Langages de programmation pour le calcul scientifique

Romaric DAVID

Ecole d'Automne Informatique Scientifique
2 Octobre 2008

Les langages compilés

Langages compilés $\Rightarrow$ famille fondamentale des langages de programmation.
En calcul scientifique, on trouve principalement :

- Fortran(s)
- C++
- C

Ces langages sont intéressants par les bibliothèques de haut niveau disponibles (ne **jamais** commencer à écrire un programme sans regarder les API existantes). En particulier :

Fortran	Lapack, Scalapack, MPI
C++	Boost, MPI
C	Lapack, MPI
Scilab,Matlab	C, Fortran, Lapack, MPI, PVM

Bibliothèques des langages compilés

À la lecture du tableau précédent, on remarque que

- Certaines bibliothèques ne sont disponibles que pour certains langages. $\Rightarrow$ Est-on lié à ces langages si on veut les utiliser ?

Bibliothèques des langages compilés

À la lecture du tableau précédent, on remarque que

- Certaines bibliothèques ne sont disponibles que pour certains langages. $\Rightarrow$ Est-on lié à ces langages si on veut les utiliser ?
- Certaines bibliothèques sont disponibles dans plusieurs langages. $\Rightarrow$ Est-ce que le travail pour les écrire a été dupliqué ?

Bibliothèques des langages compilés

À la lecture du tableau précédent, on remarque que

- Certaines bibliothèques ne sont disponibles que pour certains langages. $\Rightarrow$ Est-on lié à ces langages si on veut les utiliser ?
- Certaines bibliothèques sont disponibles dans plusieurs langages. $\Rightarrow$ Est-ce que le travail pour les écrire a été dupliqué ?

Réponse doublement négative !

Architecture d'un code utilisant des langages compilés

Dans un code de calcul, il y a de fortes probabilités que l'on passe par la séquence suivante :

- Lecture des données d'entrée + Pré (éventuellement sous forme graphique)

Architecture d'un code utilisant des langages compilés

Dans un code de calcul, il y a de fortes probabilités que l'on passe par la séquence suivante :

- Lecture des données d'entrée + Pré (éventuellement sous forme graphique)
- Enchainement des étapes de calcul

Architecture d'un code utilisant des langages compilés

Dans un code de calcul, il y a de fortes probabilités que l'on passe par la séquence suivante :

- Lecture des données d'entrée + Pré (éventuellement sous forme graphique)
- Enchainement des étapes de calcul
- Post + Écriture des données de sortie (éventuellement sous forme graphique)

Architecture d'un code utilisant des langages compilés

Dans un code de calcul, il y a de fortes probabilités que l'on passe par la séquence suivante :

- Lecture des données d'entrée + Pré (éventuellement sous forme graphique)
- Enchainement des étapes de calcul
- Post + Écriture des données de sortie (éventuellement sous forme graphique)

Les compilateurs et les processeurs sont conçus pour traiter (optimiser, exécuter) les étapes de calcul. Le code d'E/S ne bénéficiera pas des optimisations et est largement source d'erreurs.

Architecture d'un code utilisant des langages compilés

Dans un code de calcul, il y a de fortes probabilités que l'on passe par la séquence suivante :

- Lecture des données d'entrée + Pré (éventuellement sous forme graphique)
- Enchainement des étapes de calcul
- Post + Écriture des données de sortie (éventuellement sous forme graphique)

Les compilateurs et les processeurs sont conçus pour traiter (optimiser, exécuter) les étapes de calcul. Le code d'E/S ne bénéficiera pas des optimisations et est largement source d'erreurs.
Faut-il utiliser des langages compilés pour le code non-calcul ?

Quelques éléments de réponse

Nous avons identifié deux problèmes :

- Le besoin d'utiliser des fonctionnalités d'un langage A dans un langage B
- L'utilisation pas forcément optimale du compilateur/processeur pour des tâches de gestion (non calcul)

Pour s'attaquer à ces problèmes, il faut savoir :

- Interfacer des langages différents entre eux
- Utiliser autre chose que les langages compilés pour la gestion

Quelques éléments de réponse

Nous avons identifié deux problèmes :

- Le besoin d'utiliser des fonctionnalités d'un langage A dans un langage B
- L'utilisation pas forcément optimale du compilateur/processeur pour des tâches de gestion (non calcul)

Pour s'attaquer à ces problèmes, il faut savoir :

- Interfacer des langages différents entre eux
- Utiliser **des langages de haut-niveau** (interprétés ?) pour la gestion

Principe I

Interfaçage de langages consiste à permettre l'appel de méthodes écrites et compilées depuis un langage A vers un langage B. Pour cela, il faut que :

- Les symboles (fonctions) soient accessibles depuis B. Il faut connaître leur nom
- Connaître la représentation des types de données dans A et dans B. Type atomiques et composées (ex : les tableaux).
- Connaître les conventions d'appel (par adresse, par valeur, mix des deux)
- Définir des fonctions d'interface qui savent convertir le cas échéant les données en minimisant les copies (surtout pour les tableaux)

Principe II

Exemple : les tableaux multidimensionnels.
Les tableaux sont rangés de manière contiguë (aplatie) en mémoire. L'ordre du rangement des éléments ("en ligne" ou "en colonne") dépend du langage. Considérons un tableau 2D `t` utilisé dans un programme C et Fortran.
La notation dans le code source est identique :

- `t[i][j] en C`
- `t(i,j) en Fortran`

En mémoire, on trouvera dans l'ordre :

- C : t[0][0],t[0][1],...,t[0,n-1],t[1][0],t[1][1],...,t[n-1,n-1] (le dernier indice varie le plus vite, rangement en lignes)
- Fortran : t(1,1),t(2,1),...,t(n,1),t(2,1),t(2,2),...,t(n,n) (le dernier indice varie le plus lentement, rangement en colonnes)

Principe III

$\Rightarrow$ Ordre et indices changent

Pour une fonction de nom func, les symboles présents dans le fichier objet n'ont pas le même nom suivant que le code a été compilé en C ou Fortran.

- C : func
- Fortran : _func

Les choses se compliquent en C++, où le type des arguments de la fonction est codé dans le nom du symbole.

On doit prendre en compte tous ces détails lorsque l'on réalise des interfaces. Il est néanmoins possible d'interfacer C et Fortran (assez courant) et Fortran et C++ (cf. cours T. Dumont)

Le choix du langage utilisé pour piloter l'ensemble de ces routines reste à faire. S'agit-il du langage utilisé pour réaliser l'interface graphique ? D'un autre langage ?

Séparer contrôle et calcul I

Beaucoup de langages de haut niveau (Tcl, Python, Perl) sont extensibles par la programmation de modules disposant d'interfaces clairement définies. $\Rightarrow$ cible de choix lors de la mise en place d'interfaces entre langages.
Le schéma d'un code de calcul utilisant des langages de haut niveau pour les routines de gestion pourrait être le suivant :

- HN : Lecture des données d'entrée + Pré (éventuellement sous forme graphique)

Séparer contrôle et calcul I

Beaucoup de langages de haut niveau (Tcl, Python, Perl) sont extensibles par la programmation de modules disposant d'interfaces clairement définies. $\Rightarrow$ cible de choix lors de la mise en place d'interfaces entre langages.
Le schéma d'un code de calcul utilisant des langages de haut niveau pour les routines de gestion pourrait être le suivant :

- HN : Lecture des données d'entrée + Pré (éventuellement sous forme graphique)
- HN : Enchainement des étapes de calcul (BN)

Séparer contrôle et calcul I

Beaucoup de langages de haut niveau (Tcl, Python, Perl) sont extensibles par la programmation de modules disposant d'interfaces clairement définies. $\Rightarrow$ cible de choix lors de la mise en place d'interfaces entre langages.
Le schéma d'un code de calcul utilisant des langages de haut niveau pour les routines de gestion pourrait être le suivant :

- HN : Lecture des données d'entrée + Pré (éventuellement sous forme graphique)
- HN : Enchainement des étapes de calcul (BN)
- HN : Post + Écriture des données de sortie (éventuellement sous forme graphique)

Séparer contrôle et calcul II

Dans cette optique de programmation :

- Une étape du calcul $\Leftrightarrow$ une routine du langage BN

Séparer contrôle et calcul II

Dans cette optique de programmation :

- Une étape du calcul $\Leftrightarrow$ une routine du langage BN
- En profiter pour uniformiser le format des données d'entrée

Séparer contrôle et calcul II

Dans cette optique de programmation :

- Une étape du calcul $\Leftrightarrow$ une routine du langage BN
- En profiter pour uniformiser le format des données d'entrée
- Définir des jeux de tests + scénarios d'exécution

Séparer contrôle et calcul II

Dans cette optique de programmation :

- Une étape du calcul $\Leftrightarrow$ une routine du langage BN
- En profiter pour uniformiser le format des données d'entrée
- Définir des jeux de tests + scénarios d'exécution
- Simplfier les appels (diminuer le nombre de paramètres, paramètres par défaut,...)

Séparer contrôle et calcul II

Dans cette optique de programmation :

- Une étape du calcul $\Leftrightarrow$ une routine du langage BN
- En profiter pour uniformiser le format des données d'entrée
- Définir des jeux de tests + scénarios d'exécution
- Simplfier les appels (diminuer le nombre de paramètres, paramètres par défaut,...)

Python est un langage de haut niveau ayant rencontré un succès important dans la communauté scientifique. Il existe des générateurs d'interfaces automatiques (ou au moins largement guidés) pour intégrer des fonctions C, Fortran. L'interfaçage Python/Fortran sera présenté dans la suite de la journée.

Avant de céder la parole...

- BN : Penser routines plutot que programmes
- HN : Penser structuration et enchainement de routines