Développement de code pour le calcul scientifique : bibliothèques spécialisées

B. DUSSOUBS
Institut Jean Lamour – UMR 7198 UL–CNRS

Plan

- Présentation (rapide) de l'IJL
- Des bibliothèques numériques – Pourquoi ?
- Optimisation – Interfaçage
- Utilisation
- Liste des principales bibliothèques
- Conclusions

Plan

- Présentation (rapide) de l'IJL
- Des bibliothèques numériques – Pourquoi ?
- Optimisation – Interfaçage
- Utilisation
- Liste des principales bibliothèques
- Conclusions

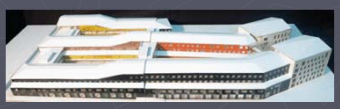
Institut Jean Lamour

- Crée au 1er janvier 2009 par la fusion de 5 laboratoires
  - Matériaux, métallurgie, nanosciences , plasmas, surfaces, interaction matériau-vivant…
  - 23 équipes de recherche divisées en 4 départements
    - P2M : Physique de la Matière et des Matériaux
    - CP2S: Chimie et Physique des Solides et des Surfaces
    - SI2M : Science et Ingénierie des Matériaux – Métallurgie
    - N2EV : Nanomatériaux, Electronique Et Vivant
  - 8 centres de compétences et 3 services communs (équipes techniques)
    - Microscopie(s), Analyse X et gamma, Calcul scientifique et Informatique, etc.
    - Cryogénie, Métallographie, etc.
  - Environ 500 personnes (170 chercheurs/enseignants-chercheurs, 100 BIATSS, > 150 doctorants)

Institut Jean Lamour

- L'IJL s'installe entre 2015 et 2017 dans un nouveau bâtiment
  - Campus ARTEM – Molitor, Nancy
  - 28.000 m² SHAUN
  - Regroupement de la plupart des équipes
  - Voisinage de l'Ecole des Mines de Nancy, de l'Ecole Nationale Supérieure d'Arts de Nancy, et de l'ICN Business School
- Pour de plus amples infos :

  http://www.ijl.univ-lorraine.fr



Plan

- Présentation (rapide) de l'IJL
- Des bibliothèques numériques – Pourquoi ?
- Optimisation – Interfaçage
- Utilisation
- Liste des principales bibliothèques
- Conclusions

Pourquoi utiliser des bibliothèques ?

- Capitaliser et réutiliser les développements
  - Réduire le temps de développement
  - Assurer la portabilité (type de variables, systèmes, etc.), la lisibilité
  - Assurer une meilleure performance / une performance optimisée
  - Fiabiliser les développements, assurer leur stabilité
  - Pouvoir disposer d'une communauté d'utilisateurs et de leur support (stabilité, correction de bugs, mutualisation de moyens)

Quels besoins peuvent-elles résoudre ?

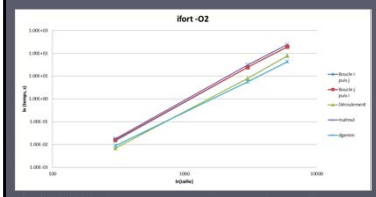
- Algèbre Linéaire (matrices denses, creuses)
- Résolution de systèmes linéaires (directe, itérative)
- Résolution d'EDO
- Calcul de valeurs propres
- Transformée de Fourier rapide
- Générateurs aléatoires
- Optimisation
- Utilisation d'outils système (mesure du temps, threads)
- Gestion des communications (MPI, etc.)
- Gestion des E/S (fichiers normalisés HDF5, XML), Visualisation

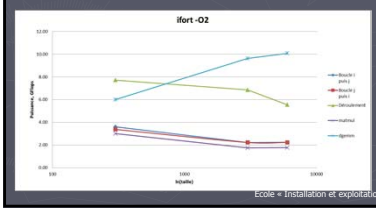
Exemple

- Multiplication de deux matrices A(n,n) et B(n,n)
  - « A la main » basique
  - « A la main » plus évolué
  - Fonction intrinsèque au langage
  - Bibliothèque BLAS (Basic Linear Algebra Subroutines)
- Ordres de grandeur
  - Pour C(i,j) : n multiplications, n additions
    → Pour C : $2n^3$ opérations
  - Avec réels double précision

n	Mémoire / matrice	N opérations
100	78 ko	2 Millions
300	703 ko	54 Millions
1000	7,8 Mo	2 Milliards
3000	70 Mo	54 Milliards
10000	780 Mo	2 000 Milliards

Exemple





- Dès que n augmente, BLAS plus efficace
- Seule version dont l'efficacité augmente avec n
- Dérouler la boucle toujours plus efficace que version de base
- Fonction intrinsèque inefficace
- NB : à corréler avec options de compilation

Premier exemple de bibliothèque

- BLAS : Basic Linear Algebra Subroutines ➔ Opérations algébriques de bas niveau
  - Level 1 : opérations sur les vecteurs $y = \alpha x + \beta y$
  - Level 2 : opérations matrices/vecteurs $y = \alpha A x + \beta y$
  - Level 3 : opérations matrices/matrices $C = \alpha A B + \beta C$
- En fortran 77
- Diffusée en 1979
- Base de très nombreuses bibliothèques ultérieures
- Différentes implémentations : autres langages, parallèles ...
- `http://www.netlib.org/blas`

Bibliothèque BLAS

- Contenu
  - Détermination de constantes machines
  - Procédures écrites pour réels/complexes simple/double précision (quadruple précision en niveau 2)
  - Niveau 1 : échange, copie, multiplication par un scalaire, produit scalaire, norme, etc.
  - Niveau 2 :
    - multiplication matrice-vecteur pour une matrice quelconque, symétrique, bande, hermitienne, triangulaire, etc.
    - gère également les opérations faisant intervenir la transposée et/ou le conjugué
  - Niveau 3 :
    - multiplication matrice-matrice pour une matrice quelconque, symétrique, bande, hermitienne, triangulaire, etc.
    - cas de dgemm ➔ Double GEneral Matrix Multiplication

Plan

- Présentation (rapide) de l'IJL
- Des bibliothèques numériques – Pourquoi ?
- Optimisation – Interfaçage
- Utilisation
- Liste des principales bibliothèques
- Conclusions

Problématique de l'optimisation

- Obtenir de bonnes **performances**
- Sur une grande variété de **matériels**
- Prenant en compte les **caractéristiques** de la machine
- Tout en restant **portable** !

  ➔ soit déjà implémenté (bibliothèques constructeurs)
  ➔ soit à faire soi-même

Bibliothèques constructeurs

- Certaines bibliothèques très utilisées sont optimisées sur les différentes architectures par les constructeurs
  - exemple : BLAS et LAPACK (cf. Intel MKL)
  - il faut les utiliser lorsqu'elles existent
    Avantages : performances optimales pour les machines concernées, rien à réécrire dans le code
- D'autres bibliothèques conçues par les constructeurs ne sont pas portables : leur interface change d'un constructeur à l'autre.
  - cas des FFTW

    Inconvénients : non portables, prévoir des interfaces différentes selon les machines

Optimisation des BLAS standard

- ATLAS Automatically Tuned Linear Algebra Software
  - API BLAS en C et Fortran 77, implémentant quelques fonctionnalités de LAPACK
  - Automated Empirical Optimization of Software (AEOS) : **compilation adaptive** qui ajuste les paramètres en fonction des caractéristiques de la machine (teste à l'installation les tailles de blocs optimales) ➔ recompilation OBLIGATOIRE.
  - `http://math-atlas.sourceforge.net/`
- GOTO BLAS (OpenBLAS) développement de K. GOTO
  - Ré-implémentation des BLAS
  - Prise en compte des caractéristiques des architectures plus modernes jusqu'à l'Intel Nehalem (2008)
  - `http://www.tacc.utexas.edu/tacc-software/gotoblas2`

Optimisation des BLAS standard

- OpenBLAS
  - continuation du développement des Goto BLAS
  - inclut des optimisations pour le processeur Intel SandyBridge
  - `http://xianyi.github.io/OpenBLAS`
- Pour ces bibliothèques
  - autant que possible, ne pas utiliser les binaires fournis par les développeurs
  - on doit **recompiler** les bibliothèques sur son propre système afin de les optimiser
    - ➔ savoir quelles sont les caractéristiques du système
    - ➔ connaître les options « utiles » de son compilateur

Pourquoi interfacer les bibliothèques ?

- Existence de nombreuses bibliothèques Fortran très optimisées
  - si on utilise d'autres langages (C, Python, etc.) on doit interfacer la bibliothèque pour profiter de son optimisation
- Souci d'utiliser le meilleur des différents langages
- Attention
  - possibles différences entre les types
  - possibles différences dans la gestion des E/S
  - possibles différences dans la gestion de la mémoire

Appel croisé Fortran et C/C++

- Echanges de données : types de données communs ou pas
  - entiers
  - flottants (réels simple précision) et réels double précision
  - chaines de caractères (char(0) pour un caractère en C)
- Attention au stockage et à l'indexation des tableaux
  - stockage colonnes et indice débutant à 1 pour Fortran
  - stockage lignes et indice débutant à 0 pour C/C++
- Autres détails amusants
  - sensibilité à la casse : NON=non en Fortran… mais OUI ≠ oui en C

Appel croisé Fortran et C/C++

- En pratique
  - définir un prototype pour la fonction appelée (en C ce sont les fichiers .h)
  - Comme un appel de fonction c depuis le C++ : précéder le prototype de la fonction par le mot clé extern C
  - attention au nom de la fonction fortran dans le code C, en minuscule et terminé par « _ »
  - Passage des paramètres par adresse
  - Des options de compilation utiles : -assume nounderscore (Intel F90), -fno-underscoring (gfortran)

Appel d'autres langages depuis Python

- Appels de code C++ depuis Python
  - SWIG Simplified Wrapper and Interface Generator, appels de programmes en C et C++ depuis des langages de programmation haut niveau (Python, PHP, Perl ...) `http://www.swig.org/`
  - BoostPython interopérabilité C++/Python `www.boost.org/doc/libs/1_42_0/libs/python/doc/index.html`
  - Pyrex permet de mixer du code Python et C `http://www.cosc.canterbury.ac.nz/greg.ewing/python/Pyrex`
  - Cython (basé sur Pyrex) compile Python et permet d'écrire des extensions en C `http://cython.org`

Appel d'autres langages depuis Python

- Appels de code Fortran depuis Python
  - f2py Fortran to Python interface generator
    `http://cens.ioc.ee/projects/f2py2e/`
  - PyFort, création d'extensions python à partir de routines Fortran
    `http://sourceforge.net/projects/pyfortran/`

Plan

- Présentation (rapide) de l'IJL
- Des bibliothèques numériques – Pourquoi ?
- Optimisation – Interfaçage
- Utilisation
- Liste des principales bibliothèques
- Conclusions

Installation

- Quand on a identifié la bibliothèque qui nous intéresse, plusieurs possibilités
  - Elle est fournie sous forme de binaire : à éviter (sauf peut-être Windows)
  - Elle est packagée dans les distributions Linux
    - Installation via le gestionnaire de paquet de la distribution
      ex : `apt-get install libfftw3-3`
    - Attention, dans ce cas, l'installation se fait sous forme binaire : pas de compilation (potentiellement moins bonnes performances)
  - Elle est fournie sous forme d'archive de ses sources
    - cas idéal (mais plus compliqué…)
    - téléchargement d'un fichier tar.gz ou autre
    - `configure, make, make install`
    - compilation à la main (il y a quand même en général un Makefile sinon…)

Principes généraux

- Différents éléments de bibliothèques
  - /usr/lib ➔ libtruc.so, libtruc.a, libtruc.so3, etc.
  - extension .a (.lib Windows) ➔ bibliothèque statique
  - extension .so<x> (.dll Windows) ➔ bibliothèque dynamique
- Compilation en dynamique
  - l'exécutable ne contient que le programme mais il faut que le fichier .so soit installé sinon on ne peut pas utiliser le programme
- Compilation en statique :
  - l'exécutable contient le programme + une copie du fichier .a (.lib). On pourra utiliser le programme même si sur un système il n'y a ni les .a (.lib) ni les .so (.dll)

En pratique

- Edition de liens (ld)
  - compilation du code avec référence aux bibliothèques utilisées par les options
    - `-L/chemin/vers` : pour préciser le chemin
    - `-ltruc` si la bibliothèque a été installée de façon classique (recherche de libtruc.so puis de libtruc.a)
    - Attention à l'ordre d'appel des bibliothèques
- Exécution du programme
  - on peut voir les dépendances d'un code avec l'outil ldd
  - la variable d'environnement LD_LIBRARY_PATH permet d'indiquer l'emplacement de bibliothèques
  - En général, le système va chercher dans `/usr/lib`, puis `lib` (défini dans `/etc/ld.so.conf`).

Utilisation des bibliothèques

- Bibliothèque statique
  - extension .a (.lib Windows)
  - peut contenir un ou plusieurs fichiers objets (.o)
  - débute par un index (consultable par la commande linux `nm`)

```
> nm -s /usr/lib/libtoto.a
> Archive index:
  suba in sub.o
  foncb in fonc.o

  sub.o:
           U lecdon
           U basename
  00000000 T suba
           U subb
  foncb.o:
           U close
  00000000 T foncb
```

  - T : symbole défini dans l'archive ; U : symbole défini ailleurs
  - l'option --defined-only permet de lister seulement les T

Utilisation des bibliothèques

- Bibliothèque statique
  - à la compilation, si la bibliothèque est statique, le code du fichier objet est inclus dans l'exécutable.
  - on peut alors exécuter le programme sur un ordinateur qui ne comporte pas la bibliothèque (utile dans le cas d'une bibliothèque commerciale)
  - l'exécutable occupe plus d'espace disque.
    → si les SP de la bibliothèque sont très utilisées et très standards, on évitera la version statique de la bibliothèque

Utilisation des bibliothèques

- Bibliothèque dynamique (partagée)
  - si la bibliothèque comporte l'extension .so (souvent suivi du numéro de version) → bibliothèque partagée (shared object).
  - à l'édition des liens, une portion de code est insérée, elle est utilisée lorsque l'application est exécutée pour lancer l'éditeur de liens dynamique (localise les bibliothèques partagées et les charge en mémoire)
  - le code du SP ne sera plus dupliqué dans l'exécutable
    - exécutable plus compact
    - moins de place mémoire (si plusieurs applications utilisent en même temps le code, une seule instance en mémoire)
    - nécessité de disposer sur la machine de la bibliothèque .so
    - code entièrement chargé en mémoire donc attention à ce qu'on met dans une bibliothèque partagée

Création de bibliothèques

- Bibliothèque statique
  - compilation du/des fichier(s) objet(s) avec l'option -c
  - utilisation de la commande `ar` pour générer la bibliothèque
    - option r pour ajouter
    - option t pour lister
  - ```
    > gcc -c sub.c
    > ar r libtoto.a sub.o
    > ar t libtoto.a
    > sub.o
    ```
  - création de l'index et stockage dans l'archive par commande `ranlib`
  - utilisation de `nm` pour vérifier les symboles définis et indéfinis dans l'archive
  - ```
    > ranlib liboutils.a
    > nm -s liboutils.a
    ```

Création de bibliothèques

- Appel de la bibliothèque
  - option `-L/chemin/vers/les_bibs` pour indiquer le répertoire qui contient la/les bibliothèques
  - puis `-ltoto` si fichier libtoto.a
  - option `-I/chemin/vers/les_entetes` pour indiquer où se trouvent les en-têtes (prototypes)
  - `> gcc monprog.c –I $HOME/include -L $HOME/lib -ltoto -o monprog`

Création de bibliothèques

- Bibliothèque dynamique
  - compilation du/des fichier(s) objet(s) avec les options –c –fPIC (Position Independent Code, code non relogeable)
  - compilation de la bibliothèque avec l'option –shared
  - `> gcc -c –fPIC sub.c`
    `> gcc –shared –o libtoto.so sub.o`

Création de bibliothèques

- Appel de la bibliothèque
  - par défaut, une bibliothèque dynamique se place dans `/usr/lib`
  - si on n'a pas les droits, on place la bibliothèque dans un répertoire local (ex. `/lib`)
  - on compile en utilisant l'option `Wl,-rpath,$HOME/lib`
    - ajoute le répertoire précisé à la liste des répertoires dans lesquels l'éditeur de liens dynamiques cherche les bibliothèques dynamiques
    - `-Wl` indique qu'il faut passer `–rpath,$HOME/lib` à l'éditeur de liens (option utilisée lors de la dernière étape de la compilation).
    - `$HOME/lib` est donc explicité 2 fois dans la ligne de commande
    - `> gcc monprog.c -I $HOME/include -L $HOME/lib -ltoto -Wl,-rpath,$HOME/lib -o monprog`

Création de bibliothèques

- Appel de la bibliothèque
  - la commande `ldd` permet de connaître les dépendances dynamiques de l'exécutable
    - `> ldd monprog`

```
        libtoto.so => /home/toto/lib/libtoto.so (0x40016000)
        libc.so.6 => /lib/libc.so.6 (0x40023000)
        /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

  - libtoto.so : bibliothèque créée située dans le répertoire indiqué à droite de la flèche
  - libc : bibliothèque standard du C
  - ld-linux.so : éditeur de lien dynamique

Gestion des extensions

- les bibliothèques sous linux ont des extensions .so ou .so.n1 voire .so.n1.n2
- une bibliothèque partagée ne peut pas être effacée et remplacée par une nouvelle si la mise à jour est majeure (les applications qui l'utilisaient ne fonctionneraient plus), c'est-à-dire lorsque n1 a changé
- dans le cas d'une modification mineure, il est possible de remplacer l'ancienne version par la nouvelle, lorsque n2 a changé
- plusieurs versions de la même bibliothèque partagées peuvent être présentes dans le système (pour une gestion plus fine ➔ ***modules***)

Gestion des extensions

- Une bibliothèque a plusieurs noms
  - nom réel ➔ le fichier qui contient le code (ex. libm-2.2.4.so)
  - nom d'objet partagé (soname) ➔ lien symbolique vers la bibliothèque (le nom réel)
    - composé du nom de la bibliothèque et de son extension majeure (ex. libm.so.6)
    - dans l'exécutable, c'est le soname qui est stocké
  - nom de lien utilisé par le compilateur ➔ nom de la bibliothèque sans indication de version majeure et mineure (ex. libm.so)
    c'est un lien symbolique vers le soname

Gestion des extensions

- Une bibliothèque a plusieurs noms
  - > locate libm.so
    > /usr/lib/libm.so
    /lib/libm.so.6
  - > ls -la /usr/lib/libm.so
    > lrwxrwxrwx 1 root root 21 oct 2014 09:16
    /usr/lib/libm.so -> ../../lib/libm.so.6
  - > ls -la /lib/libm.so.6
    > lrwxrwxrwx 1 root root 21 oct 2014 10:21
    /lib/libm.so.6 -> libm-2.2.4.so*
  - > ls -la /lib/libm-2.2.4.so
    > -rwxr-xr-x 1 root root 21 oct 2014
    /lib/libm-2.2.4.so

Gestion des extensions

- Exemple où deux versions majeures sont présentes
  - > ls -l libstdc++.so.*
    > lrwxrwxrwx 1 root root 18 oct 2014 09:16
    libstdc++.so.3 -> libstdc++.so.3.0.4
    -rwxr-xr-x 1 root root 14 fev 2012
    libstdc++.so.3.0.4
    lrwxrwxrwx 1 root root 18 oct 2014 17:13
    libstdc++.so.5 -> libstdc++.so.5.0.0
    -rwxr-xr-x 1 root root 16 aou 2011 11:52
    libstdc++.so.5.0.0
  - le soname pour la version majeure 3 pointe sur le nom réel libstdc++.so.3.0.4
  - le soname de la version 5 pointe vers le nom réel libstdc++.so.5.0.0
  - le nom de lien pointe vers la dernière version majeure

Les modules

- Sur un cluster, plusieurs versions d'une même bibliothèque peuvent être installés
  - Soit qu'on veut tester différentes versions (performance, correction de bugs)
  - Soit que des logiciels installent/utilisent des versions spécifiques
- Je viens d'installer un nouveau logiciel qui a besoin d'une bibliothèque spécifique (libc, libqt, etc.) qui est déjà présente en plusieurs versions sur mon système
  - Quelle version sera utilisée ? ➔ la 1ère que le système « trouvera »
  - Est-ce que cela marchera ? ➔ pas sûr…
- Solution : Modules d'environnement (≠ modules du noyau)

Les modules : utilisation

- But : définir un environnement spécifique (chemins des logiciels, des bibliothèques, etc.) pour un logiciel donné
- Exemple

```
> module avail
> ------------------------- /cm/shared/modulefiles ----------------------------------
acml/gcc/64/4.3.0                  fftw3/gcc/64/3.2.2
acml/gcc/mp/64/4.3.0               fftw3/open64/64/3.2.2             mpiexec/0.84_427
acml/gcc-int64/64/4.3.0            gcc/4.3.4                         mvapich/gcc/64/1.1
acml/gcc-int64/mp/64/4.3.0         globalarrays/gcc/openmpi/64/4.2   mvapich/open64/64/1.1
acml/open64/64/4.3.0               globalarrays/open64/openmpi/64/4.2 mvapich2/gcc/64/1.2
acml/open64-int64/64/4.3.0         hdf5/1.6.9                        mvapich2/open64/64/1.2
blacs/openmpi/gcc/64/1.1patch03    hpl/2.0                           netcdf/gcc/64/4.0.1
blacs/openmpi/open64/64/1.1patch03 intel-cluster-checker/1.3         netcdf/open64/64/4.0.1
blas/gcc/64/1                      intel-cluster-runtime/2.1         netperf/2.4.5
blas/open64/64/1                   intel-tbb/ia32/22_20090809oss     open64/4.2.2.2
bonnie++/1.96                      intel-tbb/intel64/22_20090809oss  openmpi/gcc/64/1.3.3
cmgui/5.0                          iozone/3_326                      openmpi/open64/64/1.3.3
default-environment                lapack/gcc/64/3.2.1               scalapack/gcc/64/1.8.0
fftw2/gcc/64/double/2.1.5          lapack/open64/64/3.2.1            scalapack/open64/64/1.8
fftw2/gcc/64/float/2.1.5           mpich/ge/gcc/64/1.2.7             sge/6.2u3
fftw2/open64/64/double/2.1.5       mpich/ge/open64/64/1.2.7          torque/2.3.7
fftw2/open64/64/float/2.1.5        mpich2/smpd/ge/gcc/64/1.1.1p1
```


Les modules : utilisation

- Par la commande `module load` je charge uniquement la bibliothèque que je veux
  - `> module load torque` (suffit : pas d'homonymie)
    `> module load mvapich/open64/64/1.1`
  - Pour lister les modules que j'ai chargés : `module list`
  - Pour décharger un module : `module unload <module(s)>`
  - Pour décharger tous les modules : `module purge`
  - Peut être scripté dans des fichiers batch pour torque/maui, slurm, etc.
- Permet de (re)définir les variables d'environnement comme $PATH, $MANPATH, $LD_LIBRARY_LOAD, etc.

Les modules

- Intérêt supplémentaire : commande utilisable par chaque utilisateur, sans droits root
- Mais c'est un outil qui doit être installé (ce n'est pas une commande du système)

  ➔ `http://modules.sourceforge.net`

Créer un module

- Template fourni à écrire en Tcl
- Exemple : nouveau module pour le compilateur gcc 4.6.2 que j'ai installé dans mon home
  - En tant que root, taper :

```
> cd /cm/shared/modulefiles
> mkdir compilateurs
> cp modules compilateurs/gcc-4.6.2
```

  - Permet de définir un dossier compilateurs dans lequel sera installé entre autres le nouveau gcc sous forme d'un template
  - Dans la commande `module list` apparaîtra la nouvelle ligne `compilateurs/gcc-4.6.2`
  - Editer le template

Créer un module

```
#%Module1.0#####################################################################
## modules compilateurs/gcc-4.6.2
##
## modulefiles/compilateurs/gcc-4.6.2.
##
proc ModulesHelp { } {
      global version modroot
      puts stderr "compilateurs/gcc-4.6.2 – definit un environnement
            pour GCC 4.6.2 installe dans mon home"}
module-whatis   "environnement pour gcc-4.6.2 (C, Fortran)«
# for Tcl script use only
set     topdir          /home/dussou13/gcc-4.6.2
set     version         4.6.2
set     sys             linux86
setenv          CC              $topdir/bin/gcc
setenv          GCC             $topdir/bin/gcc
setenv          FC              $topdir/bin/gfortran
setenv          F77             $topdir/bin/gfortran
setenv          F90             $topdir/bin/gfortran
prepend-path    PATH            $topdir/include
prepend-path    PATH            $topdir/bin
prepend-path    MANPATH         $topdir/man
prepend-path    LD_LIBRARY_PATH $topdir/lib
```


Créer un module

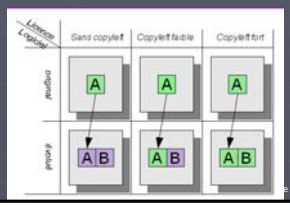
- Quelques explications
  - `> module help compilateurs/gcc-4.6.2`

```
--- Module Specific Help for 'compilateurs/gcc-4.6.2' ----
compilateurs/gcc-4.6.2 – definit un environnement pour GCC
4.6.2 installe dans mon home
```

  - `> module whatis compilateurs/gcc-4.6.2`
    `environnement pour gcc-4.6.2 (C, Fortran)`
  - Puis création de variables d'environnement et mise à jour des chemins

Problématique des licences

- **Ne pas** utiliser des briques logicielles sans connaître leur origine, leur licence
- **Ne pas** diffuser du code sans licence
- On n'a pas le droit de diffuser comme on veut
  - du point de vue juridique, seul le titulaire des droits patrimoniaux (celui qui paye) peut décider d'une diffusion du logiciel en libre ou non
- Certaines licences sont contaminantes :



Plan

- Présentation rapide du réseau Calcul Scientifique et de l'IJL
- Des bibliothèques numériques – Pourquoi ?
- Optimisation – Interfaçage
- Utilisation
- Liste des principales bibliothèques
- Conclusions

Quelques sites utiles

- Projet PLUME
  `www.projet-plume.org/logiciels_maths`
- The Guide To Avail. Mathematical Software
  `gams.nist.gov`
- Wikipedia `en.wikipedia.org/wiki/Mathematical_software`
- Arnold Neumaier
  `www.mat.univie.ac.at/~neum/software.html`
- Antoine Le Hyaric
  `www.ann.jussieu.fr/~lehyaric/freesoft/`
- Florian De Vuyst
  `www.twiki.org/cgi-bin/view/Main/FlorianDeVuyst`
- Jack Dongarra
  `www.netlib.org/utk/people/JackDongarra/la-sw.html`

Algèbre linéaire

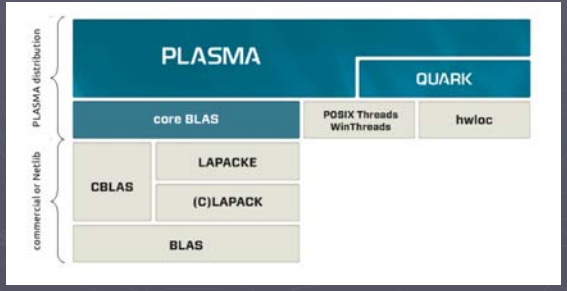
- Blitz++
  - bibliothèque générique de tableaux et vecteurs en C++
  - utilise intensivement les « Expression Templates »
  - `http://blitz.sourceforge.net/`
- Armadillo
  - bibliothèque générique en C++
  - `http://arma.sourceforge.net`
- SparseKit
  - manipulation de matrices creuses de différents formats
  - écrit en Fortran 90
  - `http://people.sc.fsu.edu/~jburkardt/f_src/sparsekit/sparsekit.html`

Systèmes linéaires, LAPACK

- Linear Algebra PACKage
  - algèbre linéaire de haut niveau
  - matrices pleines ou bandes, mais pas adapté aux matrices creuses
  - factorisations LU, Cholesky, QR et Schur, valeurs propres et vecteurs propres, décomposition en valeurs singulières ...
  - utilise intensivement les BLAS
  - base de nombreux outils numériques (Matlab, Scilab, SciPy, ...)
  - écrit en fortran 77
  - dernière version : 3.2.1, avril 2009
  - Version distribuée : ScaLaPack
  - `http://www.netlib.org/lapack` ou `/scalapack`

Systèmes linéaires

- PLASMA : Parallel Linear Algebra Software for Multicore Architecture
  - `http://www.prace-ri.eu/uploads/tx_pracetmo/plasma_magma.pdf`



Systèmes linéaires

- PETSc
  - gestion de matrices et vecteurs parallèles (basé sur MPI)
  - solveurs itératifs de systèmes linéaires parallèles
  - en C objet. Très bon support fortran. Interface Python
  - bibliothèque cohérente : on a des sous-progs, mais aussi des objets
  - `http://www.mcs.anl.gov/petsc/`

PETSc

- Des sous-programme de résolution + des classes d'objets
  - gestion des vecteurs et matrices en parallèle
  - Ex :
```
VecCreate(MPI_Comm, Vec*)
VecDestroy(Vec*)
VecSetSizes(Vec, PetscInt n, Petscint N)
etc.
```
  - différents formats de matrices : Denses, symétriques, CSR, etc.
  - matrices automatiquement partitionnées entre processus MPI par blocs de lignes
  - Ex :
```
MatCreate(MPI_Comm, Mat*)
MatSetSizes, MatSetType, MatSetFromOptions
```


PETSc

- Résolution de systèmes linéaires
  - méthodes de Krylov (KSP) et préconditionneurs (PC)
    - gradients conjugués, GMRES
    - Jacobi, méthodes de Schwartz, ILU, etc.
  - accès à de nombreuses bibliothèques externes via interface de PETSc : solveurs directs, multigrilles, etc.
- Résolution de systèmes non linéaires

Systèmes linéaires

- Trilinos
  - en C++. Interfaces Fortran 2003 et Python
  - organisé comme une collection de packages (discrétisation, géométrie, maillage, algèbre linéaire, solveurs linéaires ou non, E/S
  - plateformes parallèles
  - installation et utilisation non triviales
  - `http://trilinos.org/`
- SuperLU
  - résolution directe de très grands systèmes creux non symétriques par factorisation LU
  - plusieurs versions séquentielles et parallèles, en C
  - `http://crd.lbl.gov/~xiaoye/SuperLU/`

Systèmes linéaires

- MUMPS
  - solveur direct parallèle de systèmes linéaires creux
  - Fortran 90 + MPI
  - différents types de matrices (symétriques définies positives, symétriques générales, et non-symétriques générales)
  - très stable numériquement, très bon support utilisateurs
  - `http://mumps.enseeiht.fr/`
- PaStiX, Parallel Sparse matriX package
  - résolution de très grand systèmes linéaires creux en utilisant une méthode directe
  - En C, support GPU
  - parallélisme de type MPI et/ou Thread
  - `http://pastix.gforge.inria.fr/`

Systèmes linéaires

- HIPS, Hierarchical Iterative Parallel Solver
  - Méthodes de résolution hybride directe/itérative
  - `http://hips.gforge.inria.fr/`
- Scotch
  - partitionneur séquentiel et parallèle de graphes et re-numéroteur de matrices creuses
  - intégré dans les solveurs MUMPS et PaStiX
  - `http://www.labri.fr/perso/pelegrin/scotch/`

Systèmes linéaires

- Metis
  - partitionneur de graphes, de maillages et renumérotation de matrices creuses
  - intégré notamment dans les solveurs MUMPS et HIPS
  - `http://glaros.dtc.umn.edu/gkhome/views/metis`
- Hypre
  - résolution de très grands systèmes linéaires creux sur machine parallèle
  - préconditionneurs performants dont multi-grille sur grille structurée et
  - non structurée
  - `https://computation-rnd.llnl.gov/linear_solvers/software.php`

Valeurs et vecteurs propres

- Lapack, ScalaPack, PLASMA
  - matrices denses
- ARPACK, ARnoldi PACKage
  - calcul des valeurs propres et des vecteurs propres de matrices creuses de grande taille
  - basée sur les algorithmes itératifs de Lanczos/Arnoldi
  - écrit en fortran 77
  - `http://www.caam.rice.edu/software/ARPACK/`

Valeurs et vecteurs propres

- SLEPc
  - implémentation parallèle de recherche de valeurs propres
  - décomposition en valeurs singulières
  - construit au dessus de PETSc
  - `http://www.grycap.upv.es/slepc/`
- ELPA
  - solveur parallèle distribué pour matrices denses symétriques
  - `http://elpa.rzg.mpg.de/`

Transformée de Fourier

- FFTW
  - Fast Fourier Transform in the West
  - calcul de TF discrète en plusieurs dimensions
  - écrit en C. Interface Fortran
  - Adapte le choix de l'algorithme au matériel
  - `http://www.fftw.org/`
- FFTPACK
  - package Fortran pour la FFT
  - `http://www.netlib.org/fftpack/`

Résolution d'ODE

- ODEPACK
  - solveurs écrits en Fortran 77 pour la résolution de systèmes différentiels ordinaires
  - `http://computation.llnl.gov/casc/odepack/odepack_home.html`
- GSL (Gnu Scientific Library)
  - large choix de routines mathématiques écrites en C/C++ dont des solveurs d'ODE (bibliothèque généraliste)
  - `http://www.gnu.org/software/gsl/gsl.html`
- SUNDIALS, SUite of Nonlinear and DIfferential/ALgebraic equation Solvers
  - résolution de systèmes d'ODE et d'ADE
  - écrit en C
  - `https://computation.llnl.gov/casc/sundials/main.html`

Entrées/Sorties

- HDF5, Hierarchical Data Format
  - format standardisé d'entrées/sorties, séquentielles et parallèles
  - existence d'API en C, C++ et Fortran 90
  - `www.hdfgroup.org/HDF5/`
- XML, Extensible Markup Language
  - langage informatique de balisage générique
  - nombreuses librairies permettant la génération de fichiers en XML
  - `http://www.w3.org/XML/`

Entrées/Sorties

- NetCDF
  - bibliothèque et format de données portables
  - Supporte HDF5, peut s'utiliser en parallèle
  - `www.unidata.ucar.edu/software/netcdf/`
- MPI-IO
  - interface d'E/S parallèles incluse dans MPI-2
  - Portabilité des fichiers binaires pas assurée

Visualisation

- VTK, Visualization ToolKit
  - bibliothèque pour la visualisation de gros volumes de données 2D ou 3D
  - écrite en C++. Interface pour C, Python, Java
  - peut être utilisée soit directement via les langages C++, Python, etc., soit indirectement via des interfaces graphiques telles que Paraview ou Mayavi
  - `http://www.vtk.org/`
- OpenDX, Open Data Explorer
  - bibliothèque d'IBM pour la visualisation de données scientifiques
  - `http://www.opendx.org/`

Parallélisation

- Bibliothèques MPI
  - MPICH `http://www.mpich.org/`
  - LAM/MPI `http://www.lam-mpi.org/`
  - OpenMPI `http://www.open-mpi.org/`
- Bibliothèques de threads
  - TBB, Threading Building Blocks `http://www.threadingbuildingblocks.org/`
  - Boost Threads ➔ bibliothèques C++ généralistes offrant de nombreuses fonctionnalités : outils systèmes, threads, sérialisation, interface C++/Python, etc. `http://www.boost.org/`
  - norme POSIX : fournit un ensemble de primitives permettant de réaliser des threads, les pthreads

Bibliothèques généralistes – autres outils

- BOOST
  - bibliothèques C++ généralistes offrant de nombreuses fonctionnalités : outils systèmes, threads, sérialisation, interface C++/Python, etc.
  - `http://www.boost.org/`
- GSL
  - Gnu Scientific Library, bibliothèque généraliste en C/C++
  - `http://www.gnu.org/software/gsl/gsl.html`

Bibliothèques généralistes – autres outils

- Bibliothèques Eléments finis
  - GetFEM++, bibliothèque C++ d'éléments finis `http://home.gna.org/getfem/`
  - MELINA, bibliothèque de calculs éléments finis en Fortran `http://anum-maths.univ-rennes1.fr/melina/danielmartin/melina/`
  - OFELI, bibliothèque éléments finis orienté objet en C++ `http://www.ofeli.net/`
  - FreeFEM++, bibliothèque éléments finis orienté objet en C++ `http://www.freefem.org/ff++`

Bibliothèques généralistes – autres outils

- Outils généralistes
  - Trilinos, boîtes à outils d'algorithmes performants `http://trilinos.sandia.gov`
  - OpenFOAM, outil pour la CFD `http://www.openfoam.com`
  - etc.
- Générateurs de nombres aléatoires
  - GSL `http://www.gnu.org/software/gsl/gsl.html`
  - BOOST `http://www.boost.org/`
  - fonctions intrinsèques en Fortran et en C++

Plan

- Présentation (rapide) de l'IJL
- Des bibliothèques numériques – Pourquoi ?
- Optimisation – Interfaçage
- Utilisation
- Liste des principales bibliothèques
- Conclusions

Conclusions

- Intérêt des bibliothèques de calcul scientifique
  - efficacité, gain de temps en développement, support
  - on peut se consacrer à autre chose
- No pain no gain
  - la plupart du temps, un effort à faire (a minima recompilation)
  - certaines bibliothèques sont un peu plus difficiles à installer/maîtriser
  - si la bibliothèque de votre choix n'est pas dans le langage de votre choix, interfaçage nécessaire
  - quand on en a trouvé une qui nous convient, on a intérêt à la garder

Conclusions

- A la question « quelle est LA bibliothèque idéale pour moi »
  - pas de réponse évidente : dépend du langage, du code, etc.
  - on peut toujours commencer par la base : BLAS, LAPACK (ou ce que fournit le compilateur, e.g. MKL Intel)
  - en cas de besoin très particulier en choisir une dédiée (ex. FFTW)
  - pour aller plus loin : MUMPS (calcul parallèle)
  - pour avoir un environnement complet : PETSc

Mais de façon générale, une bibliothèque, comme un langage, nécessite un **investissement** pour en tirer la quintessence

Merci pour votre attention !

Ecole « Installation et exploitation d'un cluster de calcul » Ste Foy lès Lyon, 18-22 janvier 2016

Petit TP

- Appel C depuis fortran
  - Volontairement « rétrograde » : F77 et C (privilégiez F2003 / C++ ➔ interopérabilité)

progtest.f

```
program test
      implicit none
      integer i, j, k
      common/ijk/ i, j, k
      real*8  x
      character*50 chaine

      i = 2
      j = 3
      k = 4
      x = 123.456
      chaine = 'Exemple de chaine de caracteres'
      write(*,10) i, x
      call abc(i)
      write(*,20) i
      call doubleIJK(chaine)
      write(*,30) i, j, k
      write(*,40) chaine
10    format('prog ppal       ==> i= ',i2,5x,'x= ',e12.5)
20    format('apres abc       ==> i= ',i2)
30    format('apres doubleIJK ==> i= ',i2,' j= ', i2, ' k= ', i2)
40    format('apres doubleIJK ==> ',a32)
      end

      subroutine abc(m)
      implicit none
      integer m
      m = m * 10
      return
      end
```

doubleIJK.c

```
#include <stdio.h>

extern struct
 { int i, j, k; } ijk;

int doubleIJK(char *cc, int longueur){
cc[longueur-1] = '\0';
// caractere NULL pour terminer la chaine
printf("dans doubleIJK  ==> %s\n",cc);

ijk.i *= 2;
ijk.j *= 2;
ijk.k *= 2;
return(1);}
```

Ecole « Installation et exploitation d'un cluster de calcul » Ste Foy lès Lyon, 18-22 janvier 2016

Petit TP

- Compilez *progtest.f* et *doubleIJK.c*
  - `> gfortran -c progtest.f`
  - `> gcc -c doubleIJK.c`
  - Est-ce que cela marche ? Pourquoi ?
- Faites l'édition de lien
  - `> gfortran -o progtest doubleIJK.o progtest.o`
  - Est-ce que cela marche ? Pourquoi ?
  - Quelles sont les erreurs ? (cf commande `nm` par exemple)
- Apportez les corrections nécessaires pour que l'exécutable soit généré
  - Version 1 : en modifiant uniquement le fichier .c
  - Version 2 : en modifiant en partie le fichier .c ET la compilation
- Exécutez
  - NB : le résultat n'a aucun intérêt en soi…

Petit TP : solution

- Version 1

doubleIJK.c

```
#include <stdio.h>

extern struct
 { int i, j, k; } ijk_;

int doubleijk_ (char *cc, int longueur){
cc[longueur--] = '\0';
// caractere NULL pour terminer la chaine
printf("dans doubleIJK  ==> %s\n",cc);

ijk_.i *= 2;
ijk_.j *= 2;
ijk_.k *= 2;
return(1);}
```

- `> gfortran -c progtest.f`
- `> gcc -c doubleIJK.c`
- `> gfortran -o progtest doubleIJK.o progtest.o`
- `> ./progtest`

- Version 2

doubleIJK.c

```
#include <stdio.h>

extern struct
 { int i, j, k; } ijk;

int doubleijk (char *cc, int longueur){
cc[longueur--] = '\0';
// caractere NULL pour terminer la chaine
printf("dans doubleIJK  ==> %s\n",cc);

ijk.i *= 2;
ijk.j *= 2;
ijk.k *= 2;
return(1);}
```

- `> gfortran -c -fno-underscoring progtest.f`
- `> gcc -c doubleIJK.c`
- `> gfortran -o progtest doubleIJK.o progtest.o`
- `> ./progtest`


```
do i=1,n1
   do j=1,n1
      do k=1,n1
         c1(i,j) = c1(i,j) + a1(i,k)*b1(k,j)
      end do
   end do
end do
```

```
do i=1,n1
  do j=1,n1
      do k=1,n1,10
         s0=a1(i,k)*b1(k,j)     ; s1=a1(i,k+1)*b1(k+1,j)
         s2=a1(i,k+2)*b1(k+2,j) ; s3=a1(i,k+3)*b1(k+3,j)
         s4=a1(i,k+4)*b1(k+4,j) ; s5=a1(i,k+5)*b1(k+5,j)
         s6=a1(i,k+6)*b1(k+6,j) ; s7=a1(i,k+7)*b1(k+7,j)
         s8=a1(i,k+8)*b1(k+8,j) ; s9=a1(i,k+9)*b1(k+9,j)
        c1(i,j)=s0+s1+s2+s3+s4+s5+s6+s7+s8+s9
      end do
   end do
end do
```

```
c1=matmul(a1,b1)
```

```
call dgemm('n','n',n1,n1,n1,1.d0,a1,n1,b1,n1,0.d0,c1,n1)
```

```
#include <stdio.h>
#include <math.h>
#include <stdlib.h>
/* valeur arrondie par exces ou par defaut pour un reel entre sur la
   ligne de commande */

int main(int argc, char *argv[]){
  float nb;
  if(argc != 2){printf("usage : arrondir nb_reel\n");exit(1);}
  nb = atof(argv[1]);
  printf("arrondi par exces : %d\n", ceil(nb));
  printf("arrondi par defaut : %d\n", floor(nb));
  return 0;}

> gcc -Wall arrondir.c -o arrondir
> /tmp/ccTffpuW.o: In function `main':
> /tmp/ccTffpuW.o(.text+0x4d): undefined reference to `ceil'
> /tmp/ccTffpuW.o(.text+0x79): undefined reference to `floor'
> collect2: ld returned 1 exit status

> gcc -Wall -lm arrondir.c -o arrondir
>
```

rien = compilateur content