Génie Logiciel pour le Calcul Scientifique

Marc Tajchman
marc.tajchman@cea.fr

Table des matières

Idée : Façon de développer des application de calcul et surtout de les intégrer dans un framework. Le but est d'obtenir le plus rapidement des résultats sans avoir à tout recoder (affichage, calculs "simples", gestion de fichiers, ...) afin de se concentrer principalement sur l'algorithmique à implémenter.

1 Introduction

1.1 Notion d'API

Application Programming Interface : Indique les différentes "briques logicielle" (fonctions, classes, modules, etc) utilisables avec leur description. Pour chaque fonctionnalité, l'API documente :

- Le travail effectué (exemple : résolution d'un système linéaire)
- La méthode d'appel
- La description des entrées (types, formats et valeurs) (exemple : la matrice doit être inversible)
- La description des résultats de sortie (types et formats) (exemple : le vecteur de solutions *et* la précision de calcul ; si la sortie est une matrice, l'ordre des coefficients doit être précisé, ...)

Elle se présente sous la forme d'un ensemble de fichiers à inclure dans le code qui donne la *signature* ou *prototype* des fonctions apportées, ainsi qu'un manuel d'explication complétant les explications (ce n'est cependant malheureusement pas toujours le cas).

Attention, un code peu utiliser plusieurs API, et une API peut être commune à plusieurs code (par exemple, utiliser deux algorithmes différents pour le même calcul).

Parfois, une API *de haut niveau* est disponible, permettant d'utiliser des fonctions "simples" utilisant des paramètres par défaut (utile selon le "niveau d'expertise" de l'utilisateur).

L'intérêt d'une API réside dans la *réutilisation possible* du code : en effet, il est possible d'interchanger du code partageant la même API sans adaptations.

Il existe de nos jour de nombreuses API de confiance performantes, probablement plus qu'une implémentation à la main. Par exemple : FFTW pour la transformées de Fourier, LAPACK pour l'algèbre linéaire, MPI pour le calcul parallèle, ce dernier étant une interface dont de nombreuses librairies définissent différemment ces interfaces (OpenMPI par exemple). Avec l'achat d'une machine, il est possible que le constructeur offre une version modifiée de MPI adaptée pour cette machine ; mais toutes ces versions peuvent exécuter le même code.

Attention : Il faut réfléchir *au départ* sur le problème à résoudre et son intégration par rapport aux composantes systèmes, leurs interactions. Il est utile de vérifier sur des exemples triviaux le bon fonctionnement du code petit à petit ; mais également de vérifier que le code ne fonctionne pas lorsque les spécifications d'entrée ne sont pas respectées.

On peut commencer par utiliser "sur papier" un langage de modélisation tel ULM (*Universal Modeling Language*, un langage graphique).

1.2 Utilisation de Design Pattern

Masque de conception en français.

Par exemple :

- Factory (construction de données appartenant à une même classe générale)
- Sigleton (refuser la création de deux objets de même classe afin de garantir l'unicité)
- Iterator (parcours d'un ensemble)
- Observer (tirer des informations du code telles que l'affichage/sauvegarde ou des résultats partiels)
- Et pleins d'autres !

1.3 Réutilisation de code

Principe : écrire le moins de code possible soi-même. On utilise principalement deux voies : soit par l'utilisation de bibliothèques (*libraries*) ou l'utilisation d'une plateforme (*framework* ou *cadriciel*). Il est également possible de mélanger ces deux approches.

Librairie Ensemble logiciel réalisant un ensemble de traitement similaires ; elle ne peut pas s'exécuter seule mais est ajoutée au code afin d'utiliser ses fonctionnalités. Le plus souvent, un code utilisera plusieurs librairies. C'est le code qui gère le déroulement du calcul.

Dans un contexte HPC, il faut faire bien attention à ce que celles-ci sont capable de fonctionner en mode parallèle... Grands noms : PETSc, MUMPS, et plus récemment PASTIX et PLASMA

Framework C'est une espace de travail, de composants et de règles. Ces composants sont organisés pour être utilisés en interaction les uns avec les autres. Le développeur ajoute son code au framework et bénéficie un ensemble cohérent de composants de base. Dans ce cas, c'est le framework qui gère le déroulement de calcul. Par exemple Mathlab permet l'importation de code de différents langages sous réserve que ce dernier respecte un format précis.

Les critères de choix sont principalement :
— Le type de matrice et de système dont on a besoin (creuses denses, etc)
— Architectures visées (GPU, accélération matérielle, ...)
— Adéquation de la représentation des matrices dans le code par rapport à la librairie

De nombreuses librairies existent également pour l'écriture efficace de données, telles :
— MPI-I/O
— HDF (à utiliser dans le projet)
— SIONlib (allemand)
— ADIOS
— NetCDF

1.4 Libraires pour la gestion de maillage

Cf sildes.
Visualisation de résultats :
— VTK

Paramétrage des codes :
— INI
— JSON
— XML

1.4.1 Plateforme

Un framework apporte :
— Un ensemble de composant d'intérêt général (par exemple des librairies)
— Un ensemble de règles (normalisation des données, ensemble minimal de fonctionnalités, etc)
— Une interface utilisateur (graphique ou langage de commande) afin d'utiliser les composants