TD1 GLCS : HDF5

27/11/2020

Rendu des TDs

Vous devrez rendre vos TDs par email à l'adresse `julien.bigot@uvsq.fr` avant vendredi soir, minuit. Attention, votre mail devra comporter en fichier attaché une archive (format `tar.gz`) de votre projet ne contenant que les fichiers source, **pas d'exécutable !**

Mise en place

Commencez par installer docker sur votre machine, puis autorisez votre utilisateur à le lancer.
Sous debian / ubuntu, on obtient ce résultat avec les commandes :

```
sudo apt install docker.io
sudo gpasswd -a $USER docker
```

Une fois docker installé, téléchargez le fichier du TD : `https://raw.githubusercontent.com/jbigot/glcs_2020-2021/master/glcs-td1.sh` puis exécutez le. Cette exécution peut être un peu lente ; elle télécharge tout l'environment du TD. Elle devrait créer un répertoire `TD1` avec les fichiers du TD et ouvrir un shell dans ce répertoire.

```
wget https://raw.githubusercontent.com/jbigot/glcs_2020-2021/master/glcs-td1.sh
bash glcs_td1.sh
```

Le shell ainsi ouvert tourne dans un environment docker (container) qui comporte des bibliothèques, outils, etc... différents de ceux de votre machine personnelle. C'est dans cette environment que le TD va se dérouler. Pour vous assurez que votre shell courant tourne bien dans l'environment du TD, vérifiez que le prompt commence bien par `glcs@`.

Vous pouvez à tout moment ouvrir un nouveau shell dans cet environment en exécutant la commande :

```
bash glcs_td1.sh <chemin>/<vers>/TD1
```

1 Compilation

Le programme `ex1/ex1.cxx` utilise deux bibliothèques : `MPI` et `HDF5`. MPI est installé dans le chemin standard, mais HDF5 est installé dans `/usr/lib/x86_64-linux-gnu/hdf5/openmpi`. Le Makefile fourni ne prend pas en compte l'utilisation de ces bibliothèques et la compilation échoue.

Modifiez le Makefile en utilisant le wrapper `mpic++` et en ajoutant les options nécessaires pour compiler le programme.

2 Cmake

Dans `ex2`, pour simplifier la compilation, on va utiliser cmake. Le fichier `CMakeLists.txt` fourni supporte HDF5 mais pas MPI. En utilisant le module `FindMPI` [1] fourni par CMake, ajoutez le support de MPI.

Modifiez le `CMakeLists.txt` fourni en ajoutant la prise en charge de `MPI` pour compiler le programme.

Tous les exercices suivant devront reprendre le `CMakeLists.txt` créé pour cet exercice.

1. `https://cmake.org/cmake/help/latest/module/FindMPI.html`

3 Écriture HDF5

Jusqu'à maintenant, le code proposé créait un fichier HDF5 vide. On va modifier `ex3/ex3.cxx` pour écrire ses données dans le dataset nommé "`/data`" au sein du fichier "`my_file_r<N>.h5`". Pour simplifier cette étape, on utilisera les fonctions de HDF5 lite [2].

Modifiez le code pour que chaque rang MPI (processus) écrive dans un fichier HDF5 son bloc local de données.

Vous pourrez valider votre code à l'aide de l'outil `h5dump`. Pensez à tester votre code avec différents paramètres de taille et différents parallélismes MPI.

4 Écriture locale sans zones fantômes

En comparaison avec `ex3/ex3.cxx`, dans `ex4/ex4.cxx` on a ajouté des zones "fantômes". Le buffer alloué est plus grand que le contenu attendu. Cette technique est souvent utilisé en MPI pour permettre de récupérer les données produites par un processus voisin. Afin de ne pas écrire ces données superflues, on va devoir appliquer une sélection sur les données à écrire en mémoire (un "hyperslab" HDF5).

Modifiez le code pour que chaque rang MPI (processus) écrive dans un fichier HDF5 son bloc local de données sans les zones "fantômes".

Vous pourrez valider votre code en comparant les fichiers générés avec ceux de l'exercice précédent en utilisant l'outil `h5diff`. Les données devraient être les mêmes, quel que soit la taille des zones fantômes. Pensez à tester votre code avec différents paramètres de taille et différents parallélismes MPI.

5 Écriture parallèles

Dans `ex5/ex5.cxx`, on cherche à rendre le code indépendant de la parallélisation. On veut donc que le fichier généré ne dépende plus du nombre de rang MPI mais seulement de la taille totale $block_width * MPI_size$. Pour cela, il faudra utiliser HDF5 parallèle et créer une fenêtre ("hyperslab" HDF5) non seulement pour sélectionner les données à écrire en mémoire comme dans le , mais aussi pour

Modifiez le code pour que tous les rangs MPI (processus) écrivent en parallèle dans un même fichier HDF5 leurs blocs de données dans un tableau dont la largeur totale est $block_width * MPI_size$.

Vous pourrez valider votre code en comparant les fichiers générés avec ceux de des exercices précédents dans le cas où il n'y avait qu'un processus. Les données devraient être les mêmes, quel que soit le nombre de processus MPI. Pensez à tester votre code avec différents paramètres de taille et différents parallélismes MPI.

2. `https://support.hdfgroup.org/HDF5/doc/HL/RM_H5LT.html`