Notion d’API (application programming interface)

Ce qui permet d’utiliser une « brique logicielle » (classe, composant, module, …) : liste des fonctionnalités utilisables avec leur description

Avec pour chaque fonctionnalité :

- Description du traitement effectué
- Méthodes d’appel (fonctions C, classes C++ , routines fortran, etc)
- Description des données d’entrée (types, formats, valeurs possibles attendues)
- Description des résultats de sortie (types, formats)

Notion d'API (2)

Se présente sous la forme :

- d'un ou plusieurs fichiers à inclure dans le code qui utilise l'API (ces fichiers sont spécifiques pour un langage de programmation)
- d'un manuel d'utilisation qui complète les explications (pas obligatoire mais utile)

Un même code peut avoir plusieurs API

(suivant le langage de programmation, ou le « niveau d'expertise », ...)

Une API peut être commune à plusieurs codes

(des codes différents peuvent effectuer le même calcul avec des algo. différents)

Notion d’API (3)

Exemples :

FFTW (transformées de Fourier)

FFTW3.h

LAPACK (algèbre linéaire)

LAPACK_dsysv

MPI (calcul parallèle)

mpi.h

MPI (API complete)

MPI_Allreduce

Ne pas se lancer « tête baissée » dans la programmation

Réfléchir au problème à résoudre :

- Identifier les composantes du système
- Identifier les interactions entre ces composantes
- Vérifier que ce modèle peut représenter des cas d'utilisation du système

- C'est un processus itératif : si le modèle ne peut pas traiter un cas d'utilisation, il faut le reprendre

Peut se faire « sur le papier » ou en utilisant un langage de modélisation (exemple UML si on utilise la programmation objet, ex. C++ ou Java)

Cours UML : Cours UML

Utiliser des « masques de conception » (en anglais : « design patterns)

Exemples : Factory, Singleton, Iterator, Observer, ...

Voir aussi Cours UML

Réutilisation

Le but est d’écrire le moins possible de code soi-même pour effectuer un traitement spécifique

2 voies principales :

- Utilisation de librairies externes
- Intégration de son code dans une plateforme (appelée framework ou cadriciel)

On peut aussi mélanger les 2 approches

Librairies vs. Framework

Une librairie est un ensemble logiciel, qui réalise un certain type de traitement. Une librairie ne peut pas s'exécuter seule, mais peut être ajoutée à un code qui va utiliser ses fonctionnalités. Un code utilisera typiquement plusieurs librairies. C'est le code qui gère le déroulement du calcul.

Un framework est un espace de travail, un ensemble de composants (dont des librairies) et des règles. Ces composants sont organisés pour être utilisés en interaction les uns avec les autres. Le développeur va ajouter son code au framework et bénéficie d'un ensemble cohérent de composants de base. C'est le framework qui gère le déroulement du calcul.

Utilisation de librairies

Dans son code, identifier le besoin :

- résoudre un système linéaire,
- générer un maillage,
- écrire « efficacement » sur disque,
- gérer les paramètres du code,
- Etc.

Dans le contexte HPC, examiner en particulier les librairies capables de fonctionner en mode //.

Choisir la(les) librairie(s) parmi celles qui traitent le(s) besoin(s) identifié(s).

Librairies pour résoudre un système linéaire (1)

Il existe plusieurs librairies possibles. Pour le HPC, par exemple :

- PETSc
  https://www.mcs.anl.gov/petsc/
- MUMPS
  http://mumps.enseeiht.fr/
- PASTIX
  http://pastix.gforge.inria.fr/files/README-txt.html
- DPLASMA
  http://icl.cs.utk.edu/projectsdev/parsec/index.html

On trouvera une liste plus complète dans
http://www.netlib.org/utk/people/JackDongarra/la-sw.html

Librairies pour résoudre un système linéaire (2)

Le choix se fera sur les critères :

- Type(s) de matrices et de systèmes dont on a besoin (denses, creuses, résolutions multiples avec la même matrice ou non, etc.).
- Architectures matérielles visées (// multi-process, // multi-threads, accélérateurs – GPU, Xeon Phi, etc.).
- Adéquation entre la représentation des matrices dans le code et dans la librairie.

Librairies pour les entrées/sorties sur disque

Ci-dessous une liste de librairies qui tiennent compte des architectures parallèles :

- MPI-I/O (fait partie du standard MPI)

  http://mpi-forum.org/docs/mpi-3.1/mpi31-report/node305.htm

- HDF

  https://www.hdfgroup.org/hdf5/

- SIONlib

  http://www.fz-juelich.de/ias/jsc/EN/Expertise/Support/Software/SIONlib/_node.html

- ADIOS

  https://www.olcf.ornl.gov/center-projects/adios/

- NetCDF

  http://www.unidata.ucar.edu/software/netcdf/

Librairies pour la gestion des maillages

Il n’existe pas (à ma connaissance) de générateur de maillage parallèle open-source. En général, on procède comme suit :

- Dans une première phase, on définit un domaine géométrique
- A l’aide d’un générateur séquentiel de maillage, on construit un maillage complet
- Enfin, on utilise des découpeurs de maillages pour distribuer les maillages partiel sur les différents processus

  Le découpeur de maillage calcule aussi les informations de voisinage entre les éléments des maillages partiels voisins

Librairies pour la gestion des maillages (2)

Un deuxième procédé consiste à :

- Dans une première phase, on définit un (sous-)domaine géométrique pour chaque processus de façon à obtenir le domaine complet comme réunion des domaines partiels
- A l'aide d'un générateur séquentiel de maillage, on construit un maillage partiel sur chaque sous-domaine

La difficulté consiste à « recoller » entre eux les maillages partiels => on prend souvent le procédé précédent

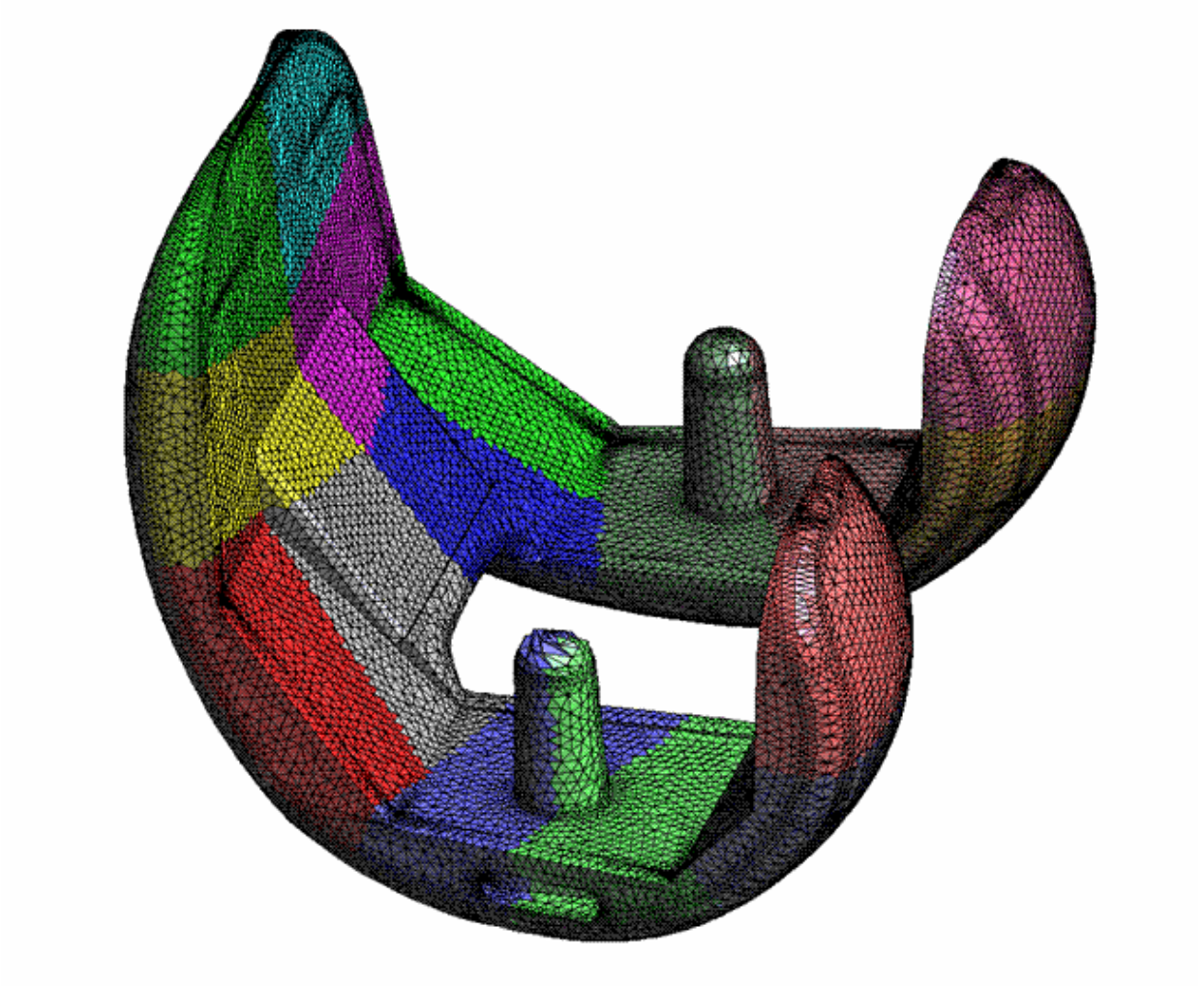
Librairies pour la génération des maillages

- Triangle (2D)
  https://www.cs.cmu.edu/~quake/triangle.html
- Tetgen (3D)
  http://wias-berlin.de/software/tetgen/
- GMSH
  gmsh.info
- NETGEN
  https://sourceforge.net/projects/netgen-mesher/

Librairies pour la découpe des maillages

- Metis/ParMetis
  http://glaros.dtc.umn.edu/gkhome/views/metis
- Scotch/PT-Scotch
  https://www.labri.fr/perso/pelegrin/scotch/scotch_fr.html
- PaMPA
  https://project.inria.fr/pampa/

Exemple de découpe de maillage



Librairies pour visualiser des résultats

- VTK

http://www.vtk.org/

Librairies pour le paramétrage des codes

- INI (nombreuses librairies disponibles), par ex.
  https://github.com/ndevilla/iniparser
- JSON (nombreuses librairies disponibles)
  http://www.json.org/
- XML (plus complexe, mais permet de valider)
  http://www.xmlsoft.org/

Avantages/inconvénient des librairies

Avantages :

- Beaucoup de choix disponibles
- Librairies souvent très bien optimisées
- Votre code est central, c’est le code qui utilise les librairies

Inconvénients :

- Beaucoup de choix disponibles
- Chaque librairie se compile et s’installe différemment
- Chaque librairie a ses propres formats de données

Intégration dans un framework (1)

Un framework apporte :

- Un ensemble de composants d'intérêt général (par exemple, contenant une des librairies vue aux slides précédents)
- Un ensemble de règles (par exemple un ensemble de types de données normalisés, un ensemble minimal de fonctionnalités que doivent assurer tout composant, etc.)
- Une interface utilisateur (graphique ou langage de commande) qui permet d'utiliser ces composants

Intégration dans un framework (2)

Pour utiliser un framework, le développeur de code doit adapter son code pour respecter les règles imposées par le framework

- Vérifier que son code propose les fonctionnalités de base de tout composant du framework.
- Transformer les données d’entrée et de sortie du code en utilisant les formats prévus par le framework.
- Supprimer tout programme principal du code.

Exemple de frameworks (1)

Matlab : offre de nombreuses fonctionnalités (dans matlab lui-même ou via des toolboxes) : algèbre linéaire, traitement d'images, visualisations, etc.

L'utilisateur peut écrire son code en langage matlab ou intégrer du code C/C++/fortran.

Très simple mais pas efficace pour le HPC
(sert pour la mise au point des algorithmes)
Cher … (voir aussi octave, scilab)

Exemple de frameworks (2)

ROOT (https://root.cern.ch/) : adapté au traitement de grandes quantités de données (calcul statistiques, visualisation, sauvegarde)

Contient un interpréteur de code C++ que l'utilisateur peut utiliser pour choisir le traitement à effectuer.

Exemple de frameworks (3)

Trilinos (https://trilinos.org/) : ensemble de composants de pré-processing et de calcul, très cohérent, avec l'accent sur la prise en compte du //

Nécessite une (très) bonne connaissance du C++.
Assez complexe à maîtriser.

Exemple de frameworks (4)

Salome (http://www.salome-platform.org/) : propose des composants pré- et post-processing (CAO, maillage, visualisation)

Propose une interface utilisateur graphique et texte (utilisant python).

Exemple d'application sur base Salome : Salome-Meca avec le code de calcul mécanique Aster (http://www.code-aster.org/spip.php?article7).

Mais pas ou peu d'utilisation dans le HPC.

TD 1 : Utilisation d'une librairie

Ecrire un programme principal en C ou en C++ qui :

- Demande à l'utilisateur un entier positif ***n***
- Crée un vecteur de taille ***n*** et l'initialise :

$$v_k = sin(10\ k\ \pi/n)$$

- Calcule la transformée de Fourier
- Ecrit le vecteur et sa transformée de Fourier dans un fichier

On utilisera la librairie fftw3 (à installer si elle n'est pas déjà disponible sur votre machine) : http://www.fftw.org/