HDF5

Entrées/Sorties portables, hierarchiques, auto-documentées

Fabrice Roy

Hierarchical Data Format V5

- portable : lisible partout et par toutes les API et tous les lecteurs ;
- hierarchique : structure interne en arborescence ;
- auto-documentée : vous pouvez décrire vos données par des métadonnées.

Portable

Un fichier HDF5 est lisible :

- quelle que soit la machine ;
- quelle que soit l'application ;
- quel que soit le language.

Avantage du fichier texte et du fichier binaire, avec en bonus la compression des données

Hierarchique

Le contenu d'un fichier HDF5 peut être vu comme un systéme de fichier UNIX

- il y a un groupe / à la racine ;
- on peut créer des groupes dans / et des groupes dans les groupes ;
- on peut écrire des datasets (données) et des attributs (métadonnées) dans les groupes.

groupe ~ répertoire

Auto-documenté

On peut (doit ?) décrire les données avec des métadonnées :

- les métadonnées et les données sont rassemblées dans le même fichier ;
- on peut associer dans l'arborescence de données la description du contenu ;
- on peut aussi, par ex., utiliser les métadonnées pour la provenance.

Comment utiliser HDF5 avec Fortran

Prérequis : installer la bibliothèque.

- **`use hdf5`**
- appeler **`h5open_f`** avant tous les autres appels HDF5
- appeler **`h5close_f`** apres tous les autres appels HDF5

Pour plus de simplicité : compiler avec h5fc (si HDF5 séquentiel) ou h5pfc (si HDF5 parallèle)

Quelques considérations générales sur l'API Fortran

Il peut être utile de retenir que :

- tous les noms de subroutines ont la forme h5xxx_f (h5open_f ou h5close_f par ex.)
- les noms des subroutines pour manipuler les fichiers ont la forme h5fxxx_f
- les noms des subroutines pour manipuler les dataset ont la forme h5dxxx_f
- les noms des subroutines pour manipuler les attributs ont la forme h5axxx_f
- toutes les subroutines retournent un code d'erreur.

Créer, ouvrir et fermer un fichier

Les fichiers ouverts sont désignés par des ID de type integer(hid_t)

- créer : **`h5fcreate_f(nom, H5F_ACC_TRUNC_F, id, erreur)`**
- ouvrir : **`h5fopen_f(nom, H5F_ACC_RDONLY_F, id, erreur)`**
- fermer : **`h5fclose_f(id, erreur)`**
- **`H5F_ACC_TRUNC_F, H5F_ACC_RDONLY_F`** : constantes qui définissent les propriétés du fichier

Valeurs possibles pour les propriétés

en création

- **`H5F_ACC_TRUNC_F`** : équivalent de unknown en Fortran, i.e. efface le fichier s'il existe déjà
- **`H5F_ACC_EXCL_F`** : équivalent de new en Fortran, i.e. erreur si le fichier existe déjà

en ouverture

- **`H5F_ACC_RDWR_F`** : lecture/écriture
- **`H5F_ACC_RDONLY_F`** : lecture seule

Créer, ouvrir et fermer un groupe

Groupe ~ répertoire UNIX

- créer : **`h5gcreate_f(id, nom, groupe_id, erreur)`**
- ouvrir : **`h5gopen_f(id, nom, groupe_id, erreur)`**
- fermer : **`h5gclose_f(groupe_id, erreur)`**

id = id du parent ; groupe_id = id du groupe créé / ouvert / fermé

Interlude : h5dump

h5dump permet de voir le contenur d'un fichier hdf5

- **`h5dump fichier.h5`** : affiche tout le fichier
- **`h5dump -A fichier.h5`** : affiche les attributs dans le fichier
- **`h5dump -d nom_dataset fichier.h5`** : affiche le contenu d'un dataset

Attribut

Les attributs :

- sont des données simples, i.e. un nombre ou une chaîne de caractères
- servent à écrire des métadonnées
- sont attachés à un objet

Écrire un attribut (1)

C'est compliqué...

- créer un dataspace : **`h5screate_simple_f(rang,dim,space_id,erreur)`**
- créer l'attribut : **`h5acreate_f(id,nom,type,space_id,attr_id,erreur)`**
- enfin écrire l'attribut : **`h5awrite_f(attr_id,type,metadata,dim,erreur)`**
- fermer l'attribut : **`h5aclose_f(attr_id,erreur)`**
- fermer le dataspace : **`h5sclose_f(space_id,erreur)`**

id dans h5acreate_f = id de l'objet auquel on attache l'attribut

Quelques précisisions

rang = nb de dimensions de la donnée

- rang = 1 pour scalaire et tableau 1D
- rang = 2 pour tableau 2D

dim = taille de la donnée

- tableau de dimension rang
- dim(1)=1 pour un scalaire

Quelques autres précisions

Types HDF5

- pour un reel : **`h5kind_to_type(kind,H5_REAL_KIND)`**
- pour un entier : **`h5kind_to_type(kind,H5_INTEGER_KIND)`**
- pour une chaîne de caractères : plus compliqué
- **`h5tcopy_f(H5T_NATIVE_CHARACTER, type_id, erreur)`**
- **`h5tset_size_f(type_id, len, erreur)`** ou len=longueur de la chaîne

Écrire un dataset

Comme un attribut !

- créer le dataspace : **`h5screate_simple_f(rang,dim,space_id,erreur)`**
- créer le dataset : **`h5dcreate_f(id,nom,type,space_id,dset_id,erreur)`**
- écrire le dataset : **`h5dwrite_f(dset_id,type,data,dim,erreur)`**
- fermer le dataset : **`h5dclose_f(dset_id, erreur)`**
- fermer le dataspace : **`h5sclose_f(space_id,erreur)`**

Lire un attribut

Plus simple que l'écriture

- ouvrir l'attribut : **`h5aopen_name_f(id,nom,attr_id,erreur)`**
- lire le type : **`h5aget_type_f(attr_id, type_id, erreur)`**
- lire l'attribut : **`h5aread_f(attr_id, type_id, metadata, dim, erreur)`**
- fermer l'attribut : **`h5aclose_f(attr_id, erreur)`**

id = id de l'objet auquel l'attribut est attaché

Lire un dataset

Plus simple aussi

- ouvrir le dataset : **`h5dopen_f(id, nom, dset_id, erreur)`**
- lire le dataset : **`h5dread_f(dset_id, type, data, dim, erreur)`**
- fermer le dataset : **`h5dclose_f(dset_id, erreur)`**

id = id du contenant (groupe ou fichier)

type peut être donné par h5kind_to_type

HDF5 parallèle

Il est possible d'écrire ou lire dans un même fichier avec plusieurs processus MPI

- les processus écrivent / lisent des morceaux du même dataset
- plusieurs pinceaux sont possibles
- réduit le nombre de fichiers
- les fichiers parallèles sont identiques aux fichiers séquentiels

HDFView

Outil graphique : explorateur de fichier HDF5

- plus lisible que h5dump
- permet de vérifier la structure facilement
- donne quelques infos statistiques
- permet d'exporter

Interface simplifiée développée au LUTH

Comme c'est quand même assez compliqué, on a développé une interface Fortran

- **`Hdf5_write_attr`** / **`Hdf5_read_attr`** pour écrire / lire un attribut
- **`Hdf5_write_data`** / **`Hdf5_read_data`** pour écrire / lire un dataset
- **`Hdf5_write_mpi_data`** / **`Hdf5_read_mpi_data`** pour écrire lire un dataset en parallèle
- + subroutines pour gérer les fichiers et groupes

On peut partager si vous voulez utiliser ou même améliorer cette interface.