Entrées/sorties en calcul scientifique

Romaric David
david@unistra.fr
Direction Informatique
25 Septembre 2014

UNIVERSITÉ DE STRASBOURG

Plan

- Introduction
- Hdf5
- TP
- Fonctionnalités avancées
- Conclusion

Introduction

- Centre de calcul de l'Université de Strasbourg
- http://hpc.unistra.fr
- 300 Serveurs HPC, Linux, Infiniband, ...
- Soutien à la recherche par :
  - la mise à disposition de ressources de calcul ;
  - l'accompagement sur l'utilisation de ces ressources
  - la formation aux utilisateurs
- Équipe de 3 personnes

Introduction

- Les codes de calcul utilisent ou produisent des données
- Ces données doivent pouvoir être traitées :
  - immédiatement, par l'auteur pour la publication
  - plus tard, par l'auteur
  - quelques temps après la publication, pour d'autres
- Les données se doivent donc d'être portables entre systèmes (ce n'est pas le cas par défaut), compactes (big data !) et documentées,
- Pour cela, il faut faire quelques efforts

Introduction

- Les données écrites dans le format le plus portable que l'on puisse imaginer sont écrites en mode texte :
  - Lisible universellement
  - Très volumineux. 1 double précision = 16 chiffres = 16 octets au lieu de 8 (64 bits)
- Pour disposer d'un format compact, on peut écrire directement en mode binaire (= copie de la mémoire)
  - Compact
  - Pas du tout universel

Introduction

- Dans notre liste au père noël, nous souhaitons donc un format et binaire et portable et documenté
- Il existe des solutions !
- À ce jour, une grande partie de ces solutions s'appuie sur Hdf5, objet de ce cours

Plan

- Introduction
- Hdf5
- TP
- Fonctionnalités avancées
- Conclusion

HDF5

- Hierarchical Data Format 5 (http://www.hdfgroup.org/HDF5) est une bibliothèque permettant de réaliser des i/o pour les codes scientifiques (notez que l'usage se répand)
- Fonctionnellement équivalent à read et write mais :
  - Fichiers portables entre machines et systèmes
- Fonctionnalités avancées :
  - Description des données (dimensionnalité, commentaires) : dataspace
  - Compression des données à la volée
  - Compréhensible par des outils de visualisation

HDF5

- Les données sont nommées : chaque dataset (le container des données) a un nom
- Les données sont organisées : les datasets sont organisées en arborescence, à la manière d'un système de fichiers :

```
/groupe/sous_groupe/[....]/dataset1
/groupe/sous_groupe/[....]/dataset2
```

- API sophistiquée en C,C++, Fortran 90
- Disponible en python via h5py et pytables

HDF5 – et plus encore

- Les fichiers Hdf5 sont visualisables avec Visit, Paraview et d'autres outils
- Vous n'avez aucune contraintes sur les noms des groupes, datasets, etc....
- HDF5 est donc une très bonne bibliothèque de stockage bas un niveau, pas un format de fichier à proprement parler. Si on vous donne un fichier Hdf5 sans le lexique des Datasets, vous serez perdus
- Il existe des formats de fichiers haut niveau spécifiques qui utilisent Hdf5 (NetCDF, Silo, Cgns)

HDF5 – En pratique

- Le package hdf5 comprend différents wrappers dont :
  - `h5cc`
  - `h5c++`
  - `h5ffc`
- Permettent d'ajouter les chemins vers les fichiers d'en-tête et les bibliothèques
- Pour voir ce que font ces wrappers :
  - `h5cc -show`
- Pour afficher en mode texte un fichief hdf5 :
  - `h5dump`

Plan

- Introduction
- Hdf5
- TP
- Fonctionnalités avancées
- Conclusion

HDF5 – TP1 - 01_write.c

Nous écrivons un tableau bi-dimensionnel dans un fichier. Nous décidons de le placer dans un groupe (comme le serait un fichier dans un répertoire)

- Première étape : Créer le fichier

```
file_id = H5Fcreate(FILE, H5F_ACC_TRUNC,
  H5P_DEFAULT, H5P_DEFAULT);
```

Propriétés de création du fichier
Laissez les défauts !

Propriétés d'accès au fichier
Laissez les défauts !

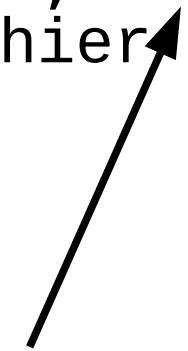
- 2ème étape : Indiquer l'emplacement du groupe

```
groupe_id = H5Gcreate(file_id, "/mon_groupe",
H5P_DEFAULT,H5P_DEFAULT,H5P_DEFAULT);
```

- Pas mal de propriétés par défaut pour un groupe
- À quoi sert tout ce charabia ?
  - On peut enrichir un groupe de commentaires afin d'aider par la suite à la compréhension du fichier
  - Un élément de l'organisation des données
  - Les paramètres ayant ici une valeur par défaut sont optionnels en F90 (et en python !)

- Vous pouvez ensuite décrire le groupe pour expliquer les raisons de sa création :

```
H5Gset_comment(groupe_id, ".","Exemple de groupe simple de mon fichier hdf5");
```



Emplacement par rapport à `groupe_id`

- Une fois crée un emplacement dans le fichier, nous allons l'utiliser pour y placer un dataset
- Tout comme un répertoire peut contenir plusieurs fichiers, un groupe peut contenir plusieurs datasets
- Il faut auparavant décrire les dimensions des données à stocker = dataspace

```
dataspace_id= H5Screate_simple(2, dims, NULL);
```

- Il s'agit simplement d'un descripteur de dimensions
- Dernier argument : liste des dimensions max dans chaque direction

- Une fois les dimensions créees, il ne reste qu'à indiquer où on écrira les données dans le fichier = créer le dataset

```
dataset_id=H5Dcreate(file_id,"/mon_groupe/dset",H5
  T_NATIVE_INT,
  dataspace_id,H5P_DEFAULT,H5P_DEFAULT,H5P_DEFAULT
  );
```

- Paramètres importants :
  - emplacement du dataset dans le fichier
  - `H5T_NATIVE_INT` : le type de données

    (la première fois où on l'indique)

- Une fois que l'on a décrit le dataset (= dimensions + type de données + emplacement dans le fichier), il ne reste qu'à l'écrire (1 ligne) :

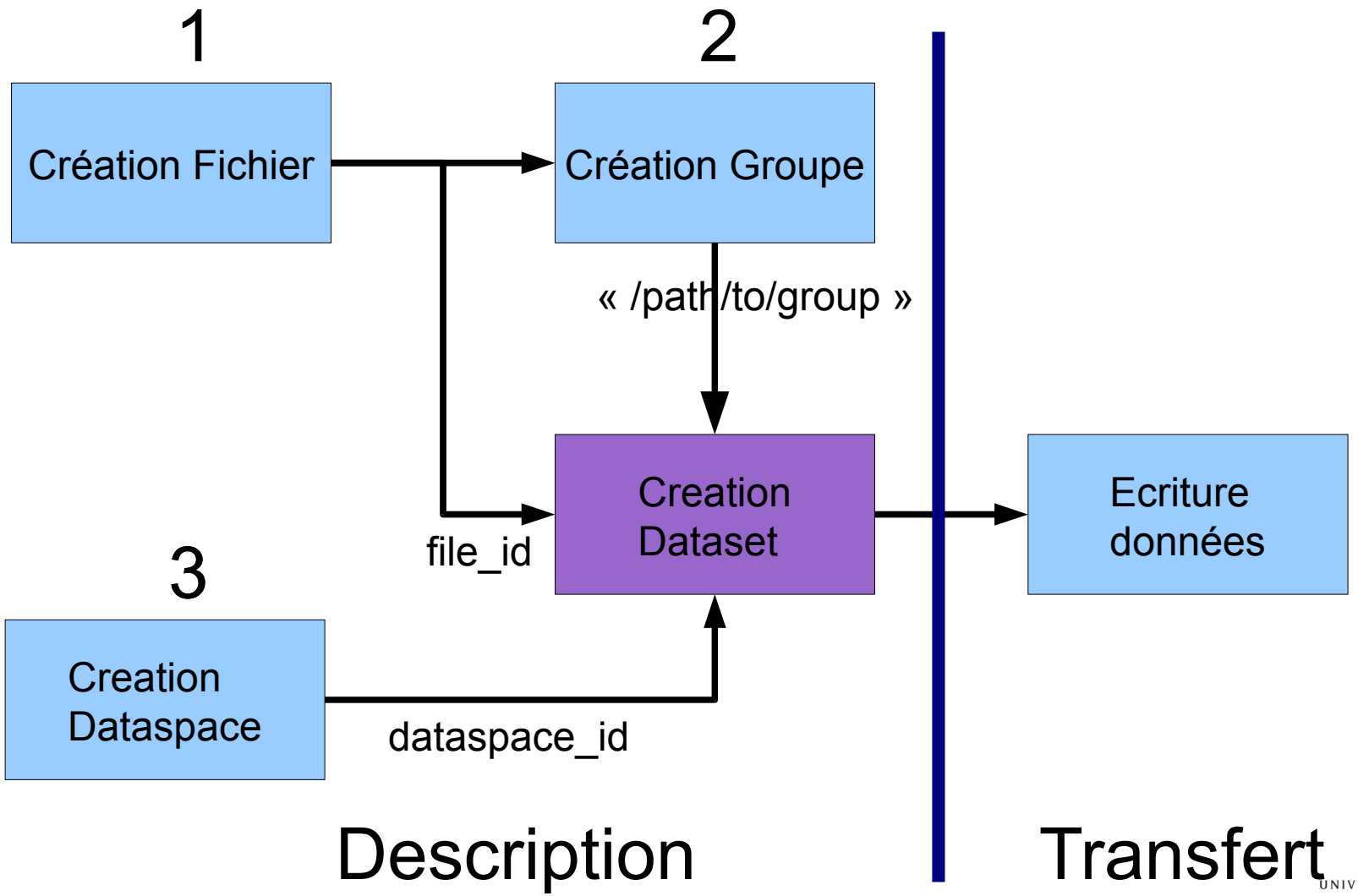
```
H5Dwrite (dataset_id, H5T_NATIVE_INT, H5S_ALL,
  H5S_ALL, H5P_DEFAULT, dset_data);
```

- `H5T_NATIVE_INT` : type des données **en mémoire**

  À repréciser car HDF5 peut convertir les types

- `H5S_ALL` : l'étendue des données que l'on transfère de la mémoire et vers le fichier. Ici : tout
- `dset_data` : les données en mémoire (pointeur)

HDF5 – Le chemin



HDF5 - TP1

- Dans le fichier, augmentez la taille du tableau et reportez les modifications lors de la création du Dataspace
- Exécutez à nouveau le programme
- Notez que pour vous n'avez aucun besoin de modifier le programme externe utilisé pour lire les données

HDF5 - TP2

- Nous allons à présent écrire des données compressées
- Pour cela, nous allons manipuler les propriétés du dataset
- La compression travaille sur des blocs (chunks) de données, il nous faut définir la taille des blocs
- Quelle taille choisir ?

  « ni trop grande, ni trop petite »

- Dans le code précédent, placez les lignes suivantes juste avant la création du Dataset :

```
properties = H5Pcreate(H5P_DATASET_CREATE);
/* Taille du chunk */
chunk_dims[0]=100;
chunk_dims[1]=100;

H5Pset_chunk(properties, 2, chunk_dims);
/* 9 : niveau de compression le plus eleve */
H5Pset_deflate(properties, 9);
```

HDF5-TP2

- Il vous faut ensuite **associer les propriétés** au Dataset

```
dataset_id = H5Dcreate(file_id,
  "/mon_groupe/dset",H5T_NATIVE_INT,
  dataspace_id,
H5P_DEFAULT,properties,H5P_DEFAULT);
```

- Nous avons modifié les propriétés de **création** du Dataset
- Que remarquez-vous à l'exécution du programme ?
- Affichez les données avec `h5dump`

HDF5 - TP3

- Pour l'instant dans notre fichier nous avons stocké un jeu de données...
- Tout ça pour ça...
- Oui mais :
  - Nous avons compressé un dataset avec peu d'efforts
  - Nous n'avons as eu à écrire le lecteur spéficique à notre format
  - Nous avons décrit nos données (ce qui constitue toujours un bon exercice)
- Il est possible de stocker plein de datasets différents dans un fichier

HDF5 - TP3

- Le programme 01-write-several.c comprend à présent 2 tableaux
- Ajoutez les instructions HDF5 pour écrire le dataset dans le groupe précédent
- Le type de données correspondant est `H5T_NATIVE_FLOAT`

HDF5 - TP4

- Comme l'enchaînement des étapes nécessaires dans HDF5 est un peu complexe, il existe des raccourcis permettant la création de datasets simples
- Le seul effort de description à faire consiste à donner les dimensions des tableaux de données
- Ces fonctions sont regroupées dans l'API *lite*
- `http://www.hdfgroup.org/HDF5/Tutor/h5lite.html`

HDF5 – TP4

- Nous utilisons le fichier `02-read-lite.c`
- Dans ce fichier, indiquez le nom du dataset que nous voulons lire depuis le fichier
- Compilez et testez le programme.
- Comparez la sortie à celle de `h5dump`
- Quels sont les pré-supposés qui sont faits dans ce programme ?
- Une lecture complète sans connaissance à priori du contenu du fichier vous est proposée dans `03-complete-read.c`

Plan

- Introduction
- Hdf5
- TP
- Fonctionnalités avancées
- Conclusion

Foncionnalités avancées

Hyperslabs

- Supposez que vos Datasets représentent des tableaux multi-dimensionnels
- Vous pouvez extraire des tranches de tableau avant de les sauvegarder ⇒ gain de temps et de place à l'écriture
- Cela est décrit par les *hyperslabs* de Hdf5
- Non détaillé ici mais sachez que cela existe
- La complexité réside dans la description du rangement des données en mémoire

Fonctionnalités avancées

Images

- Hdf5 permet également de stocker des images dans des fichiers.
- Utile pour associer un Dataset à une image
- Les images sont stockées sous la forme d'une suite de valeurs numériques sur 24 bits.
- Chaque valeur représente en 3 x 8 bits, les composants R, G, B de l'image

Fonctionnalités avancées

Montage de fichiers

- Supposez que vous disposiez de datasets issus de différentes sources :

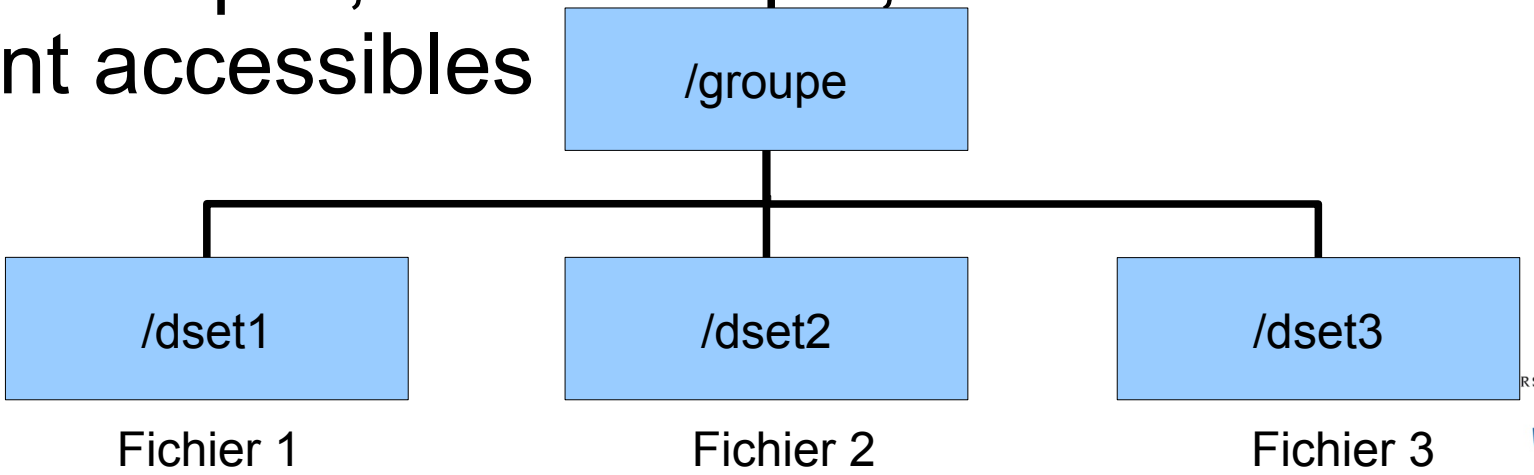
Fichier 1 Fichier 2 Fichier 3

- Si vous avez besoin de réaliser un post-traitement sur l'ensemble de ces fichiers, vous pouvez les combiner dans hdf5, sous le même groupe

Fonctionnalités avancées

Montage de fichiers

- Vous écrivez alors un reader spécifique, qui décrit le squelette et un espace de nommage global
- Dans ce fichier squelette, vous *montez* les fichiers séparés
- Vous disposez alors de l'équivalent fichier global dans lequel, sans recopie, toutes les données sont accessibles



Fonctionnalités avancées

Visualisation de données

- Les fichiers écrits avec HDF5 sont lisibles directement par un grand nombre de logiciels de visualisation :
  - Ceux basés sur Vtk : Visit, Paraview
  - Des logiciels commerciaux: Tecplot, Ensight, Idl, Matlab
  - Des suites logicielles comme Octave, Sage, Scilab
- Et ceci sans autre effort que de connaître le nom des datasets à visualiser

TP

- Nous utilisons scilab pour visualiser des données au format Hdf5
- Par exemple, tapez dans scilab :

```
h5read(« dset_simple.h5 », « /mon_groupe/dset »)
```

- Cela vous renvoie l'intégralité du contenu du tableau
- Nous visualisions à présent les données :

```
z=h5read(«dset_simple.h5», «/mon_groupe/dset»)
```

TP

- Il ne reste plus qu'à afficher ces données en 3d : z=f(x,y) ⇔ surface
- Dans scilab, nous tapons :

```
sz=size(z)
plot3d(sz(2),sz(1),double(z))
```

- Que s'affiche-t-il à l'écran ?

Plan

- Introduction
- Hdf5
- TP
- Fonctionnalités avancées
- Conclusion

Conclusion

- Hdf5 vous permet de stocker vos données de manière portable et structurée
- Le travail à réaliser pour utiliser l'API est un travail de description qui peut nécessiter de bien se représenter les données en mémoire
- Au vu de la grande popularité de Hdf5, je vous recommande de vous y intéresser
- Il ne vous reste qu'à documenter le nom de vos datasets