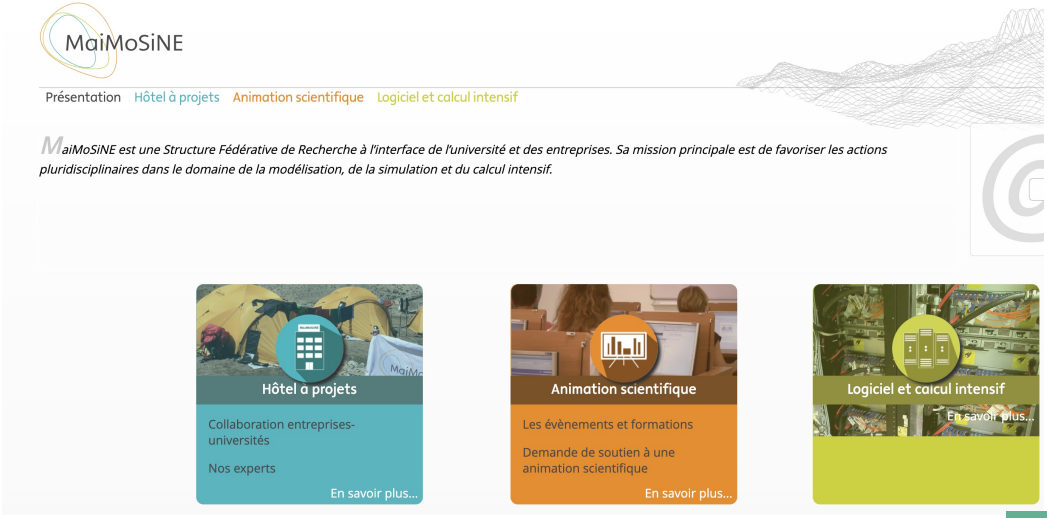
- Interaction forte avec les laboratoires : postes mutualisés, lettres de mission
- 2ème étage du batiment IMAG.
- *https://gricad.univ-grenoble-alpes.fr/*
- **Contact :** gricad-contact@univ-grenoble-alpes.fr

Quelques services proposés par GriCAD

- **Accès à des plateformes de calcul**, de grille et de stockage associées, mutualisation financière pour l'achat d'infrastructure.
- **Accompagnement et conseils dans le périmètre du calcul et du traitement de la donnée**, expertises, aide au montage de projets ...
- **Gestion, design et ingéniérie de la donnée**, DMP, stockage, diffusion, FAIRisation des données, archivage ...
- **Service de cloud**, machines virtuelles à la demande (NOVA)
- **Plateforme de notebooks**, *https://jupyterhub.u-ga.fr*
- **La forge logicielle Gitlab**
- **Former et animer la communauté sur le calcul intensif et la donnée**

MaiMoSiNE

Maison de la Modélisation et de la Simulation Nanosciences et Environnement



Organisation du module et planning

Objectifs

⇒ Définir ce qu'on entend par visualisation (3D) en calcul scienfitique.

⇒ Fournir de "bonnes" pratiques et conseils pour le choix et l'utilisation d'outils de visualisation et de formats de données.

⇒ Illustrer à travers des exemples l'utilisation d'outils standards : HDF5, Paraview.

Plan

1. Introduction et généralités sur la visualisation.
2. Formats de fichiers/données, structures de données.
   - Tutoriel/démo HDF5, VTTK
3. Visualisation
   - Un rapide aperçu de Gnuplot et Matplotlib
   - VTK et Paraview
4. Travail sur des serveurs distants
5. IO parallèles

Infos pratiques

- **Salle, accès et contacts** : voir
  https://pole-calcul-formation.gricad-pages.univ-grenoble-alpes.fr/ced/infos_pratiques/.
- **Accès aux documents** - Tous les cours sont accessibles sur le site :
  https://pole-calcul-formation.gricad-pages.univ-grenoble-alpes.fr/ced/
  Remarque: ces documents seront mis à jour au fur et à mesure des séances. Pensez à toujours récupérer la dernière version.
  Pour accéder aux exercices et solutions, il sera nécessaire de posséder un compte sur la plateforme https://gricad-gitlab.univ-grenoble-alpes.fr et de faire partie du projet ced2022/data-visu/trainings.

Rappel des pré-requis indispensables

- Etre à l'aise avec un système unix/linux (WSL pour Windows ?)
- Savoir utiliser la plateforme gricad-gitlab et git (pour récupérer les exercices et démos)
- pour la partie IO parallèles : quelques notions de MPI
- Et surtout, connaître son identifiant agalan et le password associé !

Pour ces différents points, voir entre autres les modules disponibles dans *https://pole-calcul-formation.gricad-pages.univ-grenoble-alpes.fr/ced/plans_modules/*.

Tour de table

Merci de vous présenter brièvement (école doctorale, sujet de thèse ...)

- Quelle est votre expérience en gestion de données/visualisation, quels outils connaissez-vous déja ?
- Qu'attendez vous de ce module (Culture générale, un cas d'usage concret ...) ?

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

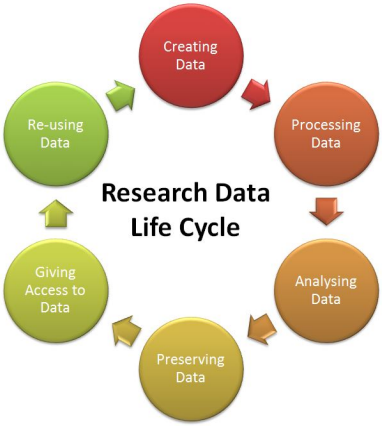
11 MPI-IO et ParallelHDF5

12 TPs

La gestion des données

- La question de la gestion des données de la recherche est liée à plusieurs problématiques plus ou moins récentes
  1. Le volume des données explose
  2. Mouvement de l'open science, open data, reproductibilité de la recherche
  3. Contraintes règlementaires sur les données (DMP, RGPD, Licences)
- Cela influe sur la manière dont les données vont être stockées, transférées, partagées, analysées, publiées

Le DMP et le cycle de vie des données



- Le **DMP**, ou **PGD** (Data Management Plan ou Plan de Gestion des Données) décrit comment sont gérées les données au cours d'un projet de recherche
- C'est une demande institutionnelle et une aide à l'organisation d'un projet
- L'Objectifs est que les données servent **facilement** au projet mais aussi éventuellement à d'autres collègues (open data), y compris pour permettre d'assurer la reproductibilité
- Un site pour vous aider : https://dmp.opidor.fr/

Il faut réfléchir aux points suivants

- **Quelles données** vont être générées / traitées dans le projet : type des données, volumétrie, format ...
- Comment les **organiser et les documenter** pour faciliter le partage entre collègues, la diffusion et la valorisation, la réutilisation ultérieure ?
- Quels **traitements et analyses** seront réalisés sur les données, quels outils utiliser , ...
- Où va-t-on les **stocker** ? Pendant la vie active des données, après ? Comment assurer la sauvegarde ?
- Quelles données **diffuser** ? Où les partager (entrepôts, data paper, supplementary materials de publi ...) ?
- Et les questions de **droits** : à qui appartiennent les données, quelle licence pour le partage...

Les principes FAIR pour vous aider

Findable

Données **faciles à trouver**.

- identifiant unique et pérenne
- décrites par des métadonnées riches ;
- enregistrées ou indexées dans une source interrogeable

Interoperable

Facile à combiner avec d'autres jeux de données, par les humains et les systèmes informatiques

- formats libres et ouverts
- mise à disposition du code source si le logiciel de traitement existe
- standards de métadonnées et vocabulaire standardisés

Accessible

Données ou au moins méta-données facilement accessibles.

- entrepôt de confiance, pérenne, certifié
- définir les conditions d'accès et la licence de diffusion
- si possible en open access
- si embargo ou accès restreint : méta-données accessibles

Reusable

Prêtes à être **réutilisables** pour une future recherche y compris via des méthodes informatiques

Stockage

Stocker ses données, quelques concepts

- Données chaudes vs données froides
- Stockage, sauvegarde, archivage
- Les solutions de site (SUMMER, Mantis, Bettik), les solutions nationales (Huma-Num pour les SHS, CINES, ..)
- Les entrepôts de données - Disciplinaires, génériques (ex Zenodo, projet RDG https://www.data.gouv.fr)

La cellule data de l'UGA

Si vous avez des questions sur les problématiques de gestion de données

- Mail cellule data UGA : mailto:sos-data@univ-grenoble-alpes.fr
- Site web cellule data : https://scienceouverte.univ-grenoble-alpes.fr/

Pour aller plus loin ...

Un module de deux jours les 27 avril et 04 mai 2023 : "Comment bien gérer ses données pour faciliter son travail de doctorat ? Enjeux et bonnes pratiques"

- *https://adum.fr/script/formations.pl?mod=3508755&site=UDG*
- *https://adum.fr/script/formations.pl?mod=3508758&site=UDG*
- Support 2022 : *https://scienceouverte.univ-grenoble-alpes.fr/services/formation-et-seminaires/formation-doctorale-gestion-des-donnees-de-la-recherche-2022/*

Deux modules complémentaires spécifiques :

- sur le RGPD le 30 mai 2023 (avec la DPO mutualisée du site) : "Savoir intégrer des pratiques respectueuses du RGPD à toutes les étapes de la thèse" *https://adum.fr/script/formations.pl?mod=3508564&site=UDG*
- sur la diffusion des données le 11 mai 2023 : "Comment bien diffuser ses données à l'issue de sa thèse ?" *https://adum.fr/script/formations.pl?mod=3508561&site=UDG*

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

Qu'est ce qu'un système de fichier ?

Rôles

- Organiser le stockage des données sur un support physique
- Gérer l'espace de nommage et les attributs

Types d'information traitées

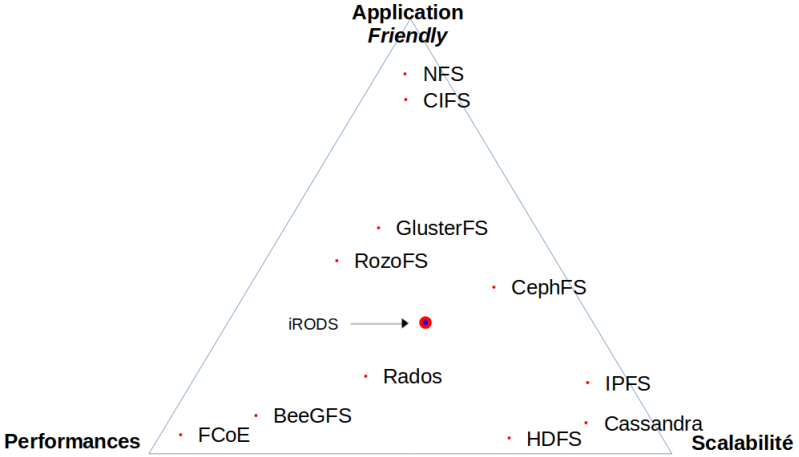
- **Données**: contenu des fichiers
- **Méta-données**: ensemble d'informations sur le fichier
  (nom, position du fichier sur le support physique, taille, attributs (propriétaire, groupe, ...))
- **Journal**: enregistrement des opérations avant leur exécution pour assurer la cohérence en cas de crash (coupure d'alimentation par exemple)

Différents types de systèmes de fichiers dans le monde du HPC[1]

- **Systèmes de fichiers séquentiels: Ext4, XFS, BtrFS,...**
  Utilisés en local sur les serveurs et les noeuds, parfois au dessus d'un système plus bas niveau (système RAID) permettant d'exploiter plusieurs disques pour des raisons de fiabilité et de performance.
- **Systèmes de fichiers partagés: NFS,NFSv4+RDMA, SSHFS,...**
  En général une couche réseau au dessus d'un système de fichiers, permettant de s'y connecter par le réseau.
- **Systèmes de fichiers distribués: BeeGFS, Lustre, Ceph,...**
  Partagés par définition, ils parallélisent les performances de plusieurs supports physiques et serveurs en `strippant` les fichiers (blocs distribués sur plusieurs serveurs)
- **Systèmes de fichiers objet: Irods, Swift, rados,...**
  Les objets peuvent être des fichiers, des blocks. Ils sont stockés sur des serveurs différents et les méta-données sont stockées à la manière d'une base de donnée.
  L'interface n'est pas POSIX (on doit généralement faire des `put` et des `get`).

[1] High Parallel Computing

Systèmes de fichiers distribués et performances



[2]Benoit Parrein, 12/12/16, Proposition de classification, ANF CNRS *Des données au BigData*

Un conseil essentiel

A éviter à tout prix
Sources de pertes de performances

BEAUCOUP de PETITS fichiers! $\longrightarrow$ Saturation du/des serveur(s) de métadonnées

- Un système de fichiers qui gère beaucoup de petits fichiers va passer plus de temps à gérer les meta-données et le journal que les données proprement dites!
- Le nombre et la taille dépendent de la configuration, mais on peut généralement considérer que 10000 fichiers de quelques k-octets peuvent déja poser des problèmes.

Les accès aléatoires ou concurrents sur des petits blocks peuvent poser problème!

- Gestion des “locks” (Conflits d'accès sur la même portion de fichier).
- Préférer les écritures et lectures séquentielles.

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

Quelques questions à se poser avant de simuler, de post-traiter, de visualiser ...

Choisir le bon outil c'est avant tout répondre à un certain nombre de contraintes.

Liées au problème traité *: que veut-on exactement voir/montrer à partir des simulations?*

- *Distribution* spatiale/temporelle des données : visualisation de courbes 2D, sur des grilles ou maillage 3D, animation en fonction du temps ...
- *Types* des données : quelle est par exemple la *précision* attendue ?
- *Pre ou post-traitement* des données? Est-il préférable de travailler sur les sorties ou d'effectuer des calculs supplémentaires durant la simulation (analyse in-situ ...) ?

Contraintes?

Liées au hardware : *où va t'on exécuter le code ?*

- Les lectures/écritures de fichiers peuvent coûter cher en *temps CPU et en mémoire* et sont souvent un frein aux performances des codes.
- Que va t'on faire des données ? Déplacer et stocker des gigas de données n'est pas toujours possible !
- Si le volume de données est important, quel outil sera capable de traiter mes fichiers ?

Contraintes?

Liées au software

- limites/possibilités du *langage de programmation* : quels sont les formats disponibles ?
- facilité d'implementation et de *maintenance*, *portabilité* : mes résultats seront ils exploitables par d'autres personnes, sur d'autres machines ?

Ou encore ...

- *Habitudes* du laboratoire, de la communauté (exemple : netcdf pour les softs de météo)
- ...

Quelques bonnes pratiques

- Estimer le volume des sorties avant de lancer le code ! Prendre en compte les limites liées au CPU, à la mémoire et au réseau.
- Limiter le volume/la fréquence des sorties : se restreindre à une zone de l'espace, sous-échantillonner en temps, estimer le coût d'un post-traitement durant la simulation, réduire la précision...
- Prévoir les sorties en fonction de l'outil de visualisation qui sera utilisé
- Choisir un format qui sera compréhensible/ exploitable par d'autres personnes.
- Profiler la simulation et estimer l'impact des entrées/sorties.
- Eviter de déplacer les données .
- En résumé : inutile de générer des résultats s'ils ne sont pas exploitables !

Envisager d'autres options :

- analyse/visualisation à la volée (Visit, Paraview).
- Travail en parallèle ou à distance (voir plus loin).

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

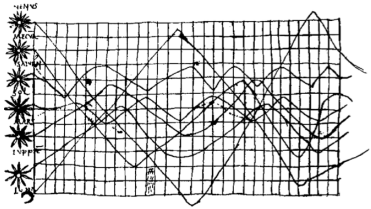
Quelle visualisation

« Visualisation de données » : Représentation visuelle de données brutes. cette terminologie recouvre des domaines divers et peut prêter à confusion.

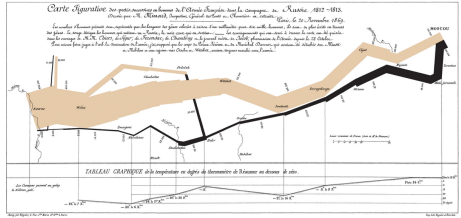
Quelles sont les types de visualisations qui existent, et pour quoi faire ?

Types de visualisations - graphiques

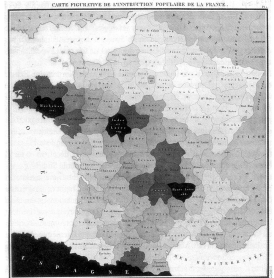
Les séries temporelles existent depuis l'antiquité. 1800 : essor des graphiques avec les statistiques



Mouvement des planètes au cours du temps, V eme siècle

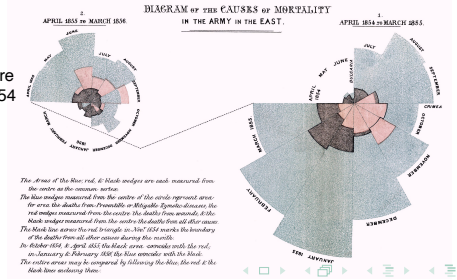


Flux - armée de Napoléon en Russie, 1832



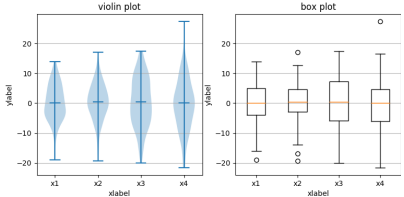
Carte choroplète (Heat map)
Instruction populaire, 1826

Circulaires
Causes de mortalité, guerre de Crimée, 1854



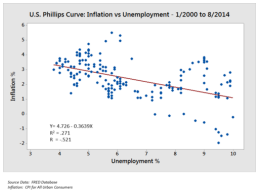
Types de visualisations - graphiques

Boites à moustaches

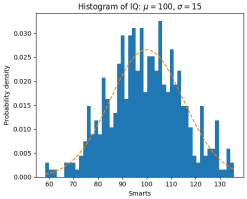


Araignée

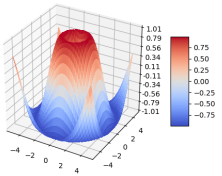
Nuages de points



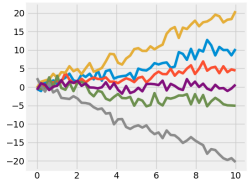
Diagrammes en bâton, histogrammes



Courbes 3D



Courbes, plot

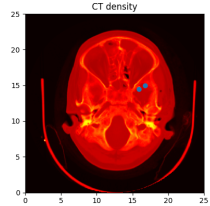


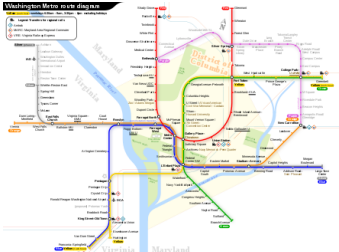
Images : matplotlib gallery, R gallery, wikipedia

Types de visualisations

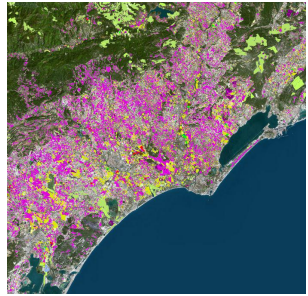
Réseaux

Images





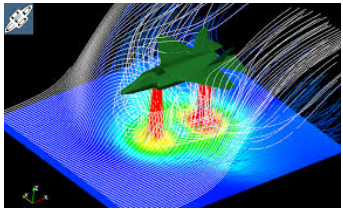
Données géolocalisées

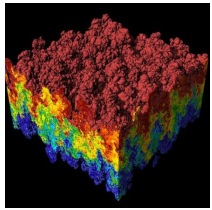


Images : wikipedia, www.data.gouv.fr

Types de visualisations

Visu 3D





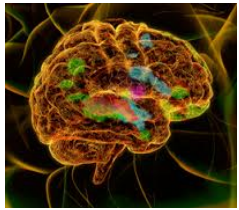


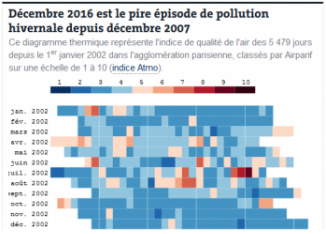
Image de synthèse, Animation
(c'est la visu qui génère les données)

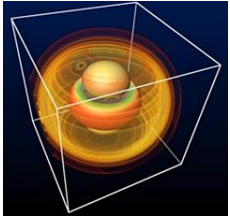
Images : wikipedia, blender gallery

Quelle visualisation

Se dégagent quelques grandes tendances

- « dataviz » vs « visualisation scientifique » vs « computer graphics »





Simulation d'ondes gravitationnelles

- Finalités : expliquer, explorer, créer

Dans la suite du cours, on se concentre sur la visualisation scientifique dans un but plutôt exploratoire.

Une incursion du coté de l'ergonomie

Quelques éléments pour améliorer l'ergonomie des graphiques, tirés de l'intervention B. Gorelik à euroscipy :

- Etre clair sur la finalité - mode « explanatory » vs « exploratory »
- Le moins de texte possible
- Choisir un type de graphique pertinent (bonne synthèse sur www.r-graph-gallery.com)
- Le titre contient la conclusion - « so what » vs « what »

https://fr.slideshare.net/borisgorelik/three-most-common-mistakes-in-data-visualization-and-how-to-avoid-them

Les données en visu 3D

Nous parlerons ici de visualisation de données scientifiques, issues de simulations ou expérimentales.

Souvent, il s'agit de comprendre des phénomènes physiques. Les domaines les plus fréquemment concernés sont la médecine, les sciences de la terre et de l'univers, la mécanique des fluides, l'industrie aéronautique et automobile.

Les données sont numériques, potentiellement volumineuses. On cherche à en extraire de l'information visuelle.

Les étapes d'une visu 3D

- Génération du **maillage**
- Simulation, calcul : génération des données
- Transformation des données : filtres, …
- Gestion de la **scène** - lumière, caméra, ajout d'objets, ...
- Génération de l'image 2D : le **rendu**
- Interactions - zoom, rotation, …
- Création d'images, de films Création d'images, de films

Un peu de vocabulaire

(sources : Wikipédia)

- Maillage (mesh) : Un maillage est la discrétisation spatiale d'un milieu continu, ou aussi, une modélisation géométrique d'un domaine. On parle aussi de grille
- Dataset, Le jeu de données
- Rendu - Calcul de l'image 2D d'une scène définie dans un environnement 3D, comportant des objets et des sources de lumière et vue d'un point de vue précis.
- API OpenGL : Open Graphic Library . OpenGL permet de déclarer la géométrie d'objets sous forme de **points**, de **lignes**, de **polygones**, de bitmaps et de textures. OpenGL effectue ensuite des calculs en vue de déterminer l'image à l'écran.
- Primitives graphiques : cellules 0,1 et 2D. OpenGL est basée sur ces primitives.
- GPU : Graphical Processing Unit, placé sur la **carte graphique,** calcule le rendu.

Librairies et outils

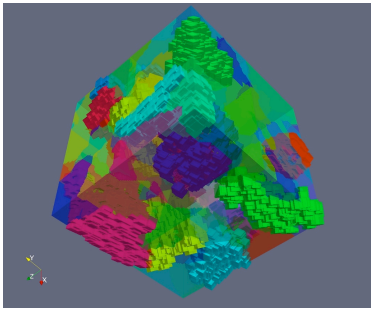
- Le choix de l'outil va dépendre du domaine, du type de données, de l'objectif de la visualisation, et des ressources techniques ou financières
- Un outils peut réaliser plusieurs étapes de la visualisation, ou être spécialisé.
- Attention à la pérennité, en particulier dans le cas de l'Open Source
- VTK et ses outils dérivés comme ParaView sont largement utilisés.
- On peut enchainer plusieurs outils et utiliser python comme langage de glue

Types d'outils - échantillon

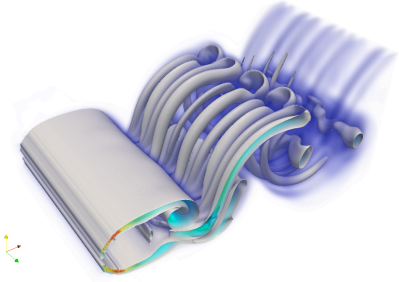
Maillage	Gmsh, OpenMesh, ViennaMesh, Netgen/NGSolve, Meshlab
Visu 3D	VTK
	ParaView, VisIt, Mayavi
Scene 3D	Blender, sketchup, freeCad
	OGRE
Rendu	Yafaray, POVray, sunFlow, RenderMan (pixar), LuxRender (—> LuxCoreRender), Cycle Render

Exemples de visualisation simple

Utilisation ‘simple’ de ParaView



Orientation des cristaux dans un matériaux ferro-magnétique. (Image Renaud Pipet)



Ecoulements autour d’un cylindre. (Image Choé Mimeau)

Exemple de visualisation - Overflow

Exemple de film réalisé avec Paraview

Stage de Mathis Blache réalisé au LJK et à l'IGE

Visualisation d'un overflow dans le détroit du Danemark

Données entrées format netcdf,
de 5Go à + de 100 Go

↓

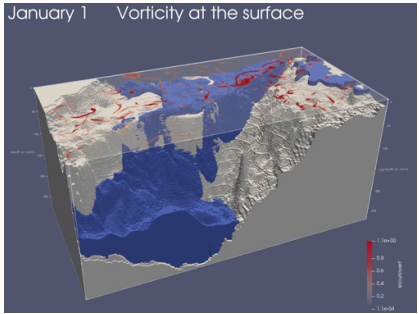
1) En amont de la visu, filtrer et ajuster les données

↓

2) Paraview : extraire le profil du fond marin, trouver les bonnes valeurs de densité, ajouter la vorticité

↓

3) Script qui enchaine les actions pour traiter les données sur un mois → 4) Monter le film (lib ffmpeg)



Exemple de visualisation - La goutte

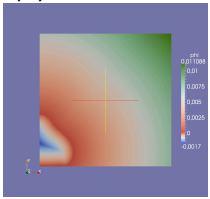
Exemple de visualisation utilisant plusieurs outils.

Réalisé dans le cadre d'une thèse au LJK par Roland Denis :

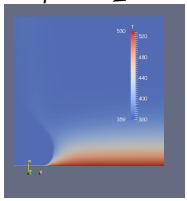
Modélisation et simulation de l'effet Leidenfrost dans les micro-gouttes

Simulation --> données dans fichier format vti

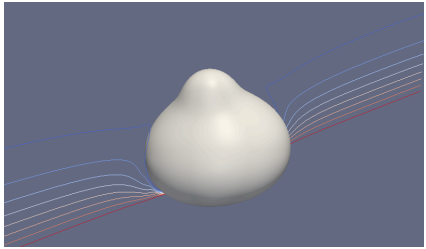
phy



temp

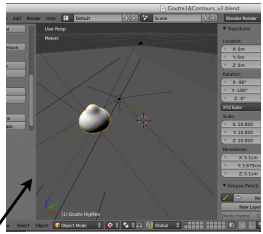


Paraview : maillage goutte + lignes de champ température



Export format .x3d

Blender : mise en place de la scène



Luxrender : calcul du rendu

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données
- Représentation des données sur le disque
- Formats de fichiers
- HDF5
- Structures de données 3D

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

Représentation des données sur le disque ?

Deux "représentations" possibles des données lors de l'écriture de fichiers (mais de multiples formats de fichiers possibles !)

ASCII : fichier texte classique, où un caractère = 1 byte. Les données sont "formatées" avant d'être écrites dans le fichier.

+ lisible
+ portable
- coût en mémoire, écriture pas optimale

Binaire : écriture 'directe', i.e. telle que dans la mémoire[3]

- pas très lisible (...), possibles problèmes de portabilité.
+ I/O plus efficaces, gain de place sur le disque.

[3] Voir la Norme IEEE 754 pour les flottants. Pour plus d'infos sur la représentation des nombres, voir par exemple le cours de Fortran de l'IDRIS : http://www.idris.fr/data/cours/lang/fortran/fortran_base/IDRIS_Fortran_cours_presentation.pdf

Exemples

En fortran,

```
!! texte
open(unit=20, file='fic', form='formatted')
write(10, "(F10.4)"), y
close(10)
!! binaire
open(unit=10, file='fic.bin', form='unformatte
write(10), x
close(10)
```

En C

```
int var[100];
...
// texte
FILE * fichier2 = fopen("fic.txt", "w");
for(int i=0;i<10;++i)
(fichier2, "%d\t", var[i]);
fclose(fichier2);
// binaire
FILE * fichier = fopen("fic.bin", "wb");
fwrite(&var, sizeof(int), 10, fichier);
fclose(fichier);
```

Exemples

En python (voir l'exemple demo_io.py fourni dans les TD)

```
# Array into an ascii file
with open('out_txt', 'w') as txtfile:
   for v in velocity.flat:
   val = "{:.10f}".format(v) + "\t"
   txtfile.write(val)

# Array into a binary file
np.save("txt.bin", v)
```

"Format" de fichier ? Structure de données ?

Une manière d'écrire les données dans un/des fichier(s), sur le disque :
quelles variables, dans quel ordre, avec quelle structure, avec quelle précision ...
Chaque code peut avoir son **format "maison"**, défini par les développeurs (exemples classiques : un fichier texte de paramètre avec des lignes de commentaires, des colonnes de chiffres etc.)

- Compréhensible par d'autres utilisateurs ?
- Optimal ?
- Exploitable par un outil de visualisation ?

En général, non ...
Autres alternatives : des librairies dédiées, fournissant des formats de données 'auto-documentés', entres autres, telles que NetCDF, HDF5.

Un petit exemple

On considère:

- un domaine 3D cubique, avec une discrétisation régulière en espace (résolution : $N_x \times N_y \times N_z$)
- la vitesse d'un fluide dans le domaine, $\mathbf{v}(x, y, z, t)$ un champs vectoriel,
- la température de ce fluide, $T(x, y, z, t)$, un champ scalaire.

On suppose que ces deux grandeurs sont données par un calcul (qui ne nous intéresse pas ici) à chaque pas de temps et en chaque point de la grille. On a donc en mémoire les variables suivantes:

- pour chaque pas de temps, 4 tableaux de réels, de dimension $N_x \times N_y \times N_z$.
- un vecteur contenant les différentes valeurs du temps

Exemple (2)

Supposons qu'on souhaite visualiser le champ de vitesse dans tout le domaine et le max de la température en fonction du temps.
Première étape : estimer la taille des variables dans la RAM.
Un tableau de réels en double précision : $N_x \times N_y \times N_z \times 8$ Bytes.

résolution	taille en mémoire (RAM)	fichier texte/binaire	temps CPU écriture texte/binaire
256^3	0.5 GB	224/128 MB	3.4/0.05 s
512^3	4.2 GB	1.8/1 GB	27/0.5 s
1024^3	34 GB	13/8 GB	6min/7s

A noter que dans cet exemple, on ne sauve qu'un seul champ dans les fichiers et ceci pour un seul pas de temps !

Exemple (3)

Quelques commentaires :

- → la RAM, le disque et le temps CPU deviennent rapidement un problème quand on augmente la résolution : la question "*où vont tourner les calculs* ?" prend tout son sens !
- → Un pré-traitement des données durant la simulation (calcul du max de température par exemple) sera vite indispensable.
- → On ne pourra pas sortir les variables à tous les pas de temps.
- → Le binaire c'est bien mais comment exploiter les fichiers générés ?
- → Quel format "sympathique" peut-on choisir pour ensuite visualiser les données sur la grille 3D ?

HDF5 et NetCDF

- HDF5 : Hierarchical Data Format, *http://www.hdfgroup.org/HDF5/*
- NetCDF : Network Common Data Form, *http://www.unidata.ucar.edu/software/netcdf/*

Ces librairies fournissent des outils pour structurer et sauvegarder des données et méta-données.
On peut également citer VTK (Visualization Toolkit, *http://www.vtk.org*), pour la création d'images et la visualisation.

HDF5 et NetCDF (2)

- *Haut niveau d'abstraction* pour manipuler les données,
- représentation structurée de *données complexes* éventuellement *volumineuses*,
- *méta-données* associées aux données,
- *interface* dans de nombreux langages (API en fortran, C, C++, Python ...),
- un format de fichier *portable*, compressé, utilisable par de nombreuses librairies de visualisation,
- capable de gérer les *entrées/sorties parallèles*.

\+ efficace, lisible, facile à maintenir et à exploiter,

\- un nouveau langage à apprendre.

HDF5, principes de fonctionnement

Organisation des données selon une arborescence type 'système de fichiers' (d'où le Hierarchical dans HDF5).

- Chaque 'niveau' de l'arborescence est appelé *groupe*.
- Les données sont stockées sous forme de *Datasets*, des tableaux.
- A chacun de ces objets sont associées des *méta-données*, contenant une description de l'objet (nom, type ...)

En pratique : un objet "fichier" dans lequel on va créer des groupes et sous-groupes puis ajouter des datasets.

Suite de l'exemple, en hdf5

On complète notre tableau précédent:

résolution	taille en mémoire	fichier texte/binaire/hdf5	temps CPU écriture texte/binaire/hdf5
256^3	0.5 GB	224/128 MB	3.4/0.05/0.05 s
512^3	4.2 GB	1.8/1 GB	27/0.5/0.5 s
1024^3	34 GB	14/8 GB	3min35/4 s/4 s

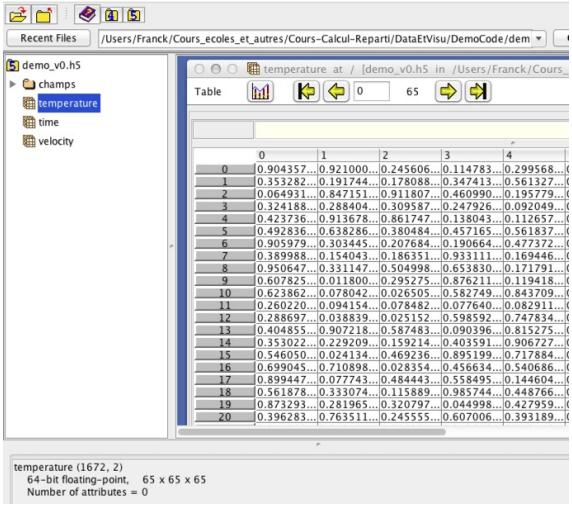
⇒ Gain en temps.

⇒ Gain en portabilité/comprehensibilité : les variables sont auto-documentées grâce aux meta-données !

⇒ Lecture/écriture et manipulation des datasets assez simple.

⇒ Comment exploiter nos fichiers h5 ?

Un outil pratique : HDFView. Permet de visualiser et d'éditer un fichier hdf5.



TP/démo écriture de fichiers et hdf5.

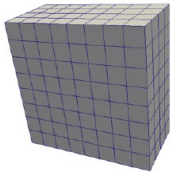
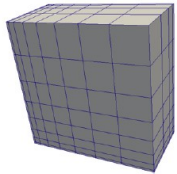
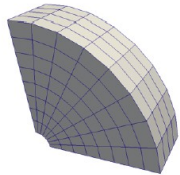
Rendez vous dans le projet data-visu/trainings et suivez les instructions de la première partie
Important : il faut que vous ayez un compte sur la plateforme
gricad-gitlab.univ-grenoble-alpes.fr et que vous soyez inscrit au projet (si ce n'est pas le cas, faites nous signe !)

Structure des données 3D

- Globalement, les données 3D sont composées de:
  - Une grille (mesh - grid) : géométrie (points) + topologie (connectivité entre les points)
  - Des données associées a des éléments de la grille : soit les points, soit les cellules
  - Eventuellement des données globales à la grille
  - Les données sont le plus souvent des scalaires, des vecteurs ou des tenseurs (matrice 3x3)
- Points abordés :
  - Les géométries de grilles possibles
  - Exemples d'implémentation avec les formats VTK, XDMF/HDF5
  - Formats de fichiers

Types de grilles : Les grilles structurées

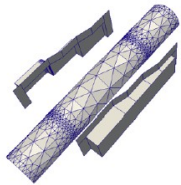
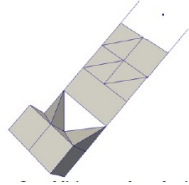
La connectivité entre les cellules est implicite

Uniform Rectilinear Grid /Image Data Axes perpendiculaires ; espacements constant	Rectilinear Grid Axes perpendiculaires	Structured Grid / Curvilinear Grid
		
DIMENSIONS nx ny nz ORIGIN x y z SPACING Sx Sy Sz	DIMENSIONS nx ny nz X_COORDINATES ... Y_COORDINATES .... Z_COORDINATES	DIMENSIONS nx ny nz POINTS n .....

Types de grilles : Les grilles non structurées

La connectivité entre les cellules est définie explicitement

```
POINT n
…
CELLS N size
  (number or points) i,j,k,l,… + cell type
 …
```

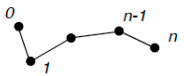
Polydata Grid	Unstructured Grid
Combinaison de cellules 0D, 1D ou 2D	Combinaison de tout type de cellule
	

Cellules 0D, 1D, 2D - exemple VTK

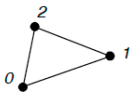
VTK_VERTEX (=1)

VTK_POLY_VERTEX (=2)

VTK_LINE (=3)

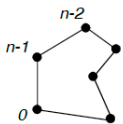


VTK_POLY_LINE (=4)

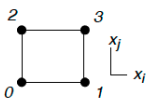


VTK_TRIANGLE(=5)

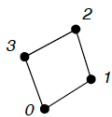
VTK_TRIANGLE_STRIP (=6)



VTK_POLYGON (=7)

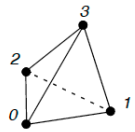


VTK_PIXEL (=8)

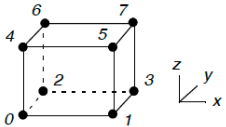


VTK_QUAD (=9)

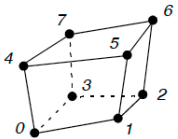
Cellules 3D - exemple VTK



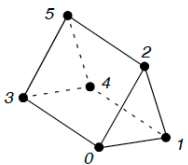
VTK_TETRA (=10)



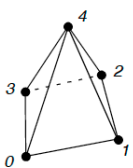
VTK_VOXEL (=11)



VTK_HEXAHEDRON (=12)



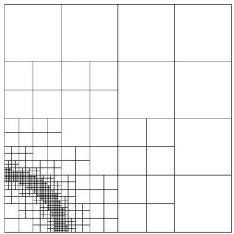
VTK_WEDGE (=13)



VTK_PYRAMID (=14)

Groupements de datasets

- Series temporelles
- Multi-block : Association de n'importe quels types de datasets
- Multi-piece : Association de datasets similaires - Souvent liés au calcul parallèle
- Hiérarchique - Ex AMR Dataset (Adaptative Mesh Refinement) : Collection de grilles régulières, à différent degrés de précision



Formats de fichier VTK et XML

Un grand nombre de formats de fichier pour les données 3D, dont :

- VTK - Visualization ToolKit
  - Destiné à la visualisation des données 3D
  - Ancien format texte (legacy) : **.vtk**. Nouveau format type xml, dépend du type de la grille : **.vti .vts, .vtu, .vtp**
  - Pour relire les données, il faut utiliser la librairie VTK. Elle contient un adaptateur vtk-numpy
  - *https://vtk.org/wp-content/uploads/2015/04/file-formats.pdf*
- XDMF + HDF5
  - Plus généraliste que VTK
  - Extension .xmf ou .xdmf
  - La description contenue dans le fichier .xmf permet aux outils de visu de lire les données contenues dans les fichiers hdf5
  - *http://www.xdmf.org/index.php/XDMF_Model_and_Format*

Structured Grid VTK

VTK Legacy

```
# vtk DataFile Version 2.0
Test simple rectilinear grid
ASCII

DATASET STRUCTURED_POINTS
DIMENSIONS 3 3 3
ORIGIN 0.0 0.0 0.0
SPACING 1 1 1

POINT_DATA 27
SCALARS scalarName int 1
LOOKUP_TABLE default
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
1 2 3 4 5 6 7 8 9
```

VTK XML

```
<VTKFile type="ImageData" version="0.1" byte_order="LittleEndian">
  <ImageData   WholeExtent="0 2 0 2 0 2"  Origin="0 0 0"
                                                Spacing="1 1 1">
    <Piece Extent="0 2 0 2 0 2">
      <PointData Scalars="scalarName">
        <DataArray type="Int32" Name="scalarName" format="ascii">
            1 2 3 4 5 6 7 8 9
            1 2 3 4 5 6 7 8 9
            1 2 3 4 5 6 7 8 9
        </DataArray>
      </PointData>
    </Piece>
  </ImageData>
</VTKFile>
```

Structured Grid XDMF

```
<?xml version="1.0" encoding="utf-8"?>
<Xdmf xmlns:xi="http://www.w3.org/2001/XInclude" Version="3.0">
  <Domain>
    <Grid Name="Grid" GridType=« Uniform">

      <Geometry Origin="" Type="ORIGIN_DXDYDZ">
        <DataItem DataType="Float" Dimensions="3" Format=« XML">
          0 0 0
        </DataItem>
        <DataItem DataType="Float" Dimensions="3" Format=« XML">
          1 1 1
        </DataItem>
      </Geometry>

      <Topology Dimensions="3 3 3" Type=« 3DCoRectMesh"/>

      <Attribute Center="Node" Name="scalar1" Type="Scalar">
        <DataItem DataType="Int" Dimensions="3 3 3" Format="HDF" Precision="4">
          h5filename.h5:scalar1
        </DataItem>
      </Attribute>

    </Grid>
  </Domain>
</Xdmf>
```

VTK Unstructured Grid

```
<VTKFile type="UnstructuredGrid" version="1.0" byte_order="LittleEndian"
header_type="UInt64">
  <UnstructuredGrid>
    <Piece NumberOfPoints="18" NumberOfCells="5">

      <Points>
        <DataArray type="Float32" Name="Points" NumberOfComponents="3" format=« ascii">
          0 0 0  1 0 0  2 0 0  0 1 0  1 1 0   2 1 0  0 0 1   1 0 1  2 0 1
          0 1 1  1 1 1  2 1 1  0 1 2  1 1 2  2 1 2  0 1 3  1 1 3  2 1 3
        </DataArray>
      </Points>

      <Cells>
        <DataArray type="Int64" Name="connectivity" format="ascii">
          0 1 4 3 6 7 10 9
          1 2 5 4 7 8 11 10
          6 10 9 12
          5 11 10 14
          15 16 17 14 13 12
        </DataArray>
        <DataArray type="Int64" Name="offsets" format="ascii">
          8 16 20 24 30
        </DataArray>
        <DataArray type="UInt8" Name="types" format="ascii">
          12 12 10 10 7
        </DataArray>
      </Cells>

    </Piece>
  </UnstructuredGrid>
</VTKFile>
```

Ecriture des fichiers VTK

- 1 - Le programme écrit directement le fichier. (print, template)
- 2 - Utilisation de la librairie VTK qui inclut des Writers. Il faut construire les objets vtk. VTK contient des outils pour les conversions numpy : vtk.numpy_interface ou vtk.utils.numpy_support
- 3 - Librairies Python « standalone » qui écrivent des fichiers vtk à partir des tableaux numpy.
  - pyeVTK - tout type de grilles sauf polydata, binaires
  - uvw - universal vtk writer - grilles structurées, binaire, ascii
  - meshio pour les grilles non structurées. Convertisseur très simple entre différents formats de maillages
- 4 - Surcouche python à vtk : pyvista (facilite aussi la lecture)
- 5 - Passer par des app. VTK plus complexes (mayavi, paraview/pvpython)

Ecriture des fichiers XDMF

- 1 - Comme pour VTK, solution basique : le programme écrit directement le fichier - en passant de préférence par un template.
- 2 - Utilisation de la librairie principale VTK qui inclut des Writers XDMF. Dans les anciennes versions, parfois pas accessible.
- 3 - Utilitaires python, seul meshio pour les grilles non structurées propose une sauvegarde xdmf/hdf5.

Remarques

Un peu plus d'outil pour la gestion des fichiers vtk que xdmf

xdmf : le fichier xdmf des méta-données s'ouvre facilement quelque soit le volume des données.

vtk : avec l'extension on sait quel type de données on traite

Attention, pas de polydata avec le xdmf (que du unstructured)

Lecture écriture de fichier, TP/DEMO

- Exemple d'écriture de fichiers vtk et xdmf à partir de données générées avec python (numpy)
- Utilisation des librairies vtk, pyevtk, et pyvista
- Exemple de lecture de fichiers vtk avec les librairies vtk et pyvista.

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation
- Quelques outils standards de visualisation 2D
- VTK et Paraview pour la visualisation 3D
- Visualisation distante

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

Gnuplot en 2 minutes (1)

- Logiciel libre, disponible facilement sur la plupart des systèmes,
- graph 2D ou 3D,
- en ligne de commande ou via des scripts,
- capable de lire des fichiers textes uniquement,
- logiciel assez simple à prendre en main pour les fonctionnalités de bases et très complet, offrant de nombreuses possibilités pour les post-traitement de données.

`http://www.gnuplot.info` - Doc facile à trouver, nombreux exemples et tutoriaux disponibles.

Gnuplot en 2 minutes (2)

Principe :

1. Définition d'un 'terminal', i.e. où sera effectué le rendu de la courbe
   - listes des terminaux disponibles :

     ```
     set terminal
     ```

   Quelques exemples : pdf, png, postscript, X11, latex ...
2. Choix (si nécessaire) du fichier de sortie

   ```
   set output "nom_du_fichier"
   ```
3. Séquence de commandes décrivant le plot

Exemples gnuplot

Exemple de lecture d'un fichier de données, sortie en pdf:

```
set terminal pdf
set output 'monfichier.pdf'
plot "data.txt" u 1:2
```

Sortie latex:

```
set terminal latex
set output 'image.tex'
plot [-3.14:3.14] sin(x)
```

Exécution d'un script

```
gnuplot -persist monscript.gp
```

Matplotlib en 2 minutes

- Package python, avec tous les avantages associés,
- graph 2D ou 3D (très complet pour les types de graph 2D),
- bonne documentation, nombreux exemples,
- très similaire à la visu sous Matlab,
- nombreux 'backends' disponibles

Pour aller plus loin

- page de matplotlib : *http://matplotlib.org*
- Les formations du groupe calcul CNRS *http://calcul.math.cnrs.fr/spip.php?article164*

Exemple

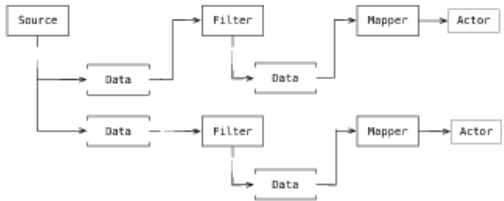
```
import matplotlib.pyplot as plt
import numpy as np
plt.ioff()
time = np.arange(0, 10)
x = np.random.rand(len(time))
plt.plot(time, x)
plt.xlabel('blabla')
plt.savefig('image.png')
plt.show()
```

VTK - the Visualisation ToolKit

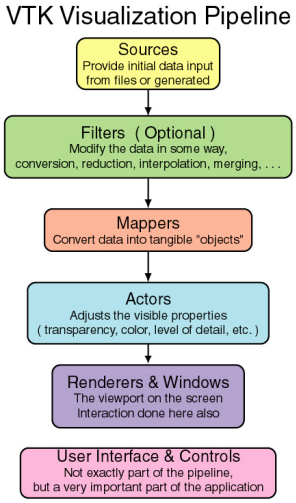
- Base de bon nombre de visualisations scientifiques 3D
- Bibliothèque C++ spécialisée dans la visualisation de données 2D/3D et le traitement d'image
- Développé à partir de 83 par le centre de recherche de General Electric (USA)
- Open source (licence BSD), mais maintenu et supporté par la société Kitware, qui propose aussi des formations
- Repose sur le standard openGL

VTK

- Plus de 1100 classes (code orienté objet), de nombreux algorithmes de visualisation —> puissant, mais pas toujours facile de trouver la bonne information
- Mapping python, java, C#
- Librairie javascript dérivée : vtk.js pour visu web
- Principe du pipeline : système de type « data-Flow », avec lazy update



VTK - Pipeline + système de rendu



vtkReader

vtkDataSet
vtkFilter

vtkMapper

vtkActor

vtkRenderer

vtkInteractor

(Le schéma provient de http://www.cs.uic.edu/~ibell/CS526/Tutorial/Tutorial.html#architecture)

VTK - Exemple python

```
#!/usr/bin/env python

import vtk
filename = "myfile.stl"

#reader
reader = vtk.vtkSTLReader()
reader.SetFileName(filename)

#Filters
cutter = vtkCutter()
vtkCutter.SetInputConnection(reader
.GetOutputPort())
....

#mapper
mapper = vtk.vtkPolyDataMapper()
mapper.SetInputConnection(cutter.Ge
tOutputPort())

# actor
actor = vtk.vtkActor()
actor.SetMapper(mapper)
```

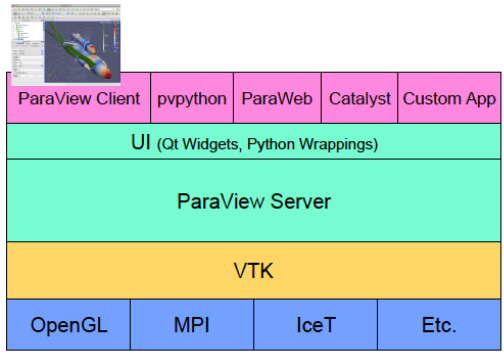
```
# Create a rendering window and
renderer
ren = vtk.vtkRenderer()
renWin = vtk.vtkRenderWindow()
renWin.AddRenderer(ren)

# Create an interactor
iren =
vtk.vtkRenderWindowInteractor()
iren.SetRenderWindow(renWin)

# Assign actor to the renderer
ren.AddActor(actor)

# Enable user interface interactor
iren.Initialize()
renWin.Render()
iren.Start()
```

VTK et les ressources machine

- Chargement des données (vtkReader): Toutes les données sont chargées en mémoire. Attention à la RAM
- Transformation des données, filtres : utilise du CPU, de la RAM pour les filtres nécessitant de dupliquer les données
- Calcul de l'image, phase de rendu : utilise le GPU, la carte graphique.

ParaView

Logiciel open source de visualisation de données basé sur VTK, développé principalement par le Sandia National Laboratories (USA) et la société Kitware, depuis 2000.



ParaView, vue globale de l'architecture

ParaView

- Architecture client server, permet de faire du traitement à distance
- Contient :
  - Interface graphique
  - Scripting python, tcl
  - Visualisation ‘A la volée’ (Catalyst)
- Comme VTK, utilise le parallélisme —> visualisation de données volumineuses
- Ajout possible de pluggins
- Lecture d’un grand nombre de formats de fichiers

VTK - ressources

- Point d'entrée du wiki : *http://www.vtk.org/Wiki/VTK*
- Exemples : *https://lorensen.github.io/VTKExamples/site/*
- Tutoriels : *http://www.vtk.org/Wiki/VTK/Tutorials*
- Doc technique : *http://www.vtk.org/doc/nightly/html/*
- Les sources : *https://gitlab.kitware.com/vtk/vtk*
- Datasets, description + schéma des classes :
  http://www.vtk.org/doc/nightly/html/classvtkDataSet.html
- Files format : *http://www.vtk.org/VTK/img/file-formats.pdf*

ParaView - ressources

- Télécharger le code et/ou des jeux de données :
  http://www.paraview.org/download/
- Télécharger le tutoriel et ses données :
  http://www.paraview.org/Wiki/The_ParaView_Tutorial
- Wiki : *http://www.paraview.org/Wiki/ParaView*
- Paraview python :
  http://www.paraview.org/Wiki/ParaView/Python_Scripting
- User guide, une version community
  http://www.paraview.org/paraview-guide/

  https://docs.paraview.org/en/latest/

TP et démo Paraview

TP/Démo VTK, Paraview.

Introduction

Contexte

- La visualisation est souvent nécessaire pour exploiter un résultat de simulation.
- Elle peut être 2D ou 3D et peut nécessiter une carte graphique puissante.

Contraintes

- Accès à des données volumineuses, fastidieuses à transférer,
- Visualisation en cours de calcul
  (pour contrôler le déroulement de la simulation, éventuellement intervenir dessus),
- Accès à des logiciels spécifiques pas toujours disponibles sur une station locale
  (licences, compatibilité de librairies,...)

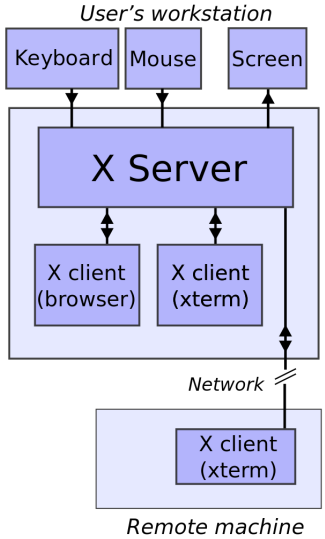
Solution

Avoir la possibilité de **visualiser à distance**.
$\longrightarrow$ **Lancer une application de visualisation sur un serveur distant disposant d'un accès direct aux données et aux librairies.**

2 Points à résoudre

- Accès à un système d'affichage ("display")
- Utilisation de l'accélération graphique

Le système graphique "X window system"



- X, X11, X window, X.org
- Architecture client/serveur,

Un environnement graphique a plusieurs niveaux:

- **X**: gère les écrans et périphériques (clavier/souris)
- **Window manager**: gère les fenêtres
- **Session manager**: gère l'environnement utilisateur (permet le glisser/déplacer par exemple)

Une variable d'environnement:

```
DISPLAY=hostname:display.screen
```

$\longrightarrow$ **Protocole réseau lourd,**
non adapté à l'accélération graphique (processeur local à la station, mais pas au serveur sur lequel tourne l'application)

Accélération graphique

- OpenGL : Un standard graphique permettant l'exploitation d'accélérateurs matériel
- Appels directs entre l'application et l'accélérateur matériel qui gère l'affichage, ce qui nécessite que l'application tourne sur la même machine que le serveur X !
- Si on veut déporter l'écran, il faut alors utiliser VirtualGL (méthode 3 vue plus loin)

TP/Démo visu distante

Vous pouvez maintenant passez au TPs/démo décrits dans le répertoire datavisu du projet outils-devel-tps.

1 Contexte et présentation du module
2 En amont, le Plan de Gestion des Données
3 Système de fichiers
4 Anticiper le post-traitement et la visualisation
5 Visualisation en calcul scientifique, quelques généralités
6 Structure et écriture des données
7 Outils de visualisation
8 Introduction aux Entrées-Sorties parallèles avec MPIIO
9 NetCDF et NCtools
10 HDF5 et H5tools
11 MPI-IO et ParallelHDF5
12 TPs

Entrées-Sorties parallèles : MPI-IO

Définition :

Interface de haut niveau de parallélisation des entrées-sorties pour éviter un goulet d'étranglement.

Fonctionnalités

- Gestion de fichiers
- Lecture/Ecriture individuelles et collectives

Mise en œuvre

1. Ouverture collective d'un fichier.
2. Définition sur les fichiers de une ou plusieurs vues par chaque processus, qui correspondent aux zones suceptibles d'être accédées par ce processus.
3. Appel aux sous-programmes spécifiques de lecture/écriture, bloquante ou non-bloquante, individuels ou collectifs.

Ecritures individuelles // via des déplacements explicites

Exemple simple en Fortran: version bloquante non collective MPI_File_write_at

```
program write_at
  use mpi_f08
  implicit none
  integer, parameter :: nb_valeurs=10
  integer :: i,rang,code,nb_octets_entier
  integer(kind=MPI_OFFSET_KIND) :: position_fichier
  integer, dimension(nb_valeurs) :: valeurs
  type(MPI_Status) :: statut
  type(MPI_File) :: fileid
  call MPI_Init(code)
  call MPI_Comm_rank(MPI_COMM_WORLD,rang,code)
  valeurs(:)= (/(i+rang*100,i=1,nb_valeurs)/)
  print *, "Ecriture processus",rang,":",valeurs(:)
  call MPI_File_open(MPI_COMM_WORLD,"donnees.dat",MPI_MODE_WRONLY+MPI_MODE_CREATE, MPI_INFO_NULL,fileid,code)
  call MPI_Type_size(MPI_INTEGER,nb_octets_entier,code)
  position_fichier=rang*nb_valeurs*nb_octets_entier
  call MPI_File_write_at(fileid,position_fichier,valeurs,nb_valeurs,MPI_INTEGER, statut,code)
  call MPI_File_close(fileid,code)
  call MPI_Finalize(code)
end program write_at
```

Il existe une version non bloquante (MPI_File_iwrite_at) et une version collective (MPI_File_write_at_all).

Ecritures individuelles // via des déplacements explicites

Exemple simple en C: version bloquante et collective MPI_File_write_at_all

```c
#include <stdio.h>
#include <mpi.h>
int main(int argc, char* argv[])
{
  int code = MPI_Init(&argc,&argv);
  int rang, i, nb_octets_entier;
  code = MPI_Comm_rank(MPI_COMM_WORLD,&rang);
  int nb_valeurs=10;
  int valeurs[nb_valeurs];
  for (int i=0;i<nb_valeurs;i++)
    valeurs[i]= i + rang * 100;
  printf("Ecriture processus %d : %d ... %d \n",rang, valeurs[0], valeurs[nb_valeurs-1]);
  MPI_File fileid;
  code = MPI_File_open(MPI_COMM_WORLD,"donnees.dat",MPI_MODE_WRONLY+MPI_MODE_CREATE, MPI_INFO_NULL,&fileid);
  code = MPI_Type_size(MPI_INT,&nb_octets_entier);
  MPI_Offset position_fichier;
  position_fichier=rang*nb_valeurs*nb_octets_entier;
  MPI_Status statut;
  code = MPI_File_write_at_all(fileid,position_fichier,&valeurs,nb_valeurs,MPI_INT,&statut);
  code = MPI_File_close(&fileid);
  code = MPI_Finalize();
}
```

Ecritures individuelles // via des déplacements explicites

```
> mpif90 -o writeat writeat.f90 # ou mpicc -o writeat writeat_all.c
> mpirun -np 2 ./writeat
Ecriture processus 1 : 101 102 103 104 105 106 107 108 109 110
Ecriture processus 0 :   1   2   3   4   5   6   7   8   9  10

> ls -l donnees.dat # 20 entiers de 4octets = 80 octets
-rw-r--r-- 1 lecoinal l-isterre 80 Dec  8 13:42 donnees.dat

> od -A d -i donnees.dat # affichage du contenu d'un fichier
                         # sous forme de nombres entiers (-i) et
                         # affiche les décalages dans la base décimale (-A d)
0000000           1           2           3           4
0000016           5           6           7           8
0000032           9          10         101         102
0000048         103         104         105         106
0000064         107         108         109         110
0000080
```

Ecritures individuelles // via des déplacements explicites

```
> od --endian=big -A d -i donnees.dat # swap input bytes according specified order
0000000    16777216    33554432    50331648    67108864
0000016    83886080   100663296   117440512   134217728
[...]
> od -A d -f donnees.dat # floats
0000000           1e-45           3e-45           4e-45           6e-45
0000016           7e-45           8e-45           1e-44         1.1e-44
[...]
```

Cette écriture dans un fichier binaire "maison" présente des inconvénients:

- aucune métadonnée associée au fichier (datatype, ...)
- pas de garantie de portabilité, ne facilite pas l'interopérabilité des données, nécessite de fournir une documentation liée à vos données, ...

⇒ **Utilisation de MPI-IO en "boite noire/wrapper" avec des librairies (par exemple HDF5 ou NETCDF)**

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

NetCDF

NetCDF (Network Common Data Format) est un format de fichier qui permet de stocker des variables à une ou plusieurs dimensions. On y trouve les variables avec leurs dimensions ainsi que les attributs. Exemple d'un champ de température:

```
netcdf example {  // example of CDL notation for a netCDF dataset
   dimensions:         // dimension names and lengths are declared first
           lat = 5, lon = 10, level = 4, time = unlimited;
   variables:          // variable types, names, shapes, attributes
           float   temp(time,level,lat,lon);
                       temp:long_name     = "temperature";
                       temp:units         = "celsius";
```

Cela permet donc de comprendre le sens de la variable et donne des informations pour pouvoir la lire et l'utiliser.

NetCDF

Le format NetCDF est à ce sens dit autodescriptif (contrairement à un fichier binaire écrit avec Fortran ou C)
Il est developpé par Unidata à l'UCAR (University Cooperation for Atmospheric Research):
https://www.unidata.ucar.edu/software/netcdf
Pour créer des fichiers NetCDF, on peut utiliser différents languages, C/C++/Fortran/Python. Chaque fichier NetCDF est portable d'une machine à l'autre et dispose de conventions pour le format.

NCtools

- ncdump
- nco : ncks, nccatted, ...
- ncgen
- ncrename
- nccopy

https://www.unidata.ucar.edu/software/netcdf/workshops/most-recent/utilities

NCtools

Voir le contenu du fichier netcdf

```
  ncdump -h simple_xy_par2.nc

netcdf simple_xy_par2 {
dimensions:
        x = 4 ;
        y = 4 ;
variables:
        int data(x, y) ;
        int data2(x, y) ;
}
```

NCtools

On peut utiliser des outils comme nco pour extraire une varaiable

```
ncks -v data2 simple_xy_par2.nc -o data2.nc
```

NCtools

```
  ncdump -h data2.nc
netcdf data2 {
dimensions:
        x = 4 ;
        y = 4 ;
variables:
        int data2(x, y) ;

// global attributes:
  :history = "Wed Apr  3 22:21:37 2019: ncks -v data2 simple_xy_par2.nc -o data2.nc" ;
  :NCO = "netCDF Operators version 4.7.6 (Homepage = http://nco.sf.net, Code = http://github.c
}
```

NCtools

On peut rajouter un attribut à une variable

```
 ncatted -a "unit",data2,a,c,"kg" data2.nc -o data2_att.nc
 ncdump -h data2_att.nc
netcdf data2_att{
dimensions:
        x = 4 ;
        y = 4 ;
variables:
        int data2(x, y) ;
                data2:unit = "kg" ;

// global attributes:
 :history = "Wed Apr  3 22:28:12 2019: ncatted -a unit,data2,a,c,kg data2.nc -o data2_att.nc
  Wed Apr  3 22:21:37 2019: ncks -v data2 simple_xy_par2.nc -o data2.nc" ;
 :NCO = "netCDF Operators version 4.7.6 (Homepage = http://nco.sf.net, Code = http://github.co
}
```

NCtools

On peut changer le nom d'un variable

```
 ncrename -v data2,data3 data2_att.nc data3.nc
 ncdump -h data3.nc
netcdf data3 {
dimensions:
        y = 4 ;
        x = 4 ;
variables:
        int data3(x, y) ;
                data3:unit = "kg" ;

// global attributes:
                :history = "Wed Apr  3 22:33:00 2019: ncrename -v data2,data3 data2_att.nc dat
 Wed Apr  3 22:28:12 2019: ncatted -a unit,data2,a,c,kg data2.nc -o data2_att.nc
 Wed Apr  3 22:21:37 2019: ncks -v data2 simple_xy_par2.nc -o data2.nc" ;
 :NCO = "netCDF Operators version 4.7.6 (Homepage = http://nco.sf.net, Code = http://github.co
}
```

NCtools

On peut convertir en un format txt .cdl

```
 ncdump data2_att.nc > data2_att.cdl
 more data2_att.cdl
netcdf data2_att{
dimensions:
        x = 4 ;
        y = 4 ;
variables:
        int data2(x, y) ;
                data2:unit = "kg" ;
}
```

NCtools

```
// global attributes:
 :history = "Wed Apr  3 22:44:52 2019: ncatted -a unit,data2,a,c,kg data2.nc -o data2_att.nc
  Wed Apr  3 22:21:37 2019:
  ncks -v data2 simple_xy_par2.nc -o data2.nc" ;
 :NCO = "netCDF Operators version 4.7.6 (Homepage = http://nco.sf.net, Code = http://github.co
data:

 data2 =
  1000, 1000, 1000, 1000,
  1001, 1001, 1001, 1001,
  1002, 1002, 1002, 1002,
  1003, 1003, 1003, 1003 ;
```

NCtools

Il est possible de créer le code en fortran ou en c qui permetterai de créer le fichier netcdf correspondant

```
ncgen -l f77 data2_att.cdl >data2_att.f
ncgen -l c data2_att.cdl >data2_att.c
```

NCtools

On peut générer un fichier netcdf à partir d'un txt

```
 ncgen  data2_att.cdl -o data2_att_ncgen.nc
 ncdump -h data2_att_ncgen.nc
netcdf data2_att_ncgen {
dimensions:
        x = 4 ;
        y = 4 ;
variables:
        int data2(x, y) ;
                data2:unit = "kg" ;

// global attributes:
 :history = "Wed Apr  3 22:44:52 2019: ncatted -a unit,data2,a,c,kg data2.nc -o data2_att.nc"
 "Wed Apr  3 22:21:37 2019: ncks -v data2 simple_xy_par2.nc -o data2.nc" ;
 :NCO = "netCDF Operators version 4.7.6 (Homepage = http://nco.sf.net, Code = http://github.co
}
```

NCtools

On peut faire la différence entre deux fichiers

```
 ncdiff data2_att_ncgen.nc data2_att.nc -o diff.nc
 ncdump diff.nc
netcdf diff {
dimensions:
        x = 4 ;
        y = 4 ;
variables:
        int data2(x, y) ;
                data2:unit = "kg" ;
}
```

NCtools

```
// global attributes:
 :history = "Wed Apr  3 22:52:38 2019: ncdiff data2_att_ncgen.nc data2_att.nc -o diff.nc",
 "Wed Apr  3 22:44:52 2019: ncatted -a unit,data2,a,c,kg data2.nc -o data2_att.nc",
 "Wed Apr  3 22:21:37 2019: ncks -v data2 simple_xy_par2.nc -o data2.nc" ;
 :NCO = "netCDF Operators version 4.7.6 (Homepage = http://nco.sf.net, Code = http://github.co
data:

 data2 =
  0, 0, 0, 0,
  0, 0, 0, 0,
  0, 0, 0, 0,
  0, 0, 0, 0 ;
```

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

HDF5 : Définition

Hierarchical Data Format
Il s'agit d'un autre format de données “intelligent“ pour stocker et organiser des données.
C'est à la fois:

- une API (interface de programmation applicative)
- un modèle de données
- un format de fichier

... pour la gestion des entrées-sorties.

HDF5 : Caractéristiques

- Modèle de données auto-descriptif permettant la gestion de dataset complexe
- Format de données portable
- Format binaire, avec possibilité de compression additionnelle (avec ou sans perte) sur les datasets
- h5tools : outils en ligne de commande pour la gestion des fichiers et données HDF5
- Supporte les entrée-sorties parallèles (si associé à MPI-IO)
- API en C, C++, Fortran 90, Python (et Java, MatLab, ...)

https://portal.hdfgroup.org/display/HDF5
https://docs.hdfgroup.org/hdf5/develop/
http://docs.h5py.org/en/stable/

HDF5 : Modèle de données auto-descriptif, portable et interopérable

- Possibilité d'organiser les datasets de façon hiérarchique, de créer des groupes (analogue à un système de fichier Linux/UNIX)

```
> h5ls -r fichier.h5
/                        Group
/grp1                    Group
/grp1/dset1              Dataset {59, 2536}
/grp1/dset2              Dataset {59, 2536}
/grp2/dset1              Dataset {59, 2536}
/grp3/subgrp/dset1       Dataset {4, 5, 6}
[...]
```

HDF5 : Modèle de données auto-descriptif, portable et interopérable

- Possibilité d'ajouter des métadonnées à un groupe ou à un dataset

```
> h5ls -rv ufinal.h5
/                        Group
    Attribute: domain\ lengths {2}
        Type:      native double
        Data:  1, 1
    Attribute: origins {2}
        Type:      native double
        Data:  0, 0
/u                       Dataset {128/128, 128/128}
[...]
```

HDF5 : Modèle de données auto-descriptif, portable et interopérable

Le type de données et l'endianness de chaque dataset sont renseignés et stockés automatiquement à l'écriture des données, dans le header de chaque dataset. L'utilisateur n'a pas à remplir manuellement ces "métadonnées".

```
> h5ls -rv fichier.h5
/                        Group
/grp1                    Group
/grp1/dset1              Dataset {59, 2536}
   Type:      native int
/grp1/dset2              Dataset {59, 2536}
   Type:      native double

> h5dump -H -d /grp1/dset2 fichier.h5
HDF5 "fichier.h5" {
DATASET "/grp1/dset2" {
   DATATYPE  H5T_IEEE_F64LE
   DATASPACE  SIMPLE { ( 59, 2536 ) / ( 59, 2536 ) }
```

HDF5 : Organisation des données

- Possibilité de créer des liens vers un dataset d'un autre fichier HDF5 (modif noms, groupes ...)

```
h5f = h5py.File("out.h5","w")
h5f["/B946.EHZ"] = h5py.ExternalLink("/bettik/a/B946.h5", "EHZ")
h5fi = h5py.File("/bettik/b/SG.h5","r")
for name in h5fi:
  h5f["/"+name] = h5py.ExternalLink("/bettik/b/SG.h5", name)
h5fi.close()
h5f.close()
```

```
> h5ls out.h5
B946.EHZ     External Link {/bettik/a/B946.h5/EHZ}
R0101.EPZ    External Link {/bettik/b/SG.h5/R0101.EPZ}
[...]
R7010.EPZ    External Link {/bettik/b/SG.h5/R7010.EPZ}
> ls -lh out.h5 /bettik/b/SG.h5 /bettik/a/B946.h5
-r--r--r-- 1 lecointre l-isterre 6.3M Aug 20 15:55 /bettik/a/B946.h5
-r--r--r-- 1 lecointre l-isterre  28G May  7  2017 /bettik/b/SG.h5
-rw-r--r-- 1 lecointre l-isterre 176K Aug 22 15:46 out.h5
```

HDF5 : Compression sur les datasets

Les données sont écrites en binaire dans les fichiers HDF5. On peut y ajouter différents filtres de compression (avec ou sans perte). Chaque dataset peut avoir son propre filtre.

```
> h5ls -rv out.h5
/01641                   Group
    Attribute: SVNRevision scalar
        Type:      17-byte null-terminated ASCII string
        Data:  "$Revision: 148 $"
/01641/deplacement_horizontal Dataset {59/59, 2536/2536}
  Storage:   598496 logical bytes, 598496 allocated bytes, 100.00% utilization
  Type:      native int
/01641/deplacement_vertical   Dataset {59/59, 2536/2536}
  Storage:   598496 logical bytes, 161746 allocated bytes, 370.02% utilization
  Filter-0:  deflate-1 OPT {6}
  Type:      native int
/01641/energie_fenetre        Dataset {59/59, 2536/2536}
  Storage:   1196992 logical bytes, 630557 allocated bytes, 189.83% utilization
  Filter-0:  scaleoffset-6 OPT {0, 2, 129336, 1, 8, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0
  Type:      native double
```

H5tools

- h5ls et h5dump: explorer un fichier HDF5, afficher son contenu
- h5copy : recopier un dataset ou un groupe, éventuellement sous un nouveau nom, éventuellement dans un nouveau fichier HDF5
- h5diff : comparer et afficher les différences entre 2 fichiers HDF5 (ou entre 2 datasets)
- h5repack : recopier un dataset en appliquant un filtre (chunking, compression, ...)
- h5pfc/h5pcc : wrappers pour compilateurs Fortran/C avec MPI et ParallelHDF5

H5tools : h5ls

h5ls : affiche le contenu du premier niveau hiérarchique du fichier HDF5

```
> h5ls imvort.h5
00001                    Group
[...]
01641                    Group
[...]
99999                    Group
```

H5tools : h5ls

Affichage récursif:

```
> h5ls -r imvort.h5
/                            Group
[...]
/01641                       Group
/01641/deplacement_horizontal Dataset {59, 2536}
/01641/deplacement_vertical   Dataset {59, 2536}
/01641/energie_fenetre        Dataset {59, 2536}
/01641/nb_corre               Dataset {59, 2536}
[...]

> h5ls imvort.h5/01641
deplacement_horizontal   Dataset {59, 2536}
deplacement_vertical     Dataset {59, 2536}
energie_fenetre          Dataset {59, 2536}
nb_corre                 Dataset {59, 2536}
```

H5tools : h5ls

Affichage verbeux:

```
> h5ls -rv imvort.h5
[...]
/01641                   Group
    Attribute: SVNRevision scalar
      Type:      17-byte null-terminated ASCII string
      Data:  "$Revision: 148 $"
    Location:  1:1464
    Links:     1
/01641/deplacement_horizontal Dataset {59/59, 2536/2536}
    Location:  1:1576
    Links:     1
    Storage:   598496 logical bytes, 598496 allocated bytes, 100.00% utilization
    Type:      native int
[...]
```

Voir la documentation: “man h5ls“ ou “h5ls -h“

H5tools : h5dump

h5dump : Affiche le contenu d'un fichier HDF5:

```
> h5dump imvort.h5
[...]
DATASET "deplacement_horizontal" {
   DATATYPE  H5T_STD_I32LE
   DATASPACE  SIMPLE { ( 59, 2536 ) / ( 59, 2536 ) }
   DATA {
   (0,0): -700, -700, -700, -700, -700, -700, -700, -700, -700, -700,
   (0,10): -700, -700, -700, -700, -700, 0, -2, -22, -8, -12, -17, -23,
   [...]
   (58,2517): -2, -3, -6, -8, -7, -5, -3, -2, -2, -1, -1, -1, -1, -1, -1, -1,
   (58,2533): -1, -1, -1
   }
}
```

Voir la documentation: “man h5dump“ ou “h5dump -h“

H5tools : h5dump

h5dump : Affichage par sous-blocs (offset, count):

```
> h5dump -d /01641/deplacement_horizontal -s "58,2517" -c "1,4" imvort.h5
HDF5 "imvort.h5" {
DATASET "/01641/deplacement_horizontal" {
   DATATYPE  H5T_STD_I32LE
   DATASPACE  SIMPLE { ( 59, 2536 ) / ( 59, 2536 ) }
   SUBSET {
      START ( 58, 2517 );
      STRIDE ( 1, 1 );
      COUNT ( 1, 4 );
      BLOCK ( 1, 1 );
      DATA {
      (58,2517): -2, -3, -6, -8
      }
   }
}
}
```

H5tools : h5copy

```
> h5copy -f noattr -p -i imvort.h5 -o tmp.h5 -s "/01641" -d "/A"
> h5ls -r tmp.h5
/                        Group
/A                       Group
/A/deplacement_horizontal Dataset {59, 2536}
/A/deplacement_vertical   Dataset {59, 2536}
/A/energie_fenetre        Dataset {59, 2536}
/A/nb_corre               Dataset {59, 2536}
```

Voir la documentation: “man h5copy“ ou “h5copy -h“

H5tools : h5repack

```
> h5repack -v -f /A/deplacement_vertical:GZIP=6 -f /A/energie_fenetre:SZIP=8,NN tmp.h5 out.h
Objects to modify layout are...
Objects to apply filter are...
 </A/deplacement_vertical> with GZIP filter
 </A/energie_fenetre> with SZIP filter
Opening file <imvort.h5>. Searching for objects to modify...
 </A/deplacement_vertical>...Found
 </A/energie_fenetre>...Found
Making file <out.h5>...
----------------------------------------
 Type      Filter (Compression)     Name
----------------------------------------
 group                          /
 group                          /A
 dset                           /A/deplacement_horizontal
 dset      GZIP    (3.700:1)    /A/deplacement_vertical
 dset      SZIP    (1.183:1)    /A/energie_fenetre
 dset                           /A/nb_corre
```

H5tools : h5repack

```
> h5ls -rv out.h5
[...]
/A/deplacement_vertical Dataset {59/59, 2536/2536}
    Storage:   598496 logical bytes, 161746 allocated bytes, 370.02% utilization
    Filter-0:  deflate-1 OPT {6}
    Type:      native int
/A/energie_fenetre   Dataset {59/59, 2536/2536}
    Storage:   1196992 logical bytes, 1011733 allocated bytes, 118.31% utilization
    Filter-0:  szip-4 OPT {169, 8, 64, 1024}
    Type:      native double
[...]
> ls -l tmp.h5 out.h5
-rw-r--r-- 1 lecointre l-isterre  3595168 Apr  2 16:05 tmp.h5
-rw-r--r-- 1 lecointre pr-resolve 2979319 Apr  2 16:42 out.h5
```

H5tools : h5diff

```
> h5diff -v tmp.h5 out.h5
file1     file2
    x       x    /A/deplacement_horizontal
    x       x    /A/deplacement_vertical
    x       x    /A/energie_fenetre
    x       x    /A/nb_corre
dataset: </A/deplacement_horizontal> and </A/deplacement_horizontal>
0 differences found
dataset: </A/deplacement_vertical> and </A/deplacement_vertical>
0 differences found
dataset: </A/energie_fenetre> and </A/energie_fenetre>
0 differences found
dataset: </A/nb_corre> and </A/nb_corre>
0 differences found
```

H5tools : h5repack et h5diff

```
> h5repack -v -f /A/deplacement_vertical:GZIP=6 -f /A/energie_fenetre:SOFF=2,DS tmp.h5 out.h
[...]
 dset     GZIP    (3.700:1)    /A/deplacement_vertical
 dset     SCALEOFFSET    (1.898:1) /A/energie_fenetre

> h5diff -v tmp.h5 out.h5
[...]
dataset: </A/energie_fenetre> and </A/energie_fenetre>
size:            [59x2536]            [59x2536]
position         energie_fenetre energie_fenetre difference
----------------------------------------------------------
[ 0 15 ]           0.00242615      0               0.00242615
[ 0 16 ]           0.0413142       0.0401086       0.00120557
[ 0 17 ]           0.148966        0.150109        0.0011425
[...]
148737 differences found
```

H5tools : h5diff

h5diff : on peut autoriser une tolérance absolue (-d) ou relative (-p):

```
> h5diff tmp.h5 out.h5
dataset: </A/energie_fenetre> and </A/energie_fenetre>
148737 differences found

> h5diff -d 1e-3 tmp.h5 out.h5
dataset: </A/energie_fenetre> and </A/energie_fenetre>
119232 differences found

> h5diff -d 1e-2 tmp.h5 out.h5
>
```

1 Contexte et présentation du module

2 En amont, le Plan de Gestion des Données

3 Système de fichiers

4 Anticiper le post-traitement et la visualisation

5 Visualisation en calcul scientifique, quelques généralités

6 Structure et écriture des données

7 Outils de visualisation

8 Introduction aux Entrées-Sorties parallèles avec MPIIO

9 NetCDF et NCtools

10 HDF5 et H5tools

11 MPI-IO et ParallelHDF5

12 TPs

Ecriture collective : MPI-IO pHDF5 : write1D

```c
#include "hdf5.h"
#include "mpi.h"

#define FILENAME        "out.h5"
#define DSETNAME        "u"
#define DIM0            10

int main (int argc, char *argv[])
{
  // == Initialize MPI context ==
  int code = MPI_Init(&argc, &argv);
  // Get rank of current process in MPI_COMM_WORLD
  // and the number of involved processes.
  int rank, nbprocs;
  MPI_Comm comm = MPI_COMM_WORLD;
  code = MPI_Comm_rank(comm, &rank);
  code = MPI_Comm_size(comm, &nbprocs);

  // allocation: each MPI processus defines its own array of 10 int
  int datawrite[DIM0];
  for(int i = 0; i< DIM0 ; ++i)
    {
      datawrite[i] = 10*rank+i;  // {0,...,9} sur P0 et {10,...,19} sur P1
    }
  printf("[%d] %d ... %d\n", rank, datawrite[0], datawrite[DIM0-1]);

```

MPI-IO pHDF5 : write1D

```
27  // Initialize C interface for HDF5
28  herr_t errcode = H5open();
29  // - Create the file with parallel file access -
30  // Properties of the file (w mode)
31  hid_t plist_id = H5Pcreate(H5P_FILE_ACCESS);
32  // Set file properties
33  // Property : collective opening (by all mpi processes) of the file
34  MPI_Info info  = MPI_INFO_NULL;
35  errcode = H5Pset_fapl_mpio(plist_id, comm, info);
36  // Create file
37  hid_t file_id = H5Fcreate(FILENAME, H5F_ACC_TRUNC, H5P_DEFAULT, plist_id);
38  // Close properties
39  errcode = H5Pclose(plist_id);
40
41  // Describe a 1D array of elements on this MPI processor
42  hsize_t count[1] = {DIM0};
43  hid_t memspace = H5Screate_simple(1, count, NULL);
44  // - Map this processor's elements into the shared file -
45  hsize_t dimsfi[1] = {nbprocs * DIM0};
46  hid_t filespace = H5Screate_simple(1, dimsfi, NULL);
47  // Create the dataset with default properties
48  hid_t dset_id = H5Dcreate(file_id, DSETNAME, H5T_NATIVE_INT, filespace, H5P_DEFAULT, H5P_DEFAULT, H5P_DEFAULT);
49  hsize_t offset[1] = {rank * DIM0};
50  errcode = H5Sselect_hyperslab(filespace, H5S_SELECT_SET, offset, NULL, count, NULL);
51
52  // Enable collective buffering to write collectively in dataset
53  hid_t dxpl_id = H5Pcreate(H5P_DATASET_XFER);
54  errcode = H5Pset_dxpl_mpio(dxpl_id, H5FD_MPIO_COLLECTIVE);
55  // Write the dataset collectively
56  errcode = H5Dwrite(dset_id, H5T_NATIVE_INT, memspace, filespace, dxpl_id, datawrite);
57
```

MPI-IO pHDF5 : write1D

```
58    // Close resources
59    errcode = H5Pclose(dxpl_id);
60    errcode = H5Dclose(dset_id);
61    errcode = H5Sclose(filespace);
62    errcode = H5Sclose(memspace);
63    errcode = H5Fclose(file_id);
64
65    // Close C interface for HDF5
66    errcode = H5close();
67
68    // === Close MPI ===
69    code = MPI_Finalize();
70
71 }
```

Environnement avec MPI + ParallelHDF5

```
  > source /applis/site/guix-start.sh
  > refresh_guix ced_io
  > guix package -p $GUIX_USER_PROFILE_DIR/ced_io -I
nco                         4.9.8         out             /gnu/store/3avqnwjrsqz2pbdg1fqmj9hv
netcdf                      4.7.4         out             /gnu/store/2hfsmfzxj33izz2w4xcp7fkx
zlib                        1.2.11        out             /gnu/store/8qv5kb2fgm4c3bf70zcg9l6h
gfortran-toolchain          10.3.0        out             /gnu/store/glkrwmfxhqq11j526isr6iqs
gcc-toolchain               11.3.0        out             /gnu/store/y26qpm6anmwx10p6as563v44
openmpi                     4.1.3         out             /gnu/store/wl70bxn62spd6i059fgs1sq9
hdf5-parallel-openmpi        1.10.7        fortran         /gnu/store/4ynzmgazim1gm1cyknr484h
hdf5-parallel-openmpi        1.10.7        out             /gnu/store/mdj7g931xrsj5v93bpbz45r
python-mpi4py               3.0.3         out             /gnu/store/31kcljbfirzn2jzs8y5cjr59
python-h5py-mpi             2.10.0        out             /gnu/store/bzwpwywpd5296y8msml7bhn6
python-matplotlib           3.5.1         out             /gnu/store/inlr765y2qwvlkr5f8zzdhrp
python                      3.9.9         out             /gnu/store/sz7lkmic6qrhfblrhaqaw0fg
cmake                       3.21.4        out             /gnu/store/nvqiq707xrc3blc044l9mpaa
```

Environnement avec MPI + ParallelHDF5

```
  > which h5pcc
/var/guix/profiles/per-user/<yourlogin>/ced_io/bin/h5pcc

  > h5pcc --show
gcc -L/gnu/store/mdj7g931xrsj5v93bpbz45rxb2agf1qj-hdf5-parallel-openmpi-1.10.7/lib /gnu/stor

  > which gcc
/var/guix/profiles/per-user/<yourlogin>/ced_io/bin/gcc
```

MPI-IO pHDF5 : write1D

```
> h5pcc -o write1D write1D.c
> mpirun -n 4  ./write1D
[0] 0 ... 9
[2] 20 ... 29
[1] 10 ... 19
[3] 30 ... 39

> h5ls -v out.h5
Opened "out.h5" with sec2 driver.
IntArray                 Dataset {40}
Location:  1:800
Links:     1
Storage:   160 logical bytes, 160 allocated bytes, 100.00% utilization
Type:      native int

> ls -l out.h5
-rw-r--r-- 1 lecoinal ondes 2304 avril  5 15:11 out.h5
```

Environnement avec MPI + ParallelHDF5

```
  > which h5pfc
/var/guix/profiles/per-user/<yourlogin>/ced_io/bin/h5pfc

  > h5pfc --show
gfortran -I/gnu/store/mdj7g931xrsj5v93bpbz45rxb2agf1qj-hdf5-parallel-openmpi-1.10.7/include

  > which gfortran
/var/guix/profiles/per-user/<yourlogin>/ced_io/bin/gfortran
```

MPI-IO pHDF5 : write1D

```
  > h5pfc -shlib -I$GUIX_PROFILE/include -o wr1D write1D.f90
  > mpirun -n 4  ./wr1D
 [ 2 ]  20 21 22 23 24 25 26 27 28 29
 [ 3 ]  30 31 32 33 34 35 36 37 38 39
 [ 0 ]   0  1  2  3  4  5  6  7  8  9
 [ 1 ]  10 11 12 13 14 15 16 17 18 19
  > h5dump out.h5
HDF5 "out.h5" {
GROUP "/" {
   DATASET "u" {
      DATATYPE  H5T_STD_I32LE
      DATASPACE  SIMPLE { ( 40 ) / ( 40 ) }
      DATA {
      (0): 0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18,
      (19): 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34,
      (35): 35, 36, 37, 38, 39
      }
   }
}
```

1 Contexte et présentation du module
2 En amont, le Plan de Gestion des Données
3 Système de fichiers
4 Anticiper le post-traitement et la visualisation
5 Visualisation en calcul scientifique, quelques généralités
6 Structure et écriture des données
7 Outils de visualisation
8 Introduction aux Entrées-Sorties parallèles avec MPIIO
9 NetCDF et NCtools
10 HDF5 et H5tools
11 MPI-IO et ParallelHDF5
12 TPs

TPs

Vous pouvez passer aux différents tps pour manipuler MPI-IO et HDF5 dans un contexte de lecture / écriture collectives de datasets 1D / 2D.