FONCTIONS UTILISATEURS C#

Commissariat à l’énergie atomique et aux énergies alternatives - www.cea.fr

Fonctions utilisateurs en C#

En 2008, il a été décidé de permettre aux utilisateurs d'étendre les fonctionnalités d'un code par l'ajout de fonctions utilisateurs

- **Deux objectifs « métiers »**
  - Ne pas avoir à modifier les plug-ins lors des évolutions du code
  - Pouvoir utiliser le plug-in tel quel dans plusieurs codes

- **Deux objectifs « informatiques »**
  - Performances raisonnables par rapport au C++ (max 5x plus lent)
  - Pas de restriction sur l'utilisation (ex: support du multithreading)

- **Ces objectifs impliquent**
  - De découpler les structures de données du plugin de celles du code
  - De limiter à l'utilisateur l'accès au code
    - Ne rendre accessible que la partie nécessaire au plugin

Fonctions utilisateurs en C#

- **C++ envisagé mais non retenu car:**
  - Langage non sur (débordement de tableaux, gestion mémoire complexe) et trop complexe pour les utilisateurs
  - Nécessité de mettre en place une chaîne de compilation par plateforme
  - Trop fortement couplé au code : non pérenne en cas de changement dans le code

- **Au départ choix du langage python pour les premiers plugins**

  ➔ **abandonné car:**
  - Pas performant (entre 10 et 100x plus lent)
  - Forte adhérence aux versions de Python
  - Pas de support du multi-threading
  - Langage interprété: les erreurs ne se voient qu'au moment de l'exécution.

Fonctions utilisateurs en C#

Le choix du C# a été fait en 2010

- **Choix du C#**
  - A l'origine (2000), langage conçu par Microsoft (C# = (C++) ++) similaire dans ces concepts à Java et dans sa syntaxe au C++
  - Implémentation open-source (Mono) utilisée massivement (notamment dans les jeux vidéo et les smartphones)
  - Déjà utilisé par certains de nos outils
- **Avantages**
  - Code compilé et performant (à l'origine 2 ou 3x plus lent que le C++, moins maintenant)
  - Multi-plateforme (bytecode) et pas de chaine de compilation à maintenir car le compilateur est le même partout et le langage reste compatible avec les différentes évolutions
  - Sécurisé : pas de débordement de tableaux, pas de gestion mémoire explicite

Fonctions utilisateurs en C#

- **Les classes C++ d'Arcane sont accessibles en C# grâce à l'outil open-source 'SWIG'**
- **Respect des objectifs initiaux via le découplage par rapport au code C++**
  - les extensions écrites en 2010 restent fonctionnelles sans modification en 2020
- **Inconvénients**
  - Extensions à définir explicitement (pas forcément un inconvénient)
  - Langage légèrement différent du C++
  - Debug plus difficile (plus maintenant avec Visual Studio Code)

Fonctions utilisateurs en C#

Exemple d'équation d'État

```
using Arcane;
using System;

public class EOS_GP : Arcane.User.EOS
{
  public void Compute(TemperatureEOS x)
  {
    int nb_cell = x.NbCell;
    double adiabatic_cst = 1.4;
    double specific_heat = 2.3;
    for( int i=0; i<nb_cell; ++i ){
      double rho = x.Density[i];
      double t = x.Temperature[i];
      double e = specific_heat * t;
      double p = (adiabatic_cst-1.0) * rho * e;
      x.InternalEnergy[i] = e;
      x.Pressure[i] = p;
      x.DerivedInternalEnergy[i] = specific_heat;
      x.DerivedPressure[i] = (adiabatic_cst-1.0) * rho * specific_heat;
      x.SoundSpeed[i] = System.Math.Sqrt(adiabatic_cst*p/rho);
    }
  }
}
```

- Pas de nom de variable du code mais des champs de classe
  ➔ peut supporter plusieurs codes et les évolutions dans le nom des variables
- Séparation Entrées/Sorties
- Utilisation de tableaux simples
  ➔ indépendance à la structure de données du code

Fonctions utilisateurs en C# : Performances

- **Exemple de comparaison C++/C#**
  - Calcul de 1M d'entiers de longueur variable (avec vectorisation)
    https://medium.com/@alexyakunin/geting-4x-speedup-with-net-core-3-0-simd-intrinsics-5c9c31c47991
  - **C++ 95ms, C# 101 ms**
- **https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/csharpcore-gpp.html**

	C#	Python 3	C++
Reverse-complement	3.11	16.93	4.71
Fannkuch-redux	11.14	534.40	10.69
Spectral-norm	2.14	169.87	1.98
pidigits	2.11	3.47	1.89
fasta	2.17	63.55	1.84
K-nucleotide	5.74	72.24	3.90
Binary-trees	6.05	80.30	3.92
N-body	21.82	865.18	7.70
mandelbrot	5.60	259.50	1.51

Fonctions utilisateurs en C# : Perspectives

- **Couplage possible avec les bibliothèques d'apprentissage (Machine Learning) (i.e: https://scisharp.github.io/SciSharp)**
  - TensorFlow
  - PyTorch
  - Keras (réseau de neurones)
  - Gym (comparaison d'algorithmes d'apprentissage)
- **Couplage possible avec les notebooks Jupyter (https://jupyter.org/)**

Merci de votre attention