Bibliothèques pour le calcul scientifique

« outils, enjeux et écosystème »

Mercredi 15 mai 2013

Coordination scientifique :
– ***Philippe d'Anfray*** *(CEA)*
– ***Michel Kern*** *(INRIA, Maison de la SImulation)*
– ***Thiên-Hiêp Lê*** *(ONERA)*

Amphithéâtre Becquerel, École Polytechnique, Palaiseau

`http://www.association-aristote.fr`
`info@association-aristote.fr`

Edition du 4 prairial an CCXXI (*vulg.* 23 mai 2013)

ARISTOTE Association Loi de 1901. Siège social : CEA-DSI CEN Saclay
Bât. 474 91191 Gif-sur-Yvette Cedex.
Secrétariat : Aristote, École Polytechnique, 91128 Palaiseau Cedex.
Tél. : +33(0)1 69 33 99 66 Fax : +33(0)1 69 33 99 67
Courriel : `Marie.Tetard@polytechnique.edu`
Site internet `http://www.association-aristote.fr`

Texto

Gentes Dames, Gentils Seigneurs
Ce jour, une Histoire va vous être contée
C'est celle de la Compagnie du Logiciel
Dont les compères *Legolas*, *Magma* et *Plasma*
Ont trouvé la clef du *Cadna* du club *FreeFEM*
Dans lequel ils rencontraient le Professeur *Eigen*
Consultant les *News* dans *OpenPalm*
Sous l'œil *Multiple* de deux *Agences Gouvernementales*
Qui ostensiblement *Évitaient les Communications*.
Tout ce beau monde se retrouvait autour d'un *Scotch* millénaire.
Auparavant, faisons un raccourci *Philosophie-Simulation*
Aristote par la voix de Philippe et *Maison* par celle de Michel.

THLE

Table des matières

Chapitre 1

Programme de la journée

1.1 Introduction

Les bibliothèques logicielles scientifiques sont des composants majeurs des grands logiciels de simulation ; ce sont aussi des portes d'accès au calcul haute performance (HPC).

Ce séminaire autour des bibliothèques scientifiques permettra de découvrir ou mieux connaître de nombreuses réalisations dans ce domaine. Parmi les questions qui se posent :

- quels sont les éléments "moteurs" du projet (algorithme, domaine d'application, ...) et ses cibles ;
- quelles sont les évolutions en cours ou prévues ;
- comment inscrire ce type de projet dans la durée (par exemple, au delà d'un financement initial) ;
- y a-t-il une spécificité liée à ce travail de développement logiciel, et une reconnaissance en tant que tel (des pairs, des organismes concernés, ...) ?
- comment diffuser (Open Source, ...), la protection intellectuelle, quelles licences ;
- comment constituer et gérer une communauté d'utilisateurs (quels services ? quels retours ?)

La disponibilité de paquetages logiciels performants, validés et maintenus est un des éléments qui permet la « cristallisation » de projets scientifiques dans de nombreux domaines applicatifs (aérodynamique, climat, matériaux, chimie, *etc.*) mais aussi le maintien d'un savoir faire partagé par des acteurs d'horizons très variés. Un point-clef pour l'interdisciplinarité.

1.2 Programme

09h00-09h30	**Marie Tétard** (Aristote) *Accueil des participants, café*	
09h30-09h35	**Thiên-Hiep Lê** (ONERA)	Introduction
09h35-09h45	**Philippe d'Anfray** (CEA, Aristote)	Présentation Aristote
09h45-10h00	**Michel Kern** (Inria, Maison de la SImulation)	Présentation Maison de la SImulation
	Animateur de session : Michel Kern	
10h00-10h30	**Marc Baboulin** (Inria, Université Paris-Sud)	Fast and reliable linear system solutions on new parallel architectures
10h30-11h00	**Laura Grigori** (Inria)	How to Avoid Communication in Linear Algebra and Beyond
11h00-11h30	*Pause café*	
	Animateur de session : Michel Kern (*cont.*)	
11h30-12h00	**Gaël Guennebaud** (Inria)	Eigen : a C++ template library for linear algebra and related numerical algorithms
12h00-12h30	**Fabienne Jézéquel** (LIP6, UPMC)	Estimation d'erreur d'arrondi par la bibliothèque CADNA
12h30-13h30	*Déjeuner (salon de marbre)*	
	Animateur de session : Philippe d'Anfray	
13h30-14h00	**François-Xavier Roux** (ONERA)	Mise en oe uvre de méthodes de résolution par sous-domaines parallèles dans des codes d'éléments fini
14h00-14h30	**Laurent Plagne** (EDF)	Legolas++ Conception d'outils génériques pour les prob-lèmes linéaires creux structurés par blocs multi-niveaux
14h30-15h00	**Florent Duchaine** (Cerfacs)	OpenPALM, an open source code coupler for massively parallel multi-physics/multi-components applications and dynamic algorithms
15h00-13h30	**François Pellegrini** (LaBRI, Inria, Université Bordeaux 1)	« How to age well a 20 y.o. Scotch »
15h30-16h00	*Pause*	
	Animateur de session : Thiên-Hiêp Lê	
16h00-16h30	**Frédéric Hecht** (UPMC)	FreeFem++, un logiciel pour résoudre numériquement des équations aux dérivées partielles
16h30-17h00	**Christophe Calvin** (CEA/DEN/DANS/DM2S)	Les bibliothèques scientifiques massivement parallèles : utilisation dans les grands codes de simulation pour l'énergie nucléaire
17h00-17h30	**Patrick Moreau** (Inria)	Mais qui est l'éditeur de ces bibliothèques ?
17h30	Clôture, fin du séminaire	

Compte-rendu de la journée

Ce compte-rendu a été réalisé par Fabien Nicolas de l'agence Umaps, « Communication de la recherche et de l'innovation », http://www.umaps.fr.

Bibliothèques pour le calcul scientifique

« outils, enjeux et écosystème »

Aristote est une société savante créée dès 1988 par L'Inria, le CEA, EDF et le CNES. Elle fonctionne par mélange des genres, croisant les regards pour apporter des visions nouvelles sur les derniers développements et nouveaux usages des technologies de l'information. Avec son comité de pilotage, Aristote a organisé de nombreux séminaires comme celui sur le Big Data, celui sur le thème de la « Sécurité et Mobilité », ou encore ceux sur le « Bâtiment intelligent » ou le Green IT. Elle contribue ainsi à tisser des liens entre le monde académique et celui de l'industrie. Le sujet d'aujourd'hui porte sur les bibliothèques numériques qui sont des composants majeurs des grands logiciels de simulation.

Présentation : Maison de la simulation

Michel Kern (Inria, Maison de la simulation)

D'après le rapport PITAC 2005, la simulation numérique est devenue le troisième pilier pour produire de la « bonne science », après la théorie et l'expérience. Cette importance croissante, Michel Kern et la Maison de la Simulation la considèrent comme un défi. L'informatique scientifique doit être capable de traiter de vastes volumes de données en travaillant pour exploiter les nouvelles architectures parallèles, dotées de GPU et d'autres technologies émergentes, pour résoudre d'énormes systèmes d'algèbres linéaires. Il est nécessaire d'être créatif techniquement et se forcer à être pluridisciplinaire.

Les moyens de calculs seront décisifs. Il y a désormais deux machines européennes dans le « top 10 mondial », mais seuls certains de nos chercheurs savent s'en servir. La Maison de la Simulation à la spécificité d'être un organisme national les aidant à utiliser au mieux ces nouvelles architectures. Elle est issue d'un projet commun CEA, CNRS, Inria, UPS, UVSQ et dispose d'une unité de service et de recherche située sur le plateau de Saclay. Ses trois axes de développement sont les laboratoires de recherche pluridisciplinaires, les unités de service d'aide au développement et une partie animation et formations en sciences, notamment *via* des cours dans deux masters. Il s'agit de cristalliser en un lieu les transferts de compétences entre chercheurs permanents ou invités et ingénieurs, pour concrétiser leurs visions.

La réactivité est de mise pour les supercalculateurs : il suffit de 6 ans pour que la dernière machine du « top 500 » rattrape la plus puissante. C'est le seul délai disponible pour s'adapter et innover, alors que la durée de vie de certains codes de simulation est de plus de 20 ans. Sans parler des progrès algorithmiques, qui ne sont pas moins importants que les progrès hardware. Ils sont responsables d'au moins la moitié de l'accroissement des performances. Il s'agit aussi de développer de nouveaux codes adaptés aux architectures disponibles dans des centres nationaux comme l'IDRIS. La maison de la Simulation dispense une formation initiale, mais aussi une formation continue à l'utilisation des méthodes de calcul ; son équipe est constituée d'une trentaine de personnes et traite des sujets qui vont de la climatologie, aux mathématiques appliquées en passant par la fusion.

Fast and reliable linear system solutions on new parallel architectures

Marc Baboulin (Université Paris-Sud/Inria)

Vers l'an 2000, les constructeurs ont arrêté la course au gigahertz. Trop d'énergie était dissipée sous forme de chaleur. À la place, la bataille se joue à présent sur l'installation des GPU et des processeurs multicœur. Mais pour Marc Baboulin, chercheur à l'Université Paris-Sud et à l'Inria, ces progrès génèrent une augmentation de l'hétérogénéité de l'architecture à l'intérieur du HPC et les rendent plus difficiles à utiliser.

Un des axes majeurs à améliorer est le coût des communications. Les performances des échanges entre les processeurs n'ont pas suivi l'envolée de la puissance : sur des machines modernes, des centaines de calculs peuvent être effectués le temps d'envoyer un seul message.

Le but aujourd'hui est d'accélérer les simulations numériques tout en gardant la précision des résultats. Pour cela, il faut trouver un moyen de tirer avantage de l'hétérogénéité. C'est-à-dire non seulement des multicœurs, mais aussi des GPUs, dont l'apport aux HPC augmente constamment et accélère tous les 6 mois. De composants spécialisés, ils sont désormais devenus très utiles pour les calculs en tous genres. Les progrès se feront à l'avenir en minimisant les communications et les transferts de tâche, ce qui passe par une meilleure répartition des taches entre CPU et GPU, planifiée pour que chacun fasse ce pour quoi il est le plus performant.

C'est dans ce contexte que s'inscrit MAGMA (*Matrix Algebra on GPU and Multi-core Architecture*), une bibliothèque numérique qui vise à conserver l'interface habituelle de LAPACK, mais en fournissant des algorithmes efficaces sur les nouvelles architectures. Le projet rencontre un vrai succès avec 15 000 téléchargements depuis le début et 8000 accès au site par mois sur l'année 2013.

Le principe de MAGMA va être de mettre la matrice sur le GPU, pour réduire le goulot d'étranglement des communications GPU/CPU. C'est un parallélisme niveau BLAS, mais en remplacant les appels aux BLAS par des appels à CUBLAS tout en gardant une « conception LAPACK ». On transfère au CPU les tâches qui n'exploitent pas vraiment le GPU, utilisant ainsi les deux pour les tâches auxquelles ils excellent. L'asynchronisme CPU-GPU est alors une nécessité.

Si l'on prend le cas classique de la factorisation, qui sert de benchmark pour le « top 500 », avec MAGMA la matrice va dans le GPU pour sa partie la plus coûteuse, mais chaque bloc est ensuite envoyé au CPU, au fur et à mesure, pour qu'il continue la factorisation.

Comme le fait remarquer une personne du public, le plafonnement de la performance lorsque l'on augmente le nombre de *threads*, fait qu'il vaudrait mieux démarrer sans pivotage puis changer de méthode. C'est une résolution à la demande, qui préfère des solveurs rapides plutôt qu'ultra-fiables.

Le problème réside dans la communication : celle-ci prend plus d'un tiers du temps. Plusieurs approches ont été tentées, notamment en créant une méthode sans aucun pivot, *via* de la *randomisation*. Le *tournament pivoting* était déjà plus rapide pour de grandes matrices. En mêlant les méthodes, on gagne vraiment en performance. On fait également des systèmes en précision mixte : certaines étapes sont en précision simple, d'autres en double, plus coûteuse. Mais le tout s'approche des performances de la précision simple pour un coût final bien plus faible.

Ces nouvelles méthodes permettent au HPC de traiter de plus grandes tailles de problème sans augmenter la partie matérielle. En termes de performance, le solveur *randomisé* multiplie par 2,5 la performance par rapport à MKL (*Math Kernel Library*). Cela va aussi vite que la méthode de référence, Cholesky, ce qui était l'objectif fixé.

Le paysage changeant des architectures oblige à de nouvelles méthodes, mais il est également plus difficile proposer une solution unique. Ainsi la *randomisation* est très prometteuse, mais il y a beaucoup de travail théorique à faire. La Maison de la Simulation y travaille notamment avec des statisticiens. Développer une bibliothèque pour le calcul scientifique devient de plus en plus technique. À cause de cela, les informaticiens sont plus impliqués et les numériciens sont un peu mis de côté, d'où une rigueur mathématique parfois mise en danger.

How to Avoid Communication in Linear Algebra and Beyond

Laura Grigori (Inria)

L'équipe de Laura Grigori, à l'Inria, travaille également sur la réduction des communications, mais leur approche se base depuis 2008 sur une nouvelle classe d'algorithmes d'algèbre linéaire.

Leur motivation est connue : non seulement le volume des données généré par les simulations croit de façon exponentielle, mais si cela ne suffisait pas, le temps de calcul de chaque donnée augmente encore plus. En astrophysique, l'équipe de Laura Grigori travaille par exemple sur les images de l'univers jeune. En 89, KOBÉ produisait 10 gigabytes et avait besoin d'une puissance d'un téraflops pour calculer une image. Aujourd'hui PLANCK produit 1 térabyte de données et a besoin de 100 pétaflops pour calculer une image.

Ces besoins croissants sont soutenus par l'augmentation de la puissance des machines, du moins en termes d'opérations flottantes. Mais l'efficacité des algorithmes va au contraire plutôt en diminuant. La multiplication des cœurs et des mémoires entraîne trop de communications et dégrade la performance de manière drastique. Il y a un vrai problème de latence, car on ne peut pas envoyer des données plus vite que la vitesse de la lumière. Il faut donc prendre le problème plus en haut dans le computing stack, ce qui demande de changer le paradigme de nos algorithmes : ne plus se contenter de réduire les opérations flottantes, mais élaguer aussi les communications.

Ces algorithmes de *communication avoiding* utilisent des progrès théoriques pour minimiser le volume des communications (pour la bande passante) et le nombre de messages (pour la latence). Laura Grigori précise que leurs modèles prennent également en compte la distance entre les processeurs, élément qui peut impacter la durée des communications d'un facteur 1 à 5. Cela augmente encore la complexité de l'optimisation. Mais les calculs supplémentaires sont de toute façon largement amortis par les gains de temps.

Les bases de ces algorithmes sont la nécessité de résoudre des systèmes linéaires ou de calculer les valeurs et vecteurs propres. D'un côté les méthodes directes de calculs sont stables et précises, mais utilisées seulement pour les petites matrices, donc pas telles quelles dans les simulations. Aujourd'hui on se tourne plutôt vers des solutions itératives, qui raffinent une solution initiale. Pour les problèmes de moindres carrées, où l'on veut minimiser une norme, la méthode directe passe par une factorisation encore plus chère, même si elle est plus stable. On rencontre ensuite le problème de pivotage évoqué par Marc Baboulin. Et le problème est le même pour d'autres calculs, par exemple sur des harmoniques sphériques.

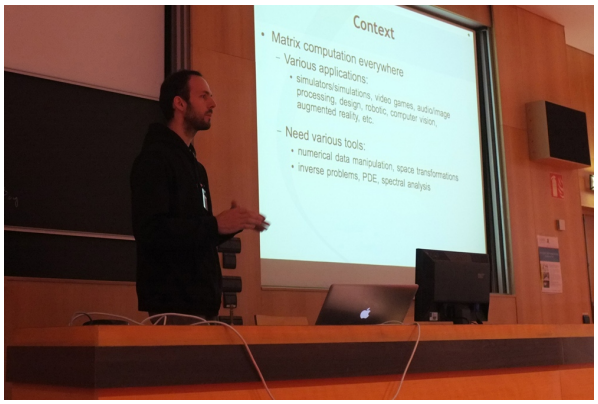
Ces problèmes, connus depuis au moins 95 avec les papiers sur le *memory walk*, ont été jusque là résolus par différentes approches, par exemple dans ScaLAPACK. Du tuning tout d'abord, qui ne fait que cacher les communications par les calculs. Le *ghosting* qui, de son côté, créé de fortes redondances de données et n'est pas adapté en algèbre linéaire, où elles sont dépendantes. Le *scheduling* quant à lui change l'ordre des opérations pour cacher les communications. Cela n'apporte qu'une accélération insuffisante, à peine d'un facteur 2. De Plus, pour leurs problèmes d'astrophysique l'équipe de Laura Grigori a bien constaté qu'avec le parallélisme les communications augmentent et il n'y a plus assez de calculs pour les cacher.

Des résultats de recherche démontrent, pour tout l'algèbre linéaire dense, le nombre minimum de communications entre deux mémoires (l'une rapide et l'autre lente) et permettent ainsi d'avoir des bornes minimales en *bandwith* et *latency*. La borne minimale sur le nombre de messages est de l'ordre de racine carrée du nombre de processeurs, et non pas de la taille de la matrice. Des progrès sont donc possibles en terme de nombres de messages. Ils changent donc les algorithmes pour chaque opération, tout en les gardant, en théorie, stables. Pour éviter les communications, il faut utiliser ces algorithmes progressifs qui seraient intégrés dans `ScaLAPACK` ou `LAPACK`. Par exemple, un scheduling statique est déjà efficace, mais les cœurs ne sont souvent pas utilisés. Mais si la planification est totalement dynamique, c'est encore plus lent. La méthode développée par Laura Grigori est donc un hybride qui n'a qu'une quantité optimale de planification dynamique, à peine 10%. Cela suffit à combler les temps libres par du dynamique et à passer les performances au-dessus de `MKL`.

À chaque fois il faut s'adapter à l'architecture machine pour optimiser les communications. Pour les gros problèmes, si l'on utilise beaucoup de cœurs, réduire les communications augmente beaucoup les performances.

`Eigen` : a C++ template library for linear algebra and related numerical algorithms

Gaël Guennebaud (`Inria`)



Les matrices sont partout : dans les jeux vidéos, les simulations, l'audio, la vidéo... Le besoin en outils est fort et varié. On vise de grosses performances, parfois même sur des machines légères

Des solutions comme `MatLab` sont limitées, car si elles sont *user-friendly*, elles ne font que des mathématiques pures et pas de logiciels. Sans compter qu'elles sont vraiment lentes pour gérer de petits objets. Cela reste donc limité au prototypage. Les bibliothèques d'`HPC` sont bien plus optimisées, mais spécialisées. Une application demande d'en faire cohabiter un certain nombre. Leurs performances sont difficiles d'accès et surtout destinées aux problèmes de grande taille.

Entre ces deux approches, Gaël Guennebaud, de l'`Inria`, a voulu construire un intermédiaire. Il se nomme `Eigen` et exploite les avantages des deux approches pour en créer une troisième, complémentaire, utilisable sous `MatLab` ou dans le contexte `HPC`. Cet outil est construit en pur `C++` moderne, c'est-à-dire en *template*. Rien à configurer, pas de binaires à compiler. Et il est en `OpenSource`. De plus en plus de caractéristiques viennent s'y ajouter, comme la décomposition matricielle, un module de géométrie, des matrices creuses et des solveurs. Tout cela unifié dans une seule `API` (*all in one*).

Son originalité est d'être optimisé pour les petits objets (à taille fixe ou *malloc free*) comme pour les gros. Ses performances sont ainsi comparables à celles des meilleurs du marché. `Eigen` propose des types de scalaires rationnels, complexes, des intervalles, des scalaires *AutoDiff* ou encore du symbolique avec des gradients, des jacobiens, *etc.*

Le plus intéressant dans `Eigen` est pour les matrices pleines : le code permet d'inscrire directement des expressions, grâce aux *templates*. Par exemple pour faire une addition de trois nombres, au lieu de

passer par des variables temporaires, `Eigen` va utiliser des *templates* pour construire une expression littérale du calcul et l'évaluer grâce à une simple boucle finie prenant deux fois moins d'accès mémoire que la méthode classique. Regrouper ces opérations réduit les variables temporaires et en cumulant de telles améliorations, `Eigen` offre une `API` de haut niveau comme `MatLab`, mais avec de véritables performances. Cela facilite aussi le déroulage des boucles et la vectorisation.

On fait de l'analyse d'expressions *top-down* pour les mettre dans des formes canoniques optimisables. Si cela n'est pas possible, on utilise alors des *templates*. Plusieurs itérations forment un *tree-optimizer* qui a pour mission de prendre une expression brute et de rendre une expression efficace. Il remet les parenthèses de façon optimale et d'autres modifications pour optimiser les accès à la mémoire. Par exemple, un produit comme `Matrice x Matrice x Vecteur` devient `Matrice x (Matrice x Vecteur)`, ce qui est bien moins coûteux que de commencer par multiplier les deux matrices.

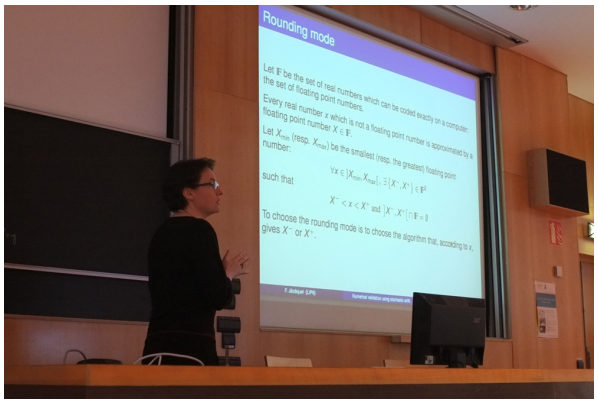
Dans les premières versions, le soutien de `KDE` a donné à `Eigen` beaucoup de visibilité sous `Linux`. Il constitue un vrai succès, notamment apprécié dans la communication, vision artificielle et le graphisme. Mais encore aujourd'hui, il est surtout développé à l'`Inria`, par Gaël Guennebaud et un ingénieur à plein temps. Le projet n'a pas immédiatement un modèle économique, il s'agit d'un projet créé sur son temps libre par Gaël Guennebaud, qu'il essaye de financer pour avoir quelqu'un d'autre à plein temps. Mais un des intérêts d'une bibliothèque scientifique est que ses utilisateurs sont aussi ses développeurs.

La question des licences est toujours importante. Dans la mesure où il s'agit d'une bibliothèque de bas niveau, il faut que cela puisse être réutilisé dans des logiciels propriétaires, mais que les modifications puissent rester libres. La licence `GNU LGPL` (*Lesser General Public Licence*) est venue naturellement... mais est devenue un boulet qu'`Eigen` a trainé pendant quatre ans. Elle n'est pas aussi *open* qu'elle le laisse paraître et a pour énorme problème que les industriels... n'en veulent pas. Mais depuis 2012 il existe aussi la `MPL2` (*Mozilla Public Licence 2*) qui, elle, n'a que des avantages du point de vue de Gaël Guennebaud : acceptée des industriels, elle est générique, simple et avec une bonne réputation. Ce fut un très bon choix.

Le développement se poursuit avec des ajouts à venir `AVX`, `CUDA`, plus de parallélisme, du traitement de matrices creuses ou encore de l'optimisation non linéaire.

Estimation d'erreur d'arrondi par la bibliothèque `CADNA`

Fabienne Jézéquel (`LIP6-UPMC`)



Le credo de Fabienne Jézéquel et de son laboratoire, le `LIP6-UPMC`, est l'amélioration dans les codes scientifiques de l'estimation et du contrôle de l'erreur. Leur bibliothèque se nomme `CADNA`. Pour rappel, la norme `IEEE 754` définit quatre méthodes d'arrondi d'un nombre réel pour obtenir un nombre flottant sur un ordinateur. Chaque nombre réel `X` est encadré par deux flottants `X-` ou `X+`, qui constituent des arrondis possibles, et l'on peut également arrondir au flottant le plus proche entre `X` et `0` ou au flottant le plus proche de `X`. Ce dernier mode d'arrondi est celui par défaut. Une opération d'arrondi est effectuée après chaque opération.

Mais sur l'ordinateur, les opérateurs arithmétiques sont en réalité des approximations des véritables opérateurs mathématiques. Trop souvent, elles sont utilisées en oubliant qu'elles ne possèdent ni la pro-

priété d'associativité ni celle de distributivité. Par exemple, ce n'est pas parce qu'un flottant est inférieur à un autre que les valeurs mathématiques associées ont la même relation.

Les algorithmes fonctionnels sur le papier amènent alors à des erreurs lors des calculs sur ordinateur. Le modèle établi par le LIP6-UPMC donne pour commencer un théorème qui établit le nombre de bits significatifs entre le résultat numérique et le résultat mathématique réel. On y voit que la perte de précision, dans un calcul, ne dépend pas du nombre de bit, du format utilisé. Ainsi un même calcul effectué en précision simple ou double perdra le même nombre de chiffres significatifs, par exemple 10. Mais la précision simple n'ayant que 8 chiffres significatifs à la base, il n'en restera plus aucun.

Pour estimer l'ampleur de l'erreur dans nos calculs, Fabienne Jézéquel décrit plusieurs méthodes. La première est l'analyse d'inverse et fournit des bornes d'erreurs pour un coût faible. Elle peut être utilisée entre autres pour l'arithmétique linéaire. La deuxième méthode est l'arithmétique linéaire, qui fournit des bornes sûres à 100%. revanche, elle surévalue l'erreur, jusqu'à la considérer infinie si elle n'est pas adaptée à chaque calcul.

La troisième méthode est l'arrondi aléatoire, qui évalue le nombre de chiffres corrects sur le résultat. C'est celle développée pour CADNA. Elle est fondée sur la méthode CESTAC et consiste à exécuter le même calcul plusieurs fois avec différentes propagations d'erreurs d'arrondis. Dans les différents résultats finaux obtenus, les chiffres qui diffèrent sont considérés comme non significatifs. Concrètement, à chaque opération il s'agit de prendre aléatoirement soit l'arrondi supérieur du nombre obtenu, soit son arrondi inférieur. À la fin, on obtient un échantillon de résultats, avec une distribution gaussienne sur laquelle on effectue des tests statistiques pour avoir une estimation du nombre de chiffres significatifs.

CESTAC est basée sur un modèle du premier ordre : pour éviter que des termes du second ordre ne deviennent prépondérants et que le calcul n'ait plus aucun chiffre significatif, un contrôle dynamique est nécessaire. C'est ce que permet cette méthode. De plus, elle ne requiert que 2 ou 3 exécutions, car la performance croît en fonction de racine carrée du nombre de tests ; plus d'itérations seraient donc du gaspillage.

Pour les calculs itératifs, le défi classique est de choisir quand s'arrêter. Notamment, le choix optimal serait de pouvoir s'arrêter quand la variabilité du résultat ne se situe plus dans les chiffres significatifs. C'est ce que permet CADNA. La bibliothèque redéfinit par exemple un « zéro informatique » et l'utilise à la place du zéro arithmétique. On récupère ainsi certaines propriétés comme la transitivité. Il devient ainsi possible de reconstruire différents opérateurs élémentaires.

CADNA est une bibliothèque pour Fortran ou C++. En pratique, elle fournit de nouveaux types numériques stochastiques, comme single-st ou double-st et surcharge toutes les opérations élémentaires. Le coût des trois exécutions nécessaires est d'environ 10 fois le coût initial, voire plus pour des détections fines. L'équipe de Fabienne Jézéquel travaille à gérer en parallèle ces trois exécutions pour les multi-cœurs. Cela fonctionne déjà avec MPI, avec CUDA ou encore Open MP (en cours d'implémentation).

L'utilisation de CADNA consiste à la déclarer, à l'initialiser, à substituer les types stochastiques aux types numériques correspondants, puis à changer les données si besoin et à changer les formats pour n'afficher que les chiffres corrects. Enfin, on appelle une fonction pour avoir la liste des instabilités détectées. Les bons résultats ne sont souvent pas ceux que l'on croit, et il faut exécuter CADNA pour voir que des résultats qui « tombaient pile » n'avaient en fait aucun chiffre significatif ! Par exemple, la liste des instabilités peut montrer qu'il y avait de fortes pertes de précision (*cancellations*), ainsi certaines opérations font en fait perdre 4 chiffres significatifs d'un seul coup...

Alors, quand arrêter les itérations ? Avec CADNA, le critère devient simple : quand deux itérés successifs sont égaux au sens de CADNA. Dans d'autres algorithmes, il suffit de s'arrêter quand l'objectif est atteint numériquement au sens de CADNA, alors qu'auparavant il fallait attendre d'atteindre des précisions

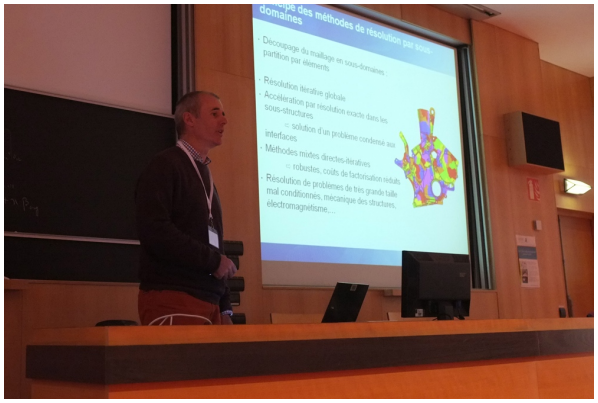
de l'ordre de 10^{-15}. Pour de meilleures approximations, `CADNA` permet de surveiller les erreurs à mesure que l'on diminue le pas, pour garder l'erreur de méthode inférieure à l'erreur de calcul.

C'est avantageux à utiliser « dans la vraie vie », car cela donne des possibilités de surveillance des codes de simulations de grande taille. Il devient possible de contrôler les instabilités.

L'inconvénient est le coût temps et mémoire, ainsi que la difficulté d'utiliser cette bibliothèque avec d'autres bibliothèques. Même si l'adaptation n'est pas longue, c'est toujours cela de plus à faire.

Mise en œuvre de méthodes de résolution par sous-domaines parallèles dans des codes d'éléments finis

François-Xavier Roux (`ONERA`)



Le problème majeur que François-Xavier Roux, de l'`ONERA`, souhaite soulever est celui des systèmes implicites de très grande taille et mal conditionnés, comme ceux rencontrés en électromagnétisme. Sa méthode, très adaptée au calcul parallèle, est la décomposition en sous-domaines grâce aux éléments finis.

Cela consiste à décomposer le domaine de calcul en sous-domaines, où l'on peut assembler des matrices locales, connues et donc toujours solubles. Seules les interfaces sont à ajuster pour assurer la cohérence. En effet, les solutions locales doivent être des traces de la solution globale. La question est donc comment les raccorder au reste du problème. En représentant cette condition-là, on formule les contributions de chaque sous-domaine. La méthode la plus ancienne pour y parvenir est celle du complément de Schur. On peut ainsi calculer le résidu à l'interface et trouver les valeurs à imposer pour fixer l'équilibre.

Un des intérêts est qu'il n'y a pas de points interfaces multiples. Lambda est la condition entre les différents domaines, et si vous faites de la mécanique des structures, lambda est la force d'interaction entre les domaines. Un point multiple appartient simplement à trois interfaces, et chacune apporte une contrainte, une force qui est indépendante. En pratique, les applications sont très liées à l'électromagnétisme et à l'élasticité. Cela permet d'ailleurs souvent de poser des conditions physiques qui aident beaucoup à la résolution.

La solution des problèmes locaux n'est connue qu'à une constante près. Mais ce problème devient un avantage, car il permet de s'accorder aux interfaces. Par exemple, cette constante que l'on ne connaît pas peut être le mouvement de corps rigides de chaque sous-domaine, et on la calcule simplement pour assurer le minimum de déplacement à l'interface. Le problème global est donc de très petite taille. À chaque itération, on a juste à résoudre un problème très grossier. Il faut seulement assurer une bonne transmission d'infos à chaque interaction. La résolution se fait naturellement avec le solveur et simplement trouver les constantes assure la cohérence. En fait, on fait du multi-grille sans le savoir, comme M. Jourdain faisait de la prose. Chaque sous-domaine est lié à un processus qui ne calcule que sa partie de la base de données.

La théorie de ces méthodes semble un peu compliquée, mais d'un point de vue mise en œuvre elles sont assez faciles. Dans chaque domaine, les seuls prérequis sont la matrice de rigidité, les conditions aux

limites locales issues des conditions aux limites globales, les contraintes et les interfaces. Et cela suffit. Pour le faire sur un objet existant, il faut un maillage. Le découpage peut être réalisé « à la main » ou par un partitionneur de maillage. D'après François-Xavier Roux, l'outil plus connu pour cela est METIS, mais le meilleur reste vraiment Scotch, qui sera abordé plus tard dans le séminaire. En revanche, la méthode ne permet pas de partir de la matrice globale, car on ne sait pas retrouver les matrices locales.

La méthode de sous-domaines peut être très simple si l'on travaille au niveau élémentaire. Et justement, d'après François-Xavier Roux, la physique ne va pas au-delà du niveau des éléments. L'interprétation mécanique est très utile ici pour déterminer les conditions.

Pour ce qui est du passage à l'échelle, (*scalability*), il y a de plus en plus de cœurs sur les machines, et donc localement il est intéressant d'avoir des solveurs directs.

François-Xavier Roux préfère utiliser des solveurs conçus par d'autres, car il vaut mieux être spécialiste de ce que l'on fait ! La meilleure façon de faire un code qui vieillit bien reste de ne pas l'écrire, et il faut donc utiliser au maximum les bibliothèques préexistantes et ouvertes.

Legolas++ Conception d'outils génériques pour les problèmes linéaires creux structurés par blocs multi-niveaux

Laurent Plagne (EDF)

À l'EDF, Laurent Plagne travaille sur un type de problème très particulier, les matrices creuses structurées par blocs multi-niveaux. Ces objets exotiques sont générés par des problèmes de neutroniques et consistent en des matrices très creuses, dont on peut exprimer le pattern en peu d'éléments. Legolas++ est un outil pour les décrire.

De façon générale, l'intérêt d'une bibliothèque est de séparer les disciplines. Mais Laurent Plagne, paradoxalement, travaillait non pas avec des équipes pluridisciplinaires, mais avec des chercheurs pluridisciplinaires. Ceux-ci devaient alors tout apprendre de tous les domaines !

Les bibliothèques existantes n'étaient pas adaptées pour manipuler des objets aussi spécifiques et ils ont pensé que le C++ serait le langage idéal. Le monde du HPC requiert une haute performance et l'outil en lui-même ne doit donc pas générer de pertes. C'est actuellement le cas de Legolas++.

Son principe est de reconstruire des patterns compliqués à partir de petits patterns de base. Il permet même de reconstruire des matrices non stockées. Les opérations pour cela sont d'une part la composition verticale qui accole des matrices, de l'autre la composition horizontale qui les additionne. Ces deux opérations permettent de reconstruire des matrices assez complexes à partir d'un nombre fini de blocs. Un solveur parallèle permet alors de tout résoudre.

La performance obtenue est comparable « à la concurrence ». En utilisant les bons outils, Legolas++ obtient par exemple, sur une machine multi-cœur à mémoire partagée, un *speed-up* de 10,3.

Laurent Plagne fait ensuite un rappel sur la manipulation des unités SMID, unités multi-vectorielles dans les multi-cœurs et dans GPU. Leur utilisation efficace a des conséquences importantes sur la conception des codes. En particulier, il va falloir adapter le placement des données selon l'architecture. Une

unité SMID est comme un baby-foot : comme la ligne de joueurs qui fait effectuer le même mouvement à plusieurs figurines, le SMID est capable d'effectuer en même temps des instructions identiques sur des données différentes. Il est facile de bouger la poignée pour faire tourner les joueurs, mais il est difficile d'exploiter les 5 joueurs en mettant 5 balles à leurs pieds. Il s'agit d'un type d'algorithme contraint.

Ainsi par exemple, le parallélisme SMID est difficile à trouver si le problème comprend des matrices tridiagonales. À deux cœurs c'est simple : chacun s'occupe d'un morceau du problème. Mais le SMID a besoin de régularité au plus bas niveau, par exemple que les flottants à traiter soient rangés les uns à côté des autres. Il faut donc réorganiser la diagonale, et mettre ainsi les joueurs en face des balles. C'est particulièrement facile si l'on utilise un *wrapper* vers le logiciel Eigen, car les codes sont alors les mêmes et les performances sont alors vraiment très bonnes. En combinant la transformation de structure et le parallélisme SMID *via* Eigen, on obtient 2,6 gigaflops et un *speed-up* de 4,3 (les 4 opérations simultanées et tous les contrôles de boucles divisés par 4). Si en plus on utilise toute la machine, donc du TBB sur les gros blocs, le *speed-up* obtenu est 22,1. Le SMID est donc difficile à éviter, mais demande pourtant de grosses modifications du code. L'analogie avec le baby-foot continue lorsque l'on fait remarquer la tendance du jeu à croître ces derniers temps.

Cette méthode développée dans Legolas++ était le cadre de la thèse de Wilfried Kirschenmann en 2012, qui visait à aller vers des noyaux de calcul intensif pérennes.

Laurent Plagne souligne combien il a été utile que le C++ soit multi-paradigme car le SMID est vraiment un problème et pousse à des solutions peu élégantes. Malheureusement cela a tendance à devenir de plus en plus important, et il espère que l'avenir s'en détachera.

L'intérêt général est de construire des bibliothèques sémantiquement les plus homogènes possible et qui s'appuient sur d'autres bibliothèques. Pour Legolas++, les perspectives de Laurent Plagne sont d'intégrer davantage du travail de Wilfried Kirschenmann, mais il s'intéresse aussi à une thèse de Benoit Lizet, dont les travaux chez EADS sont prometteurs.

OpenPALM, OpenSource code coupler for massively parallel multi-physics/ multi-components applications and dynamic algorithms

Florent Duchaine (Cerfacs)

OpenPALM est un coupleur co-developpé par le Cerfacs et l'ONERA depuis 2011, dont le développement a été très fructueux. Florent Duchaine va parler de ce qu'il peut apporter, par exemple en mécanique des fluides, pour réaliser des couplages multi-physiques, mais aussi pour de l'algorithmique dynamique.

OpenPALM n'est pas un nom très bien choisi. En effet, il cache plusieurs bibliothèques et surtout il ne met en avant que le côté OpenSource et l'origine Cerfacs, mais il oublie l'importance de la partie ONERA, appelée CWIPI. Beaucoup de personnes ont travaillé sur ce produit, permanents ou stagiaires.

Le principe est d'utiliser du couplage de code pour traiter un système dans sa globalité. Il y a d'un côté les aspects couplages physiques, et de l'autre les aspects informatiques avec des échanges de données

qui sont le cœur du travail du Cerfacs. Florent Duchaine montre l'exemple d'un étage de turbine avec une partie fixe et une partie mobile, traitée chacune par une instance du même code qui sont couplées ensemble. D'autres applications sont le réchauffement climatique ou le couplage thermique, pour le spatial entre autres.

En plus de gérer le parallélisme et de prendre garde à être le moins intrusif possible dans les codes, il est important de prêter attention à l'évolution des modèles couplés. Pour qui est-il fait, a-t-il une durée de vie courte — étude ponctuelle — ou bien va-t-il servir, par exemple, à plusieurs étudiants de thèse successifs ?

Pour la conception des couplages, on ne repart en général pas de zéro. Mais les gens choisissent souvent de fusionner des codes existants. Bien que cette approche puisse être très efficace si cela est bien fait, ce n'est pas évolutif ou flexible. Florent Duchaine préfère alors parler d'assemblage. Il est ensuite possible de faire de l'échange de fichier et divers raffinements, mais au-delà de deux modèles couplés ou pour l'interpolation, c'est une épine dans le pied.

Un vrai coupleur pour le Cerfacs est en fait une sur-bibliothèque encapsulant tout ce qui va être MPI et communication, qui va être caché à l'utilisateur au travers de primitives simples et d'une interface graphique. Transférer des informations entre de nombreux codes et gérer les séquences d'exécution doit devenir le plus simple possible.

Oasis et OpenPALM sont ainsi des coupleurs développés par le Cerfacs. Leur philosophie est de garder des performances élevées pour ces instruments, avec des communications courtes, mais aussi d'écrire du code le plus portable possible, donc en C, C++, Fortran et MPI, et enfin le plus flexible possible, pour pouvoir changer des sous-parties pour faire autre chose au besoin.

Comment fonctionne OpenPALM ? Au contraire des couplages statiques qui démarrent séparément et fusionnent à la fin, le couplage dynamique d'OpenPALM repose sur des codes qui ne sont lancés que sous conditions. Nul ne sait donc avant l'exécution ce qui va être utilisé dans le code.

PrePALM est une interface de *monitoring* et de *post-processing* pour OpenPALM. PALM s'occupe de l'ordonnancement (*scheduling*), avec les appels de primitives. Là-dessus vient se greffer CWIPI, qui s'occupe du maillage. PALM date de 1996 : il s'agissait d'une demande d'Météo France pour faciliter la mise en œuvre et les tests d'algorithmes d'assimilation de données.

Leur interface graphique est entièrement dynamique, chaque bouton est une unité dans PALM et peut être un programme parallèle. L'utilisateur dessine alors des branches dans l'interface et cela seul créée déjà du parallélisme de tâches. C'est aussi *via* l'interface graphique que les communications vont être adressées, sachant qu'il y a un code couleur et que l'interface va protester si le tout n'est pas cohérent. Dans le reste, il s'agit de faire du remapping d'objets sur les différents processeurs. Le code observe quelles sont les parties qui attendent, quelles sont celles qui bloquent les autres, et facilite ainsi le prototypage d'applications complexe de couplage de code. Le but est de favoriser des applications massivement parallèles, tandis que l'interface de PrePALM confère un gain de généricité.

OpenPALM est par exemple utilisé en océanographie. La librairie PALM est utilisée depuis des années par Mercator (sur les machines de calcul de Météo France) pour prévoir l'état de l'océan toutes les semaines. Il peut aussi prédire la propagation des fronts de feu de forêt, avec PALM, où prévoir un été dans 100 ans, ce qui est un projet RTRA/STAE ACCLIMAT. Et bien d'autres, dans l'aéronautique notamment. Pour l'anecdote, OpenPALM a déjà tourné sur 12000 cœurs pour résoudre un problème de *conjugate heat transfer*.

Le produit est ouvert et Florent Duchaine encourage tout le monde à aller sur le site internet du Cerfacs et à le télécharger.

How to age well a 20 y.o. Scotch

François Pellegrini (Université Bordeaux 1/LaBRI/Inria)

Comment faire un développement long terme pour une bibliothèque scientifique, comment parvenir à ce que le code ne tourne pas à la piquette ? Après 20 ans de travail sur Scotch, François Pellegrini est bien placé pour répondre à cette question.

À propos de ce que fait Scotch, il rappelle qu'il est basé sur les graphes, qui sont des outils primordiaux pour l'analyse. Constitués simplement de points reliés entre eux par des arrêtes, ils sont utiles pour faire énormément de choses, par exemple de l'analyse du réseau routier ou de la répartition de communications entre serveurs. Beaucoup de problèmes NP-difficiles consistent à trouver la coupe minimale pour un *flow* maximal, et les algorithmes de Scotch le permettent.

Les deux problèmes principaux étaient la renumérotation de matrices creuses et la décomposition de domaines, pour les méthodes itératives comme a décrit François-Xavier Roux. Cela donne deux objectifs, de faire du partitionnement-arête et du partitionnement-sommet.

Mais au-delà, et au contraire de tous ses concurrents, Scotch gère depuis le début un sur-problème nommé le placement statique. Pour la répartition des sommets sur le graphe, couper simplement revient à oublier que les machines ne sont pas équidistantes. Il faut minimiser non seulement la coupe, mais aussi l'étirement des arêtes, et ainsi éviter la congestion du réseau. Cette pondération était moins importante dans les années 90, car les architectes avaient réalisé des machines presque UMA (*Uniform Memory Access*) avec 128 ou 256 nœuds. Mais aujourd'hui, parler au cœur voisin où au cœur dans l'armoire à l'autre bout de la machine n'est plus du tout du même ordre de grandeur en terme de coût.

Le deuxième objectif de Scotch était de gérer des graphes de plus d'un milliard de sommets distribués sur un millier de processeurs. C'est aujourd'hui fait à Livermore. Cela craque un peu aux membrures, mais cela tient. Il s'agit à présent de surenchérir : le but actuel est de 10^{15}, soit un billion de sommets, distribués sur un million de processeurs. Cela pose la question de la gestion des architectures massivement non uniformes, à la fois comme cibles, mais aussi en redistribution à la volée, donc de la hiérarchie sur la machine même où s'exécute Scotch. Il faut s'orienter vers des algorithmes asynchrones pour relâcher les contraintes. C'est d'ailleurs cette partie théorique qui pousse François Pellegrini à dire que 20 ans après, il continue à faire de la science et pas de l'ingénierie. Aujourd'hui, un certain nombre de leurs algorithmes ont déjà été reformulés pour le contexte *multithread*.

Dans le fichier d'en-tête de Scotch, on voit que la première ligne de code date de décembre 1992. Cela « sèche » un peu les étudiants de François Pellegrini, quand ils voient que le code est plus vieux qu'eux ! Mais au début, le logiciel était hautement privatif (« propriétaire »). On a commencé à montrer le code source en 1999, et à partir de la version 4.0, en 2006 on a pu la sortir sous licence LGPL. La 5.0 quant à elle est passée sous CeCILL-C1.

Des logiciels et des chercheurs

Maintenant se pose la question des logiciels et de la science. Le logiciel est une fin en soi, par exemple un code fonctionnel est une preuve mathématique de faisabilité. Mais c'est aussi un moyen. Ils

permettent aussi de « calculer des démonstrations », comme celle du théorème des 4 couleurs. Mais s'il s'agit de science, il est très problématique que le code source ne soit pas joint avec les publications dans les revues. Les listes de résultats et de performances peuvent être écrites à la main, les montrer est juste un argument d'autorité. Il est impossible de refaire l'expérience ou... les 20 années de développement, pour vérifier l'affirmation ! Cela viole donc le principe de réfutabilité de Popper, sortant du même coup du domaine de la science. Il faut donc mettre en place des démarches dans ce sens, avec des licences certes restrictives, mais en respectant la démarche scientifique.

Que faire quand on a produit le logiciel ? Il peut arriver que le logiciel ne soit plus utile à la recherche, mais qu'une communauté d'utilisateur s'en serve tout de même. Comment le maintenir pour des autres utilisateurs ? Car sinon, c'est tout le temps et l'argent investi qui a été gaspillé. Et un thésard, cela coûte cher ! avec le temps que prend la conception d'un logiciel...

Les licences libres permettent aux gens intéressés, si l'auteur originel ne travaille plus son logiciel, de pérenniser l'investissement existant. Sinon la maintenance applicative empêche le chercheur de faire son travail, et de valoriser sa recherche.

De plus, comment et sous quelle forme valoriser la masse de travail que représente l'intégration d'un code dans une solution logicielle ? Cela prend souvent la forme d'un contrat avec l'industrie, mais celle-ci revend alors un logiciel à un public qui l'a pourtant déjà financé *via* la recherche... Certains instituts font plus attention et mettent un ingénieur sur le développement logiciel : c'est leur métier.

À propos des stagiaires, François Pellegrini fait remarquer que puisqu'ils sont « sans contrats », ils ne cèdent pas leurs droits d'auteurs ! Il est donc illégal de revendre un code comprenant du travail effectué par un stagiaire. Il faut vraiment prendre garde à garder une trace de toutes les contributions, pour pouvoir ensuite gérer les droits. D'ailleurs attention, il n'est pas toujours évident pour le chercheur de donner son code à son nouvel employeur : il faut l'accord de tous, par exemple l' `Inria` et les autres partenaires du développement.

La question suivante est bien sûr celle du choix de la licence. Pour libérer `Scotch` de sa licence `LGPL`, François Pellegrini a dû négocier avec son directeur. Lorsqu'il a finalement obtenu le feu vert, il a sauvegardé précautionneusement le *mail*, car négocier auprès de tous les ayants-droits peut vraiment être difficile. Mais être un logiciel libre est un avantage compétitif. Tout d'abord, la communauté crée alors spontanément des adaptations du code. `Scotch` a tout à coup été repris sous `Debian`, `Ubuntu`, (paquetages *aptget*) sans que jamais il ne rencontre les auteurs. Deuxième avantage, cela lui a permis des contacts avec les industriels qui font du libre et qui ne veulent pas utiliser du privatif.

De son côté, François Pellegrini a même du mal à évaluer l'étendue de son périmètre d'utilisateur tant sa communauté vit indépendamment de lui... sauf quand il y a une bogue, dans quel cas tout tout le monde se réveille et il voit clairement qui utilise son travail. Récemment, il y a même un de ses compétiteurs qui veut venir tester ses codes sous `Scotch`, ce qu'il prend comme un aboutissement de 20 ans de travail.

Au niveau de la licence, il y a une bonne raison de dire non à la `CeCILL-C` : elle est incompatible avec la `GPL`. Les gens doivent créer des exceptions à leur licence pour se connecter avec la `CeCILL-C`. Mais la `LGPL` permet un *fork* juridique vers la `GPL` : du coup une modification faite sous `GPL` ne peut pas être réintégrée sous `LGPL`. Or les modèles économiques de ces licences sont différents. François Pellegrini a donc préféré se placer en `CeCILL-C`.

Le point majeur reste que la définition de la politique de licence doit se faire dès le début du projet. Il faut gérer les aspects légaux en séparant les parts publiques et privées, donc en traçant toutes les contributions. Sans quoi plus personne ne contrôle le devenir du code.

En conclusion, François Pellegrini conseille d'être paranoïaque avec la qualité. Trop planifier de l'informatique n'est pas un problème. Le vrai problème, c'est quand le code qui a été bâclé sur un coin

de table est toujours dans la chaîne de production 20 ans après. Il faut se dire que l'on n'écrit jamais du code jetable. Pour cela, il faut faire respecter les conventions de nommage, bien rédiger les docs et ne pas laisser le style du rédacteur transparaître.

Une excellente technique est celle du sous-marin. Le code doit être compartimenté pour garder la bogue dans les sections dont elle provient, comme l'eau est confinée dans le compartiment du sous-marin d'où provient la fuite. C'est de cette manière que l'on peut savoir efficacement d'où vient la bogue, notamment quand c'est le graphe d'entrée fourni par l'utilisateur qui est incorrect. En définissant et en se tenant à une routine de vérification axiomatique pour tout le code, on peut vraiment avancer. Et il faut l'exposer à tous les utilisateurs pour qu'ils puissent vérifier.

`FreeFem++`, un logiciel pour résoudre numériquement des équations aux dérivées partielles

Frédéric Hecht (`UPMC`)

`FreeFem` est un logiciel libre qui permet de résoudre les équations aux dérivées partielles, créé entre autres par Frédéric Hecht. Son nom signifie qu'il est gratuit et basé sur une *Finite Element Method*. Au contraire de `Legolas++` évoqué plus tôt dans la journée, le « ++ » ne fait pas référence au C++, mais au fait qu'il s'agit de la troisième version de `FreeFem`.

L'année dernière, l'étudiant Pierre Joliet a résolu un système avec un milliard d'inconnues, avec 6000 processeurs. Aujourd'hui, `FreeFem` est à dix milliards d'inconnues, et passe à l'échelle sur 12000 processeurs. Mais il arrive aux limites de ses possibilités dans ce domaine (*scalability*).

`FreeFem` fonctionne par méthode d'éléments finis `R2` et `R3`, avec un langage dédié. Comme François Pellegrini, il pense que programmer un logiciel n'est définitivement pas le moyen pour avoir un poste. Mais cela peut tout de même aider à se vendre ! `FreeFem` n'a été pensé que pour la recherche de Frédéric Hecht, mais il est utilisé pourtant par d'autres. Il fonctionne sous `Windows`, `Mac` et `Unix`. Un seul paradigme de parallélisation est implanté, `MPI`, car c'est un petit logiciel. Le code ne gère que les triangles et les simplexes, surtout parce que Frédéric Hecht, de son propre aveu, est « paresseux et veut une structure simple ». Il gère des valeurs réelles ou complexes, récupére toutes les matrices et fait... tout ce que l'on peut attendre d'un logiciel d'éléments finis. Son avantage par rapport à d'autres est l'interpolation automatique des données sur un maillage et sur un autre.

Tout a commencé en 1987, avant même `Scotch`, avec `MacFem/PCFem` écrit par Olivier Pironneau et payant. Donnant un cours de compilation, Olivier Pironneau a vu le moyen d'écrire un langage spécifique pour les éléments finis. En 1992, réécriture de `FreeFem` en C++ par Olivier Pironneau et D. Bernardi, F. Hecht et C. Prudhomme. En 1996, une réécriture a fait passer à plusieurs maillages et en 1998, on passe à `FreeFem++`, nouveau noyau éléments finis et nouveau langage utilisateur. En 1999, il y a eu du travail sur `FreeFem 3d` par S. Del Pino, mais qui est aujourd'hui au point mort.

En 2008, une `ANR` a permis `FreeFem++` (`v3`) et une écriture d'un noyau éléments finis pour prendre en compte les cas multidimensionnels : `1d`, `2d`, `3d`, ... Mais pour des raisons techniques, la `2d` fait toujours

appel à l'ancien noyau. Un jour `Freefem v4` verra le jour et ce vieux noyau disparaitra... mais ce jour-là, il y aura des bogues.

À tous ses étudiants, Frédéric Hecht donne le même conseil : quand on code, il faut utiliser au maximum les assertions. Cela permet de traquer les bogues, et elles peuvent être retirées pour la version finale du logiciel. Chercher une bogue est difficile, c'est la vie du programmeur !

`FreeFem` permet aujourd'hui du prototypage d'algorithme, des expériences numériques, pour de la programmation scientifique en parallèle. Il a une *mailing-list* de 400 membres qui résolvent leurs problèmes sans intervention systématique de Frédéric Hecht et le logiciel est enseigné dans de nombreuses écoles. Mais ne voulant pas s'occuper des aspects juridiques, Frédéric Hecht a choisi la solution de facilité : « celui qui compile est responsable ! »

Des démonstrations de `FreeFem++` montrent que, *via* entre autres l'équation de Laplace et une forme faible de l'équation de Poisson, il permet de mailler la peau d'un modèle de poisson pris sur `Internet` et d'y faire de l'élasticité linéaire. Il s'agit de passer des déformations aux contraintes. Il est possible de faire du Stokes, car Frédéric Hecht y a passé beaucoup de temps dans sa carrière. Dans `FreeFem`, il ne prend les conditions limites que sur des bords, les conditions ponctuelles sont impossibles, car cela générait trop de bogues et ne servirait à rien. Frédéric Hecht continue sa démonstration de `FreeFem` en faisant calculer en temps réel à son `MacBook` un écoulement `2D` issu d'un cylindre.

Initialement destiné aux étudiants, `FreeFem` est rapidement devenu utile pour la recherche, où il ne faut pas gaspiller du temps à recoder sans cesse les problèmes ! `FreeFem` peut utiliser le parallélisme jusqu'à un million de processeurs, et c'est un outil bien plus rapide que `Matlab`.

Les bibliothèques scientifiques massivement parallèles : utilisation dans les grands codes de simulation pour l'énergie nucléaire

Christophe Calvin (`CEA/DEN/DANS/DM2S`)

Contrairement aux autres intervenants, Christophe Calvin n'est pas venu à ce séminaire pour présenter une bibliothèque, mais plutôt pour parler de comment le `CEA` utilise différentes bibliothèques, et pourquoi dans certains cas il n'en utilise pas.

Le `CEA` est le deuxième exploitant nucléaire en France, avec ses réacteurs expérimentaux, propulsions navales et d'autres systèmes nucléaires électrogènes. Les échelles de temps sont donc longues et les codes associés doivent avoir une vraie longévité. Ces cycles de vie de 20 ou 30 ans impliquent de toujours penser à l'architecture logicielle. Et les activités obligent à couvrir un spectre très large de sciences.

Une des plates-formes majeures est `SALOME`, un coupleur d'applications co-développé avec `EDF`.

Parmi les applications utilisant des bibliothèques, il y a notamment `EUROPLEXUS`, code de dynamique rapide ; ses grandes caractéristiques : Intégration temporelle explicite, grands déplacements et rotations, interaction fluide, etc. Les aspects parallèles ont été traités souvent dans le cadre de projets `ANR`, mais la parallélisation a été faite sur tout le code de A jusqu'à Z. C'est un code industriel et il était donc hors de question de créer plusieurs versions successives.

via l'ANR, les premières parallélisations de RepDyn étaient pures MPI. Mais on voyait que certains cas de simulation se passaient mal, notamment à cause des modifications des maillages et de modifications des sous-domaines. Un rapprochement s'est alors effectué avec une équipe de l'Inria de Grenoble, pour commencer à mixer de l'échange explicite de message avec du *multi-treading*. L'idée était aussi de faire du parallélisme dynamique au niveau du multi-treading, ce qui a été fait en utilisant la bibliothèque KAAPI. Cela fonctionne bien, même si la gestion des changements rapides des propriétés des matériaux, dans les réacteurs, sont perfectibles.

Christophe Calvin résume donc qu'il s'agit d'un parallélisme à mémoire distribuée dynamique, un solveur distribué pour le calcul des forces de liaison globales, le calcul avancé à mémoire partagé, le tout concrétisant une approche hybride à grande échelle.

Un exemple plus concret de code utilisant un outil externe est l'application de CFD/DNS nommée Trio_u, architecturée autour de modules. Ses utilisations se trouvent par exemple dans le code MC2, pour en particulier la simulation d'écoulement dans les cœurs de réacteur. Mais Trio_u fait aussi de la simulation de turbulence, une CFD plus classique. MPCUBE de son côté utilise Trio_u pour du stockage en milieu poreux. Ce sont des systèmes linéaires creux qu'il faut résoudre. On fait un conditionnement des matériaux, qui dépend de la physique, de la discrétisation et des éléments numériques.

Le CEA écrivait son propre solveur, mais ils se sont rendu compte que leurs problèmes étaient trop divers. Ils avaient fait des essais de multi-grille, mais en fin de compte pourquoi tout faire soi-même alors que d'autres le font mieux ? Ils ont simplement créé une interface pour accéder à un outil externe, en l'occurrence PETSc.

L'avantage des grosses simulations est que l'on peut prendre le temps de faire des essais.

APOLLO3 est de son côté un code collaboratif avec Areva et EDF. Il s'agit d'un code de dégénération pour du transport neutronique, avec une approche probabiliste comme Tripoli, par la méthode de Monte-Carlo qui est la méthode de référence pour la physique des réacteurs. Elle est gourmande, il faut tout modéliser depuis le cœur jusqu'aux enceintes de confinement, en passant par les particules émises. Le CEA s'est basé sur la librairie externe ROOT pour créer son outil.

Des applications dans la mécanique existent aussi, avec le code CAST3M, en développement depuis 1983. La principale version est à mémoire partagée, sans vraiment de parallélisme. Christophe Calvin souligne qu'il vaut mieux parfois délaisser le parallélisme et faire des efforts sur l'analyse numérique, ce qui permet de réduire la consommation mémoire et d'obtenir des accélérations. Mais ce code CAST3M n'utilise aucune bibliothèque, ce qui reste une question ouverte à l'heure actuelle. N'y aurait-il pas intérêt à en tester une, ne serait-ce que pour montrer quelles approches sont moins performantes sans avoir à toutes les développer ?

Récemment un excellent papier de Benovo propose des méthodes itératives très intéressantes. Il ne faut pas exclure les nouvelles possibilités.

Les codes du CEA ont la grosse contrainte d'être non dédiés à un projet particulier et de devoir être utilisés tant en recherche et développement que dans le secteur industriel.

Dernière catégorie de code au CEA, les applications OpenSource utilisées à l'extérieur. Tout le code CEA n'est pas libre, mais il y a notamment SALOME, URANIE (dédié au traitement des incertitudes) ou Snorky 3d (Pour des écoulements complexes 3D instationnaire, visualisés en temps réel.)

En conclusion, les situations au CEA sont assez contrastées : des applications utilisent des bibliothèques externes avec succès, et des bibliothèques qui ne sont en général pas uniquement des bibliothèques numériques parallèles. Mais d'autres applications restent totalement en interne. Mais la DEN est aussi fournisseur d'outils pour la communauté.

Les raisons principales qui justifient de ne pas utiliser d'outils externes sont les contraintes de l'adaptation de l'outil, le manque de maitrise du code pour les techniciens en interne, notamment sur le long terme. Et enfin, il y a une très (trop) grande optimisation de certaines parties des applications développées en interne, qui rend délicate voire quasi impossible, l'utilisation de bibliothèques externes. Pourtant, l'utilisation de bibliothèques externes va devenir de plus en plus nécessaire, car les architectures des machines ne vont pas en se simplifiant et il est impossible de pouvoir tirer parti de toutes les machines. Les nouveaux algorithmes devront s'adapter aux nouveaux matériels, de façon dynamique.

Mais qui est l'éditeur de ces bibliothèques ?

Patrick Moreau (Inria)

Ces bibliothèques Open Source qu'utilise la recherche, finalement qui les édite ? Patrick Moreau n'est pas un scientifique, mais à l'Inria il a constaté que vient un moment dans la vie des logiciels où « la corde se tend » pour les logiciels issus de la recherche. À mesure qu'ils sont de plus en plus utilisés par des industriels, des enseignants et des utilisateurs externes qui ont des exigences de pérennité, de fonctionnalité et d'absence de bogues, il devient nécessaire d'allier la recherche et la satisfaction des utilisateurs. C'est une équation paradoxale qui oblige les chercheurs à travailler le week-end et sur leur temps libre... Rajouter des ingénieurs, cela ne tient qu'un temps. Plus la diffusion du logiciel prend de l'ampleur, plus cela devient intenable.

Patrick Moreau pense lui aussi qu'il faut choisir une licence très tôt pour son code OpenSource. Toutefois il faut tout de même attendre d'en savoir assez sur le projet pour pouvoir bien choisir sa licence !

Plusieurs phases dites OSRL (*OpenSource Readiness Levels*) peuvent ensuite être distinguées. En phase OSRL 1, seuls les collègues du créateur utilisent le logiciel. En ORSL 2, une communauté d'utilisateurs externes se forme. La phase ORSL 3 est optionnelle, il s'agit de la possibilité que ces mêmes utilisateurs contribuent au logiciel, même si ce n'est que par des remarques. En phase OSRL 4, les industriels se joignent au cortège. L'impact augmente alors et « la corde se tend ». Les industriels ne peuvent pas prendre le risque que le chercheur ne corrige pas les bogues ! L'open-source devient la condition d'existence sur le marché. La phase OSRL 4 est censée aboutir au transfert technologique, où l'industrie entretient le code et le pérennise. Mais pour qu'une entreprise puisse ainsi exploiter un logiciel, il est indispensable qu'elle ait suffisamment de business *via* ce code précis. Car s'il s'agit juste d'un ajout à son activité, pourra-t-elle vraiment y investir et le garder viable ?

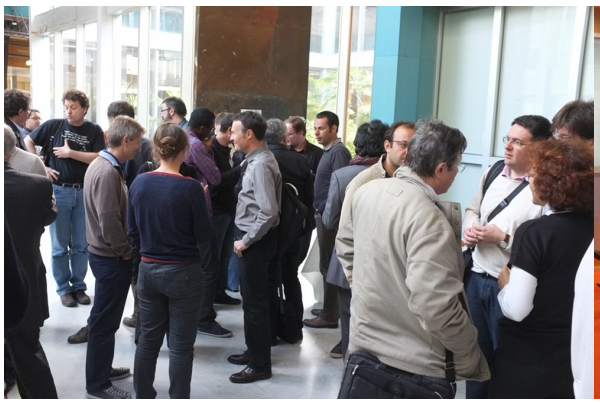
Les cycles sont longs dans ce domaine, il peut se passer 10 à 15 ans entre le dépôt en OpenSource et le transfert vers l'industrie. La phase de transition OSRL 4 est problématique. Pour bien la gérer, une idée intéressante est le consortium. Un consortium est un partenariat ou une association, mais c'est surtout des gens qui se mettent ensemble temporairement pour faire quelque ; viser un but précis. C'est un moyen pertinent pour permettre la maturation d'un logiciel. Par exemple, l'Inria a créé le Scilab, CAML ou encore MONOLIX. Le dernier né est PHARO, où les partenaires ont prit le temps de créer un consortium plus « professionnalisé ».

Pourquoi décider de faire un consortium ? L'`Inria` investit si le jeu en vaut la chandelle, s'il y a création d'emploi et transfert, ce qui est sa mission. De façon générale, il faut se demander s'il y a un potentiel d'utilisateurs suffisant et si cela peut être financé. C'est-à-dire y-a-t-il une part assez importante de l'activité de l'utilisateur qui peut être payable. Il faut également qu'il y ait déjà des utilisateurs actuels, qu'il reste des domaines d'utilisations à explorer, et enfin que les attentes autour du logiciel soient modérées.

Enfin le logiciel doit avoir une généricité, pour permettre à des membres qui sont pourtant concurrents de s'associer pour le réaliser. Patrick Moreau fait l'analogie avec l'achat d'une voiture : il y a 30 ans, les gens ouvraient le capot pour regarder le moteur. Aujourd'hui, ils regardent la consommation. Les constructeurs automobiles se mettent donc ensemble pour développer de meilleurs moteurs, qui ne sont plus des arguments en soi.

Parfois il faut remettre le projet en question, un consortium n'est pas ou plus adapté. Remettre les accords sur la table tous les six mois est une bonne idée. Dès le démarrage du consortium, il faut avoir des idées, des conditions de sortie. Y a-t-il des possibilités de création de *start-up*, le développement pourra-t-il ensuite être repris par un des acteurs ?

Enfin, Patrick Moreau aborde la question de quel consortium construire ? Il ne doit pas forcément porter sur tout le logiciel, il est possible de n'y mettre que la partie générique. Pour quelle offre les membres payeront-ils ? Il faut écouter les clients. En ce qui concerne le montage du consortium, il faut faire simple avec une association de 1901, et juste un contrat d'adhésion. Sans oublier de protéger les marques et logos ! Cela fait également partie de la valeur du consortium.



Chapitre 2

Présentations

2.1 Philippe d'Anfray (CEA, Aristote)

Présentation Aristote

Nouvelles technologies de l'information et de la communication

Association Aristote

Technologies des réseaux et NTICs

Bibliothèques pour le calcul scientifique outils, enjeux et écosystème

École Polytechnique, 15 mai 2013

Qui ? Créée "informellement" en 1984 par l'INRIA, le CEA, EDF et le CNES, **Aristote** est depuis 1988 une association loi 1901 (34 adhérents en 2013).

Quoi ? **Aristote** est une société savante qui regroupe des organismes et des entreprises impliqués dans les derniers développements et les nouveaux usages des technologies de l'information.

Pourquoi ? **Aristote** contribue à tisser des liens entre le monde académique et celui de l'industrie et des services à travers ses activités de transfert de technologie et de veille scientifique. L'idée est de croiser regards et cultures, recherche fondamentale et retours d'expérience pour apporter des éclairages nouveaux aux problématiques abordées.

Aristote propose des groupes de travail qui sont des lieux d'échanges privilégiés autour des NTICs.

- **PIN** pérennisation des informations numériques ;
- **Gus'G** Grilles et HPC ;
- **Calcul Hybride** ;
- **e-Laboratoire** travail collaboratif, arts numériques.
- Cloud ; Big Data.

Aristote organise ponctuellement des sessions de formations (PIN, Cloud-HPC, *etc.*)

Aristote organise chaque année, à l'École Polytechnique, un cycle de séminaires.

- À la poursuite des Big Data (mars 2013)
- Sécurité et Mobilité (février 2013)
- CFD Workflow Meshing, Solving, Visualizing -ONERA- (octobre 2012)
- Le bâtiment intelligent ; source de valeurs (septembre 2012)
- Green IT & Cloud (juin 2012)
- Logiciel Libre et communautés la clef du transfert ? -OWF-Plume-Systématic- (mai 2012)

Aristote

Comment ? Un **comité de programme** se réunit chaque mois pour organiser les séminaires, discuter de nouveaux thèmes ou groupes de travail, *etc.*

La participation est ouverte à tous (inscription en ligne).

aristote

15 mai 2013 5 / 9

Les bibliothèques numériques

... sont des composants majeurs des logiciels de simulation et des portes d'accès au HPC.

- quels sont les éléments "moteurs" de ces projets ; les cibles ; les évolutions en cours ou prévues ;
- comment inscrire ce type de projet dans la durée ;
- y a-t-il une spécificité et une reconnaissance en tant que telle de cette activité ;
- comment diffuser (Open Source, ...), la protection intellectuelle, quelles licences ;
- comment constituer et gérer une communauté d'utilisateurs.

Ces bibliothèques permettent la "cristallisation" de projets scientifiques et le maintien d'un savoir faire partagé. Un point-clef pour l'interdisciplinarité.

Programme 15 mai 2013, matin

9h00-9h30	**M. Tétard** (Aristote) : accueil	
9h30-9h45	**Ph. d'Anfray** (Aristote) : Présentation Aristote	
9h45-10h00	**Michel Kern** (Maison de la simulation) : Présentation Maison de la simulation	
	Session 1 (*Chair* : M. Kern (Maison de la simulation)	
10h00-10h30	**Marc Baboulin** (Université Paris-Sud/Inria)	Fast and reliable linear system solutions on new parallel architectures
10h30-11h00	**Laura Grigori** (Inria)	How to Avoid Communication in Linear Algebra and Beyond
11h00-11h30	*Pause café*	
	Session 1 (*Cont.*)	
11h30-12h00	**Gaël Guennebaud** (Inria)	Eigen : a C++ template library for linear algebra and related numerical algorithms
12h00-12h30	**Fabienne Jézéquel** (LIP6-UPMC)	Estimation d'erreur d'arrondi par la bibliothèque CADNA
12h30-13h30	*Déjeuner*	

Programme 15 mai 2013, après-midi

	Session 2 (*Chair* : Ph. d'Anfray (CEA)	
13h30-14h00	**François-Xavier Roux** (ONERA)	Mise en œuvre de méthodes de résolution par sous-domaines parallèles dans des codes d'éléments finis
14h00-14h30	**Laurent Plagne** (EDF)	Conception d'outils génériques pour les problèmes linéaires creux structurés par blocs multi-niveaux
14h30-15h00	**Florent Duchaine** (Cerfacs)	OpenPALM, an open source code coupler for massively parallel multi-physics/multi-components applications and dynamic algorithms
15h00-15h30	**François Pellegrini** (Université Bordeaux 1/LaBRI/Inria)	How to age well a 20 y.o. Scotch
15h30-16h00	*Coffee break*	
	Session 3 (*Chair* : Thiên-Hiệp Lê (ONERA)	
16h00-16h30	**Frédéric Hecht** (UPMC)	FreeFem++, un logiciel pour résoudre numériquement des équations aux dérivées partielles
16h30-17h00	**Christophe Calvin** (CEA/DEN/DANS/DM2S)	Les bibliothèques scientifiques massivement parallèles : utilisation dans les grands codes de simulation pour l'énergie nucléaire
17h00-17h30	**Patrick Moreau** (Inria)	Mais qui est l'éditeur de ces bibliothèques ?
17h30	**Clôture**	

Les bibliothèques numériques

2.2 Michel Kern (Inria, Maison de la SImulation)

Présentation Maison de la SImulation

MAISON DE LA SIMULATION

11/05/2013

Les grands enjeux du HPC

La simulation est devenue le *troisième pilier* de la science, au même titre que la théorie et l'expérience

> Computational science has become the third pillar of the scientific enterprise, a peer alongside theory and physical experiment.
> Rapport PITAC 2005

- Les grands défis à venir pour le HPC sont partagés par une grande partie de la communauté scientifique :
  - Visualisation et exploitation de grands volume de données
  - Parallélisme massif, technologies émergentes (GPU, multi-cœurs,…)
  - Algèbre linéaire
  - Tolérances aux pannes
  - …
- Être créatif sur les aspects techniques et maintenir le cap sur la science
- Pouvoir réagir et s'adapter rapidement
- Nécessité de construire des équipes pluri-disciplinaires et de développer les partenariats
- Besoin de mutualisation des ressources humaines :
  - économie d'échelle
  - Mutualisation et diffusion des compétences.

Contexte et Evolution des Moyens de calcul

Echelle Nationale :

- **Création du GENCI**
  - → Financement pérenne des centres nationaux

Echelle Européenne : le Projet **PRACE**

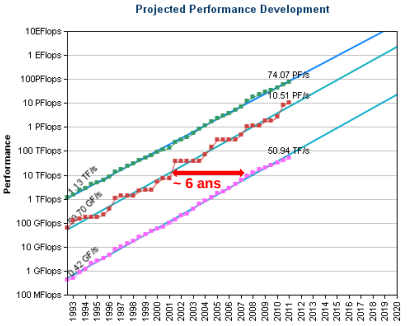
Partnership for Advanced Computing in Europe

→ Développement d'une infrastructure de Européenne de classe mondiale pour le calcul haute performance

2011 : Installation de la machine Curie au TGCC

La Loi de Moore

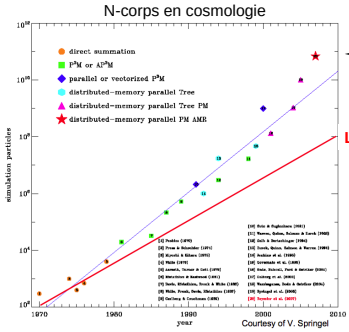
Utiliser un supercalculateur impose une très grande réactivité et une forte capacité d'innovation, notamment avec les évolutions techniques prévisibles.



La Loi de Moore pour les Applications

Les progrès « *algorithmiques* » contribuent très significativement à l'accroissement de performance des applications.

N-corps en cosmologie



550 Milliard de particules sur Curie

Loi de Moore

Maison de la Simulation

❑ Création en Janvier 2011 :

- Projet commun CEA/CNRS/INRIA/UPS/UVSQ
- La MdS est une Unité de Service et de Recherche
- Mise en place d'un comité de pilotage et d'un conseil scientifique
- Localisé sur le plateau de Saclay (Bât. Digitéo / CEA)
- Affectation de personnel : 35% CEA, 35% CNRS, 15% INRIA, 7.5% UPS et 7.5% UVSQ

Accompagner, soutenir et stimuler les communautés scientifiques afin de tirer le meilleur parti des moyens de calcul disponibles

➢ Laboratoire de recherche pluridisciplinaire autour de la simulation numérique

➢ Unité de service ouverte sur les communautés offrant notamment une expertise et une aide aux développements applicatifs de haut niveau

➢ Un pôle d'enseignement et d'animation scientifique en calcul intensif

Laboratoire

Centre d'excellence multidisciplinaire dans le domaine de la simulation

Création au sein d'un même laboratoire d'équipes pluridisciplinaires à même de relever les grands défis du calcul intensif

Chercheurs permanents : thématiques *transverses* (mathématiques appliquées, informatique, algorithmique,..) ainsi que dans quelques domaines où le calcul représente un enjeu majeur.

- un lien étroit avec leurs communautés d'origine
- mise en place de projets pluridisciplinaires
- transfert de compétences

Chercheurs accueillis sur projets

Ingénieurs : conception, réalisation et diffusion d'outils numériques

Contribuer au développement et à l'exploitation de codes numériques novateurs ayant vocation à être utilisés sur les équipements nationaux (GENCI) et européens (PRACE)

Service et soutien à la communauté

Fournir une expertise et une aide aux développements applicatifs de haut niveau pour l'utilisation des grands moyens de calculs, notamment ceux mis à disposition dans le cadre de GENCI et du projet PRACE

- **Accueillir des chercheurs et/ou des équipes et leur fournir le support matériel et humain pour :**
  - le développement et l'optimisation de codes applicatifs
  - la préparation, la réalisation et l'exploitation de simulations de type « grand challenge »
- **Soutenir les équipes de recherches impliquées dans la simulation notamment par l'attribution de bourses doctorales et postdoctorales**
- **Contribuer à l'animation scientifique autour du HPC.**
- **Collaboration étroite avec GENCI et les centres de calculs nationaux**
- **Volonté d'Inria de créer un réseau national**

Formation et animation scientifique

Formation initiale :

Contribuer, en collaboration avec les partenaires, à assurer une implication forte de la MdS dans les masters "*HPC*" du plateau de Saclay.

- M2S :
  - Etablissement partenaire
  - Mise en place est hébergement du site web
  - Accueil des certains cours et participations à 3 modules
- MIHPS
- Utilisation de la plateforme par deux autres Masters

Formation continue : **(6 semaines de formation, ~150 personnes formées)**

Former les chercheurs et ingénieurs à l'utilisation des grands moyens de calculs.

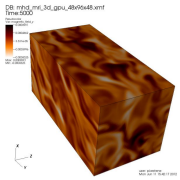
- Labélisation PRACE Advanced Training Center (PATC) :
  - Candidature française portée par la Maison de la Simulation et regroupant les trois centres de calcul nationaux et Inria
  - Dix formations/an
- Sessions de formation
- Organisation d'ateliers, de conférences et d'écoles. Soutien au CECAM Ile-de-France

Personnel

- Equipe de direction : E. Audit, A. Bui et M. Kern
- 5 permanents : 1 secrétaire, 4 ingénieurs/chercheurs
- 12 doctorants : Math. appli., Fusion, Géoscience, Astrophysique, Science de la vie,…
- 3 post-docs
- 3 ingénieurs HPC en CDD
- 1 CDD pour travailler dans PRACE
- 2 CDD en cours de recrutement pour l'ANR *ANEMOS*

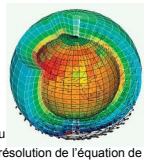
Activités scientifiques - I

La Maison de la Simulation est ouverte à toutes les communautés.



- **Fusion :**
  - 1 doctorat (+ 1 en octobre 2012), 1 post-doctorant
  - 1 ANR (Uni. Nice, IRFM, INRIA/Bordeaux, MdS)
    Instabilité MHD dans les Tokomaks
  - Mutualisation du cluster de calcul
- **Astrophysique :**
  - Développement d'un code sur GPU en lien avec une ERC (demande de temps de calcul sur Curie)
  - Un doctorat en co-tutelle
- **Nanoscience :**
  - un CDD ingénieur en collaboration avec un postdoctorant du programme transverse nanoscience.
  - optimisation du code DP du LSI → demande PRACE

Activités scientifiques - II



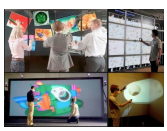
- **Climatologie :**
  - un CDD PRACE/CEA
  - Optimisation du coupleur OASIS dans le cadre de PRACE
  - Représentation de la communauté française dans PRACE
- **Mathématiques appliquées :**
  - Deux doctorats en collaboration avec les universités de Lille-I et de Pau
  - Collaboration en émergence avec Paris-sud sur l'algèbre linéaire et la résolution de l'équation de Poisson
- **Modélisation du vivant :**
  - Electrophysiologie avec EPI CARMEN d'Inria Bordeaux
  - Optimisation des solveurs linéaires, passage à l'échelle
- **Collaborations avec la Direction de l'Energie Nucléaire du CEA :**
  - collaboration et mutualisation d'expertise dans le domaine de la visualisation
  - deux thèses en partenariat DEN/Maison de la Simulation
  - organisation de séminaires

Deux équipements phares

Digiscope :

Mise en place d'une infrastructure haute performance pour la visualisation interactive et collaborative sur le plateau de Saclay

- Mur d'image stéréoscopique, cluster de visualisation
- Visualisation de données de simulation issues des grands centres de calcul.
- Partenariat avec plusieurs laboratoires du plateau spécialiste de visualisation et d'interaction



Equip@Meso

Renforcer la « base » de la pyramide est devenu nécessaire après le développementdes Tier-1 (GENCI) et Tier-0 (PRACE)
Installation en France de 10 méso-centres avec l'objectif affiché de développer l'usage du HPC et de pousser les utilisateurs vers les Tier-1 et les Tier-0.

- Projet coordonné par GENCI
- Coordination nationale des activités de formation
- Méso–équipement de calcul (mutualisé avec l'IRFM) :

Conclusions

- Outil scientifique : **Ordinateur + Applications**
- Développer en France une communauté des *Sciences pour et par la Simulation* (Computational sciences)
- La Maison de la Simulation est un lieu d'expertise et d'accueil que les chercheurs et les laboratoires doivent s'approprier

2.3 Marc Baboulin (Inria, Université Paris-Sud)

Fast and reliable linear system solutions on new parallel architectures

Fast and reliable linear system solutions on new parallel architectures

Marc Baboulin
Université Paris-Sud/Inria

Abstract

Recent years have seen an increase in peak ``local" speed through parallelism in terms of multicore processors and GPU accelerators. At the same time, the cost of communication between memory hierarchies and/or between processors have become a major bottleneck for most linear algebra algorithms.

In this presentation we explain how hybrid multicore+GPU systems can be used efficiently to enhance performance of linear algebra libraries. We illustrate this approach by considering hybrid factorizations where we split the computation over a multicore and a graphic processor and where the amount of communication is significantly reduced.

Next we describe a randomized algorithm that accelerates factorization of general or symmetric indefinite systems on multicore or hybrid multicore+GPU systems. Randomization prevents the communication overhead due to pivoting, is computationally inexpensive and requires very little storage. The resulting solvers outperform existing routines while providing us with a satisfying accuracy.

Fast and reliable linear system solutions on new parallel architectures

Marc Baboulin

Université Paris-Sud
Chaire Inria Saclay Île-de-France

Séminaire Aristote - Ecole Polytechnique

15 mai 2013

Motivations

Hardware trends in HPC

- Power issues and the move towards multicore
- Hybrid GPU-accelerated systems

Impact on existing software ?

- Increase of heterogeneity and data-communication costs
- Must rethink the design of numerical libraries

How to speed up numerical simulations ?
(while maintaining accuracy)

Outline

1. Taking advantage of parallel multicore-GPU architectures

Outline

1. Taking advantage of parallel multicore-GPU architectures
2. Accelerating linear system solutions with randomization

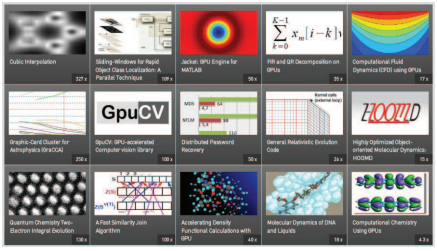
Outline

1. Taking advantage of parallel multicore-GPU architectures
2. Accelerating linear system solutions with randomization
3. Conclusion

Outline

1. Taking advantage of parallel multicore-GPU architectures
2. Accelerating linear system solutions with randomization
3. Conclusion

Why GPU-based computing



- Most HPC applications report high speedups with GPUs.
- Top 500, November 2012:
  62 systems with accelerators (vs 58 in June 2012 and 39 in Dec. 2011).
  #1 and #8 systems use NVIDIA GPUs.

Designing algorithms for multicore+GPU

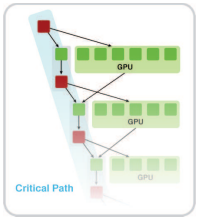
- Exploit strengths of each architectural component
- Minimize communication and data transfers
- Properly schedule the tasks execution over the CPU and the GPU
- **MAGMA**: Matrix Algebra on GPU and Multicore Architectures
  (U. Tennessee, U. California Berkeley, INRIA, U. Colorado...)
  LAPACK-style interface.

  [MB, Demmel, Dongarra, Tomov, Volkov, SC 2008]
  [MB, Dongarra, Tomov, PARA 2008]
  [Tomov, Dongarra, MB, J. PARCO 2010]
  [MB, Donfack, Dongarra, Grigori, Rémy, Tomov, ICCS 2012]
  [MB, Rémy, Sosonkina, Rozoy, PARCO 2013, submitted]

- 15,000 downloads, 8,000 hits per day in 2013
  Used by MathWorks, CRAY...

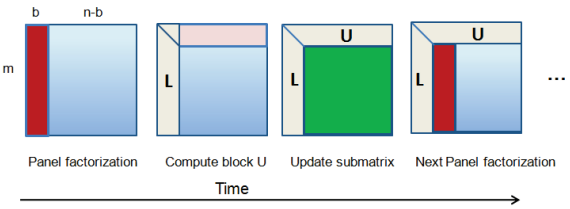
Principles of hybrid implementation

1. BLAS-level parallelism where the matrix resides on the GPU (BLAS calls replaced by CUBLAS)
2. Offload to the CPU small kernels that are inefficient for the GPU
3. Use asynchronism between CPU and GPU whenever possible



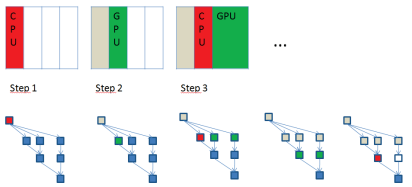
Example: LU factorization (general linear systems)

- Decompose an input matrix A into a product $L \times U$
- Block algorithm that iterates over blocks of columns (panels)
- At each iteration: factorize panel then update trailing submatrix



Hybrid version for LU factorization

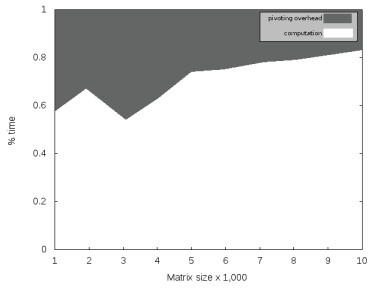
-Matrix transferred to the GPU
-Panel downloaded and factored by CPU using partial pivoting
-Updates performed by the GPU
-Look-ahead technique



Task splitting in hybrid LU factorization (4 panels)

More details in [Tomov, Dongarra, MB, PARCO 2010]

Communication overhead due to pivoting

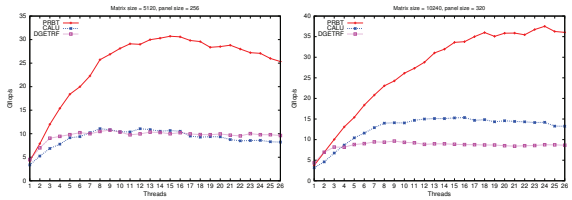


Cost of partial pivoting in LU factorization (MAGMA)
1 × Quad-Core Intel Core2 Q9300 @ 2.50 GHz - GPU C2050 @ 1.15 GHz

Other techniques

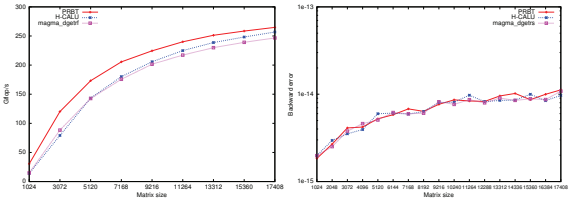
- Communication in pivoting can be reduced by using tournament pivoting [Grigori, Demmel, Xiang, SIMAX 2011]
  We developed a hybrid version → H-CALU solver
  [MB, Donfack, Dongarra, Grigori, Rémy, Tomov, ICCS 2012]
- We can remove completely the pivoting by preprocessing the system by randomization ($\mathcal{O}(n^2)$ flops) → PRBT solver
  [MB, Dongarra, Herrmann, Tomov, ACM TOMS 2013]

Performance for panel factorization



Comparison of CPU multi-threaded panel factorizations
(4× 12-Core AMD Opteron 6172 Magny-Cours @ 2.1 GHz)

Performance/accuracy of hybrid LU implementations



Performance results

Componentwise backward error
($\omega = \max_i \frac{|Ax-b|_i}{(|A|\cdot|x|+|b|)_i}$)

Experiments on AMD (16 threads) + NVIDIA Fermi Tesla S2050

Mixed precision algorithms

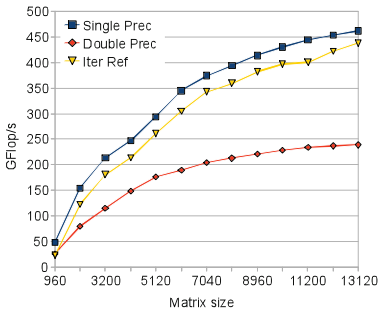
- Bulk of the computation in 32-bit arithmetic
- Postprocess the 32-bit solution by refining it into a solution that is 64-bit accurate
  Can be performed on the GPU
- Problem must be "not ill-conditioned"
- Software details in:
  M. Baboulin, A. Buttari, J. Dongarra, J. Kurzak, J. Langou, J. Langou, P. Luszczek, S. Tomov,
  Accelerating scientific computations with mixed precision algorithms.
  Computer Physics Communications , Vol. 180, No 12, pp. 2526-2533 (2009).
- Interest if: single precision is significantly faster than double precision and "cheap" iteration steps

Mixed precision algorithms

Example of the LU factorization

1:	$LU \leftarrow PA$	(ε_s)	$\mathcal{O}(n^3)$
2:	solve $Ly = Pb$	(ε_s)	$\mathcal{O}(n^2)$
3:	solve $Ux_0 = y$	(ε_s)	$\mathcal{O}(n^2)$
	do $k = 1, 2, \ldots$		
4:	$r_k \leftarrow b - Ax_{k-1}$	(ε_d)	
5:	solve $Ly = Pr_k$	(ε_s)	
6:	solve $Uz_k = y$	(ε_s)	
7:	$x_k \leftarrow x_{k-1} + z_k$	(ε_d)	
	stopping criterion		
	done		

Mixed precision



Performance for mixed precision LU-based solver on Fermi (C2050)

Outline

Randomization algorithms for HPC applications

- Randomized algorithms are gaining ground in HPC
- Can outperform deterministic methods while still providing accurate results
- Objective: addressing larger problems and/by performing less computation and/or communication
- Examples: random sampling for least squares, low rank matrix approximation...
- **In this talk**:
  RBT for dense linear systems → less communication

Application: symmetric indefinite linear systems

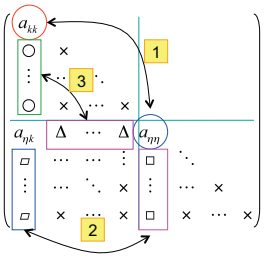
- Symmetric Indefinite (dense) linear system $Ax = b$
- Applications: least-squares via augmented system method, Maxwell equations in electromagnetics, optimization problems...
- Factorization $A = LDL^T$ and solve successively

$$Lz = b, \quad Dy = z, \quad L^T x = y$$

- Not stable – to ensure stability pivoting is usually required
- Requires $n^3/3$ flops (half the cost of LU)
- No **parallel** implementation for such systems in public domain libraries (MKL, very recently: Aasen LTL^T)

Symmetric pivoting

- To maintain symmetry, columns and rows must be interchanged
- Compromise data locality
- Increase data dependence



How to avoid pivoting

No pivoting by randomizing instead:

- For general systems (LU factorization):
  Initially proposed by [Parker, 1995]
  Revisited in [MB, Dongarra, Herrmann, Tomov, ACM TOMS 2013]
- Transform the original matrix into a matrix "sufficiently random" so that, with a probability close to 1, pivoting is not needed

How to avoid pivoting with symmetric randomization?

Symmetric Random Butterfly Transformation (SRBT)

$$Ax = b \equiv \underbrace{U^T A U}_{A_r} \underbrace{U^{-1} x}_{y} = \underbrace{U^T b}_{c}$$

1. Compute $A_r = U^T A U$ with U random (recursive butterfly) matrix
2. Factorize A_r without pivoting (LDL^T)
3. Solve $A_r y = U^T b$ then $x = Uy$

How to avoid pivoting with symmetric randomization?

Symmetric Random Butterfly Transformation (SRBT)

$$Ax = b \equiv \underbrace{U^T A U}_{A_r} \underbrace{U^{-1}x}_{y} = \underbrace{U^T b}_{c}$$

1. Compute $A_r = U^T A U$ with U random (recursive butterfly) matrix
2. Factorize A_r without pivoting (LDL^T)
3. Solve $A_r y = U^T b$ then $x = Uy$

Requirements :

- Randomization must be cheap
- LDL^T with no pivoting should strive for a "Cholesky speed"
- Accuracy must be similar to Bunch-Kaufman (LAPACK)

Random Butterfly Transformation

- **Butterfly matrix**:

$B = \frac{1}{\sqrt{2}} \begin{pmatrix} R & S \\ R & -S \end{pmatrix}$, with R and S random diagonal

Random Butterfly Transformation

- **Butterfly matrix**:

$B = \frac{1}{\sqrt{2}} \begin{pmatrix} R & S \\ R & -S \end{pmatrix}$, with R and S random diagonal

- **Recursive butterfly matrix** of depth d :

$$U = \underbrace{\begin{pmatrix} & & \\ & \ddots & \\ & & \end{pmatrix}}_{2^{d-1} \text{ butterflies of size } \frac{n}{2^{d-1}}} \times \ldots \times \underbrace{\begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}}_{2 \text{ butterflies of size } \frac{n}{2}} \times \underbrace{\begin{pmatrix} & & \\ & & \\ & & \end{pmatrix}}_{1 \text{ butterfly of size } n}$$

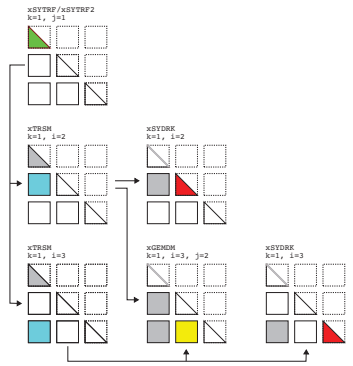
Applying randomization

- Tiled SRBT algorithm
- $A_r = U_1^T \times U_2^T \times \cdots \times (U_d^T \times A \times U_d) \times \cdots \times U_2 \times U_1$
  We compute recursively $A_r^{(i-1)} = U_i^T A^{(i)} U_i$.
- Tiled decomposition (d=2):

$$U_2^T A^{(2)} U_2 = \begin{bmatrix} B_1^T & \\ & B_2^T \end{bmatrix} \begin{bmatrix} A_{11} & A_{12} \\ A_{21} & A_{22} \end{bmatrix} \begin{bmatrix} B_1 & \\ & B_2 \end{bmatrix} = \begin{bmatrix} B_1^T A_{11} B_1 & B_1^T A_{12} B_2 \\ B_2^T A_{21} B_1 & B_2^T A_{22} B_2 \end{bmatrix}$$

  Elementary operation is $B_i^T A_{ij} B_j$

- $A_r = U^T A U$ requires $\approx 2dn^2$ flops



Tiled LDL^T Algorithm (3 tiles)

Numerical issues

Condition number ?

- Choosing the random values in $[e^{-1/20}, e^{1/20}]$, we get

$$cond_2(A_r) \leqslant 1.2214^d cond_2(A)$$

- In practice, $d = 2$: $cond_2(A_r) \leqslant 1.5\ cond_2(A)$

Stability of LDL^T?

- "Average" growth factor expressed in [Parker, 95]
- Iterative refinement is systematically added
- Backward error (available from IR process) is sent back
- Future work: probabilistic error bounds

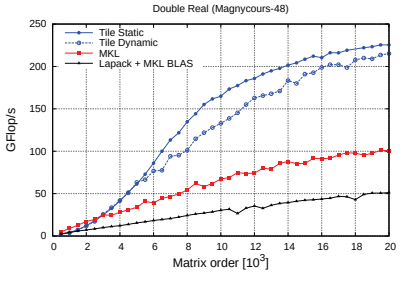
Accuracy Comparison

Matrix	Cond A	No Pivoting	Pivoting	SRBT (IR)
condex	$1 \cdot 10^2$	$5 \cdot 10^{-15}$	$6 \cdot 10^{-15}$	$6 \cdot 10^{-15}$ (0)
fiedler	$7 \cdot 10^5$	Fail	$2 \cdot 10^{-15}$	$9 \cdot 10^{-15}$ (0)
orthog	$1 \cdot 10^0$	$8 \cdot 10^{-1}$	$1 \cdot 10^{-14}$	$3 \cdot 10^{-16}$ (1)
randcorr	$3 \cdot 10^3$	$4 \cdot 10^{-16}$	$3 \cdot 10^{-16}$	$5 \cdot 10^{-16}$ (0)
augment	$5 \cdot 10^4$	$7 \cdot 10^{-15}$	$4 \cdot 10^{-15}$	$2 \cdot 10^{-16}$ (1)
prolate	$6 \cdot 10^{18}$	$8 \cdot 10^{-15}$	$8 \cdot 10^{-16}$	$2 \cdot 10^{-15}$ (0)
toeppd	$1 \cdot 10^7$	$5 \cdot 10^{-16}$	$7 \cdot 10^{-16}$	$2 \cdot 10^{-16}$ (0)
$\lvert i-j \rvert$	$7 \cdot 10^5$	$2 \cdot 10^{-15}$	$2 \cdot 10^{-15}$	$1 \cdot 10^{-14}$ (0)
max(i,j)	$3 \cdot 10^6$	$2 \cdot 10^{-14}$	$2 \cdot 10^{-15}$	$1 \cdot 10^{-14}$ (0)
Hadamard	$1 \cdot 10^0$	0	0	$7 \cdot 10^{-15}$ (0)
rand0	$2 \cdot 10^5$	$1 \cdot 10^{-12}$	$7 \cdot 10^{-14}$	$1 \cdot 10^{-15}$ (1)
rand1	$2 \cdot 10^5$	Fail	$1 \cdot 10^{-13}$	$1 \cdot 10^{-15}$ (1)
rand2	$1 \cdot 10^5$	Fail	$5 \cdot 10^{-14}$	$1 \cdot 10^{-15}$ (1)
rand3	$8 \cdot 10^4$	$4 \cdot 10^{-13}$	$7 \cdot 10^{-14}$	$1 \cdot 10^{-15}$ (1)

Componentwise backward error ($n = 1024$, tile size=8)

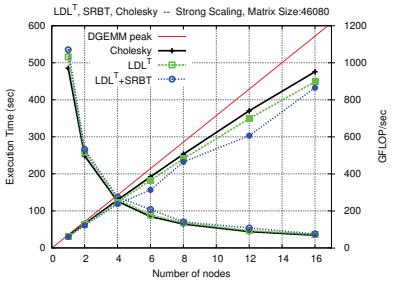
$\omega = \max_i \frac{|Ax-b|_i}{(|A|\cdot|x|+|b|)_i}$

Performance results



Performance of SRBT-LDL^T against MKL and LAPACK (double precision)
(4× 12-Core AMD Opteron 6172 Magny-Cours @ 2.1 GHz)

Comparison with Cholesky



Performance on clusters of multicore, matrix size: 46080
(16 $\times$ 2 quadcores Nehalem Xeon @ 2.27GHz, Infiniband 20G).

Concluding remarks

- Changing architectural and computational landscape
  $\rightarrow$ difficult to propose a unique solver for each type of problem (e.g. LU)
- Randomized algorithms are very promising but $\cdots$
  Requires background in linear algebra, statistics and sometimes the underlying physical problem.
  Need for more research on stability and accuracy issues
- More error analysis tools in new libraries
  Contrary to the time of LAPACK, software for new architectures cannot be easily developed by numerical analysis practitioners
  $\rightarrow$ additional expertise for numerical validation

2.4 Laura Grigori (Inria)

How to Avoid Communication in Linear Algebra and Beyond

How to Avoid Communication in Linear Algebra and Beyond

Laura Grigori
Inria

Abstract

The cost of moving data in an algorithm can surpass by several orders of magnitude the cost of performing arithmetics, and this gap has been steadily and exponentially growing over time. In this talk I will argue that this communication problem needs to be addressed by the numerical software community directly at the mathematical formulation and the algorithmic design level. This requires a paradigm shift in the way the numerical algorithms are devised, which now need to aim at keeping the number of communication instances to a minimum, while retaining their numerical efficiency.

Communication avoiding algorithms provide such a novel perspective on designing algorithms that provably minimize communication in numerical linear algebra. The novel numerical schemes employed, the speedups obtained with respect to conventional algorithms, as well as their impact on applications in computational science will be also discussed.

Avoiding communication in linear algebra

Laura Grigori
ALPINES
INRIA Rocquencourt - LJLL, UPMC

Plan

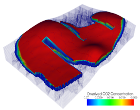
- Motivation
- Selected past work on reducing communication
- Communication complexity of linear algebra operations
- Communication avoiding for dense linear algebra
  - LU, QR, Rank Revealing QR factorizations
  - Often not in ScaLAPACK or LAPACK (YET !)
  - Algorithms for multicore processors
- Communication avoiding for sparse linear algebra
  - Iterative methods and preconditioning
- Conclusions

Page 2

Data driven science

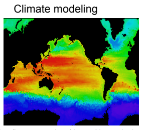
Numerical simulations require increasingly computing power as data sets grow exponentially

CO2 Underground storage



Source: T. Guignon, IFPEN

Climate modeling

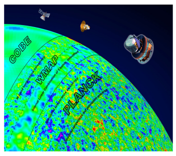


http://www.epm.ornl.gov/chammp/chammp.html

Figures from astrophysics:
- Produce and analyze multi-frequency 2D images of the universe when it was 5% of its current age.
- COBE (1989) collected 10 gigabytes of data, required 1 Teraflop per image analysis.
- PLANCK (2010) produced 1 terabyte of data, requires 100 Petaflops per image analysis.
- CMBPol (2020) is estimated to collect .5 petabytes of data, will require 100 Exaflops per image analysis.

Source: J. Borrill, LBNL, R. Stompor, Paris 7

Astrophysics: CMB data analysis



http://www.scidacreview.org/0704/html/cmb.html

Page 3

Motivation - the communication wall

- Runtime of an algorithm is the sum of:
  - #flops x **time_per_flop**
  - #words_moved / **bandwidth**
  - #messages x **latency**
- Time to move data >> time per flop
  - Gap steadily and exponentially growing over time

Annual improvements			
Time/flop		Bandwidth	Latency
59%	Network	26%	15%
	DRAM	23%	5%

- Performance of an application is less than 5% of the peak performance

Page 4

Motivation

- The communication problem needs to be taken into account higher in the computing stack
- A paradigm shift in the way the numerical algorithms are devised is required
- Communication avoiding algorithms - a novel perspective for numerical linear algebra
  - Minimize volume of communication
  - Minimize number of messages
  - Minimize over multiple levels of memory/parallelism
  - Allow redundant computations (preferably as a low order term)

Page 5

The role of numerical linear algebra

- Challenging applications often rely on solving linear algebra problems
- Linear systems of equations

  Solve $Ax = b$, where $A \in \mathbf{R}^{nxn}$, $b \in \mathbf{R}^n$, $x \in \mathbf{R}^n$
  - Direct methods

    $PA = LU$, then solve $P^T LUx = b$
  - Iterative methods
    - Find a solution x_k from $x_0 + K_k(A, r_0)$, where $K_k(A, r_0) = span\{r_0, A r_0, \ldots, A^{k-1} r_0\}$ such that the Petrov-Galerkin condition $b - Ax_k \perp L_k$ is satisfied, where L_k is a subspace of dimension k and $r_0 = Ax_0 - b$.
    - Convergence depends on $\kappa(A)$ and the eigenvalue distribution (for SPD matrices).

Page 6

Least Square (LS) Problems

- Given $A \in \mathbf{R}^{m \times n}, b \in \mathbf{R}^m$, solve $\min_x \|Ax - b\|_2$.
- Any solution of the LS problem satisfies the normal equations: $A^T Ax = A^T b$
- Given the QR factorization of A

$$A = Q\begin{bmatrix} R \\ 0 \end{bmatrix} \quad \begin{array}{l} A \text{ is } m \times n \text{ real matrix}, m \geq n \\ \text{where } R \text{ is } n \times n \text{ upper triangular matrix} \\ Q \text{ is } m \times m \text{ orthogonal matrix} \end{array}$$

if $rank(A) = rank(R) = n$, then the LS solution is given by $Rx = (Q^T b)(1:n)$

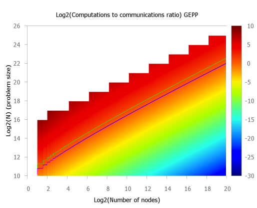
- The QR factorization is column-wise backward stable

$$\left\|(A - \hat{Q}\hat{R})(:,j)\right\|_2 \leq \sqrt{n} f(\varepsilon,n) \|A(:,j)\|_2$$

Page 7

Previous work on reducing communication

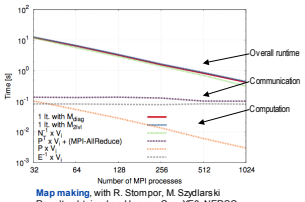
- Tuning
  - Overlap communication and computation, at most a factor of 2 speedup
- Ghosting
  - Store redundantly data from neighboring processors for future computations
- Scheduling
  - Cache oblivious algorithms for linear algebra
    - Gustavson 97, Toledo 97, Frens and Wise 03, Ahmed and Pingali 00
  - Block algorithms for linear algebra
    - ScaLAPACK, Blackford et al 97



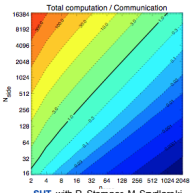
Page 8

Communication in CMB data analysis

- Map-making problem
  - Find the best map *x* from observations *d*, scanning strategy *A*, and noise N^{-1}
  - Solve generalized least squares problem involving sparse matrices of size 10^{12}-by-10^7
- Spherical harmonic transform (SHT)
  - Synthesize a sky image from its harmonic representation
    - Computation over rows of a 2D object (summation of spherical harmonics)
    - Communication to transpose the 2D object
    - Computation over columns of the 2D object (FFTs)



Map making, with R. Stompor, M. Szydlarski
Results obtained on Hopper, Cray XE6, NERSC



SHT, with R. Stompor, M. Szydlarski
Simulation on a petascale computer

Communication Complexity of Dense Linear Algebra

- Matrix multiply, using $2n^3$ flops (sequential or parallel)
  - Hong-Kung (1981), Irony/Tishkin/Toledo (2004)
  - Lower bound on Bandwidth = Ω (#flops / $M^{1/2}$)
  - Lower bound on Latency = Ω (#flops / $M^{3/2}$)
- Same lower bounds apply to LU using reduction
  - Demmel, LG, Hoemmen, Langou 2008

$$\begin{pmatrix} I & -B \\ A & I \\ & I \end{pmatrix} = \begin{pmatrix} I & \\ A & I \\ & I \end{pmatrix} \cdot \begin{pmatrix} I & -B \\ & I \\ & AB \end{pmatrix} \cdot \begin{pmatrix} I & -B \\ & I \\ & I \end{pmatrix}$$

- And to almost all direct linear algebra [Ballard, Demmel, Holtz, Schwartz, 09]

2D Parallel algorithms and communication bounds

- If memory per processor = n^2 / P, the lower bounds become
  #words_moved ≥ Ω (n^2 / $P^{1/2}$), #messages ≥ Ω ($P^{1/2}$)

Algorithm	Minimizing #words (not #messages)	Minimizing #words and #messages
Cholesky	ScaLAPACK	ScaLAPACK
LU	ScaLAPACK uses partial pivoting	[LG, Demmel, Xiang, 08] [Khabou, Demmel, LG, Gu, 12] uses tournament pivoting
QR	ScaLAPACK	[Demmel, LG, Hoemmen, Langou, 08] uses different representation of Q
RRQR	ScaLAPACK	[Branescu, Demmel, LG, Gu, Xiang 11] uses tournament pivoting, 3x flops

- Only several references shown, block algorithms (ScaLAPACK) and communication avoiding algorithms

LU factorization (as in ScaLAPACK pdgetrf)

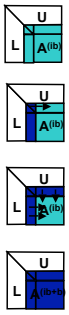
LU factorization on a P = P_r x P_c grid of processors
For ib = 1 to n-1 step b
$A^{(ib)}$ = A(ib:n, ib:n) #messages

(1) Compute panel factorization $O(n \log_2 P_r)$
- find pivot in each column, swap rows

(2) Apply all row permutations $O(n/b(\log_2 P_c + \log_2 P_r))$
- broadcast pivot information along the rows
- swap rows at left and right

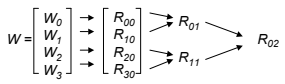
(3) Compute block row of U $O(n/b \log_2 P_c)$
- broadcast right diagonal block of L of current panel

(4) Update trailing matrix $O(n/b(\log_2 P_c + \log_2 P_r))$
- broadcast right block column of L
- broadcast down block row of U



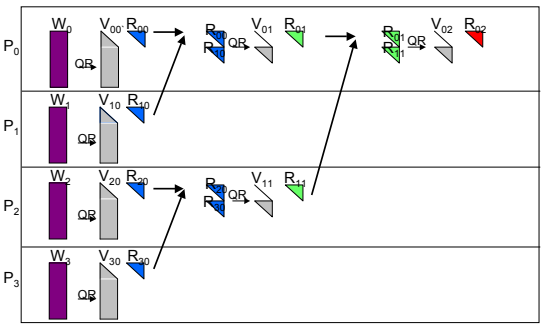
TSQR: QR factorization of a tall skinny matrix using Householder transformations

- QR decomposition of m x b matrix W, m >> b
  - P processors, block row layout
- Classic Parallel Algorithm
  - Compute Householder vector for each column
  - Number of messages ∝ b log P
- Communication Avoiding Algorithm
  - Reduction operation, with QR as operator
  - Number of messages ∝ log P



J. Demmel, LG, M. Hoemmen, J. Langou, 08

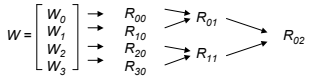
Parallel TSQR



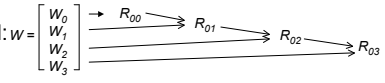
References: Golub, Plemmons, Sameh 88, Pothen, Raghavan, 89, Da Cunha, Becker, Patterson, 02

Flexibility of TSQR and CAQR algorithms

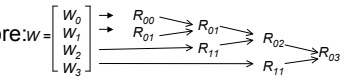
Parallel:



Sequential:

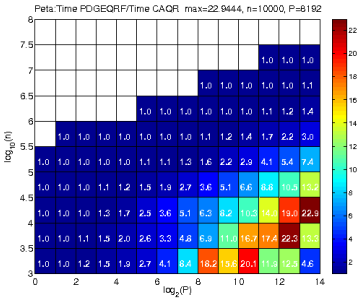


Dual Core:



Reduction tree will depend on the underlying architecture, could be chosen dynamically

Modeled Speedups of CAQR vs ScaLAPACK



Petascale
up to 22.9x

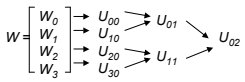
IBM Power 5
up to 9.7x

"Grid"
up to 11x

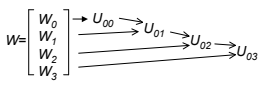
Petascale machine with 8192 procs, each at 500 GFlops/s, a bandwidth of 4 GB/s.
$\gamma = 2 \cdot 10^{-12} s, \alpha = 10^{-5} s, \beta = 2 \cdot 10^{-9} s / word.$

Obvious generalization of TSQR to LU

- Block parallel pivoting:
  - uses a binary tree and is optimal in the parallel case



- Block pairwise pivoting:
  - uses a flat tree and is optimal in the sequential case
  - used in PLASMA for multicore architectures and FLAME for out-of-core algorithms and for multicore architectures



Stability of the LU factorization

- The backward stability of the LU factorization of a matrix A of size n-by-n

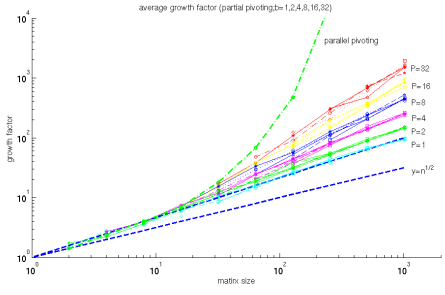
$$\|L\|\cdot|U|\|_\infty \le (1+2(n^2-n)g_w)\|A\|_\infty$$

depends on the growth factor

$$g_W = \frac{\max_{i,j,k}|a_{ij}^k|}{\max_{i,j}|a_{ij}|}$$ where a_{ij}^k are the values at the k-th step.

- $g_W \le 2^{n-1}$, but in practice it is on the order of $n^{2/3}$ -- $n^{1/2}$
- Two reasons considered to be important for the average case stability [Trefethen and Schreiber, 90] :
  - the multipliers in L are small,
  - the correction introduced at each elimination step is of rank 1.

Block parallel pivoting



- Unstable for large number of processors P
- When P=number rows, it corresponds to parallel pivoting, known to be unstable (Trefethen and Schreiber, 90)

Tournament pivoting - the overall idea

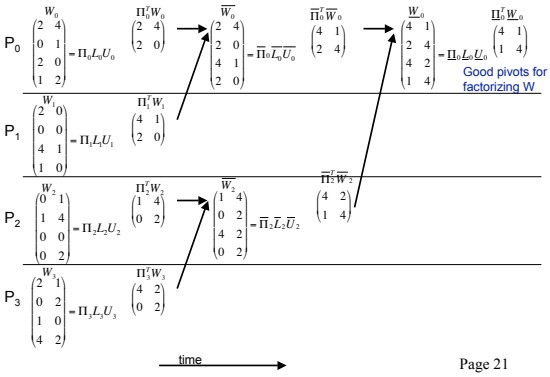
- At each iteration of a block algorithm

$$A = \begin{pmatrix} \tilde{A}_{11} & \tilde{A}_{21} \\ A_{21} & A_{22} \end{pmatrix} \begin{matrix} b \\ n-b \end{matrix}, \text{ where } W = \begin{pmatrix} A_{11} \\ A_{21} \end{pmatrix}$$

  - Preprocess W to find at low communication cost good pivots for the LU factorization of W, return a permutation matrix P.
  - Permute the pivots to top, ie compute PA.
  - Compute LU with no pivoting of W, update trailing matrix.

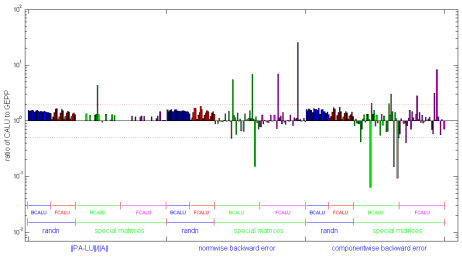
$$PA = \begin{pmatrix} L_{11} & \\ L_{21} & I_{n-b} \end{pmatrix} \begin{pmatrix} U_{11} & U_{12} \\ & A_{22} - L_{21}U_{12} \end{pmatrix}$$

Tournament pivoting

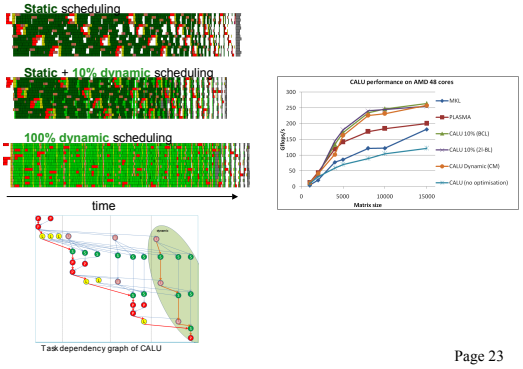


Stability of CALU (experimental results)

- Results show ||PA-LU||/||A||, normwise and componentwise backward errors, for random matrices and special ones
  - See [LG, Demmel, Xiang, 2010] for details
  - BCALU denotes binary tree based CALU and FCALU denotes flat tree based CALU



Lightweight scheduling for CALU



Conclusions

- Introduced a new class of communication avoiding algorithms that minimize communication
  - Attain theoretical lower bounds on communication
  - Minimize communication at the cost of redundant computation
  - Are often faster than conventional algorithms in practice
  - Many previous references, only a few given in the talk
- Remains a lot to do for sparse linear algebra
  - Communication bounds, communication optimal algorithms
  - Numerical stability of s-step methods
  - Preconditioners - limited by the memory size, not flops
- And BEYOND

Collaborators, funding

Collaborators:

- A. Branescu, INRIA, S. Donfack, INRIA, A. Khabou, INRIA, M. Jacquelin, INRIA, S. Moufawad, INRIA, H. Xiang, University Paris 6
- J. Demmel, UC Berkeley, B. Gropp, UIUC, M. Gu, UC Berkeley, M. Hoemmen, UC Berkeley, J. Langou, CU Denver, V. Kale, UIUC

Funding: ANR Petal and Petalh projects, ANR Midas, Digiteo Xscale NL, COALA INRIA funding

Further information:

http://www-rocq.inria.fr/who/Laura.Grigori/

References

Results presented from:

- J. Demmel, L. Grigori, M. F. Hoemmen, and J. Langou, *Communication-optimal parallel and sequential QR and LU factorizations*, UCB-EECS-2008-89, 2008, published in SIAM journal on Scientific Computing, Vol. 34, No 1, 2012.
- L. Grigori, J. Demmel, and H. Xiang, *Communication avoiding Gaussian elimination*, Proceedings of the IEEE/ACM SuperComputing SC08 Conference, November 2008.
- L. Grigori, J. Demmel, and H. Xiang, *CALU: a communication optimal LU factorization algorithm*, SIAM. J. Matrix Anal. & Appl., 32, pp. 1317-1350, 2011.
- M. Hoemmen's Phd thesis, *Communication avoiding Krylov subspace methods*, 2010.
- L. Grigori, P.-Y. David, J. Demmel, and S. Peyronnet, *Brief announcement: Lower bounds on communication for sparse Cholesky factorization of a model problem*, ACM SPAA 2010.
- S. Donfack, L. Grigori, and A. Kumar Gupta, *Adapting communication-avoiding LU and QR factorizations to multicore architectures*, Proceedings of IEEE International Parallel & Distributed Processing Symposium IPDPS, April 2010.
- S. Donfack, L. Grigori, W. Gropp, and V. Kale, *Hybrid static/dynamic scheduling for already optimized dense matrix factorization* , Proceedings of IEEE International Parallel & Distributed Processing Symposium IPDPS, 2012.
- A. Khabou, J. Demmel, L. Grigori, and M. Gu, *LU factorization with panel rank revealing pivoting and its communication avoiding version*, LAWN 263, 2012.
- J. Demmel, L. Grigori, M. Gu, H. Xiang, Communication avoiding rank revealing QR factorization with column pivoting, LAWN 276, 2013.
- L. Grigori, S. Moufawad, *Communication avoiding ILU0 preconditioner*, Inria TR 8266.

2.5 Gaël Guennebaud (Inria)

Eigen : a C++ template library for linear algebra anrelated numerical algorithms

Eigen: a C++ template library for linear algebra and related numerical algorithms.

Gaël Guennebaud
Inria Bordeaux

Abstract

In this talk I will present the open-source Eigen project from both a technical and management point of view.

On the first aspect, Eigen is a versatile C++ template library covering dense and sparse linear algebra in a generic and easy to use manner. High performance is obtained through an optimal use of vector instructions. All of this is possible thanks to an advanced use of expression templates that is at the core of Eigen. Eigen aims to fill the gap between MatLab-like tools and the C/Fortran specialized libraries coming from the HPC community. Eigen is especially appreciated by the graphics, vision, and robotics communities.

On the second aspect, I will show how developing Eigen as a pure open-source project with a fully open repository and open discussions was a key in the success of Eigen. I will also briefly discuss funding and organizational issues to make the project live a long life.

Eigen

a c++ linear algebra library

Gaël Guennebaud

[http://eigen.tuxfamily.org]

Outline

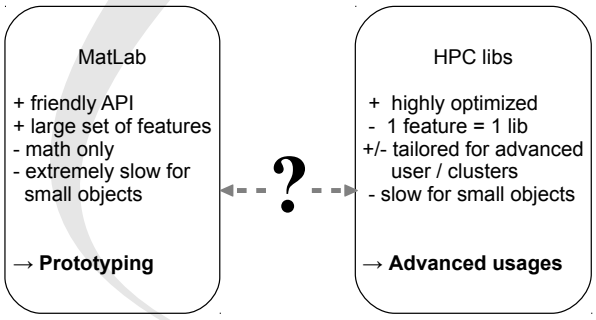
- Eigen as a library
  - Why such a library?
  - What's so special?
  - How does it work?
- Eigen as an opensource community?

Why?

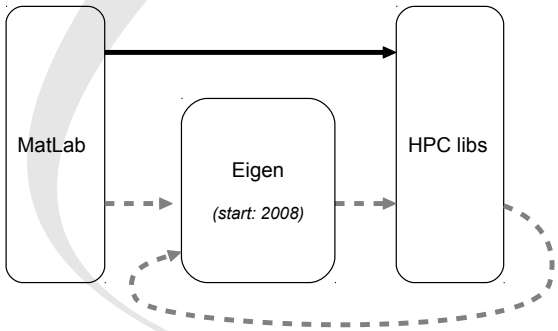
Context

- Matrix computation everywhere
  - Various applications:
    - simulators/simulations, video games, audio/image processing, design, robotic, computer vision, augmented reality, etc.
  - Need various tools:
    - numerical data manipulation, space transformations
    - inverse problems, PDE, spectral analysis
  - Need performance:
    - on standard PC, smartphone, embedded systems, etc.
    - real-time performance

Matrix computation?



Context



What?

Facts

- Pure C++ template library
  - header only, no binary to compile/install...
  - no dependency (optional only)
  - opensource: MPL2

  → **easy to install & distribute**
- Multi-platforms
  - GCC, MSVC, Intel ICC, Clang/LLVM
  - SSE(x86), NEON (ARM), Altivec (PowerPC)

9

Increasing feature set

- Core
  - Matrix and array manipulation (~MatLab, 1D & 2D)
  - Basic linear algebra (~BLAS)
- LU, Cholesky, QR, SVD, Eigenvalues
  - Matrix decompositions and linear solvers (~Lapack)
- Geometry (transformations, …)
- Sparse
  - Manipulation
  - Solvers (LLT, LU, QR & CG, BiCGSTAB, GMRES)
- WIP modules (autodiff, non-linear opt., FFT, etc.)

→ **“unified API” - “all-in-one”**

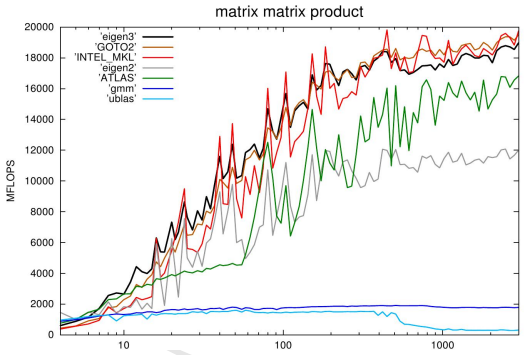
10

Optimized for both small and large objects

- Small objects
  - means fixed sizes:
    ```
    Matrix<float,4,4>
    ```
  - malloc-free
  - meta unrolling
- Large objects
  - means dynamic sizes
    ```
    Matrix<float,Dynamic,1>
    ```
  - cache friendly kernels *(perf ~ MKL)*
  - multi-threading *(OpenMP)*
- Vectorization (SIMD)
- Unified API → write generic code
- Mixed fixed/dynamic dimensions

11

Performance



12

Custom scalar types

- Can use custom types everywhere
  - Exact arithmetic (rational numbers)
  - Multi-precision numbers (e.g., via mpfr++)
  - Auto-diff scalar types
  - Interval
  - Symbolic
- Example:

```
typedef Matrix<mpreal,Dynamic,Dynamic> MatrixX;
MatrixX A, B, X;
// init A and B
// solve for A.X=B using LU decomposition
X = A.lu().solve(B);
```

13

How?

14

Expression templates

- Example:

```
m3 = m1 + m2 + m3;
```

- Standard C++ way:

```
tmp1 = m1 + m2;
tmp2 = tmp1 + m3;
m3   = tmp2;
```

15

Expression templates

- Example:

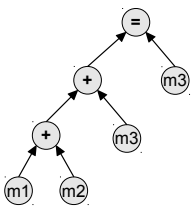
```
m3 = m1 + m2 + m3;
```

- Expression templates:
  - “+” returns an expression => expression tree
  - e.g.: A+B returns:

```
Sum<type_of_A, type_of_B> {
    const type_of_A &A;
    const type_of_B &B;
};
```

  - complete example:

```
Assign<Matrix,
       Sum< Sum<Matrix,Matrix> , Matrix > >
```



16

Expression templates

- Example:

```
m3 = m1 + m2 + m3;
```

- Evaluation:
  - Top-down creation of an evaluator
    - e.g.:

```
Evaluator<Sum<type_of_A,type_of_B> > {
  Evaluator<type_of_A> evalA(A);
  Evaluator<type_of_B> evalB(B);
  Scalar coeff(i,j) {
    return evalA.coeff(i,j) + evalB.coeff(i,j);
  }
};
```

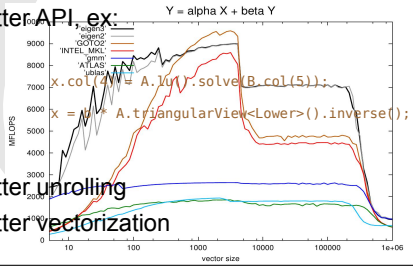
  - Assignment produces:

```
for(i=0; i<m3.size(); ++i)
  m3[i] = m1[i] + m2[i] + m3[i];
```

17

ET: Immediate benefits

- Fused operations
  - Temporary removal
  - Reduce memory accesses, cache misses
- Better API, ex:




```
X.col(4) = A.lu().solve(B.col(5));

X = A * A.triangularView<Lower>().inverse();
```

- Better unrolling
- Better vectorization

18

Cost model

- Cost Model
  - Track an approximation of the cost to evaluate one coefficient
- Control of:
  - loop unrolling (partial)
  - evaluation of sub expressions, e.g.:
    - ***(a+b)*** ** c* → *(a+b)* is evaluated into a temporary
    - enable vectorization of sub expressions

19

Top-down expression analysis

- Products
  - detect BLAS-like sub expressions
    - e.g.: `m4 -= 2 * m2.adjoint() * m3;`
      → `gemm<Adj,Nop>(m2, m3, -2, m4);`
    - e.g.: `m4.block(...) += ((2+4i) * m2).adjoint() * m3.block(...).transpose();`

```
Evaluator<Product<type_of_A,type_of_B> > {
  EvaluatorForProduct<type_of_A> evalA(A);
  EvaluatorForProduct<type_of_B> evalB(B);
};
```

20

Top-down expression analysis (cont.)

- More complex example:

```
m4 -= m1 + m2 * m3;
```

  - so far:
```
tmp = m2 * m3;
m4 -= m1 + tmp;
```
  - better:
```
m4 -= m1;
m4 += m2 * m3;
```

```
// catch R = A + B * C
Evaluator<Assign<R,Sum<A,Product<B,C> > > { … };
```

21

Tree optimizer

- Even more complex example:

```
res -= m1 + m2 + m3*m4 + 2*m5 – m6*m7;
```

  - Tree optimizer
    → `res -= ((m1 + m2 + 2*m5) + m3*m4) + m6*m7;`
  - yields:
```
res -= m1 + m2 + 2*m5;
res += m3*m4;
res += m6*m7;
```
  - Need only two rules:

```
// catch A * B + Y and builds Y' + A' * B'
TreeOpt<Sum<Product<A,B>,Y> > { … };

// catch X + A * B + Y and builds (X' + Y') + (A' * B')
TreeOpt<Sum<Sum<X,Product<A,B> >,Y> > { … };
```

22

Tree optimizer

- Last example:

```
res += m1 * m2 * v;
```

  - Tree optimizer
    → `res += m1 * (m2 * v);`
  - Rule:

```
TreeOpt<Product<Product<A,B>,C> > { … };
```

23

Community?

24

Developer Community

- Jan 2008: start of Eigen2
  - part of KDE
    - packaged by all Linux distributions
  - open repository
  - open discussions on mailing/IRC
    - 300 members, 300 messages/month
      → high quality API
- Today
  - most development @ Inria (Gaël + full-time engineer)
- Future
  → consortium... ??

25

User community

- Active project with many users
  - Website: ~30k unique visitors / month (+10% / month)
- Robotics, computer vision, graphics
  - Google street view
  - Willow garage (ROS, PCL)
  - CEA

26

License

- Initially:
  - LGPL3+
    → default choice
    → not as liberal as it might look...
- Now:
  - MPL2 (Mozilla Public License 2.0)
    - same spirit but with tons of advantages:
      - accepted by industries
      - do work with header only libraries
      - versatile (apply to anything)
      - a lot simpler
      - good reputation

27

What next?

- Work in progress:
  - AVX, CUDA
  - More parallelization
  - Sparse block matrices
  - Non linear optimization
  - Utility modules
    - Polynomial manipulation/solver
    - Autodiff

28

Conclusion

- Goal: → ideal compromise between:
  - versatility
  - ease of use
  - performance

http://eigen.tuxfamily.org

29

30

Sparse Matrix

- Representation & manipulations
  - compressed format
- Built-in solvers (Ax=b)
  - direct: simplicial Cholesky, LU with supernodes, QR
  - iterative: CG, BiCGSTAB, GMRES (ILUT)

```
1: typedef SparseMatrix<double,ColMajor> SpMat;
2: SpMat A(rows,cols);
3: A.setFromTriplets(elements.begin(), elements.end());
4: SimplicialCholesky<SpMat,Lower> chol_A(A);
5: x = chol_A.solve(b);
```

31

Supports for external libraries

- Can fallback to Intel MKL
  - Feature implemented by Intel
- Unified interface to many sparse solvers:
  - UmfPack, Cholmod, SuperLU
  - PaSTiX, Pardiso

32

Eigen vs BLAS/Lapack

- Pros:
  - C++ friendly API
  - Matrix manipulation
  - Easy to use, install, distribute, etc.
  - Custom scalar types
  - Static allocation
  - Temporary removal
  - Auto vectorization, ARM NEON
  - Higher perf for small objects
  - etc.
- Cons:
  - Covers only most common features of lapack
  - Not fully multi-threaded yet

33

Eigen & BLAS

- Call Eigen's algorithms through a BLAS/Lapack API
  - Alternative to ATLAS and GotoBlas
    - Makes sense for third party sparse solvers
  - Run the Lapack test suite on Eigen
- Fallback to existing BLAS/Lapack/etc.
  - WIP

34

Eigen as a code generator

```
#include<Eigen/Core>
using namespace Eigen;

void foo(Matrix2f& u,
         float a, const Matrix2f& v,
         float b, const Matrix2f& w)
{
  u = a*v + b*w - u;
}
```

```
movl 8(%ebp), %edx
movss 20(%ebp), %xmm0
movl 24(%ebp), %eax
movaps %xmm0, %xmm2
shufps $0, %xmm2, %xmm2
movss 12(%ebp), %xmm0
movaps %xmm2, %xmm1
mulps (%eax), %xmm1
shufps $0, %xmm0, %xmm0
movl 16(%ebp), %eax
mulps (%eax), %xmm0
addps %xmm1, %xmm0
subps (%edx), %xmm0
movaps %xmm0, (%edx)
```

35

Good Documentation

- Generated every day
  - Doxygen doc with many examples
  - Tutorial/Manual
  - Quick reference guide
  - Quick MatLab to Eigen guide
  - Discussions on advanced topics

```
source code + inline doc      3.9M
user guide + snippets         1.8M
unit tests                    1.1M
```

36

Reliability

- Extensive unit tests
  - run every days on various platforms
  - include BLAS and Lapack test suites

37

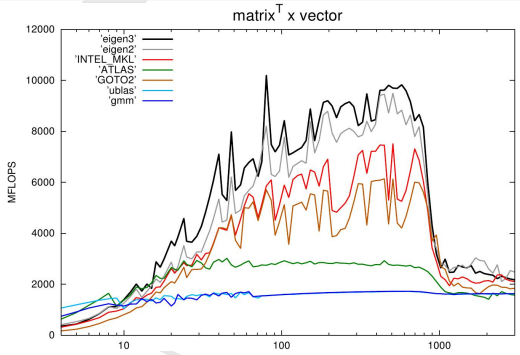
Fancy Matrices

- Diagonal view/matrix
- Triangular views
- Selfadjoint views

```
MatrixXf A, B, C;
// ...
X = A.triangularView<Upper|UnitDiag>().solve(B);     // Ax=b
A.selfadjointView<Lower>().rankUpdate(B,-1);         // A -= BB'
```

38

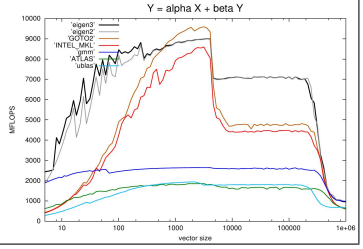
Performance



39

Expression template engine

- Lazy expression evaluation
  - fused operations
    - reduce temporaries
    - reduce memory accesses, cache misses
  - nice API
  - expression level optimizations



40

Summary

- Many unique features
- Goal: → ideal compromise between:
  - versatility
  - ease of use
  - performance
- in-between MatLab ↔ specialized numerical packages
- Main targets:
  - researchers, end user applications, embedded applications, education, etc.

41

42

Expression templates in Eigen

- Cost model:
  - controls loop unrolling
  - controls evaluation of nested expressions, e.g.:
    - *(**a+b**) * c* => (*a+b)* is evaluated into a temporary
- ! Aliasing ! => enforcing evaluation:
  - *a = a.corner(TopLeft,a.rows()-1,a.cols()-1).**eval**();*
  - *a.row(i) = a.col(j).**eval**();*
- Matrix products are not expressions:
  - *m = m * m; // works out of the box*
  - *c.**noalias**() = a * b; // avoids a useless temporary*

43

Motivations

- “Linear algebra for every body”
  - easy to use, install, distribute, etc.
  - versatility
  - offer a large set of tools
  - consistent API
  - high performance
- in-between MatLab ↔ specialized numerical packages

2.6 Fabienne Jézéquel (LIP6, UPMC)

Estimation d’erreur d’arrondi par la bibliothèque CADNA

Estimation d'erreur d'arrondi par la bibliothèque CADNA

Fabienne Jézéquel

Laboratoire d'Informatique de Paris 6 (LIP6)

Résumé

L'Arithmétique Stochastique Discrète (ASD) est une méthode automatique d'analyse d'erreur d'arrondi fondée sur une approche probabiliste. L'ASD consiste à exécuter un programme plusieurs fois de manière synchrone en utilisant un mode d'arrondi aléatoire, ce qui permet d'estimer le nombre de chiffres significatifs exacts des résultats.

La bibliothèque CADNA, qui implémente l'ASD, permet dans un code scientifique en C ou en Fortran d'estimer la qualité numérique des résultats et de détecter les instabilités numériques générées pendant l'exécution. En outre, CADNA permet de développer de nouvelles méthodologies de programmation et ainsi d'optimiser les critères de convergence des algorithmes itératifs. La bibliothèque CADNA a été utilisée pour la validation numérique de codes de taille importante dans un contexte académique ou industriel.

Estimation d'erreur d'arrondi par la bibliothèque CADNA

Jean-Marie Chesneaux, Fabienne Jézéquel,
Jean-Luc Lamotte, Jean Vignes

Laboratoire d'Informatique de Paris 6,
P. and M. Curie University, Paris, France

Séminaire Aristote
Ecole Polytechnique
15 mai 2013

Overview

- Floating-point arithmetic and round-off errors
- The CESTAC method and the stochastic arithmetic
- The CADNA software
- Contributions of CADNA in numerical methods

Rounding mode

Let $\mathbb{F}$ be the set of real numbers which can be coded exactly on a computer: the set of floating point numbers.

Every real number x which is not a floating point number is approximated by a floating point number $X \in \mathbb{F}$.

Let X_{min} (resp. X_{max}) be the smallest (resp. the greatest) floating point number:

$$\forall x \in]X_{min}, X_{max}[\ , \ \exists \{X^-, X^+\} \in \mathbb{F}^2$$

such that

$$X^- < x < X^+ \text{ and }]X^-, X^+[\cap \mathbb{F} = \emptyset$$

To choose the rounding mode is to choose the algorithm that, according to x, gives X^- or X^+.

The 4 rounding modes of the IEEE 754 standard

Rounding to zero: x is represented by the floating point number the nearest to x between x and 0.

Rounding to nearest: x is represented by the floating point number the nearest to x.

Rounding to plus infinity: x is represented by X^+.

Rounding to minus infinity: x is represented by X^-.

The rounding operation is performed after each assignment and after every elementary arithmetic operation.

Inconsistency of the floating point arithmetic

On a computer, arithmetic operators are only approximations.

- commutativity
- no associativity
- no distributivity

On a computer, order relationships are the same as in mathematics

$\Longrightarrow$ it leads to a global inconsistent behaviour.

$$X = Y \not\Rightarrow x = y \quad \text{and} \quad x = y \not\Rightarrow X = Y.$$
$$X \geq Y \not\Rightarrow x \geq y \quad \text{and} \quad x \geq y \not\Rightarrow X \geq Y.$$

Round-off error model

Let $r \in \mathbb{R}$ be the exact result of a computation of n elementary arithmetic operations.

On a computer, one obtains the result $R \in \mathbb{F}$ which is affected by round-off errors.

R can be modeled, at the first order with respect to 2^{-p}, by

$$R \approx r + \sum_{i=1}^{n} g_i(d).2^{-p}.\alpha_i$$

p is the number of bits used for the representation including the hidden bit, $g_i(d)$ are coefficients depending only on data and α_i are the round-off errors.

Remark: we have assumed that exponents and signs of intermediate results do not depend on α_i.

A theorem on numerical accuracy

The number of significant bits in common between R and r is defined by

$$C_R \approx -\log_2 \left| \frac{R-r}{r} \right| = p - \log_2 \left| \sum_{i=1}^{n} g_i(d).\frac{\alpha_i}{r} \right|$$

The last part corresponds to the accuracy which has been lost in the computation of R, we can note that it is independent of p.

Theorem

The loss of accuracy during a numerical computation is independent of the precision used.

Round-off error analysis

Several approaches

- Inverse analysis
  based on the " Wilkinson principle": the computed solution is assumed to be the exact solution of a nearby problem
  - provides error bounds for the computed results
- Interval arithmetic
  The result of an operation between two intervals contains all values that can be obtained by performing this operation on elements from each interval.
  - guaranteed bounds for each computed result
  - the error may be overestimated
  - specific algorithms
- Probabilistic approach
  - uses a random rounding mode
  - estimates the number of exact significant digits of any computed result

The CESTAC method

The CESTAC method (Contrôle et Estimation Stochastique des Arrondis de Calculs) was proposed by M. La Porte and J. Vignes in 1974.

It consists in performing the same code several times with different round-off error propagations. Then, different results are obtained.

Briefly, the part that is common to all the different results is assumed to be in common also with the mathematical results and the part that is different in the results is affected by the round-off errors.

The random rounding mode

Let r be the result of an arithmetic operation: $R^- < r < R^+$.

The random rounding mode consists in rounding r to minus infinity or plus infinity with the probability 0.5.

If round-off errors affect the result, even slightly, one obtains for N different runs, N different results on which a statistical test may be applied.

By running N times the code with the random arithmetic, one obtains a N-sample of the random variable modeled by

$$R \approx r + \sum_{i=1}^{n} g_i(d).2^{-p}.\alpha_i$$

where the α_i's are modeled by independent identically distributed random variables. The common distribution of the α_i is uniform on $[-1, +1]$.

$\Rightarrow$ the mathematical expectation of R is the mathematical result r,

$\Rightarrow$ the distribution of R is a quasi-Gaussian distribution.

Implementation of the CESTAC method

The implementation of the CESTAC method in a code providing a result R consists in:

- performing N times this code with the random rounding mode to obtain N samples R_i of R,
- choosing as the computed result the mean value $\overline{R}$ of R_i, $i = 1, ..., N$,
- estimating the number of exact significant decimal digits of $\overline{R}$ with

$$C_{\overline{R}} = \log_{10}\left(\frac{\sqrt{N}\,|\overline{R}|}{\sigma\tau_\beta}\right)$$

where

$$\overline{R} = \frac{1}{N}\sum_{i=1}^{N} R_i \quad \text{and} \quad \sigma^2 = \frac{1}{N-1}\sum_{i=1}^{N}\left(R_i - \overline{R}\right)^2.$$

τ_β is the value of Student's distribution for $N-1$ degrees of freedom and a probability level β.

On the number of runs

2 or 3 runs are enough. To increase the number of runs is not necessary.

From the model, to increase by 1 the number of exact significant digits given by $C_{\overline{R}}$, we need to multiply the size of the sample by 100.

Such an increase of N will only point out the limit of the model and its error without really improving the quality of the estimation.

It has been shown that $N = 3$ is the optimal value.

Self-validation of the CESTAC method

The CESTAC method is based on a 1st order model.

- A multiplication of two insignificant results
- or a division by an insignificant result

may invalidate the 1st order approximation.

Therefore the CESTAC method requires a dynamical control of multiplications and divisions, during the execution of the code.

The problem of stopping criteria

Let a general iterative algorithm be: $U_{n+1} = F(U_n)$, U_0 being a data.

```
WHILE (ABS(X-Y) > EPSILON) DO
   X = Y
   Y = F(X)
ENDDO
```

ε too low $\Longrightarrow$ a risk of infinite loop
ε too high $\Longrightarrow$ a too early termination.

The optimal choice from the computer point of view:
$X - Y$ **an insignificant value**.

New methods for numerical algorithms may be developed.

The concept of computed zero

J. Vignes, 1986

Definition

Using the CESTAC method, a result R is a computed zero, denoted by @.0, if

$$\forall i, R_i = 0 \text{ or } C_{\overline{R}} \leq 0.$$

This means that 0 belongs to the confidence interval.

It means that R is a computed result which, because of round-off errors, cannot be distinguished from 0.

The stochastic definitions

Definition

Let X and Y be two results computed using the CESTAC method (N-sample), X is stochastically equal to Y, noted $X \ s= \ Y$, if and only if

$$X - Y = @.0.$$

Definition

Let X and Y be two results computed using the CESTAC method (N-sample).

- X is stochastically strictly greater than Y, noted $X \ s> \ Y$, if and only if

$$\overline{X} > \overline{Y} \quad \text{and} \quad X \ s\neq \ Y$$

- X is stochastically greater than or equal to Y, noted $X \ s\geq \ Y$, if and only if

$$\overline{X} \geq \overline{Y} \quad \text{or} \quad X \ s= \ Y$$

DSA **Discrete Stochastic Arithmetic** is defined as the joint use of the CESTAC method, the computed zero and the relation definitions.

A few properties

- $x = 0 \implies X = @.0$.
- $X \ s\neq \ Y \implies x \neq y$.
- $X \ s> \ Y \implies x > y$.
- $x \geq y \implies X \ s\geq \ Y$.
- The relation $s>$ is transitive.
- The relation $s=$ is reflexive, symmetric but not transitive.
- The relation $s\geq$ is reflexive, antisymmetric but not transitive.

The CADNA library

The CADNA library implements Discrete Stochastic Arithmetic.

CADNA allows to estimate round-off error propagation in any scientific program.

More precisely, CADNA enables one to:

- estimate the numerical quality of any result
- control branching statements
- perform a dynamic numerical debugging
- take into account uncertainty on data.

CADNA is a library which can be used with Fortran or C++ programs.

CADNA can be downloaded from http://www.lip6.fr/cadna

The stochastic types

CADNA provides two new numerical types, the stochastic types (3 floating point variables `x,y,z` and a hidden variable `acc`):

- `type (single_st)` for stochastic variables in single precision stochastic type associated with real.
- `type (double_st)` for stochastic variables in double precision stochastic type associated with double precision.

All the operators and mathematical functions are overloaded for these types.

The cost of CADNA is about:

- 4 for memory
- 10 for run time.

Specific implementations of DSA

- Parallelization of CADNA
- CADNA for parallel programs
  - MPI
  - GPU
  - OpenMP (work in progress)
- SAM (Stochastic Arithmetic in Multiprecision)

How to implement CADNA

The use of the CADNA library involves six steps:

- declaration of the CADNA library for the compiler,
- initialization of the CADNA library,
- substitution of the type REAL or DOUBLE PRECISION by stochastic types in variable declarations,
- possible changes in the input data if perturbation is desired, to take into account uncertainty in initial values,
- change of output statements to print stochastic results with their accuracy,
- termination of the CADNA library.

An example proposed by S. Rump (1)

Computation of $f(10864, 18817)$ and $f(\frac{1}{3}, \frac{2}{3})$ with $f(x, y) = 9x^4 - y^4 + 2y^2$

```
      program ex1
      implicit double precision (a-h,o-z)
      x = 10864.d0
      y = 18817.d0
      write(*,*)'P(10864,18817) = ', rump(x,y)
      x = 1.d0/3.d0
      y = 2.d0/3.d0
      write(6,100) rump(x,y)
100   format('P(1/3,2/3) = ',e24.15)
      end

      function rump(x,y)
      implicit double precision (a-h,o-z)
      a=9.d0*x*x*x*x
      b=y*y*y*y
      c=2.d0*y*y
      rump = a-b+c
      return
      end
```


An example proposed by S. Rump (2)

The results:

```
P(10864,18817) =    2.00000000000000
P(1/3,2/3) =    0.802469135802469E+00
```

```
program ex1

implicit double precision  (a-h,o-z)

x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)

end

function rump(x,y)

implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```



```
program ex1
use cadna
implicit double precision  (a-h,o-z)

x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)

end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```



```
program ex1
use cadna
implicit double precision  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)

end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```



```
program ex1
use cadna
implicit double precision  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```



```
program ex1
use cadna
implicit double precision  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit double precision  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```



```
program ex1
use cadna
implicit type(double_st)  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit type(double_st)  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```



```
program ex1
use cadna
implicit type(double_st)  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ', rump(x,y)
call cadna_end()
end

function rump(x,y)
use cadna
implicit type(double_st)  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```



```
program ex1
use cadna
implicit type(double_st)  (a-h,o-z)
call cadna_init(-1)
x = 10864.d0
y = 18817.d0
write(*,*)'P(10864,18817) = ',str(rump(x,y))
x = 1.d0/3.d0
y = 2.d0/3.d0
write(*,*)'P(10864,18817) = ',str(rump(x,y))
call cadna_end()
end

function rump(x,y)
use cadna
implicit type(double_st)  (a-h,o-z)
a = 9.d0*x*x*x*x
b = y*y*y*y
c = 2.d0*y*y
rump = a-b+c
return
end
```

The run with CADNA

```
CADNA software — University P. et M. Curie — LIP6
Self-validation detection: ON
Mathematical instabilities detection: ON
Branching instabilities detection: ON
Intrinsic instabilities detection: ON
Cancellation instabilities detection: ON
P(10864,18817) = @.0
P(1/3,2/3) = 0.802469135802469E+000
CADNA software — University P. et M. Curie — LIP6
There are 2 numerical instabilities
0 UNSTABLE DIVISION(S)
0 UNSTABLE POWER FUNCTION(S)
0 UNSTABLE MULTIPLICATION(S)
0 UNSTABLE BRANCHING(S)
0 UNSTABLE MATHEMATICAL FUNCTION(S)
0 UNSTABLE INTRINSIC FUNCTION(S)
2 UNSTABLE CANCELLATION(S)
```


Contributions of CADNA

- In direct methods:
  - estimate the numerical quality of the results
  - control branching statements
- In iterative methods:
  - optimize the number of iterations
  - check if the computed solution is satisfactory
- In approximation methods:
  - optimize the integration step

In direct methods - Example

$$0.3x^2 - 2.1x + 3.675 = 0$$

Without CADNA, in single precision with rounding to the nearest:
d = -3.8146972E-06
Two complex roots
z1 = 0.3499999E+01 + i * 0.9765625E-03
z2 = 0.3499999E+01 + i * -.9765625E-03

With CADNA:
d = @.0
The discriminant is null
The double real root is 0.3500000E+01

Iterative methods: which strategy to adopt?

- problems with a solution that cannot be controlled (sequence computation):
  The following stopping criterion should be used

  IF $(x(k).eq.x(k+1))$ THEN

- problems with a solution that can be controlled:
  the solution x_s satisfies $\Psi(x_s) = 0$.
  The optimal stopping criterion should be used

  IF $(\Psi(x(k)).eq.0)$ THEN

Iterative methods: the solution cannot be controlled

$$S_n(x) = \sum_{i=1}^{n} \frac{x^i}{i!}$$

Stopping criterion
- IEEE: $|S_n - S_{n-1}| < 10^{-15}|S_n|$
- CADNA: $S_n == S_{n-1}$

	IEEE		CADNA	
x	iter	$S_n(x)$	iter	$S_n(x)$
-5.	37	6.737946999084039E-003	38	0.673794699909E-002
-10.	57	4.539992962303130E-005	58	0.45399929E-004
-15.	76	3.059094197302006E-007	77	0.306E-006
-20.	94	5.621884472130416E-009	95	@.0
-25.	105	-7.129780403672074E-007	106	@.0

Iterative methods: the solution can be controlled

The linear system $AX = B$ is solved using Jacobi method.

$$x_i^{(k+1)} = -\frac{1}{a_{ii}} \sum_{j=1, j\neq i}^{n} a_{ij} x_j^{(k)} + \frac{b_i}{a_{ii}}$$

Without CADNA
- Stop when $\max_{i=1}^{n} |x_i^k - x_i^{k-1}| < \varepsilon$
- Compute $R = B - AX^k$.

eps=1.E-3

```
niter =          35
x( 1)= 0.1699924E+01 (exact: 0.1700000E+01), r( 1)= 0.3051758E-03
x( 2)=-0.4746889E+04 (exact:-0.4746890E+04), r( 2)= 0.1953125E-02
x( 3)= 0.5023049E+02 (exact: 0.5023000E+02), r( 3)= 0.1464844E-02
x( 4)=-0.2453197E+03 (exact:-0.2453200E+03), r( 4)=-0.7324219E-03
x( 5)= 0.4778290E+04 (exact: 0.4778290E+04), r( 5)=-0.4882812E-03
x( 6)=-0.7572980E+02 (exact:-0.7573000E+02), r( 6)= 0.9765625E-03
x( 7)= 0.3495430E+04 (exact: 0.3495430E+04), r( 7)= 0.3173828E-02
x( 8)= 0.4350277E+01 (exact: 0.4350000E+01), r( 8)= 0.0000000E+00
x( 9)= 0.4529804E+03 (exact: 0.4529800E+03), r( 9)= 0.9765625E-03
x(10)=-0.2759901E+01 (exact:-0.2760000E+01), r(10)= 0.9765625E-03
x(11)= 0.8239241E+04 (exact: 0.8239240E+04), r(11)= 0.7568359E-02
x(12)= 0.3459919E+01 (exact: 0.3460000E+01), r(12)=-0.4882812E-03
x(13)= 0.1000000E+04 (exact: 0.1000000E+04), r(13)= 0.9765625E-03
x(14)=-0.4999743E+01 (exact:-0.5000000E+01), r(14)= 0.1464844E-02
x(15)= 0.3642400E+04 (exact: 0.3642400E+04), r(15)=-0.1953125E-02
x(16)= 0.7353594E+03 (exact: 0.7353600E+03), r(16)=-0.3662109E-03
x(17)= 0.1700038E+01 (exact: 0.1700000E+01), r(17)= 0.1464844E-02
x(18)=-0.2349171E+04 (exact:-0.2349170E+04), r(18)= 0.1953125E-02
x(19)=-0.8247521E+04 (exact:-0.8247520E+04), r(19)=-0.8728027E-02
x(20)= 0.9843570E+04 (exact: 0.9843570E+04), r(20)= 0.0000000E+00
```


eps=1.E-4

```
niter =        1000
x( 1)= 0.1699924E+01 (exact: 0.1700000E+01), r( 1)= 0.1831055E-03
x( 2)=-0.4746890E+04 (exact:-0.4746890E+04), r( 2)=-0.4882812E-03
x( 3)= 0.5022963E+02 (exact: 0.5023000E+02), r( 3)=-0.9765625E-03
x( 4)=-0.2453193E+03 (exact:-0.2453200E+03), r( 4)= 0.1464844E-02
x( 5)= 0.4778290E+04 (exact: 0.4778290E+04), r( 5)=-0.1464844E-02
x( 6)=-0.7573022E+02 (exact:-0.7573000E+02), r( 6)=-0.1953125E-02
x( 7)= 0.3495430E+04 (exact: 0.3495430E+04), r( 7)= 0.5126953E-02
x( 8)= 0.4350277E+01 (exact: 0.4350000E+01), r( 8)=-0.4882812E-03
x( 9)= 0.4529798E+03 (exact: 0.4529800E+03), r( 9)=-0.9765625E-03
x(10)=-0.2760255E+01 (exact:-0.2760000E+01), r(10)=-0.1953125E-02
x(11)= 0.8239240E+04 (exact: 0.8239240E+04), r(11)= 0.3173828E-02
x(12)= 0.3459731E+01 (exact: 0.3460000E+01), r(12)=-0.1464844E-02
x(13)= 0.1000000E+04 (exact: 0.1000000E+04), r(13)=-0.1953125E-02
x(14)=-0.4999743E+01 (exact:-0.5000000E+01), r(14)= 0.1953125E-02
x(15)= 0.3642400E+04 (exact: 0.3642400E+04), r(15)= 0.0000000E+00
x(16)= 0.7353599E+03 (exact: 0.7353600E+03), r(16)=-0.7324219E-03
x(17)= 0.1699763E+01 (exact: 0.1700000E+01), r(17)=-0.4882812E-03
x(18)=-0.2349171E+04 (exact:-0.2349170E+04), r(18)= 0.0000000E+00
x(19)=-0.8247520E+04 (exact:-0.8247520E+04), r(19)=-0.9155273E-03
x(20)= 0.9843570E+04 (exact: 0.9843570E+04), r(20)=-0.3906250E-02
```

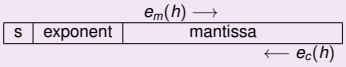
With CADNA

```
 niter =           29
x( 1)= 0.170E+01     (exact: 0.1699999E+01), r( 1)=@.0
x( 2)=-0.4746888E+04 (exact:-0.4746888E+04), r( 2)=@.0
x( 3)= 0.5023E+02    (exact: 0.5022998E+02), r( 3)=@.0
x( 4)=-0.24532E+03   (exact:-0.2453199E+03), r( 4)=@.0
x( 5)= 0.4778287E+04 (exact: 0.4778287E+04), r( 5)=@.0
x( 6)=-0.75729E+02   (exact:-0.7572999E+02), r( 6)=@.0
x( 7)= 0.349543E+04  (exact: 0.3495428E+04), r( 7)=@.0
x( 8)= 0.435E+01     (exact: 0.4349999E+01), r( 8)=@.0
x( 9)= 0.45298E+03   (exact: 0.4529798E+03), r( 9)=@.0
x(10)=-0.276E+01     (exact:-0.2759999E+01), r(10)=@.0
x(11)= 0.823923E+04  (exact: 0.8239236E+04), r(11)=@.0
x(12)= 0.346E+01     (exact: 0.3459999E+01), r(12)=@.0
x(13)= 0.10000E+04   (exact: 0.9999996E+03), r(13)=@.0
x(14)=-0.5001E+01    (exact:-0.4999999E+01), r(14)=@.0
x(15)= 0.364239E+04  (exact: 0.3642398E+04), r(15)=@.0
x(16)= 0.73536E+03   (exact: 0.7353597E+03), r(16)=@.0
x(17)= 0.170E+01     (exact: 0.1699999E+01), r(17)=@.0
x(18)=-0.234917E+04  (exact:-0.2349169E+04), r(18)=@.0
x(19)=-0.8247515E+04 (exact:-0.8247515E+04), r(19)=@.0
x(20)= 0.984356E+04  (exact: 0.9843565E+04), r(20)=@.0
```


Approximation methods

How to estimate the optimal step?

If h decreases, $X(h)$:



If $e_c(h) < e_m(h)$, decreasing h brings reliable information.

Computation should stop when $e_c(h) \approx e_m(h)$

Approximation of integrals

$I = \int_a^b f(x)dx$ is computed using a quadrature method (trapezoidal rule, Simpson's rule, ...)

Let I_n be the approximation computed with step $h = \frac{b-a}{2^n}$.

The computation stops when $I_n - I_{n+1} = @.0$.

```
DO WHILE (integold .NE. integ)
 integold = integ
 h=h/2
 ...
 integ = h * ( ... )
ENDDO
```

Using this strategy, the significant digits of the result which are not affected by round-off errors are in common with I, up to one.

Approximation methods with the CADNA library

Approximation of $\int_{-1}^{1} 20cos(20x)\,((2.7x - 3.3)x + 1.2)\,dx$ using Simpson's method.

```
n= 1 In= 0.532202672142964E+002 err= 0.459035794670113E+002
n= 2 In=-0.233434428466744E+002 err= 0.306601305939595E+002
n= 3 In=-0.235451792663099E+002 err= 0.308618670135950E+002
n= 4 In= 0.106117380632568E+002 err= 0.329505031597175E+001
n= 5 In= 0.742028156692706E+001 err= 0.1035938196419E+000
n= 6 In= 0.732233719854278E+001 err= 0.564945125770E-002
n= 7 In= 0.731702967403266E+001 err= 0.34192674758E-003
n= 8 In= 0.731670894914430E+001 err= 0.2120185922E-004
n= 9 In= 0.731668906978969E+001 err= 0.13225046E-005
n=10 In= 0.731668782990089E+001 err= 0.8261581E-007
n=11 In= 0.731668775244794E+001 err= 0.516286E-008
n=12 In= 0.7316687747607 8E+001 err= 0.3227E-009
n=13 In= 0.731668774730 3E+001 err= 0.202E-010
n=14 In= 0.731668774728 4E+001 err= 0.1E-011
n=15 In= 0.731668774728 2E+001 err= 0.1E-012
n=16 In= 0.731668774728 1E+001 err=@.0
```

The exact solution is: `7.316687747285081429939`.

Conclusion

- ☺ can be used on real life applications
- ☹ difficult to understand the numerical instabilities in large codes
- ☹ time and memory consuming
- ☺ solution for parallel programs (MPI and GPU)
- ☹ difficult to use with the libraries (BLAS, LAPACK ...)

On the probability of the confidence interval

With $\beta = 95\%$ and $N = 3$,

- the probability of overestimating the number of exact significant digits of at least 1 is 0.054%
- the probability of underestimating the number of exact significant digits of at least 1 is 29%.

By choosing a confidence interval at 95%, we prefer to guarantee a minimal number of exact significant digits with high probability (99.946%), even if we are often pessimistic by 1 digit.

2.7 François-Xavier Roux (ONERA)

Mise en oe uvre de méthodes de résolution par sous-domaines parallèles dans des codes d'éléments finis

Mise en œuvre de méthodes de résolution par sous-domaines parallèles dans des codes d'éléments finis

Abstract

On rappellera tout d'abord les propriétés des méthodes de résolution par sous-domaines sans recouvrement, du type BDD et FETI, qui peuvent être considérées comme des méthodes de résolution hybrides,mêlant approches directes, locales et itératives globales.
On parlera aussi de méthodologies de préconditionnement par projection adaptées aux méthodes de résolution par sous-domaines.

On montrera enfin que la façon la plus simple de mettre en œuvre en parallèle ces méthodes dans un code d'éléments finis, consiste à récupérer les informations au niveau élémentaire. Ce qui amène finalement à ne conserver du code initial que la partie qui concerne réellement la physique, et à la coupler au solveur par sous-domaines.

Méthodes FETI

François-Xavier Roux
Unité Calcul à Haute Performance
ONERA
THE FRENCH AEROSPACE LAB

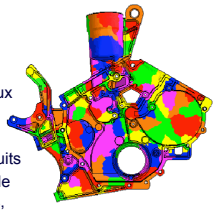
retour sur innovation

Plan

- Principe des méthodes de résolution par sous-domaines sans recouvrement
- Méthode du complément de Schur
- Mise en œuvre en parallèle
- Méthode FETI
- Préconditionneur global « grille grossière »
- Préconditionneur local optimal,
- FETI-2LM
- Mise en œuvre dans des codes existants
- Séparation physique-algèbre linéaire
- Conclusion

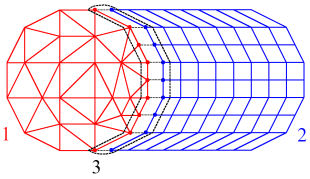
Principe des méthodes de résolution par sous-domaines

- Découpage du maillage en sous-domaines : partition par éléments
- Résolution itérative globale
- Accélération par résolution exacte dans les sous-structures
  ⇨ solution d'un problème condensé aux interfaces
- Méthodes mixtes directes-itératives
  ⇨ robustes, coûts de factorisation réduits
- Résolution de problèmes de très grande taille mal conditionnés, mécanique des structures, électromagnétisme,…



Découpage par sous-domaine d'une matrice creuse

- Graphe de la matrice, graphe du maillage
- Séparateur



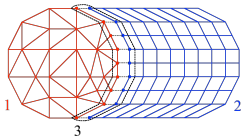
- Structure de la matrice par blocs

$$\begin{pmatrix} K_{11} & 0 & K_{13} \\ 0 & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$

Formation du système par sous-domaines

- Découpage du système en deux systèmes locaux

$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} \end{pmatrix}, \begin{pmatrix} x_1 \\ x_3^{(1)} \end{pmatrix}, \begin{pmatrix} b_1 \\ b_3^{(1)} \end{pmatrix} \qquad \begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} \end{pmatrix}, \begin{pmatrix} x_2 \\ x_3^{(2)} \end{pmatrix}, \begin{pmatrix} b_2 \\ b_3^{(2)} \end{pmatrix}$$



$$K_{33}^{(1)} + K_{33}^{(2)} = K_{33}$$

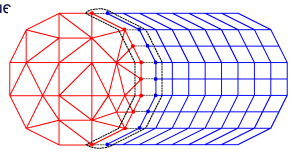
$$b_3^{(1)} + b_3^{(2)} = b_3$$

Equations locales, conditions de raccord

- Equations internes dans chaque sous-domaine

$$K_{11}\, x_1 + K_{13}\, x_3^{(1)} = b_1$$

$$K_{22}\, x_2 + K_{23}\, x_3^{(2)} = b_2$$



- Condition d'admissibilité sur l'interface

$$x_3^{(1)} = x_3^{(2)} \quad (= x_3)$$

- Condition d'équilibre à l'interface

$$K_{31}\, x_1 + K_{32}\, x_2 + K_{33}\, x_3 = b_3$$

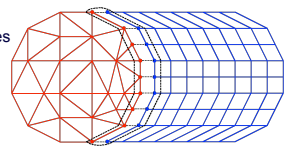
$$K_{31}\, x_1 + K_{33}^{(1)}\, x_3^{(1)} - b_3^{(1)} + K_{32}\, x_2 + K_{33}^{(2)}\, x_3^{(2)} - b_3^{(2)} = 0$$

Condensation locale: méthode du complément de Schur

- Variable d'interface: $x_3 = x_3^{(1)} = x_3^{(2)}$
- Détermination des valeurs internes dans chaque sous-domaine par résolution des équations locales

$$K_{11}\, x_1 = b_1 - K_{13}\, x_3$$

$$K_{22}\, x_2 = b_2 - K_{23}\, x_3$$



- Condition de raccord ⇨ définition du résidu à l'interface

$$K_{31}\, x_1 + K_{32}\, x_2 + K_{33}\, x_3 - b_3 =$$

$$\left(K_{33} - K_{31}K_{11}^{-1}K_{13} - K_{32}K_{22}^{-1}K_{23}\right)x_3 - \left(b_3 - K_{31}K_{11}^{-1}b_1 - K_{32}K_{22}^{-1}b_2\right)$$

Mise en œuvre

- Résolution des équations locales par une méthode directe (Gauss, Crout)

$$K_{11}\, x_1 = b_1 - K_{13}\, x_3 \qquad K_{22}\, x_2 = b_2 - K_{23}\, x_3$$

- Calcul du résidu interface local

$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} \end{pmatrix}\begin{pmatrix} x_1 \\ x_3 \end{pmatrix} - \begin{pmatrix} b_1 \\ b_3^{(1)} \end{pmatrix} = \begin{pmatrix} 0 \\ \left(K_{33}^{(1)} - K_{31}K_{11}^{-1}K_{13}\right)x_3 - \left(b_3^{(1)} - K_{31}K_{11}^{-1}b_1\right) \end{pmatrix}$$

$$\begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} \end{pmatrix}\begin{pmatrix} x_2 \\ x_3 \end{pmatrix} - \begin{pmatrix} b_2 \\ b_3^{(2)} \end{pmatrix} = \begin{pmatrix} 0 \\ \left(K_{33}^{(2)} - K_{32}K_{22}^{-1}K_{23}\right)x_3 - \left(b_3^{(2)} - K_{32}K_{22}^{-1}b_2\right) \end{pmatrix}$$

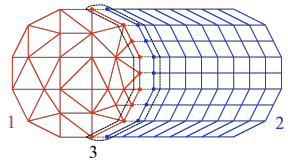
- Assemblage du résidu interface global

$$\left(K_{33}^{(1)} - K_{31}K_{11}^{-1}K_{13}\right)x_3 - \left(b_3^{(1)} - K_{31}K_{11}^{-1}b_1\right) + \left(K_{33}^{(2)} - K_{32}K_{22}^{-1}K_{23}\right)x_3 - \left(b_3^{(2)} - K_{32}K_{22}^{-1}b_2\right)$$

$$= \left(K_{33} - K_{31}K_{11}^{-1}K_{13} - K_{32}K_{22}^{-1}K_{23}\right)x_3 - \left(b_3 - K_{31}K_{11}^{-1}b_1 - K_{32}K_{22}^{-1}b_2\right)$$

Mise en œuvre en parallèle et propriétés

- Un domaine par processeur
- Résolution des problèmes locaux indépendants dans chaque processeur
- Echange des résidus interface entre domaines voisins
- Variables et résidus interface connus dans les deux domaines



- Méthode itérative sur l'interface, directe à l'intérieur
- Problème mieux conditionné, moins d'itérations que pour une méthode itérative globale
- Meilleure granularité qu'une méthode itérative globale
- Seuls les blocs internes des matrices locales sont factorisés, moins coûteux qu'une factorisation globale

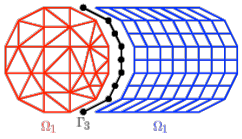
Conditions de raccord duale

- Problèmes locaux

$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} \end{pmatrix}\begin{pmatrix} x_1 \\ x_3^{(1)} \end{pmatrix}=\begin{pmatrix} b_1 \\ b_3^{(1)}+\lambda_1 \end{pmatrix} \qquad \begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} \end{pmatrix}\begin{pmatrix} x_2 \\ x_3^{(2)} \end{pmatrix}=\begin{pmatrix} b_2 \\ b_3^{(2)}+\lambda_2 \end{pmatrix}$$

- Conditions de raccord

$$x_3^{(1)} - x_3^{(2)} = 0$$
$$\lambda_1 + \lambda_2 = 0$$



Méthode FETI

- Variable d'interface : $\lambda = \lambda_1 = -\lambda_2$
- Problèmes locaux :

$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} \end{pmatrix}\begin{pmatrix} x_1 \\ x_3^{(1)} \end{pmatrix}=\begin{pmatrix} b_1 \\ b_3^{(1)}+\lambda \end{pmatrix} \qquad \begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} \end{pmatrix}\begin{pmatrix} x_2 \\ x_3^{(2)} \end{pmatrix}=\begin{pmatrix} b_2 \\ b_3^{(2)}-\lambda \end{pmatrix}$$

- Résidu à l'interface : $x_3^{(1)} - x_3^{(2)}$

Problème condensé à l'interface

- Problèmes locaux

$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} \end{pmatrix}\begin{pmatrix} x_1 \\ x_3^{(1)} \end{pmatrix}=\begin{pmatrix} b_1 \\ b_3^{(1)}+\lambda \end{pmatrix} \qquad \begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} \end{pmatrix}\begin{pmatrix} x_2 \\ x_3^{(2)} \end{pmatrix}=\begin{pmatrix} b_2 \\ b_3^{(2)}-\lambda \end{pmatrix}$$

- Condensation à l'interface

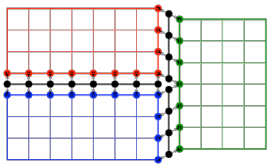
$$(K_{33}^{(1)} - K_{31}K_{11}^{-1}K_{13})\,x_3^{(1)} = b_3^{(1)} - K_{31}K_{11}^{-1}b_1 + \lambda$$
$$S^{(1)}x_3^{(1)} = c_3^{(1)} + \lambda$$
$$(K_{33}^{(2)} - K_{32}K_{22}^{-1}K_{23})\,x_3^{(2)} = b_3^{(2)} - K_{32}K_{22}^{-1}b_2 - \lambda$$
$$S^{(2)}x_3^{(2)} = c_3^{(2)} - \lambda$$

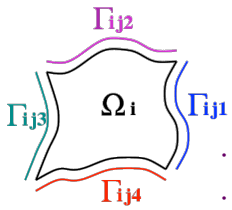
- Problème condensé à l'interface

$$x_3^{(1)} - x_3^{(2)} = 0 \;\Leftrightarrow\; \left(S^{(1)^{-1}} + S^{(2)^{-1}}\right)\lambda = -S^{(1)^{-1}}c_3^{(1)} + S^{(2)^{-1}}c_3^{(2)}$$

Raccords redondants pour les interfaces multiples

- Pas de variables interface multiples





- Découplage des calculs sur les différentes interfaces
- Mise en œuvre identique pour 2 ou n sous-domaines

Modes singuliers locaux : interprétation mécanique

- Problème de Neumann local singulier
- Mouvements de corps rigides

$$\begin{cases} x_3^{(1)} = S^{(1)^+}\left(c_3^{(1)}+\lambda\right) + N^{(1)}\alpha^{(1)} \\ N^{(1)^t}\left(c_3^{(1)}+\lambda\right) = 0 \end{cases} \qquad \begin{cases} x_3^{(2)} = S^{(2)^+}\left(c_3^{(2)}-\lambda\right) + N^{(2)}\alpha^{(2)} \\ N^{(2)^t}\left(c_3^{(2)}-\lambda\right) = 0 \end{cases}$$

- Problème condensé à l'interface mixte : résolution par projection

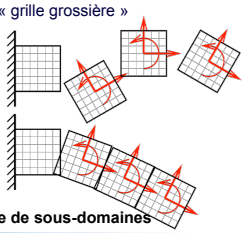
$$\begin{pmatrix} S^{(1)^*}+S^{(2)^*} & N^{(1)} & -N^{(2)} \\ N^{(1)^t} & 0 & 0 \\ -N^{(2)^t} & 0 & 0 \end{pmatrix}\begin{pmatrix} \lambda \\ \alpha^{(1)} \\ \alpha^{(2)} \end{pmatrix}=\begin{pmatrix} -S^{(1)^*}c_3^{(1)}+S^{(2)^*}c_3^{(2)} \\ -N^{(1)^t}c_3^{(1)} \\ N^{(2)^t}c_3^{(2)} \end{pmatrix}$$

- Couplage des déplacements de corps rigides locaux

$$\begin{pmatrix} N^{(1)^t}N^{(1)} & -N^{(1)^t}N^{(2)} \\ -N^{(2)^t}N^{(1)} & N^{(2)^t}N^{(2)} \end{pmatrix}$$

Préconditionneur global « grille grossière »

- Transmission entre sous-domaines voisins à chaque itération
  ⇨ interactions entre domaines éloignés prises en compte par diffusions successives
  ⇨ nombre d'itérations croissant avec le nombre de sous-domaines
  ⇨ nécessité d'un mécanisme de transfert d'information entre tous les sous-domaines, préconditionneur global « grille grossière »
- Résolution par projection de FETI
- Ajustement des mouvements de corps rigides à chaque itération
  ⇨ résolution d'un problème « grossier » global
- **Convergence indépendante du nombre de sous-domaines**



Préconditionneur local optimal

- Opérateur du problème condensé à l'interface pour FETI

$$F = \left(S^{(1)^{-1}} + S^{(2)^{-1}}\right)$$

- Préconditionneur local optimal (**convergence indépendante de la finesse du maillage**) $\tilde{F}^{-1} = \left(\frac{1}{2}S^{(1)}\frac{1}{2} + \frac{1}{2}S^{(2)}\frac{1}{2}\right)$
- Calcul du gradient préconditionné

$$K_{11}\,v_1 = -K_{13}\frac{1}{2}g \qquad \begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} \end{pmatrix}\begin{pmatrix} v_1 \\ \frac{1}{2}g \end{pmatrix}=\begin{pmatrix} 0 \\ \left(K_{33}^{(1)} - K_{31}K_{11}^{-1}K_{13}\right)\frac{1}{2}g \end{pmatrix}$$

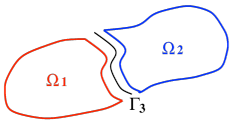
$$K_{22}\,v_2 = -K_{23}\frac{1}{2}g \qquad \begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} \end{pmatrix}\begin{pmatrix} v_2 \\ \frac{1}{2}g \end{pmatrix}=\begin{pmatrix} 0 \\ \left(K_{33}^{(2)} - K_{32}K_{22}^{-1}K_{23}\right)\frac{1}{2}g \end{pmatrix}$$

- Deux problèmes à résoudre par itération dans chaque sous-domaine

Conditions de raccord de Fourier : méthode FETI-2LM

- Système linéaire global

$$\begin{pmatrix} K_{11} & 0 & K_{13} \\ 0 & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$



- Systèmes linéaires locaux

$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} + k_1 \end{pmatrix} \begin{pmatrix} x_1 \\ x_3^{(1)} \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3^{(1)} + \lambda_1 \end{pmatrix}$$

$$\begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} + k_2 \end{pmatrix} \begin{pmatrix} x_2 \\ x_3^{(2)} \end{pmatrix} = \begin{pmatrix} b_2 \\ b_3^{(2)} + \lambda_2 \end{pmatrix}$$

- Equations de raccord aux interfaces

$$\begin{cases} x_3^{(1)} = x_3^{(2)} \\ k_1 x_3^{(1)} + k_2 x_3^{(2)} = \lambda_1 + \lambda_2 \end{cases} \Leftrightarrow$$

$$\begin{cases} \lambda_1 + \lambda_2 - (k_1 + k_2) x_3^{(2)} = 0 \\ \lambda_1 + \lambda_2 - (k_2 + k_1) x_3^{(1)} = 0 \end{cases}$$

Problème interface condensé

- Equations locales condensées à l'interface

$$(k_1 + K_{33}^{(1)} - K_{31} K_{11}^{-1} K_{13}) x_3^{(1)} = \lambda_1 + b_3^{(1)} - K_{31} K_{11}^{-1} b_1$$

$$(k_2 + K_{33}^{(2)} - K_{32} K_{22}^{-1} K_{23}) x_3^{(2)} = \lambda_2 + b_3^{(2)} - K_{32} K_{22}^{-1} b_2$$

- Matrice des équations de raccord aux interfaces

$$\begin{pmatrix} I & I - (k_1 + k_2)(k_2 + K_{33}^{(2)} - K_{32} K_{22}^{-1} K_{23})^{-1} \\ I - (k_2 + k_1)(k_1 + K_{33}^{(1)} - K_{31} K_{11}^{-1} K_{13})^{-1} & I \end{pmatrix}$$

Conditions de raccord optimales

- Conditions de Fourier optimales

$$k_1 = K_{33}^{(2)} - K_{32} K_{22}^{-1} K_{23}$$
$$k_2 = K_{33}^{(1)} - K_{31} K_{11}^{-1} K_{13}$$

- Conditions optimales = rigidité condensée du domaine opposé sur l'interface
- Interprétation par condensation locale dans les équations globales

$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} + K_{33}^{(2)} - K_{32} K_{22}^{-1} K_{23} \end{pmatrix} \begin{pmatrix} x_1 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_3^{(1)} + b_3^{(2)} - K_{32} K_{22}^{-1} b_2 \end{pmatrix}$$

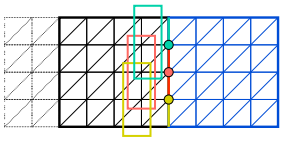
Caractéristiques de la méthode

- Problèmes locaux bien posés, même avec des découpages complexes
- Convergence en *p -1* itérations en cas de découpage en *p* tranches

- Problème : impossible en pratique de calculer exactement l'opérateur optimal (complément de Schur)
- Détermination d'un opérateur approché

Approximation creuse du complément de Schur

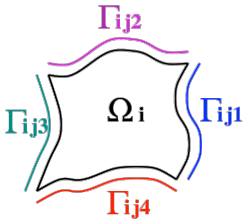
- Condensation locale sur des petites zones
- Assemblage pondéré



- Approche purement algébrique
- Mise en œuvre en « boîte » noire
- Utilisable pour tout type d'éléments
- Convergence très rapide pour des domaines très contrastés

Mise en œuvre en parallèle

- Parallélisation facile par attribution d'un sous-domaine à un processus
- Gestion des itérations par échanges de données aux interfaces
- Données nécessaires localement :
  - matrice de rigidité
  - valeurs imposées (Dirichlet)
  - contraintes
  - second membre
  - interfaces

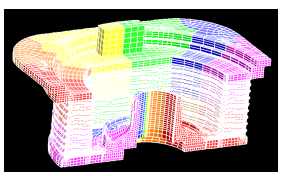


Description d'une interface :

identification du sous-domaine voisin, liste des équations

Intégration dans un code existant

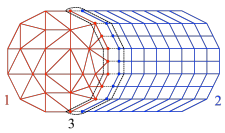
- Nécessité d'un partitionnement du maillage par éléments
- « Coloriage »
- Par l'utilisateur en fonction de critères géométriques ou physiques
- Automatique à l'aide d'un partitionneur de graphe, SCOTCH, METIS



Partitionnement algébrique

- Découpage du système en deux systèmes locaux partir du système global

$$\begin{pmatrix} K_{11} & 0 & K_{13} \\ 0 & K_{22} & K_{23} \\ K_{31} & K_{32} & K_{33} \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ x_3 \end{pmatrix} = \begin{pmatrix} b_1 \\ b_2 \\ b_3 \end{pmatrix}$$



$$\begin{pmatrix} K_{11} & K_{13} \\ K_{31} & K_{33}^{(1)} \end{pmatrix}, \begin{pmatrix} x_1 \\ x_3^{(1)} \end{pmatrix}, \begin{pmatrix} b_1 \\ b_3^{(1)} \end{pmatrix}$$

$$\begin{pmatrix} K_{22} & K_{23} \\ K_{32} & K_{33}^{(2)} \end{pmatrix}, \begin{pmatrix} x_2 \\ x_3^{(2)} \end{pmatrix}, \begin{pmatrix} b_2 \\ b_3^{(2)} \end{pmatrix}$$

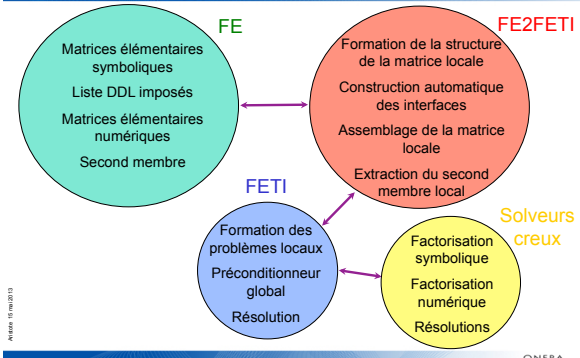
$$K_{33}^{(1)} + K_{33}^{(2)} = K_{33}$$
$$b_3^{(1)} + b_3^{(2)} = b_3$$

$$K_{33}^{(1)} \neq K_{33}^{(2)} \neq \frac{1}{2} K_{33}$$

Interfaçage au niveau élémentaire

- Autant de processus distribués (MPI) que de domaines
- Chaque processus élément fini ne traite que les éléments de sa couleur (numéro de sous-domaine)
- Couche d'interface entre le code élément fini et le solveur
  - phase symbolique : liste des degrés de liberté dans chaque élément, liste des degrés de liberté imposés, contraintes
  - phase numérique : matrice élémentaire de chaque élément
  - phase résolution : second membre assemblé ou non, valeur des degrés de liberté imposé et des contraintes
- Numérotation globale
- Tout ce qui n'est pas élémentaire peut être redondant
- **Tout ce qui est calcul élémentaire est parallélisé automatiquement**

Structure logicielle



Avantages

- Chaque couche utilise ses propres structures de données et même son propre langage
- Passage de données par tableaux
- La couche FE2FETI peut gérer complètement le parallélisme
- Le code élément fini ne fait plus d'algèbre linéaire
- **C'est la couche FE2FETI qui a vocation à s'adapter au code élément fini et pas l'inverse**
- Développements réalisés dans le cadre de collaborations multiples, logiciel mutualisé
- Utilisation des différentes bibliothèques de solveurs creux disponibles : MUMPS, PARDISO, BCS, … + petit solveur skyline interne pour les problèmes globaux et les tests

Conclusion, méthodes de sous-domaines

- Mise en œuvre très simple au niveau élémentaire
- Interprétation mécanique très utile pour déterminer les bonnes approches (conditions de Fourier, espace réduit pour préconditionnement « grille grossière »)
- Adaptation des méthodes en fonction des problèmes traités
- Généralisation aux problèmes multi-physiques fortement couplés (interaction fluide-structure)
- Couplage de codes
- Maillages non coïncidents : contraintes multi-point

Conclusion

- A chacun ses compétences
- La meilleure façon de faire un code qui vieillit bien : ne pas l'écrire…

2.8 Laurent Plagne (EDF)

Legolas++ Conception d'outils génériques pour les problèmes linéaires creux structurés par blocs multi-niveaux

Legolas++: outils génériques pour les problèmes linéaires creux structurés par blocs multi-niveaux

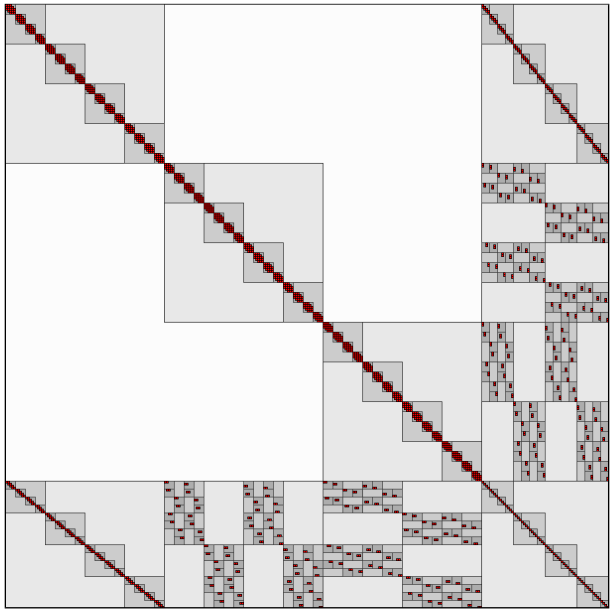
Laurent Plagne (EDF R&D)

15 mai 2013

Il existe un grand nombre de bibliothèques d'algèbre linéaire qui permettent de manipuler des matrices creuses. Pour utiliser la connaissance a priori de l'emplacement préférentiel des éléments non nuls des matrices, ces bibliothèques peuvent fournir des formats de stockage plus ou moins adaptés (skyline, Compressed Row Storage, Sparse Blocked Storage...). Ces formats résultent d'un compromis entre la généralité de ces bibliothèques, leur efficacité et le degré d'expressivité du langage utilisé pour définir leur interface.

Les matrices structurées par blocs multi-niveaux (cf. figure) ne peuvent être prises en charge par les bibliothèques HPC existantes. En l'absence de bibliothèque dédiée, les problèmes linéaires impliquant ces matrices ont jusqu'à présent été traités par des logiciels spécifiques. Malheureusement, cette solution pragmatique prive les équipes de développements des deux bénéfices attendus de l'usage des bibliothèques :

- la mutualisation des efforts de développement et d'optimisation,
- la séparation des champs sémantiques (physique, algèbre linéaire, informatique) améliorant le travail des équipes pluridisciplinaires.



La conception de la bibliothèque Legolas++ a été engagée pour tenter de résoudre ce problème.

Après avoir introduit les objets principaux qui composent Legolas++, cet exposé mettra l'accent sur le rôle de l'expressivité du langage utilisé (C++) ainsi que sur les obstacles qui doivent être surmontés pour produire des codes efficaces sur les architectures actuelles (processeurs multi-coeurs et GPU).

Legolas++

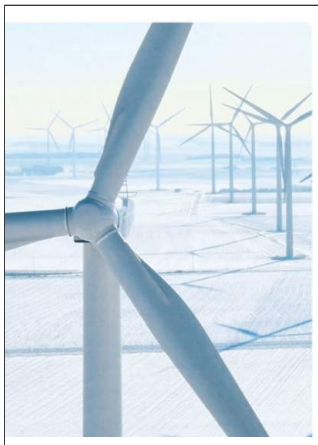
Outils génériques pour les problèmes linéaires creux structurés par blocs multi-niveaux

Laurent Plagne (EDF R&D)
laurent.plagne@edf.fr

Séminaire Aristote 15 mai 2013

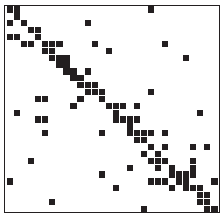
EDF CHANGER L'ENERGIE ENSEMBLE

Sommaire



1. Matrices creuses structurées par bloc multi-niveaux

Matrices creuses généralistes : bibliothèques C et F77

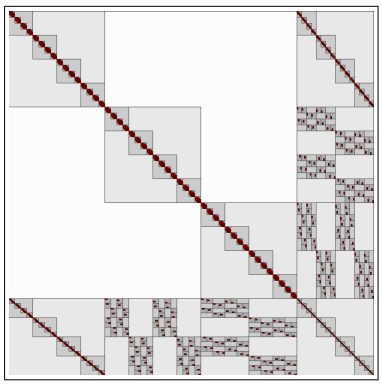


Des formats variés basés sur des tableaux C ou F77

- Skyline (1IA-1FA)
- Diagonal (1IA-1FA)
- Coordinate (2IA-1FA)
- Compressed Sparse Columns (2IA-1FA)
- Compressed Sparse Row (2IA-1FA)
- Block Compressed Sparse Rows (2IA-1FA)
- Modified Block Compressed Sparse Rows (MBSR) (OSKI)

```
SUBROUTINE mkl_dbsrgemv(transa, m, lb, a, ia, ↩
    ja, x, y)
   CHARACTER*1 transa
   INTEGER      m, lb
   INTEGER      ia(*), ja(*)
   DOUBLE PRECISION     a(*), x(*), y(*)
```


Matrices creuses très structurées : (SPN)



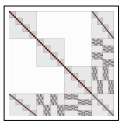
Sans bibliothèques : logiciels spécifiques

Sans outils disponibles, l'approche classique est de développer des solveurs spécifiques avec des langages procéduraux (F77/C).
Deux inconvénients :

- Pas de mutualisation.
- Pas de séparation des champs sémantiques
  `solveNeutronicScatteringProblemWithGaussSeidelMethod()`.
  - Pas d'équipe pluri-disciplinaire
    → développeurs pluri-disciplinaires.

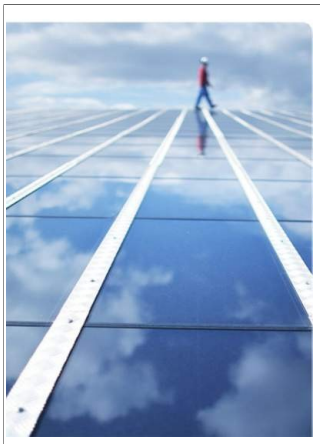
Machines parallèles multi-niveaux...

Pourquoi n'y a t-il pas d'outils ?



- Le problème est-il d'intérêt général ?
- Est-il possible de concevoir ces outils avec les langages du HPC (C/F77) ?)
  - Bibliothèque classique ? Framework ? mini-langage ?

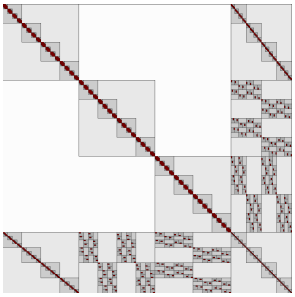
Contraintes : intensité arithmétique et granularité des opérations envisagées→ polymorphisme statique en C++.



2. Legolas++ : V0.1 Polymorphisme statique

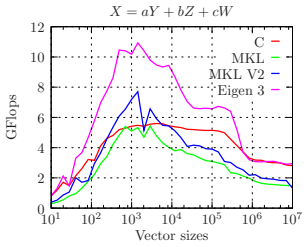
Legolas++ : principaux objectifs

- Décrire les matrices creuses structurées par bloc multi-niveaux, les vecteurs et les algorithmes associés.



Legolas++ : principaux objectifs

- Décrire les matrices creuses structurées par bloc multi-niveaux, les vecteurs et les algorithmes associés.
- Pas de perte de performance induite par l'outil.



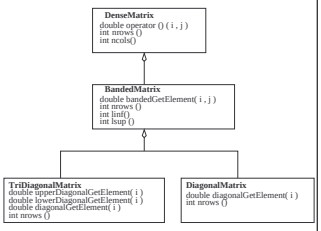
Legolas++ : principaux objectifs

- Décrire les matrices creuses structurées par bloc multi-niveaux, les vecteurs et les algorithmes associés.
- Pas de perte de performance induite par l'outil.
- Possibilité de décrire des matrices non stockées.

$$\begin{pmatrix} -4 & 1 & & & \\ 1 & -4 & 1 & & \\ & \ddots & \ddots & \ddots & \\ & & 1 & -4 & 1 \\ & & & 1 & -4 \end{pmatrix}$$

Legolas++ : principaux objectifs

- Décrire les matrices creuses structurées par bloc multi-niveaux, les vecteurs et les algorithmes associés.
- Pas de perte de performance induite par l'outil.
- Possibilité de décrire des matrices non stockées.
- La hiérarchie des matrices doit être respectée.



Legolas++ : principaux objectifs

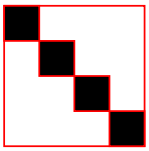
- Décrire les matrices creuses structurées par bloc multi-niveaux, les vecteurs et les algorithmes associés.
- Pas de perte de performance induite par l'outil.
- Possibilité de décrire des matrices non stockées.
- La hiérarchie des matrices doit être respectée.
- Écriture naturelle algorithmes par blocs.

```
do
  for (int i=0;i<n;i++){
    s=B[i];
    for (int j=0;j<i;j++)
      s-=A(i,j)*X[j];
    for (int j=i+1;j<n;j++)
      s-=A(i,j)*X[j];
    X[i]=s/A(i,i);
  }
}while(!iter.end(X));
```

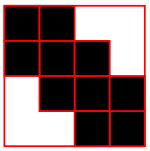

Legolas++ : systèmes creux structurés

Un langage dédié (EDSL) aux problèmes d'algèbre linéaire creux structurés

À partir de quelques structures simples :



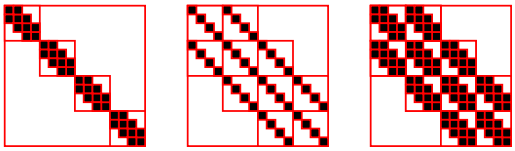
Diagonal



TriDiagonal

Composition verticale

De nouvelles structures par blocs peuvent être créées :

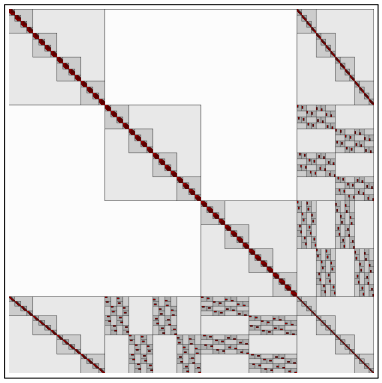


Composition horizontale

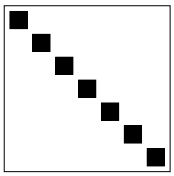
- Une matrice peut être définie comme une somme : A=B+C

- Une matrice peut être définie comme un produit : A=B*C

Matrices creuses très structurées : (SPN)



Opérateurs polymorphiques : Y=A*X

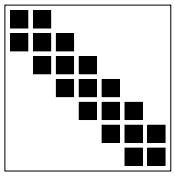


Produit Matrix-Vecteur Diagonal :

```
Mat A; Vec X,Y;

for (int i=0; i < A.nrows(); i++){
  Y[i]=A.diagGetElt(i)*X[i];
}
```

Écrit par un développeur (de Legolas++)

Opérateurs polymorphiques : Y=A*X

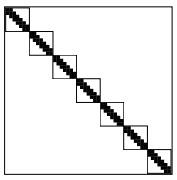


Produit Matrix-Vecteur Banded :

```
Mat A; Vec X,Y;

int linf=A.linf();
int lsup=A.lsup();
for (int i=0; i< A.nrows(); i++){
  Vec::Elt s=0.0;
  int jmin=max(i-linf,0);
  int jmax=min(i+lsup+1,size);
  for (int j=jmin; j< jmax; j++)
    s+=A.bandGetElt(i,j)*X[j];
  Y[i]=s;
}
```

Écrit par un développeur (de Legolas++)

Opérateurs polymorphiques : Y=A*X

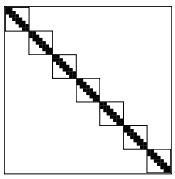


Produit Matrix-Vecteur Diagonal<Banded> :

```
Mat A; Vec X,Y;

for (int i=0; i < A.nrows(); i++){
  Y[i]=A.diagGetElt(i)*X[i];
}
```

Déjà écrit par un développeur

Opérateurs polymorphiques : Y=A*X



Produit Matrix-Vecteur Diagonal<Banded> :

```
for (int i=0; i < A.nrows(); i++){
  Mat::Elt & Ai = A.diagGetElt(i);
  Vec::Elt & Xi = X[i];
  Vec::Elt & Yi = Y[i];
  // Yi=Ai*Xi
  int linf=Ai.linf();
  int lsup=Ai.lsup();
  for (int j=0; j< Ai.nrows(); j++){
    s=0.0;
    int kmin=max(j-linf,0);
    int kmax=min(j+lsup+1,size);
    for (int k=kmin; k< kmax; k++)
      s+=Ai.bandGetElt(j,k)*Xi[k];
    Y[j]=s;
  }
}
```

PAS écrit par un développeur !

Algorithme par bloc

X=B/A ⇔ résoudre AX = B

Algorithme de Gauss-Seidel par bloc :

repeat
  for i = 1 to n do
    $S = B_i$
    for j = 1 to i − 1 do
      $S = S - A_{ij}X_j$
    for j = i + 1 to n do
      $S = S - A_{ij}X_j$
    solve : $A_{ii}X_i = S$
until converged ;

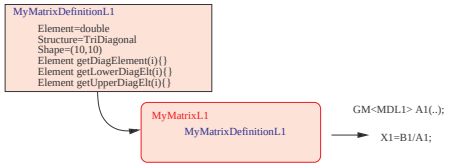
```
do {
 for (int i=0;i<n;i++){
   s=B[i];
   for (int j=0;j<i;j++)
     s-=A(i,j)*X[j];
   for (int j=i+1;j<n;j++)
     s-=A(i,j)*X[j];
   X[i]=s/A(i,i);}
}while(!iter.end(X));
```


Legolas++ : Concept MatrixDefinition

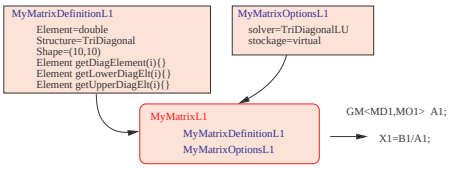
GenericMatrix< MatrixDefinition >
Une classe conforme au concept MatrixDefinition est fournie par l'utilisateur.
Cette classe doit contenir toutes les informations nécessaires pour évaluer les éléments de matrices :

- MatrixStructure ∈ { Diagonal, Banded, Dense... } (Matrix sparsity pattern).
- GetElement (Type des élements).
- Data Une classe qui contient les informations nécessaire à l'évaluation des éléments.
- Un ensemble de fonctions définies par la MatrixStructure choisie.

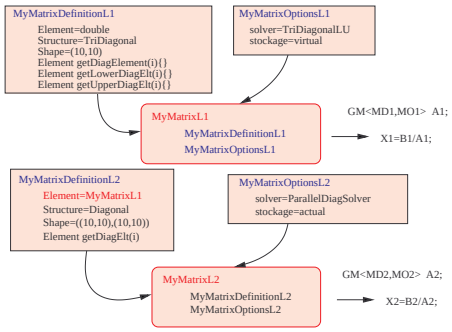
Définitions de matrices et options



Définitions de matrices et options



Définitions de matrices et options



Vecteur Legolas++ : Multi-dim Expression Template

```
for (int i=0 ; i < X.size() ; i++){
  for (int j=0 ; j < X[i].size() ; j++){
    for (int k=0 ; k < X[i][j].size() ; k++){
X+=a*Y+b*Z   ⇔     X[i][j][k]+=a*Y[i][j][k]+b*Z[i][j][k];
    }
  }
}

for (int i=0 ; i < X.size() ; i++){
  for (int j=0 ; j < X[i].size() ; j++){
    for (int k=0 ; k < X[i][j].size() ; k++){
dot(X,a*Y+b*Z)  ⇔     s+=X[i][j][k]*
                  (a*Y[i][j][k]+b*Z[i][j][k]);
    }
  }
}
```


Motivations pour un Legolas++ multi-cible

Temps et accélérations pour un calcul aux valeurs propres SP_N sur un benchmark AIEA avec 2 groupes d'énergie

RT N	SP n	Degrees of Freedom	CPU Seq. T(s)	CPU 12 Threads T(s)	CPU 12 Threads SpUp	GPU T(s)	GPU SpUp
Spatial Grid Size : 68×68×38							
0	1	1.4×10^6	2.24	0.34	6.6	0.19	11.8
0	3	2.8×10^6	7.32	1.02	7.2	0.53	13.8
1	1	$12.\times10^6$	14.8	1.61	9.2	0.67	22.0
1	3	$22.\times10^6$	43.3	5.02	8.6	1.72	25.2
2	1	$38.\times10^6$	50.9	5.25	9.7	2.47	20.6
2	3	$76.\times10^6$	152	16.7	9.0	5.63	27.0
Spatial Grid Size : 289×289×38							
0	1	$25.\times10^6$	49.9	5.79	8.6	1.58	31.6
0	3	$51.\times10^6$	146	17.1	8.6	4.01	36.4
1	1	$200.\times10^6$	330	32.2	9.9	–	–
1	3	$410.\times10^6$	930	102	9.0	–	–
2	1	$690.\times10^6$	1123	109	10.3	–	–

CPU : 2×6-cores Intel X5670 GPU : NVIDIA Tesla C2050

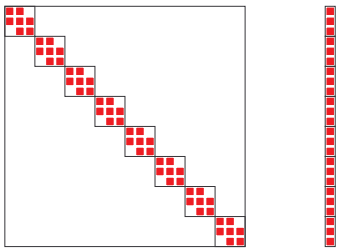
Mais le code devient trop coûteux à maintenir !

(1 version pour CPU ET 1 version pour GPU...)

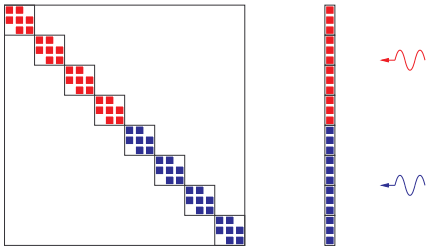
Parallélisme et vectorisation

Placement de donnée optimal sur architecture SIMD (Single Instruction Multiple Data) :
→ CPU (SSE,MMX,AltiVec),GPU :

Matrice Diagonal<TriDiagonal>

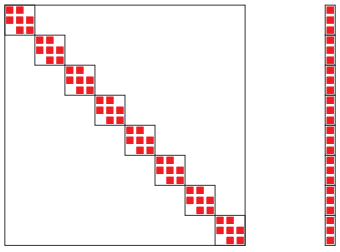


Matrice Diagonal<TriDiagonal>

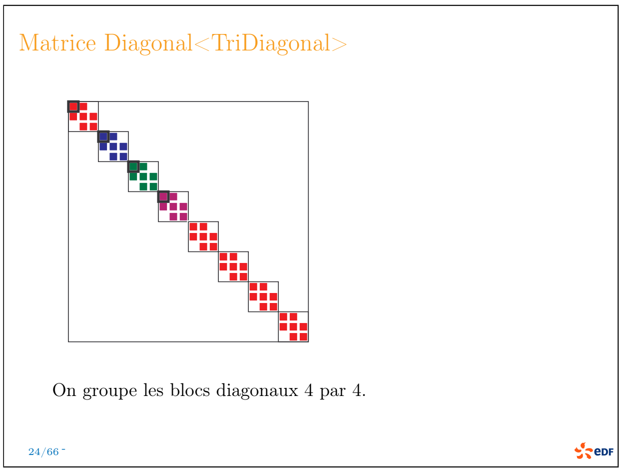


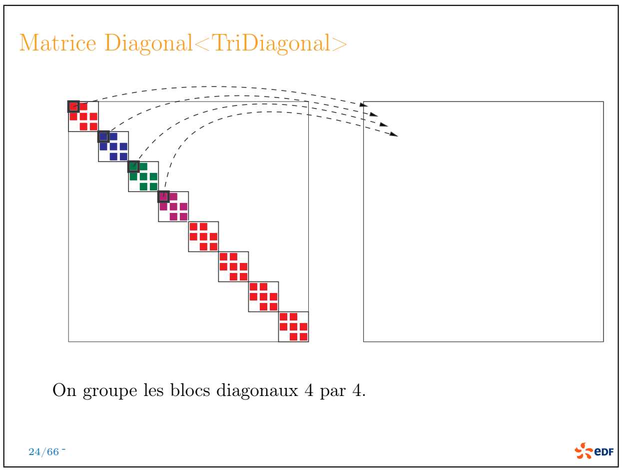
AX=B→Parallélisme à 2 coeurs.

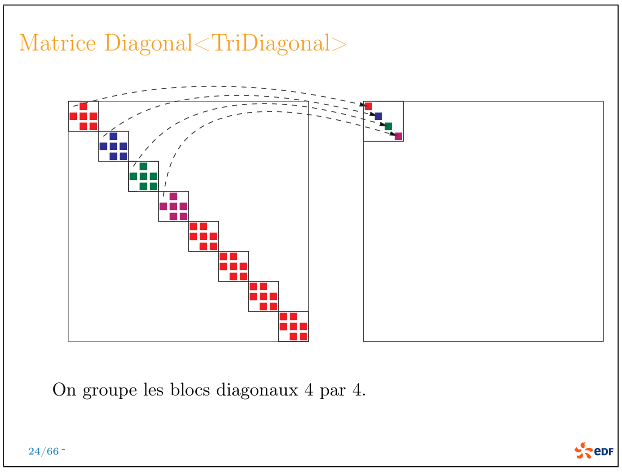
Matrice Diagonal<TriDiagonal>

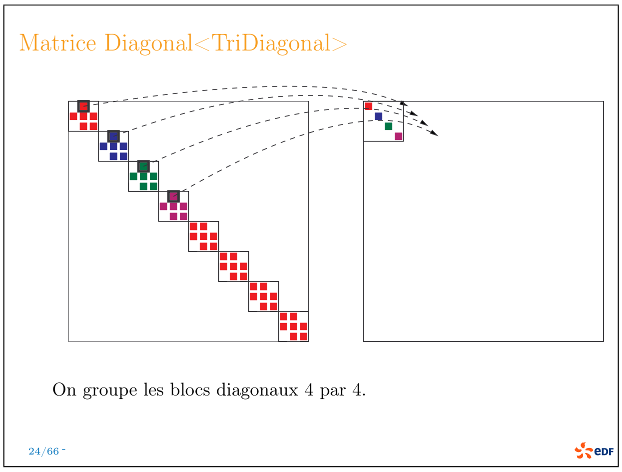


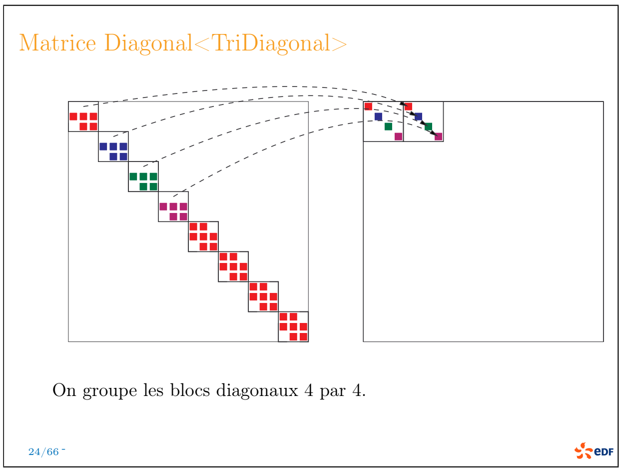
AX=B→Parallélisme SIMD (4 opérations simultanées).

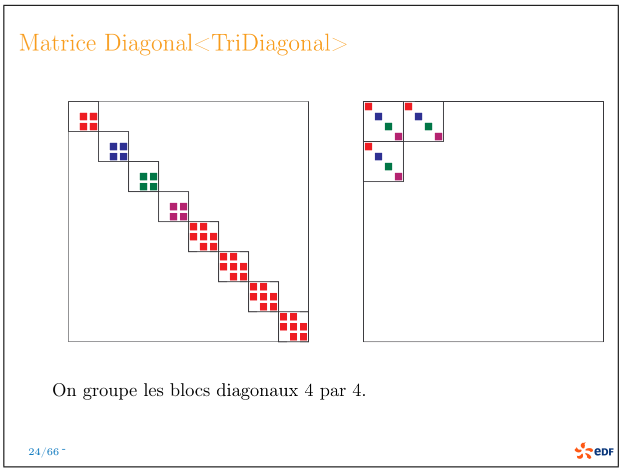

Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66

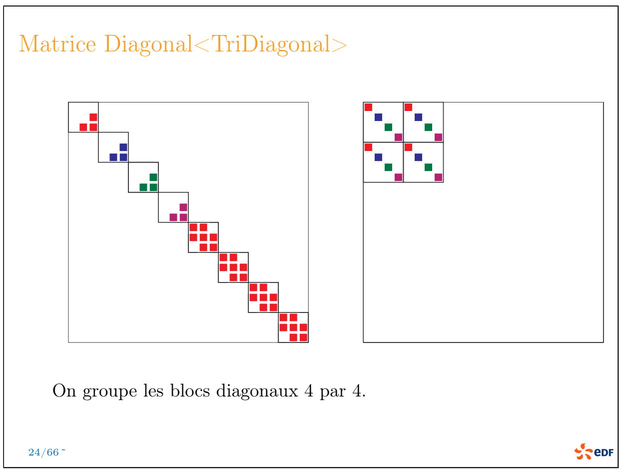

Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66

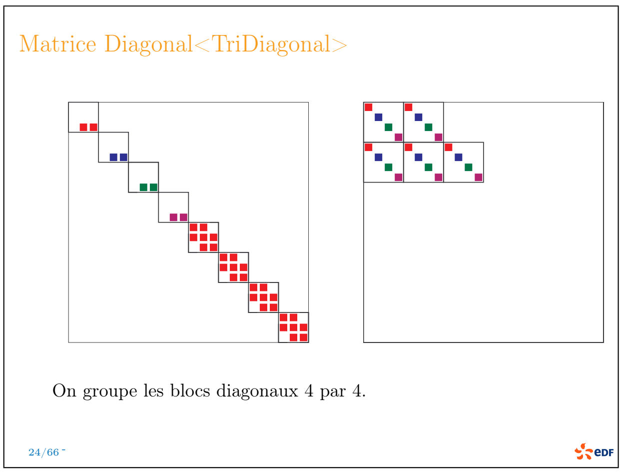

Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66


Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66

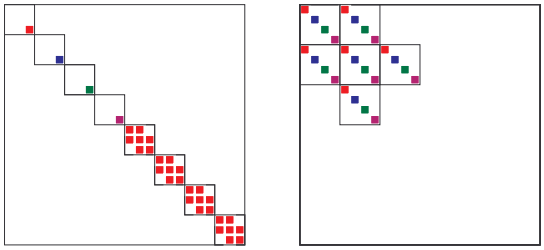

Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66


Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66


Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66

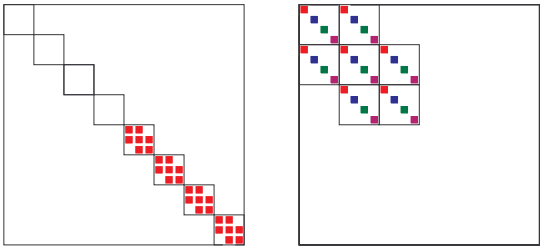

Matrice Diagonal<TriDiagonal>
On groupe les blocs diagonaux 4 par 4.
24/66

Matrice Diagonal<TriDiagonal>



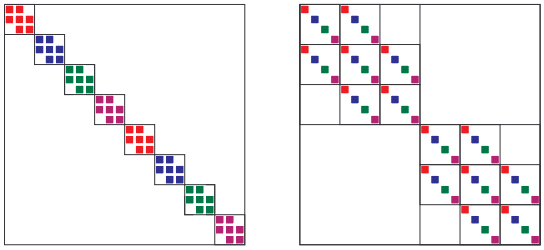
On groupe les blocs diagonaux 4 par 4.

Matrice Diagonal<TriDiagonal>



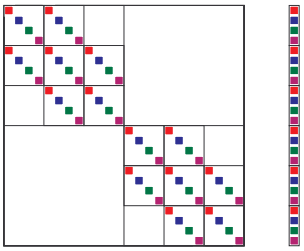
On groupe les blocs diagonaux 4 par 4.

Matrice Diagonal<TriDiagonal>



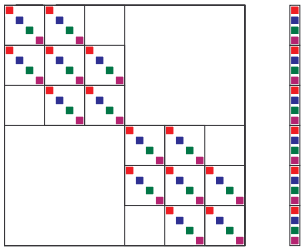
On groupe les blocs diagonaux 4 par 4.

Matrice Diagonal<TriDiagonal>



Diag<TridaDiag>→Diag< Tridiag< Diag > >

Matrice Diagonal<TriDiagonal>



`X[i][j]` $i \in [0, N_x[, j \in [0, N_y[$

→ `X[i][j][k]` $i \in [0, N_x/4[, j \in [0, N_y[, k \in [0, 4[$

Système Diagonal<TriDiagonal>

```
for (int i=0 ; i<Nx ; i++){//boucle parallelisable
  typedef Legolas::MultiVector<RT,1> V1D;
  const V1D & D(D2D[i]),U(U2D[i]),L(L2D[i]),B(B2D[i]);
  V1D & X(X2D_[i]),S(S_);

  RT s(D[0]);
  RT sm1=1.f/s;

  X[0]=B[0]*sm1;
  //Descente
  for (int j=1 ; j < Ny ; j++ ){
    S[j]=U[j-1]*sm1;
    s=D[j]-L[j]*S[j];
    X[j]=B[j]-L[j]*X[j-1];
    sm1=1.f/s;
    X[j]*=sm1;
  }
  //Remontee
  for (int j=(Ny-2) ; j >= 0 ; j-- ){
    X[j]-=S[j+1]*X[j+1];
  }
}
```

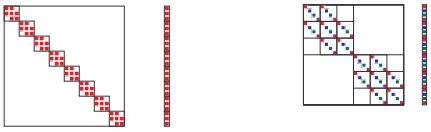

Système Diagonal<TriDiagonal<V4f> >

```
for (int b=0 ; b<Nx/4 ; b++){//boucle parallelisable
  //Interface vers des vecteurs qui renvoient des V4f
  ConstEigenWrap<RT> D(DI[b]),U(UI[b]),L(LI[b]),B(BI[b]);
  EigenWrap<RT> X(XI_[b]),S(S_);

  V4f s(D[0]);
  V4f sm1=s.inverse();

  X[0]=B[0]*sm1;
  //Descente
  for (int j=1 ; j < Ny ; j++ ){
    S[j]=U[j-1]*sm1;
    s=D[j]-L[j]*S[j];
    X[j]=B[j]-L[j]*X[j-1];
    sm1=s.inverse();
    X[j]*=sm1;
  }
  //Remontee
  for (int j=(Ny-2) ; j >= 0 ; j-- ){
    X[j]-=S[j+1]*X[j+1];
  }
}
```


Eigen Wrapper

```
template<>
struct EigenWrap<float>{
  static const int ps=4;

  typedef Legolas::MultiVector<float,1> V1D;

  typedef Eigen::Array<float, ps, 1> V4f;
  typedef Eigen::Map<V4f,Eigen::Aligned>  V4fView;
  typedef Eigen::Map<const V4f,Eigen::Aligned>  ↩
      ConstV4fView;

  V1D & v1D_;
  EigenWrap( V1D & v1D):v1D_(v1D){}

  inline V4fView operator[] (int i){ return &v1D_[i*ps] ;}
  inline ConstV4fView operator[] (int i) const { return ↩
      &v1D_[i*ps] ;}
};
```

Système Diagonal<TriDiagonal<V4f> >

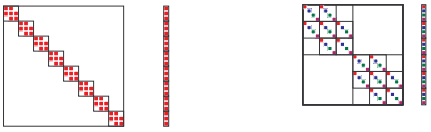


Intel Xeon E6520 2.4 GHz, 2×4 cores

- Nombre de blocs tridiagonaux : $N_x = 600$
- Taille des blocs tridiagonaux : $N_y = 800$

Implémentation	Durée du calcul (s)	GFLOPS	Speed-Up
Séquentielle	0.88	0.60	1.0

Système Diagonal<TriDiagonal<V4f> >



Intel Xeon E6520 2.4 GHz, 2×4 cores

- Nombre de blocs tridiagonaux : $N_x = 600$
- Taille des blocs tridiagonaux : $N_y = 800$

Implémentation	Durée du calcul (s)	GFLOPS	Speed-Up
Séquentielle	0.88	0.60	1.0
Parallèle (TBB)	0.13	4.2	×6.97

Système Diagonal<TriDiagonal<V4f> >

Intel Xeon E6520 2.4 GHz, 2×4 cores

- Nombre de blocs tridiagonaux : $N_x = 600$
- Taille des blocs tridiagonaux : $N_y = 800$

Implémentation	Durée du calcul (s)	GFLOPS	Speed-Up
Séquentielle	0.88	0.60	1.0
Parallèle (TBB)	0.13	4.2	×6.97
SIMD (Eigen)	0.20	2.6	×4.31

Système Diagonal<TriDiagonal<V4f> >

Intel Xeon E6520 2.4 GHz, 2×4 cores

- Nombre de blocs tridiagonaux : $N_x = 600$
- Taille des blocs tridiagonaux : $N_y = 800$

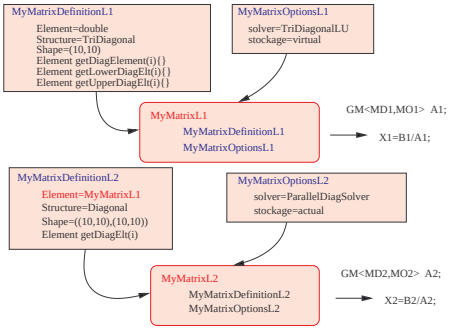
Implémentation	Durée du calcul (s)	GFLOPS	Speed-Up
Séquentielle	0.88	0.60	1.0
Parallèle (TBB)	0.13	4.2	×6.97
SIMD (Eigen)	0.20	2.6	×4.31
TBB+Eigen	0.04	13.3	×22.1

Legolas++ multi-cible (CPU,GPU)

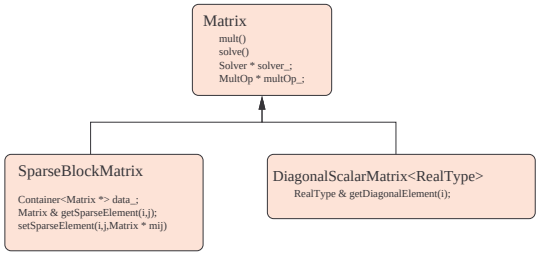
Thèse de Wilfried Kirschenmann soutenue en 2012 :
Vers des noyaux de calcul intensif pérennes
Parallélisation et vectorisation automatique sur CPU et GPU pour une classe de problèmes Legolas++ :
Diagonal< Matrices de structure identique arbitrairement compliquées impliquant des algorithmes non divergents >

3. Legolas++ : V0.2 Polymorphisme hybride

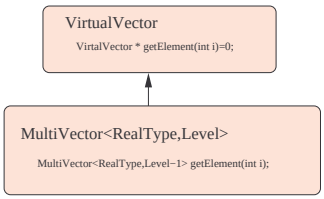
Polymorphisme statique



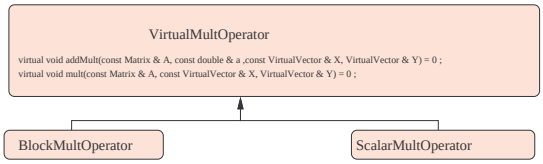
Polymorphisme hybride : classe `Matrix`



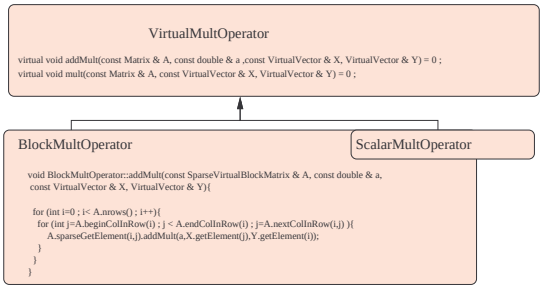
Polymorphisme hybride : classe `VirtualVector`



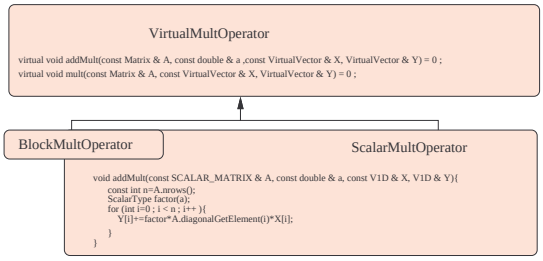
Polymorphisme hybride : classe `VirtualMultOperator`



Polymorphisme hybride : classe `VirtualMultOperator`



Polymorphisme hybride : classe `VirtualMultOperator`



Solveur SN fondé sur Legolas++ V0.2

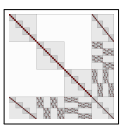
Solveur SN (ordonnées discrètes) de transport.

- Système $AX = \lambda BX$ avec 10^{12} inconnues sur un noeud à 32 coeurs SIMD en 10H.
- Parallélisme à 2 niveaux (multi-core+SIMD)
- $\simeq$50% de la crête du noeud HPC !

4. Résumé et perspectives

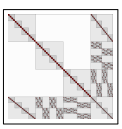
Résumé 1/2

- Legolas++
  - Matrices multi-niveaux
  - Vecteurs multi-niveaux
  - Algorithmes par bloc (récursifs)



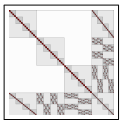
Résumé 1/2

- Legolas++
  - Matrices multi-niveaux
  - Vecteurs multi-niveaux
  - Algorithmes par bloc (récursifs)
- Polymorphisme $\rightarrow$ description de matrices creuses très structurées à partir d'un petit ensemble de structures de base.



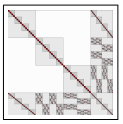
Résumé 1/2

- Legolas++
  - Matrices multi-niveaux
  - Vecteurs multi-niveaux
  - Algorithmes par bloc (récursifs)
- Polymorphisme → description de matrices creuses très structurées à partir d'un petit ensemble de structures de base.
- Fourniture d'algorithmes parallèles par Legolas++.



Résumé 1/2

- Legolas++
  - Matrices multi-niveaux
  - Vecteurs multi-niveaux
  - Algorithmes par bloc (récursifs)
- Polymorphisme → description de matrices creuses très structurées à partir d'un petit ensemble de structures de base.
- Fourniture d'algorithmes parallèles par Legolas++.
- Legolas++ multicible (prototype) (SIMD,GPU)



Résumé 2/2

Enseignements de la conception de Legolas++ (tjrs en cours)

- Le polymorphisme statique (template) est efficace mais lourd→ polymorphisme hybride (dynamique+statique).

Résumé 2/2

Enseignements de la conception de Legolas++ (tjrs en cours)

- Le polymorphisme statique (template) est efficace mais lourd→ polymorphisme hybride (dynamique+statique).
- Les solveurs HPC peuvent avoir des états internes → objets de traitement.

Résumé 2/2

Enseignements de la conception de Legolas++ (tjrs en cours)

- Le polymorphisme statique (template) est efficace mais lourd→ polymorphisme hybride (dynamique+statique).
- Les solveurs HPC peuvent avoir des états internes → objets de traitement.
- Le SIMD pose des problèmes architecturaux mais est difficilement évitable en HPC.

Résumé 2/2

Enseignements de la conception de Legolas++ (tjrs en cours)

- Le polymorphisme statique (template) est efficace mais lourd→ polymorphisme hybride (dynamique+statique).
- Les solveurs HPC peuvent avoir des états internes → objets de traitement.
- Le SIMD pose des problèmes architecturaux mais est difficilement évitable en HPC.
- Legolas++ s'appuie sur d'autres bibliothèques génériques TBB (multithread), Eigen (SIMD).

Pespectives

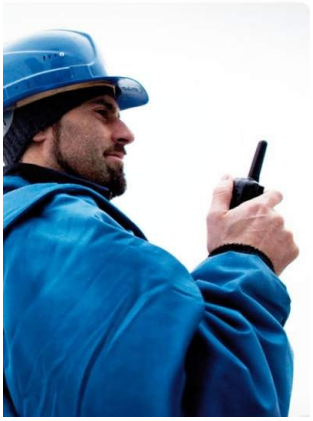
- Intégration des travaux de thèse W. Kirschenmann pour Legolas++ multi-cible.

Pespectives

- Intégration des travaux de thèse W. Kirschenmann pour Legolas++ multi-cible.
- Utilisation des travaux de thèse de B. Lizé (EADS) sur l'utilisation de StarPU pour les H-Matrix.

Fin

Merci pour votre attention !
Vos question sont les bienvenues ;)



5. Annexes

MatrixDefinition example

```
class MyMatrixDefinition :
            public Legolas::MatrixDefinition<double>{

  typedef Legolas::Diagonal  MatrixStructure;
  typedef double             GetElement;
  typedef FluxData<double>   Data;

  static GetElement diagonalGetElement(int i, const ↩
     Data & data){
    return data.diagonalGetElement(i);
  }

};
```


Example of use

```
typedef GenericMatrix<MyMatrixDefinition> ↩
   MyMatrixInterface;
MyMatrixDefinition::Data data(10,2.5);

MyMatrixInterface::Matrix  A(data);

// Available methods

 A.diagonalGetElements(i);
 A.bandedGetElements(i); A.lsup() ; A.linf();
 A.nRows(); A.nCols();
 A(i,j) ;

 X=B/A ; // solves AX=B;
 Y=A*X ;
```


Matrix options

```
 InputMatrixOptions<Actual> MatrixOptions;
// typedef GenericMatric<MyMatrixDefinition> ↩
   MyMatrixInterface
 typedef ↩
    GenericMatric<MyMatrixDefinition,MatrixOptions> ↩
    MyMatrixInterface;

 MyMatrixDefinition::Data data(10,2.5);
 MyMatrixInterface::Matrix  A(data);

 // Available methods

  A.diagonalGetElements(i);
  A.bandedGetElements(i); A.lsup() ; A.linf();
  A.nRows(); A.nCols();
  A(i,j) ;

  X=B/A ; // solves AX=B;
  Y=A*X ;
```


Matrix Options

```
typedef InputMatrixOptions<Actual,
                           TriDiagonalLU,
                           SparseMatrixVectorProduct,
                           STLVectorContainer,
                           TriDiagonalMatrixContainer,
                           TriDiagonal> MatrixOptions;

// typedef GenericMatric<MyMatrixDefinition> ↩
   MyMatrixInterface
typedef ↩
   GenericMatric<MyMatrixDefinition,MatrixOptions> ↩
   MyMatrixInterface;

MyMatrixDefinition::Data data(10,2.5);
MyMatrixInterface::Matrix  A(data);

X=B/A ; // solves AX=B;
Y=A*X ;
```

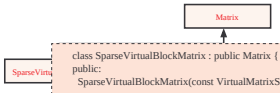

Architecture

Architecture


```
class Matrix{
public:
 Matrix(const VirtualMatrixShape & virtualMatrixShape);

 void solve(const VirtualVector & B, VirtualVector & X) const;
 void mult(const VirtualVector & X, VirtualVector & Y) const;

 int nrows( void ) const ; int ncols( void ) const ;

 void displayLatex(std::string filename, int flag=0) const ;
 virtual void iterateOverElements(Legolas::MatrixStream & matrixStream) const ;

 void setVirtualSolverPtr(VirtualSolver * solverPtr) ;
 void setVirtualMultOperatorPtr(VirtualMultOperator * multOperatorPtr) ;
private:
 VirtualMatrixShape * virtualMatrixShapePtr_;
 Matrix * father_;
 VirtualSolver * solverPtr_;
 VirtualMultOperator * multOperatorPtr_;
};
```

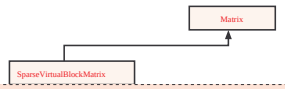
Architecture

Architecture




```
class SparseVirtualBlockMatrix : public Matrix {
public:
  SparseVirtualBlockMatrix(const VirtualMatrixShape & virtualMatrixShape) ;

  virtual void setSolverPtr(SparseVirtualSolver * sparseVirtualSolverPtr) ;
  virtual void setMultOperatorPtr(SparseVirtualMultOperator * sparseVirtualMultOperatorPtr) ;

  virtual void iterateOverElements(Legolas::MatrixStream & matrixStream) const ;


  virtual const Matrix & sparseGetElement(int i, int j) const =0;
  virtual Matrix & sparseGetElement(int i, int j) =0;

  virtual int beginRow( void ) const =0;
  virtual int endRow( void ) const =0;

  virtual int beginColInRow( int i ) const =0;
  virtual int endColInRow( int i ) const =0;

  virtual int nextColInRow(int i, int j) const =0;
};
```


Architecture

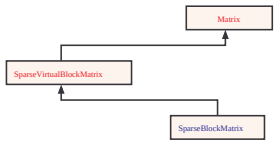



```
SparseVirtualBlockMatrix::SparseVirtualBlockMatrix(const VirtualMatrixShape & vms):Matrix(vms){
  this->setSolverPtr( new SparseGaussSeidelSolver());
  this->setMultOperatorPtr( new SparseMultOperator());
}
```


Architecture


```
void SparseMultOperator::addMult(const SparseVirtualBlockMatrix & A, const double & a,
 const VirtualVector & X, VirtualVector & Y){
  for (int i=0 ; i< A.nrows() ; i++){
    for (int j=A.beginColInRow(i) ; j < A.endColInRow(i) ; j=A.nextColInRow(i,j) ){
       A.sparseGetElement(i,j).addMult(a,X.getElement(j),Y.getElement(i));
    }
  }
}
```

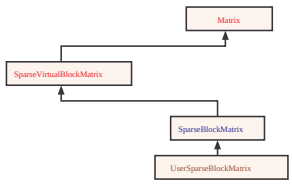

Architecture



Architecture


```
class SparseBlockMatrix : public SparseVirtualBlockMatrix {
private:
  typedef std::map<int,Matrix*>   Row;
  typedef std::vector< Row >      SparseData;
  SparseData sparseData_;
public:

  SparseBlockMatrix(const VirtualMatrixShape & virtualMatrixShape);

  void setSparseElementPtr(int i, int j, Matrix * matrixPtr);
  virtual const Matrix & sparseGetElement(int i, int j) const ;
  virtual Matrix & sparseGetElement(int i, int j) ;

  virtual int beginRow( void ) const ;
  virtual int endRow( void ) const ;
  virtual int beginColInRow( int i ) const ;
  virtual int endColInRow( int i ) const ;
  virtual int nextColInRow(int i, int j) const ;
};
```

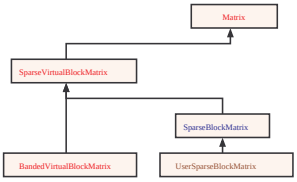

Architecture



Architecture

```
#include "UserSparseBlockMatrix.hxx"
#include "UserDiagonalMatrix.hxx"

UserSparseBlockMatrix::UserSparseBlockMatrix(const Legolas::MatrixShape<2> ms):SparseBlockMatrix(ms){
  for (int i=0 ; i<this->nrows() ; i++){
    for (int j=0 ; j<this->ncols() ; j++){
      if (i==j) this->setSparseElementPtr(i,j,new UserDiagonalMatrix<double>(ms.getSubMatrixShape(i,j),4.0));
      if (i==(j-1)) this->setSparseElementPtr(i,j,new UserDiagonalMatrix<double>(ms.getSubMatrixShape(i,j),-1.0));
      if (i==(j+1)) this->setSparseElementPtr(i,j,new UserDiagonalMatrix<double>(ms.getSubMatrixShape(i,j),-1.0));
    }
  }
}
```

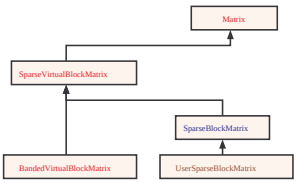
Architecture



Architecture

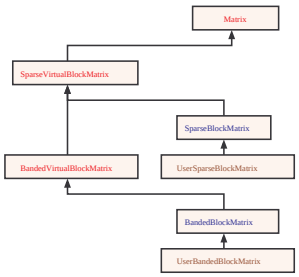
```
class BandedVirtualBlockMatrix : public SparseVirtualBlockMatrix {
public:
 BandedVirtualBlockMatrix(const VirtualMatrixShape & virtualMatrixShape);
 virtual void setSolverPtr(BandedVirtualSolver * bandedVirtualSolverPtr);
 virtual void setMultOperatorPtr(BandedVirtualMultOperator * bandedVirtualMultOperatorPtr);
 //sparse API implementation
 virtual const Matrix & sparseGetElement(int i, int j) const ;
 virtual Matrix & sparseGetElement(int i, int j) ;
 virtual int beginRow( void ) const ;
 virtual int endRow( void ) const ;
 virtual int beginColInRow( int i ) const ;
 virtual int endColInRow( int i ) const ;
 virtual int nextColInRow(int i, int j) const ;
 //Banded API definition
 virtual const Matrix & bandedGetElement(int i, int j) const =0;
 virtual Matrix & bandedGetElement(int i, int j) =0;
 virtual int linf( void ) const =0;
 virtual int lsup( void ) const =0;  };
```


Architecture

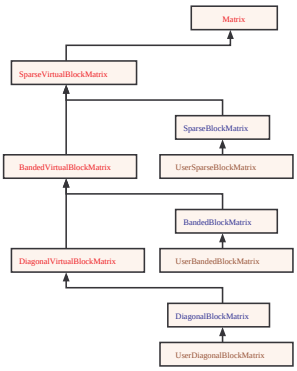



```
int BandedVirtualBlockMatrix::beginRow( void ) const { return 0;}
int BandedVirtualBlockMatrix::endRow( void ) const { return this->nrows();}
int BandedVirtualBlockMatrix::beginColInRow( int i ) const { return std::max(i-this->linf(),0);}
int BandedVirtualBlockMatrix::endColInRow( int i ) const { return std::min(i+this->lsup()+1,this->nrows());}
int BandedVirtualBlockMatrix::nextColInRow(int i, int j) const { return j+1;}
```

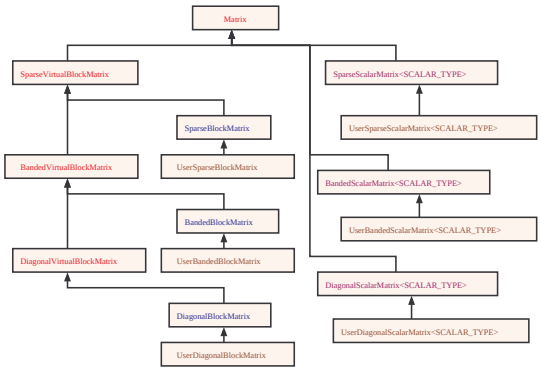

Architecture



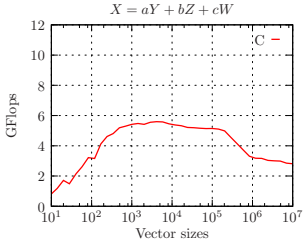
Architecture



Architecture



Intensité arithmétique, granularité et polymorphisme

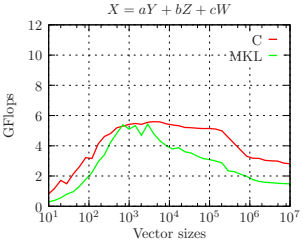


X, Y, Z, W : tableaux de `float`

```
for (int i=0; i<N; i++){
  X[i]=a*Y[i]+
       b*Z[i]+
       c*W[i];
}
```

Exécution séquentielle.
4 traversées de vecteurs.
Intel Xeon E6520 2.4 GHz
gcc 4.7.2 -O3 -funrool-loops -msse4.1

Intensité arithmétique, granularité et polymorphisme

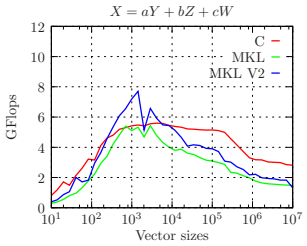


X, Y, Z, W : tableaux de `float`

```
cblas_sscal(N,0.0,X,1);
cblas_saxpy(N,a,Y,1,X,1);
cblas_saxpy(N,b,Z,1,X,1);
cblas_saxpy(N,c,W,1,X,1);
```

Exécution séquentielle.
7 traversées de vecteurs.
Intel Xeon E6520 2.4 GHz
Intel MKL 11
MKL_NUM_THREADS=1

Intensité arithmétique, granularité et polymorphisme

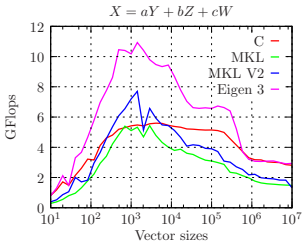


X, Y, Z, W : tableaux de `float`

```
cblas_saxpby(N,a,Y,1,0,X,1);
cblas_saxpy(N,b,Z,1,X,1);
cblas_saxpy(N,c,W,1,X,1);
```

Exécution séquentielle.
6 traversées de vecteurs.
Intel Xeon E6520 2.4 GHz
Intel MKL 11
MKL_NUM_THREADS=1

Intensité arithmétique, granularité et polymorphisme



X, Y, Z, W : tableaux de `float`

```
Eigen::VectorXf X,Y,Z,W;

X=a*Y+b*Z+c*W;
```

Exécution séquentielle.
4 traversées de vecteurs.
Intel Xeon E6520 2.4 GHz
Eigen 3 ; gcc 4.7.2 -O3 -funrool-loops

Intensité arithmétique, granularité et polymorphisme

On peut definir l'intensité arithmétique (NVidia) ou la densité de calcul (Intel) par

$$I_a = \frac{\text{Nombre d'opérations arithmétiques}}{\text{Nombre d'accès à la RAM}}$$

Si $I_a >> 1$ on peut appeler plusieurs fonctions, sinon il faut une grande expressivité pour la bibliothèque.
Si la granularité de la fonction est grande le surcoût d'un appel de fonction (4 cycles) est amorti, sinon il faut une transformation statique par le compilateur pour inliner la fonction.
→ On choisit le C++ pour implémenter notre outil pour accéder au polymorphisme statique.

2.9 Florent Duchaine (Cerfacs)

OpenPALM, an open source code coupler for massively parallel multi-physics/multi-components applications and dynamic algorithms

OpenPALM, an open source code coupler for massively parallel multi-physics/multi-components applications and dynamic algorithms

Florent Duchaine
Cerfacs

Abstract

After many years of development and regular use, the PALM (CERFACS) and CWIPI (ONERA) libraries are distributed since January 2011 under the Open Source project named OpenPALM.

Since 1996 CERFACS has been developing the PALM parallel coupler, which is currently used for more than 50 research and industrial projects ranging from operational data assimilation to multi-physics modeling, from climate change impact assessment to fluid and structure interaction. It can be defined as a dynamic coupler for its ability to deal with situations where the component execution scheduling and the data exchange patterns cannot be entirely defined before execution.

Based on the EDF libraries bft and fvm, CWIPI aims at providing a fully parallel communication layer for mesh based coupling. CWIPI functionalities involve the construction of the communication graph between distributed geometric interfaces, interpolation on non coincident meshes and exchange of coupling fields for massively parallel applications.

The presentation provides some highlights on the design of OpenPALM and on the main implementation choices and complementarities of the two libraries and a brief description of some representative applications.

O-Palm

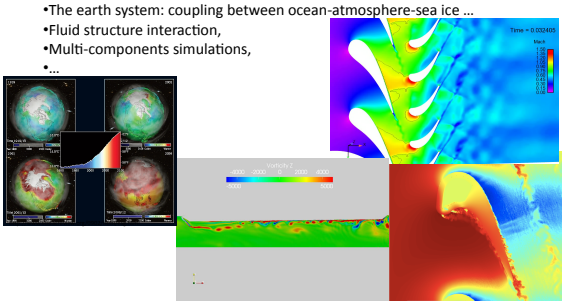
an open source code coupler for massively parallel multi-physics/multi-components applications and dynamic algorithms

Co-development CERFACS – ONERA
http://www.cerfacs.fr/globc/PALM_WEB/

T. Morel, F. Duchaine, A. Thévenin, M. Kirmse A. Piacientini (CERFACS)
E. Quémerais, A. Refloch (ONERA)

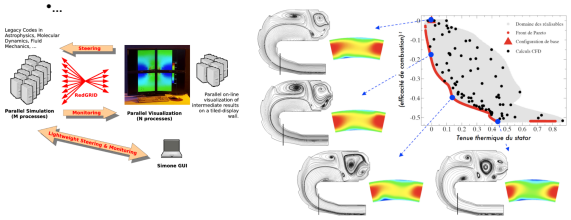
Why coupling codes?

➔To treat a global system
- •The earth system: coupling between ocean-atmosphere-sea ice …
- •Fluid structure interaction,
- •Multi-components simulations,
- •…



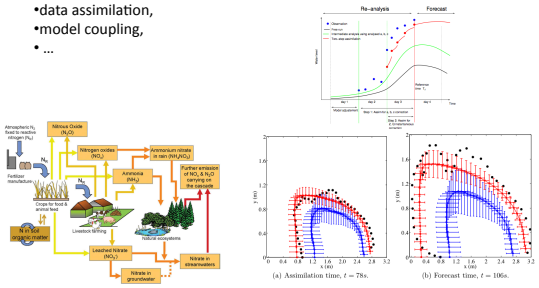
Why coupling codes?

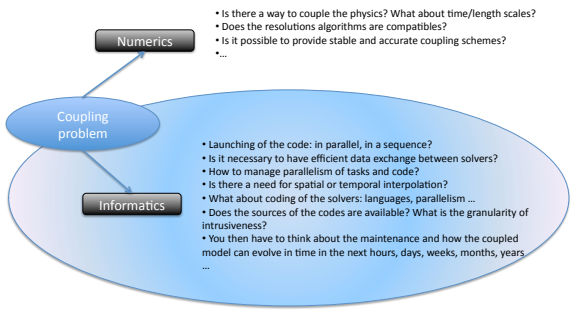
➔ To construct applications around existing codes
- •On the fly post-processing,
- •Optimization loop around a code,
- •Ensemble simulations,
- •…



Why coupling codes?

➔ To construct modular applications by assembling elementary components:
- •data assimilation,
- •model coupling,
- • …





Implementation

➔ Rewrite a new unified solver
➔ Fusion of existing codes

prog1 & prog2

```
Program prog1
…
Call sub_prog2(in, out)
…
end
```

~~Program prog2~~

```
Subroutine sub_prog2(in,out)
…
end
```

+ very efficient (direct access in memory for the transfer of data)

- integration problems (common, files, compilations tags, languages …),
- no flexibility (maintenance, evolution …),
- potential memory problems,
- parallelism of the master code imposed to the application

Assembling and not coupling !!!

Implementation

➔ Use a communication protocol (PVM, MPI, CORBA, unix pipe, socket, files …)

Prog1 send/recv **Prog2**

```
Program prog1
…
Call XXX_send(prog2, data)
end
```

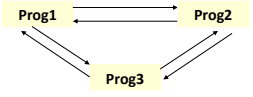
```
Program prog2
…
Call XXX_recv(prog1, data)
end
```

+ can be very efficient depending on the protocol
+ preserve existing codes

- non generic coupling: the instrumentation is specific to the application,
- need expertise in parallel computing,
- non flexible (parallelism, interpolation …),
- Potential portability troubles,
- very complex when more than 2 codes are involved.

Implementation

➔ Use a code coupler, ie a library of functionalities that facilitate ease data exchange between existing codes and execution schedule



➔ OASIS, OpenPALM, Salomé, MpCCI, CESM, ESMF, MCT, CPL7, YAC …

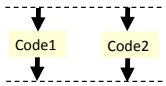
Commitments:
- Easy set-up: *non intrusive interfaces*
- Efficiency: *no loss in performances*
- Portability: *standard technical solutions*
- Flexibility: *test different configurations with a few changes*
- Genericity: *reuse components in other couplings*

ONERA O-Palm

Code coupling

Classical couplers

Static coupling



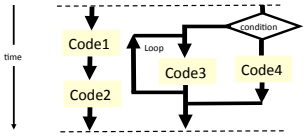
Both codes are started at the beginning of the simulation and exit together at the end of the application

Example: Ocean/Atmosphere coupling in climate modeling

MPI-1 MPMD

OpenPALM coupler

Dynamic coupling



Loops and conditional executions

Complex algorithms can be easily end efficiently implemented

Dynamic resources management (processors and memory)

MPI-2 MPMD with dynamic spawn

ONERA O-Palm

OpenPALM

Palm & CWIPI

Open Source LGPL

O-Palm

ONERA O-Palm

OpenPALM

O-Palm

PrePALM Graphic User Interface (G.U.I.):
- →Definition of the coupling scheme
- →Monitoring and Post-processing

PALM Library:
- →Drives the [parallel/sequential] execution of the components
- →Manages the data exchanges

CWIPI Library:
- →Manages fully parallel data exchanges based on distributed mesh definition
- →Interpolation between non coincident meshes

ONERA O-Palm

PALM characteristics

- 1996 : Mercator Project : Operational Oceanography
  - Data assimilation with a high resolution model
- Project Leader: Philippe Courtier, DGA MF
  - Need for a modular and efficient tool to compose assimilation algorithms → component coupling approach
- 2011 : OpenPALM, Open source - collaboration ONERA

ONERA O-Palm

PALM characteristics

How to describe complex algorithms?

IHM PrePALM :
- →Definition of the coupling scheme
- →Monitoring and Post-processing

PrePALM runs on a standard PC or on clusters, allows to define coupling algorithms and prepare compilation process

PALM library:
- →Parallel execution of components algorithms, dynamic launching by the PALM driver
- →Management of data exchanges, PALM primitives inside the codes

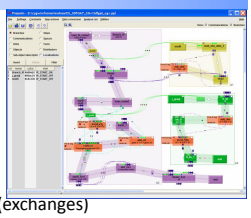
The PALM library is compiled on the different machines where the applications are executed

ONERA O-Palm

PALM characteristics

How to describe complex algorithms?

A Graphic User Interface for:
- → Representing the components
- → Handling the parallelism
- → Describing loops and conditions
- → Describing the communications (exchanges)



With all the benefits of a G.U.I. as, for instance:
- → Coherency checks on the exchanged data
- → Pictorial flow chart representation of the algorithm
- → Performance analysis, debugging and run-time monitoring

ONERA O-Palm

PALM characteristics

- Unit: computational component
  - Code
  - Module of a code
  - Data loaders
  - Pre-defined units (F77/F90 subroutines, C or C++ functions, Java, Python, Matlab and so on through F90 or C/C++ interfaces)

- Parallel unit

ONERA O-Palm

PALM characteristics

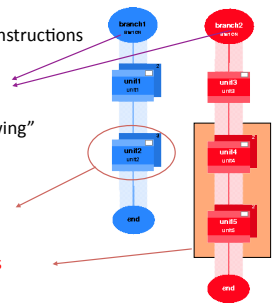
Branch: sequence of units and instructions (algorithm)

Task parallelism: branches

Parallel computing "simply drawing"

Distributed components: units

Components assembling: blocks



ONERA O-Palm

PALM characteristics

"End point" communication paradigm

→ NO explicit indication of the receiving units on the producer side, NOR of the producer on the receivers side

In the sources of the units: potential sending or reception

Announce data production: PROVIDER

```
CALL PALM_Put (…)
```

Register a data request: CLIENT

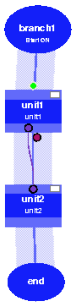
```
CALL PALM_Get (…)
```


ONERA O-Palm

PALM characteristics

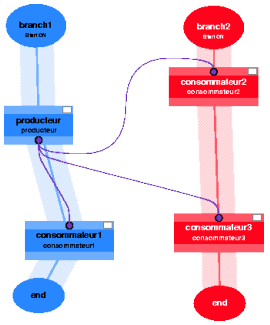
- Object: data chunk produced or received by a unit
- Communication: exchange of an object between two units

N.B.: "Packed data" for coherent data batches can simplify the data flow representation



ONERA O-Palm

PALM characteristics



Actual communications are described by connecting the plugs in the graphic interface

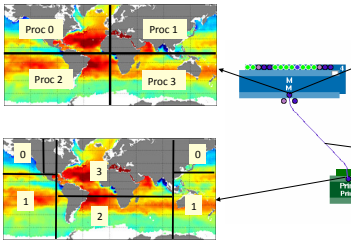
Handshaking between clients and providers

ONERA O-Palm

PALM characteristics

In parallel units ID cards, you declare only global objects

A distributor let you define how the global object is locally distributed among processors



Every processor issues a PALM_Put for its own domain

A distributor is associated to the source object

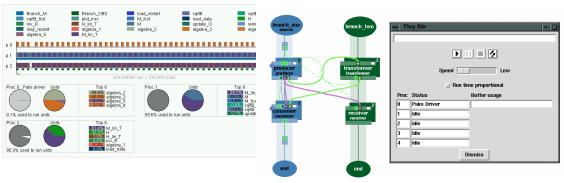
Another distributor is associated to the target object

With the communication PALM remaps the objects from the source to the target distribution

ONERA O-Palm

PALM characteristics

- A PALM output file per branch with different verbosity levels for different classes of messages
- A user defined debugging function, called on the PALM_Put and PALM_Get sides of an active communication to check the coherency of the objects
- A performance analyzer accounting for the elapsed and the CPU time
- A graphic post-mortem replay or run-time monitoring of the application

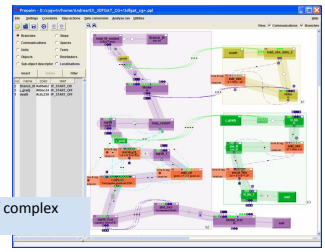


ONERA O-Palm

PALM characteristics

- A dynamic code coupler
  - ✓ Designed to do very complex algorithms,
  - ✓ With a high level of flexibility,
  - ✓ Make communications between parallel codes running at the same time or not

⇒ Code coupling,
⇒ Data assimilation,
⇒ Optimization environment,
⇒ Post-processing,
⇒ UQ,
⇒ …

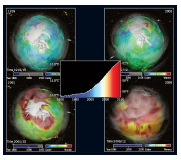


PALM facilitates the prototyping of complex applications

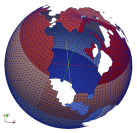
ONERA O-Palm

CWIPI Characteristics

A coupled simulation step by step, climate modeling



- Launch the parallel codes Ocean and Atmosphere
- Domain decomposition in the codes
- (*) Sharing of the parts to couple: communication graph between the codes



Atmosphere solver: n_a cores on N_a are concerned by the coupled interface

Ocean solver: n_o cores on N_o are concerned by the coupled interface

- (*) Temporal loop in the solvers: exchange of physical quantities

(*) can either be done by concentrating information on one process or in a distributed way

ONERA O-Palm

CWIPI Characteristics

HPC issues for code coupling with mesh partitioning

- data transfer between massively parallel codes
- the data can be light compared to the model (surface exchanges) or heavy (volume exchanges)
- interpolations from one model to the other(s)

Constraints

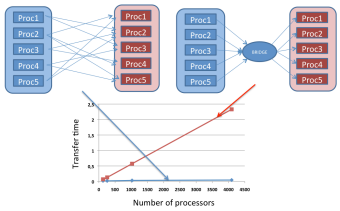
⇒ efficiency (execution time and memory consumption) of the processing strategy
⇒ scalability when the model and/or the number of core increase
⇒ precision of the interpolation results
⇒ easy to use for different types of discretizations

Two main challenges:

- information routing between parallel codes
- interpolation in parallel

CWIPI Characteristics

Direct communications are mandatory for HPC



To establish the efficiently the routing between the cores, it is necessary to consider:

- CPU time of the solutions (weak limit)
- number of communications (weak limit)
- memory allocation (strong limit)

⇒ avoid a master / slaves scheme with information merging

CWIPI characteristics

CWIPI : Library for data exchange with interpolation in parallel

- Management of the coupling between *n* parallel codes,
- Well adapted to HPC,
- Direct data exchange with interpolation of fields defined an interfaced discretized by different partitioned meshes (1D, 2D, 3D),
- Takes into account any type of elements,
- Simplified interface totally transparent to parallelism,
- Visualization of exchanges fields,
- Tunable by the user with user functions,
- Interface to C/C++, Fortran and Python

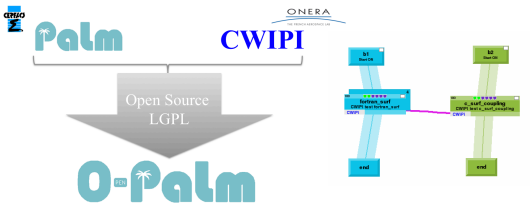
Developed at ONERA (DSNA/ELCI) since 2009

Now under LGPL within OpenPALM

CWIPI uses the FVM library (License LGPL, EDF R&D)

Gain in genericity of instrumentation thanks to PrePALM

OpenPALM

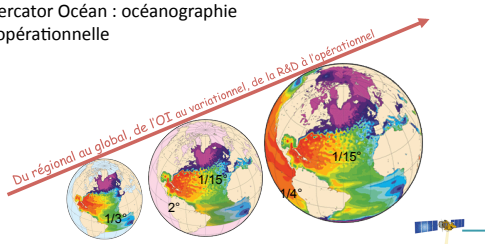


➔ Definition of complex coupling algorithms with PrePALM,
➔ Access to the PALM and CWIPI libraries

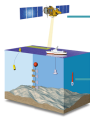
➔ A lot of users in France / Europe in labs and industries
➔ Some examples of use at CERFACS ...

OpenPALM - Applications

Mercator Océan : océanographie opérationnelle



Toutes les chaînes d'assimilation ont été développées avec PALM
Les mêmes méthodes s'appliquent aux différentes résolutions
Le passage d'une version d'OPA à une autre n'impacte pas l'assimilation
L'évolution de PALM (Research -> MP) permet d'optimiser la mémoire de l'application

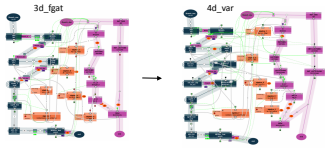


OpenPALM - Applications

Assimilation de DOnnées dans les MOdèles de Chimie Atmosphérique

Fédération des principaux acteurs de la communauté scientifique
SA, LA, LSCE, LPCE, LMD , IPSL, CERFACS, L3AB, CNRM, ACRI, CNES

Palm
-Échanger facilement des composants : modèles, opérateurs d'observation
Échanger des méthodes



L'algorithme d'assimilation est simplement redéfinit dans l'interface graphique PrePALM, aucune ligne n'est modifiée dans le code source des unités.

On ajoute certaines unités comme le modèle adjoint et le linéaire tangent

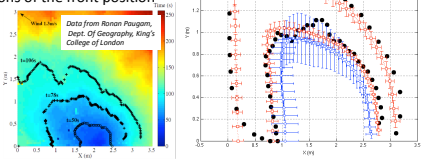
OpenPALM - Applications

Propagation model of the fire front

Rate of spread Γ
Burnt vegetation
Front
Unburnt vegetation

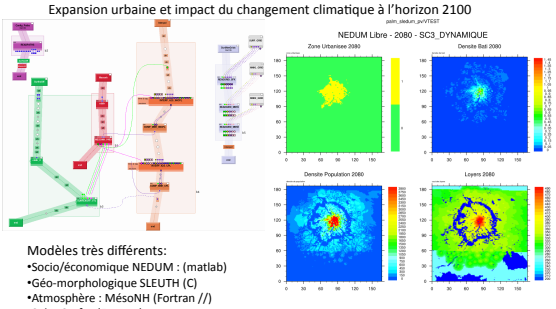
Correct model parameters (vegetation, wind) to correct the ROS assimilating front position observations with an Ensemble approach using Parasol-OpenPALM (Rochoux et al. 2012)

Airborne observations of the front position



OpenPALM - Applications

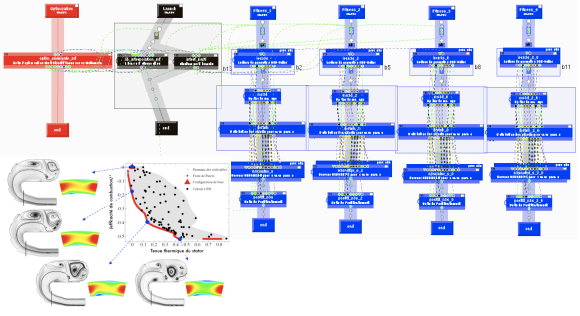
Projet RTRA/STAE ACCLIMAT : plateforme de modélisation de la ville Toulouse
Expansion urbaine et impact du changement climatique à l'horizon 2100



Modèles très différents:
•Socio/économique NEDUM : (matlab)
•Géo-morphologique SLEUTH (C)
•Atmosphère : MésoNH (Fortran //)
•Sols : Surfex (Fortran)
Interpolation géophysique

OpenPALM - Applications

Optimization loop



OpenPALM - Applications

PALM - Parasol

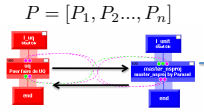
$P = [P_1, P_2..., P_n]$

$R = [R_1, R_2..., R_n]$

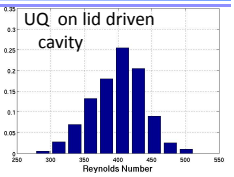
$R_1 = F(P_1)$

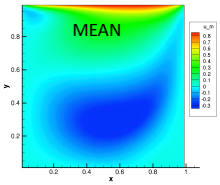
$R_2 = F(P_2)$

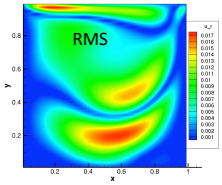
$R_n = F(P_n)$



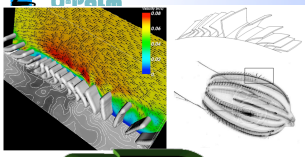
UQ on lid driven cavity

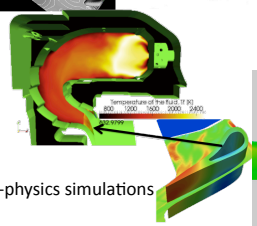


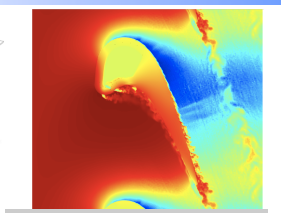


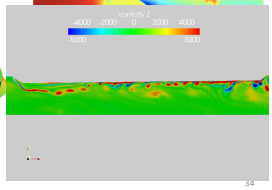


OpenPALM - Applications





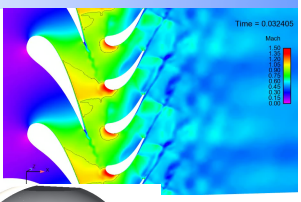


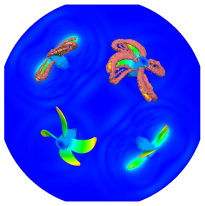


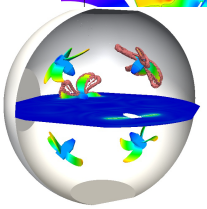
Multi-physics simulations

OpenPALM - Applications

Multi-components simulations







OpenPALM

- A constant evolution of the code in direct link with user needs
  - ➔ Parasol
  - ➔ Python interface

  - ➔ Connection to commercial codes or heterogeneous coupling via IP
  - ➔ …
- New uses as people are aware of the soft
- Tested over 12,000 cores for conjugate heat transfer applications

OpenPALM

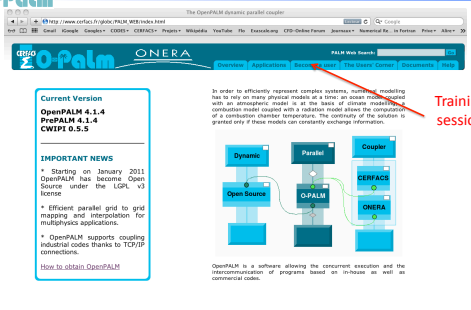
A software to manage complex applications in a modular way while respecting the performances of the codes,

An easy to use GUI to integrate, present and supervise applications,

Multiples interests :

- ➔ Facilitate evolution and maintenance of complex applications and of its components,
- ➔ Easy integration of new codes replacement, in existing applications (multi-physic coupling, collaborative development …)
- ➔ Maximize the use of intrinsic parallelism of applications and algorithms

OpenPALM



Training session

Web site: http://www.cerfacs.fr/globc/PALM_WEB

Florent.duchaine@cerfacs.fr & thierry.morel@cerfacs.fr

PALM characteristics

Some additional specificities

- PALM is based on MPI, the complete version runs with MPI-2 and a version with less dynamicity exists with MPI-1

  => Portable on all type of architectures: NEC, CRAY, IBM, SGI, BLUE GENE, Linux, Mac OS X, Windows with Cygwin

- Optimized communications

  ⇒In the standard communication scheme, the driver of PALM is informed about all the communications. Depending on the coupling algorithm, this control can be avoided

- Heterogeneous environments and commercial: advanced functionalities with IP sockets

ONERA O-Palm

CWIPI characteristics

- Data exchange of interpolated field between a source and a target meshes
  - Meshes are non coincident,
  - Meshes are differently partitioned on several processes,
- Communications are:
  - Unidirectional and synchronized,
  - Synchronized ping/pong,
  - Asynchronized,
- Parallelism:
  - Fully transparent for the user.

ONERA O-Palm

CWIPI characteristics

Type of meshes:

- 1D: edges,
- 2D: triangles, quadrangles, polygons
- 3D: tetrahedrons, pyramids, prisms, hexahedra, polyhedra

Control of the library with primitives called in the codes

Exchanges:

Source and target mesh types are the same:

Exchange through the defined interface

Source and target mesh types are not the same:

2D surface immerged in a 3D volume,

2D axi-symetric mesh immerged in a 3D volume,

…

ONERA O-Palm

CWIPI characteristics

Interpolation localization:

- Cell center of the target mesh,
- Vertex of the target mesh,
- Nodes defined by the user,
- The source data can be cell centered or vertex centered

Methods:

- For cell centered source fields:
  the target value is the value of the cell that contains the target point,
- For cell vertex source field:
  Interpolation matrix constructed with the geometric localization and barycentric coordinates
- User interpolation:
  the user can define its own methods based on the localization results

ONERA O-Palm

CWIPI characteristics

Results from the geometrical localization:

- Element of the source mesh that contains the target node,
- Closest element among potential candidates if no intersection,
- If to far away, the node is defined as non localized,

Geometric tolerance:

- User parameter (>0),
- Aims at increasing the number of candidates by increasing the research distance,
- The highest it is:
  The more the number of localized point is high,
  The more the research algorithm is slow !!

Non localized points:

- List accessible for the user,
- Be very careful: the received field is only defined on localized points, control and definition of non localized points have to be done by the user

ONERA O-Palm

CWIPI characteristics

Visualization

- Processor partitioning of the coupling interface,
- Non localized points,
- Sent and received fields

File Format

- Ensight, CGNS, MED

2.10 François Pellegrini (LaBRI, Inria, Université Bordeaux 1)

« How to age well a 20 y.o. Scotch »

"How to age well a 20 y.o. Scotch"

François Pellegrini
Université Bordeaux 1/LaBRI/Inria

Abstract

The Scotch software package is now 20 years old. The purpose of this talk is to address ways to make software last, from the technical, organisational and licensing points of view.

Most of the code produced during these years has been new code. It is the consequence both of a constant need for new features and of a set of initial design choices that enabled extensibility and maintainability.

The features of Scotch reflect the evolution of high performance architectures in this 20 year period. While Scotch has been initially designed to compute process- processor mappings, because parallel architectures in the 1980's were NUMA, this feature is little known by the public because in the 1990's hardware advances made these architectures UMA again, and plain partitioning features were sufficient. Now, machines become NUMA again, and parallel mapping features become mandatory. Scotch became free software in 2006, which considerably extended its user base, but not its contributor base, due to the technicality of such toolboxes.

We will conclude by proposing our vision for the next 20 years to come.

How to mature a 20 y.o. Scotch

François Pellegrini

EQUIPE PROJET
BACCHUS
Bordeaux
Sud-Ouest

15/05/2012

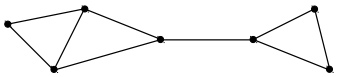
Outline of the talk

- Graph partitioning
- The Scotch project and history
- Licensing issues
- Some lessons (to be) learnt

Graph partitioning

What are graphs

- A graph is a set of vertices, linked by edges



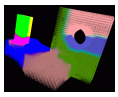
- Graphs are a versatile tool for representing problems :
  - Minimization of delivery trips
    - E.g. « Traveling Salesman Problem »
    - Search for « Hamiltonian paths »
  - Determination of maximum flow in a network
    - Search for « max flow / min cut »

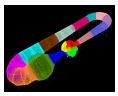
Graph partitioning (1)

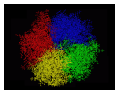
- Graph partitioning is an ubiquitous technique which has proven useful in a wide number of application fields
  - Used to model domain-dependent optimization problems
  - “Good solutions” take the form of partitions which minimize vertex or edge cuts, while balancing the weight of graph parts
- NP-hard problem in the general case
- Many algorithms have been proposed in the literature :
  - Graph algorithms, evolutionary algorithms, spectral methods, linear optimization methods, …

Graph partitioning (2)

- Two main problems for our team, in relation to sparse linear system solving (Ax = b) :
  - Sparse matrix ordering for direct methods
  - Domain decomposition for iterative methods
- These problems can be modeled as graph partitioning problems on the adjacency graph of symmetric positive-definite matrices
  - Edge separator problem for domain decomposition
  - Vertex separator problem for sparse matrix ordering by nested dissection

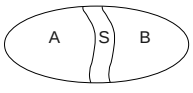


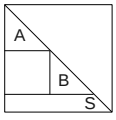


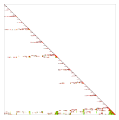


Nested dissection

- Top-down strategy for removing potential fill-inducing paths
- Principle [George, 1973]
  - Find a vertex separator of the graph
  - Order separator vertices with available indices of highest rank
  - Recursively apply the algorithm on the separated subgraphs



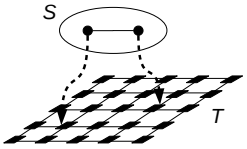




The Scotch project and history

The Scotch project (1)

- Provide a set of fast heuristic algorithms and tools for vertex and edge graph partitioning and for static mapping
- Static mapping is a generalization of the graph partitioning problem in which vertices of a source graph S have to be mapped onto vertices of a target graph T
  - Communication cost function accounts for distance



$$f_C(\tau_{S,T}, \rho_{S,T}) \stackrel{\text{def}}{=} \sum_{e_S \in E(S)} w(e_S)\,|\rho_{S,T}(e_S)|$$

The Scotch project (2)

- Previous roadmap : should handle graphs of more than a billion vertices distributed across one thousand processors — DONE !
- Current roadmap : should handle graphs of a trillion vertices distributed across one million processors
  - Account for heavily non uniform parallel architectures
    - As target architectures
    - As the machine on which we are running !
  - Asynchronous algorithms
  - Multi-threaded algorithms for hybrid parallelism

The Scotch history (1)

- Dec. 1992 : Start coding of v0.0
  - Algorithms for static mapping
- May 1994 : First published conference paper
- Jul. 1995 : Start coding of V3.0
  - First version planned to be publicly released
    - Competing non-free software MeTiS was available from the web
- Aug. 1996 : Start coding of v3.2
  - Algorithms for sparse matrix ordering
- Sep. 1996 : First website for public release of v3.0 under binary form
- Sep. 1999 : First license form for source code

The Scotch history (2)

- Nov. 2001 : Start coding of v4.0
- Oct. 2004 : Start coding of v5.0
  - Parallel versions of sparse matrix ordering code
- Feb. 2006 : Release of v4.0 as free software under LGPL
  - Project hosted by Inria Gforge
- Aug. 2007 : Release of v5.0 as free software under CeCILL-C
  - PT-Scotch parallel offspring
- Sep. 2008 : Start coding of v6.0
- Dec. 2008 : Start coding of v6.1
- Dec. 2012 : Release of v6.0
  - 20 years after coding of v0.0 started

(Free) software in science

Place of software in research

- In the world of research, one can see software :
  - As an end :
    - Demonstrator of algorithmic feasibility
    - Mathematical proof of existence
  - As a mean :
    - Self-crafted tool
    - Necessary to the obtainment of some results
  - It is usually both at the same time
- Scientific reproducibility imposes that software be available along with papers that exhibit its results
  - A policy regarding technical and legal means for accessing such software must be set up

What to do with produced software ? (1)

- A research laboratory is not supposed to be a software editor
  - A software may become useless from a research point of view but still be highly valuable from an application point of view
  - The value placed into the former development of such software must not be lost
    - Unused software is wasted money
  - Leadership on software development and maintenance may evolve
    - This has to be anticipated and encouraged
    - Free software licenses are most often a very suitable tool for this purpose

What to do with produced software ? (2)

- Application maintenance is not part of the tasks of a scientist
  - Yet, it is necessary to build and maintain a user community
  - Its cost/benefit ratio has to be carefully evaluated

What to do with produced software ? (3)

- The cost of turning research software into production-grade products can be high
- Yet, this step is necessary so as not to lose software value
- Several complementary means can be envisioned :
  - Technology transfer contracts with industry
    - But community is likely to lose further developments if the industrial version becomes privative/proprietary
  - Allocation of dedicated means by the research institution
    - Software engineers, not PhD's or post-doc's !
    - Beware of interns ! ;-)

License issues

Ownership of author's rights (1)

- Software is covered by author's rights, like many other works of the mind
  - Yet, standard author's rights do not apply
- Software authors who are civil servants or company employees see their patrimonial author's rights automatically transferred to their employer
- Only the employer can decide about :
  - Whether the software can be made publicly available or not
  - Under what license(s) it can be made available

Ownership of author's rights (2)

- Necessity to track contributions
  - Whenever handling licensing issues, author's rights must be asserted
    - Better to do it beforehand
- Beware of interns !
  - The author's rights of unpaid interns are not automatically transferred to the employer !
  - Problem of searching for the members of the “Disappeared Intern's Society”...
    - Some projects had to hire employees to re-code many critical modules

Choosing the proper license

- Select a license that is suitable to your project and acceptable by your community
  - As a civil servant, my results have to be used by the majority of the taxpayers and citizens
    - Weak copyleft licenses are interesting in this respect
- Advocate the fact of releasing your code to your employer
  - This process can be long, all the more when several institutions participated in the funding
    - In the case of Scotch: CNRS, ENSEIRB, Inria, Université Bordeaux 1
  - Find relevant arguments :
    - “My software is crap and nobody will use it anyway”
    - There already exist competitors using these licenses
    - ...

Benefits of going free software

- Inclusion of software on the form of packages within the main free software distributions
  - Increased visibility : Linux (Debian, Ubuntu), FreeBSD, …
  - Packaging done by autonomous mainteners (Debian Science, ...)
- Exclusive use within academic and/or industrial free software
  - E.g. OpenFOAM
- No contribution to the software itself
  - Expertise is scarce, mostly owned by competitors
    - Build a testbed environment that they can join !

Choosing the proper license (2)

- Within a given class, choose the license according to its own merits and to environmental constraints
- In the case of Scotch, for weak copyleft licenses :
  - LGPL allows “legal forking” towards GPL
  - Inria was my employer
  - So... CeCILL-C
    - But it is incompatible with GPL...
- Define a licensing policy from the inception of your project
  - Using a free software license reduces the impact of external contributors as long as the software is kept within the same license perimeter

Some lessons (to be) learnt

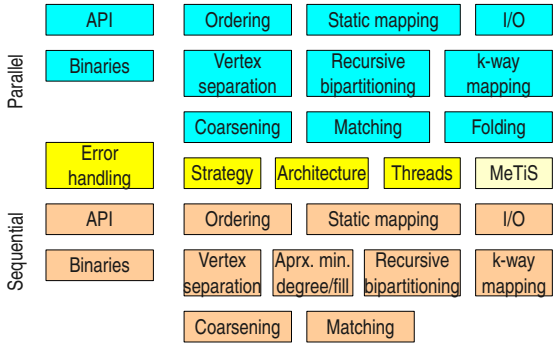
Handle legal aspects from the beginning

- Define the licensing policy from the beginning
  - Allows one to define the license compatibility matrix of envisioned third-party modules
- Separate public branch from private branch
- Track all contributions !
  - Public contributions for backporting and negotiation
  - Private contributions for attribution

Be paranoid about quality (1)

- Strict rules have to be defined and enforced since the inception of the project regarding :
  - Architectural conventions
    - The structure of the software should be clearly exposed
  - Naming conventions
    - Names should reflect architecture and function
    - A given variable or routine function should result in a single canonical name
  - Coding standards
    - For reader's and writer's sake
- Always aim at durability and extensibility !

Structure of the Scotch package (1)



Structure of the Scotch package (2)

- All data structures are defined by a C type (aka “class”)
  - `Graph` type in graph.h, etc...
- Routines are grouped by type name and function (methods)
  - `arch_*` : target architectures
  - `bgraph_*` : sequential graph bipartitioning
  - `bdgraph_*` : parallel graph bipartitioning
  - `dgraph_*` : parallel graph handling
  - `kdgraph_*` : parallel k-way static mapping
  - `vdgraph_*` : parallel vertex separation
  - `vgraph_*` : sequential vertex separation
  - ...

Structure of the Scotch package (3)

- Method files are identified by their type of computation :
  - `b?graph_bipart_xy` : edge graph bipartitioning method
  - `k?graph_map_xy` : static mapping method
  - `h?graph_order_xy` : graph ordering method
  - `v?graph_separate_xy` : vertex graph separation method
  - `hmesh_order_xy` : node mesh ordering method
  - `vmesh_separate_xy` : node mesh separation method
  - ...
  -

Be paranoid about quality (2)

- Every data structure should have an axiom checker routine attached to it
  - Written before the data structure is used !
  - Called at the end of every routine that modifies a data structure of its kind
- When used at the beginning of the library API routines, they help debug user's software
  - Eternal worshiping easily earned... ;-)

Thank you for your attention !

Any questions ?

http://scotch.gforge.inria.fr/

2.11 Frédéric Hecht (UPMC)

FreeFem++, un logiciel pour résoudre numériquement des équations aux dérivées partielles

FreeFem++, un logiciel pour résoudre numériquement des équations aux dérivées partielles

Frédéric Hecht
Université Pierre et Marie Curie

Résumé

FreeFEM++ est un logiciel libre qui permet de résoudre numériquement des équations aux dérivées partielles (EDP).

Nous ferons un tour d'horizon, sur quelques exemples comme :
- un problème à plus d'un milliard d'inconnues en 3d,
- une méthode pour simuler des condensa de Bose-Einstein,
- des remarques numériques sur les conditions aux limites pour les équations de Stokes.

FreeFem++, un logiciel pour résoudre numériquement des équations aux dérivées partielles.

F. Hecht
Laboratoire Jacques-Louis Lions
Université Pierre et Marie Curie
Paris, France

with , O. Pironneau, I. Danaila, S. Auliac, P. Jolivet

http://www.freefem.org mailto:hecht@ann.jussieu.fr

Outline

Last News, FreeFem++ pass 10^9 unknowns

The message from Pierre Jolivet

```
Machine: Curie Thin Node@CEA
Financement: PRACE project HPC-PDE (number 2011050840)
Nombre de coeurs: 6144     Mémoire par coeurs: 4 Go
Eléments finis: P3  Dimension: 2D
Précision: 1e-08
Nombres d'inconnues: 1 224 387 085
Méthode de résolution: GMRES préconditionné par une méthode de
décomposition de domaine à deux niveaux
Nombre d'itérations: 18
Temps de résolution: 2 minutes
Type de problème: équation de diffusion avec coefficients très
hétérogènes (5 ordres de grandeur de variation)
```

The FreeFem++ days, Begin of the December, 2013, UPMC, Jussieu, Paris, France

Introduction `FreeFem++` is a software to solve numerically partial differential

equations (PDE) in $\mathbb{R}^2$) and in $\mathbb{R}^3$) with finite elements methods. We used a user language to set and control the problem. The `FreeFem++` language allows for a quick specification of linear PDE's, with the variational formulation of a linear steady state problem and the user can write they own script to solve no linear problem and time depend problem. You can solve coupled problem or problem with moving domain or eigenvalue problem, do mesh adaptation , compute error indicator, etc ...

By the way, FreeFem++ is build to play with abstract linear, bilinear form on Finite Element Space and interpolation operator.

FreeFem++ is a freeware and this run on Mac, Unix and Window architecture, in parallel with MPI.

History

1987 MacFem/PCFem les ancêtres (O. Pironneau en Pascal) payant.

1992 FreeFem réécriture de C++ (P1,P0 un maillage) O. Pironneau, D. Bernardi, F. Hecht , C. Prudhomme (adaptation Maillage, bamg).

1996 FreeFem+ réécriture de C++ (P1,P0 plusieurs maillages) O. Pironneau, D. Bernardi, F. Hecht (algèbre de fonction).

1998 FreeFem++ réécriture avec autre noyau élément fini, et un autre langage utilisateur ; F. Hecht, O. Pironneau, K.Ohtsuka.

1999 FreeFem 3d (S. Del Pino) , Une première version de freefem en 3d avec des méthodes de domaine fictif.

2008 FreeFem++ v3 réécriture du noyau élément fini pour prendre en compte les cas multidimensionnels : 1d,2d,3d...

For who, for what !

For what

1. R&D
2. Academic Research ,
3. Teaching of FEM, PDE, Weak form and variational form
4. Algorithmes prototyping
5. Numerical experimentation
6. Scientific computing and Parallel computing

For who : the researcher, engineer, professor, student...

The mailing list `mailto:Freefempp@ljll.math.upmc.fr` with 410 members with a flux of 5-20 messages per day.

More than 2000 true Users (more than 1000 download / month)

The main characteristics I/II (2D)(3D)

- Wide range of finite elements : continuous P1,P2 elements, discontinuous P0, P1, RT0,RT1,BDM1, elements ,Edge element, vectorial element, mini-element, ...
- Automatic interpolation of data from a mesh to an other one (with matrix construction if need), so a finite element function is view as a function of (x, y, z) or as an array.
- Definition of the problem (complex or real value) with the variational form with access to the vectors and the matrix.
- Discontinuous Galerkin formulation (only in 2d to day).
- LU, Cholesky, Crout, CG, GMRES, UMFPack, SuperLU, MUMPS, HIPS , SUPERLU_DIST, PASTIX. ... sparse linear solver ; eigenvalue and eigenvector computation with ARPACK.
- Online graphics with OpenGL/GLUT/VTK, C++ like syntax.
- An integrated development environment FreeFem++-cs

The main characteristics II/II (2D)(3D)

- Analytic description of boundaries, with specification by the user of the intersection of boundaries in 2d.
- Automatic mesh generator, based on the Delaunay-Voronoï algorithm. (2d,3d (tetgen))
- load and save Mesh, solution
- Mesh adaptation based on metric, possibly anisotropic (only in 2d), with optional automatic computation of the metric from the Hessian of a solution. (2d,3d).
- Link with other soft : parview, gmsh , vtk, medit, gnuplot
- Dynamic linking to add plugin.
- Full MPI interface
- Nonlinear Optimisation tools : CG, Ipopt, NLOpt, stochastic
- Wide range of examples : Navier-Stokes 3d, elasticity 3d, fluid structure, eigenvalue problem, Schwarz' domain decomposition algorithm, residual error indicator ...

Laplace equation, weak form

Let a domain Ω with a partition of $\partial\Omega$ in Γ_2, Γ_e.
Find u a solution in such that :

$$-\Delta u = 1 \text{ in } \Omega, \quad u = 2 \text{ on } \Gamma_2, \quad \frac{\partial u}{\partial \vec{n}} = 0 \text{ on } \Gamma_e \tag{1}$$

Denote $V_g = \{v \in H^1(\Omega)/v_{|\Gamma_2} = g\}$.
The Basic variationnal formulation with is : find $u \in V_2(\Omega)$, such that

$$\int_\Omega \nabla u.\nabla v = \int_\Omega 1v + \int_\Gamma \frac{\partial u}{\partial n} v, \quad \forall v \in V_0(\Omega) \tag{2}$$

The finite element method is just : replace V_g with a finite element space, and the FreeFem++ code :

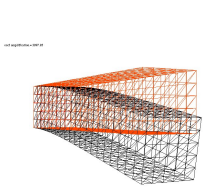
Poisson equation in a fish with FreeFem++

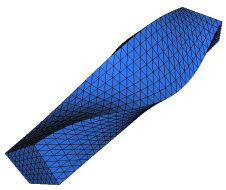
The finite element method is just : replace V_g with a finite element space, and the FreeFem++ code :

```
mesh3 Th("fish-3d.msh");                          //    read a mesh 3d
fespace Vh(Th,P1);                          //    define the P1 EF space

 Vh u,v;                    //   set test and unknown function in Vh.
macro Grad(u) [dx(u),dy(u),dz(u)]                    //   EOM Grad def
solve laplace(u,v,solver=CG) =
   int3d(Th)(  Grad(u)'*Grad(v)    )
   - int3d(Th)  ( 1*v)
   + on(2,u=2);                                          //    int on Γ2
plot(u,fill=1,wait=1,value=0,wait=1);
```

Run:fish.edp Run:fish3d.edp

Lame equation / figure





Run:beam-3d.edp Run:beam-EV-3d.edp Run:beam-3d-Adapt.edp

Stokes equation

The Stokes equation is find a velocity field $\boldsymbol{u} = (u_1, .., u_d)$ and the pressure p on domain Ω of $\mathbb{R}^d$, such that

$$\begin{aligned} -\Delta \boldsymbol{u} + \nabla p &= 0 \quad \text{in } \Omega \\ \nabla \cdot \boldsymbol{u} &= 0 \quad \text{in } \Omega \\ \boldsymbol{u} &= \boldsymbol{u}_\Gamma \quad \text{on } \Gamma \end{aligned}$$

where $\boldsymbol{u}_\Gamma$ is a given velocity on boundary Γ.
The classical variational formulation is : Find $\boldsymbol{u} \in H^1(\Omega)^d$ with $\boldsymbol{u}_{|\Gamma} = u_\Gamma$, and $p \in L^2(\Omega)/\mathbb{R}$ such that

$$\forall \boldsymbol{v} \in H^1_0(\Omega)^d, \ \forall q \in L^2(\Omega)/\mathbb{R}, \qquad \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - p\nabla.v - q\nabla.u = 0$$

or now find $p \in L^2(\Omega)$ such than (with $\varepsilon = 10^{-10}$)

$$\forall \boldsymbol{v} \in H^1_0(\Omega)^d, \ \forall q \in L^2(\Omega), \int_\Omega \nabla \boldsymbol{u} : \nabla \boldsymbol{v} - p\nabla.v - q\nabla.u + \varepsilon pq = 0$$

Stokes equation in FreeFem++

```
 ...  build mesh .... Th (3d) T2d ( 2d)
fespace VVh(Th,[P2,P2,P2,P1]);          //    Taylor Hood FE.
macro Grad(u) [dx(u),dy(u),dz(u)]          //    EOM
macro div(u1,u2,u3) (dx(u1)+dy(u2)+dz(u3))       //    EOM
VVh [u1,u2,u3,p],[v1,v2,v3,q];
solve vStokes([u1,u2,u3,p],[v1,v2,v3,q]) =
  int3d(Th)(
       ( D(u1,u2,u3):D(v1,v2,v3)   )
     - div(u1,u2,u3)*q - div(v1,v2,v3)*p
     - 1e-10*q*p )
 + on(1,u1=0,u2=0,u3=0)    +  on(2,u1=1,u2=0,u3=0);
```

Run:Stokes3d.edp

Navier Boundary condition of Stokes equations

$\boldsymbol{\tau}$ the tangent , $\boldsymbol{n}$ the normal, on Γ, g a given function, remember the boundary force $\boldsymbol{f}_\Gamma = {}^t\boldsymbol{n}\boldsymbol{\sigma}(\boldsymbol{u}, p)$.

$$\begin{aligned} \boldsymbol{u}.\boldsymbol{n} &= 0 \quad (3) \\ \boldsymbol{f}.\boldsymbol{\tau} &= \beta \boldsymbol{u}.\boldsymbol{\tau} + \boldsymbol{g}.\boldsymbol{\tau} \quad (4) \end{aligned}$$

This imply add in V.F. in RHS :

$$-\int_\Gamma \beta \boldsymbol{u}.\boldsymbol{\tau}\boldsymbol{v}.\boldsymbol{\tau} + \boldsymbol{g}.\boldsymbol{\tau} = -\int_\Gamma \beta\, {}^t\boldsymbol{u}({}^t\boldsymbol{\tau}\,{}^t\boldsymbol{\tau})\boldsymbol{v} + \boldsymbol{g}.\boldsymbol{\tau}$$

Remark, if $\boldsymbol{n} \neq \boldsymbol{e}_i$, change $\boldsymbol{u}.\boldsymbol{n} = 0$ by penalisation we have

$$O = \frac{1}{\epsilon}\boldsymbol{u}.\boldsymbol{n}; \qquad \text{Add to V.F. in RHS} - \int_\Gamma \frac{1}{\epsilon}\, {}^t\boldsymbol{u}({}^t\boldsymbol{n}\,{}^t\boldsymbol{n})\boldsymbol{v}$$

Run:NSCaraCyl-100-mpi.edp

Poisson equation with Schwarz method

- new parallel solver
- Schwarz algorithm to solve Poisson
  - Construction of the local partition of the unity
  - Construction of the overlapping,
  - the Schwarz alternating method,
  - asynchronous Send/Receive (Hard)
  - an GMRES version for Schwarz alternating method,
  - 2 levels of Precondiconner
  - Parallel visualisation for debugging

Poisson equation with Schwarz method

To solve the following Poisson problem on domain Ω with boundary Γ in $L^2(\Omega)$:

$$-\Delta u = f, \text{ in } \Omega, \text{ and } u = g \text{ on } \Gamma,$$

where f and g are two given functions of $L^2(\Omega)$ and of $H^{\frac{1}{2}}(\Gamma)$,
Let introduce $(\pi_i)_{i=1,..,N_p}$ a regular partition of the unity of Ω, q-e-d :

$$\pi_i \in C^0(\Omega) : \quad \pi_i \geq 0 \text{ and } \sum_{i=1}^{N_p} \pi_i = 1.$$

Denote Ω_i the sub domain which is the support of π_i function and also denote Γ_i the boundary of Ω_i.
The parallel Schwarz method is Let $\ell = 0$ the iterator and a initial guest u^0 respecting the boundary condition (i.e. $u^0_{|\Gamma} = g$).

$$\begin{aligned} \forall i = 1.., N_p : \quad -\Delta u_i^\ell &= f, \text{ in } \Omega_i, \quad \text{and } u_i^\ell = u^\ell \text{ on } \Gamma_i \setminus \Gamma, \ u_i^\ell = g \text{ on } \Gamma_i \cap \Gamma \quad (5) \\ u^{\ell+1} &= \textstyle\sum_{i=1}^{N_p} \pi_i u_i^\ell \quad (6) \end{aligned}$$

A Parallel Numerical experiment on laptop

We consider first example in an academic situation to solve Poisson Problem on the cube $\Omega =]0,1[^3$

$$-\Delta u = 1, \quad \text{in } \Omega; \qquad u = 0, \quad \text{on } \partial\Omega. \tag{7}$$

With a cartesian meshes $\mathcal{T}_{hn}$ of Ω with $6n^3$ tetrahedron, the coarse mesh is $\mathcal{T}^*_{hm}$, and m is a divisor of n.
We do the validation of the algorithm on a Laptop Intel Core i7 with 4 core at 2.66 Ghz with $8Go$ of RAM DDR3 at 1067 Mhz,

Run:DDM-Schwarz-Lap-2dd.edp Run:DDM-Schwarz-Lame-2d.edp
Run:DDM-Schwarz-Lame-3d.edp

Ipopt

The IPOPT optimizer in a FreeFem++ script is done with the `IPOPT` function included in the `ff-Ipopt` dynamic library. IPOPT is designed to solve constrained minimization problem in the form :

$$\text{find} \quad x_0 = \underset{x\in\mathbb{R}^n}{\operatorname{argmin}} f(x)$$
$$\text{s.t.} \quad \begin{cases} \forall i \le n,\ x_i^{\text{lb}} \le x_i \le x_i^{\text{ub}} & \text{(simple bounds)} \\ \forall i \le m,\ c_i^{\text{lb}} \le c_i(x) \le c_i^{\text{ub}} & \text{(constraints functions)} \end{cases}$$

Where ub and lb stand for "upper bound" and "lower bound". If for some $i, 1 \le i \le m$ we have $c_i^{\text{lb}} = c_i^{\text{ub}}$, it means that c_i is an equality constraint, and an inequality one if $c_i^{\text{lb}} < c_i^{\text{ub}}$.

Bose Einstein Condensate

Just a direct use of `Ipopt` interface (2day of works)
The problem is find a complex field u on domain $\mathcal{D}$ such that :

$$u = \underset{\|u\|=1}{\operatorname{argmin}} \int_{\mathcal{D}} \frac{1}{2}|\nabla u|^2 + V_{trap}|u|^2 + \frac{g}{2}|u|4 - \Omega i\overline{u}\left(\begin{pmatrix} -y \\ x \end{pmatrix}.\nabla\right)u$$

to code that in FreeFem++
use

- Ipopt interface (`https://projects.coin-or.org/Ipopt`)
- Adaptation de maillage

Run:BEC.edp

Hyper elasticity equation

The Hyper elasticity problem is the minimization of the energy $W(I_1, I_2, I_3)$ where I_1, I_2, I_3 are the 3 invariants. For example The Ciarlet Geymonat energy model is

$$W = \kappa_1(J_1 - 3) + \kappa_2(J_2 - 3) + \kappa(J - 1) - \kappa\ln(J)$$

where $J_1 = I_1 I_3^{-\frac{1}{3}}$, $J_2 = I_2 I_3^{-\frac{2}{3}}$,$J = I_3^{\frac{1}{2}}$,
let u the deplacement, when

- $F = I_d + \nabla u$
- $C = {}^tFF$
- $I_1 = \text{tr}(C)$
- $I_2 = \frac{1}{2}(\text{tr}(C)^2 - \text{tr}(C^2))$
- $I_3 = \det(C)$

The problem is find

$$u = \underset{u}{\operatorname{argmin}}\, W(I_1, I_2, I_3)$$

Hyper elasticity equation

```
fespace Wh(Th,[P2,P2]);
                                        //    methode de Newton ..
Wh [d1,d2]=[0,0];
Wh [w1,w2],[v1,v2];
for(int i=0;i<Nnewton;++i)
 {
    solve dWW([w1,w2],[v1,v2]) =
         int2d(Th)( ddW2d([d1,d2],[w1,w2],[v1,v2]) )
       - int2d(Th)( dW2d([d1,d2],[v1,v2]) -[v1,v2]'*[f1,f2] )
       + on(1,w1=0,w2=0);

     d1[] -= w1[];
     real err = w1[].linfty;
     if(err< epsNewton) break;
 }
```

Run:Hyper-Elasticity-2d.edp Run:ElasticLaw2d.edp Run:CiarletGemona.edp

Conclusion/Future

Freefem++ v3.20 is

- very good tool to solve non standard PDE in 2D/3D
- to try new domain decomposition domain algorithm

The the future we try to do :

- Build more graphic with VTK, paraview , ... (in progress)
- Add Finite volume facility for hyperbolic PDE (just begin C.F. FreeVol Projet)
- 3d anisotrope mesh adaptation
- automate the parallel tool

Thank for you attention.

2.12 Christophe Calvin (CEA/DEN/DANS/DM2S)

Les bibliothèques scientifiques massivement parallèles : utilisation dans les grands codes de simulation pour l'énergie nucléaire

Les bibliothèques scientifiques massivement parallèles : utilisation dans les grands codes de simulation pour l'énergie nucléaire

Christophe Calvin
CEA/DEN/DANS/DM2S

Résumé

Les bibliothèques scientifiques sont des outils qui peuvent permettre d'accéder a priori aisément au calcul hautes performances pour les applications scientifiques. Cependant leur intégration peut imposer des contraintes au niveau de leur utilisation dans ces applications.

Au cours de cet exposé nous aborderons différentes problématiques liées aux bibliothèques scientifiques massivement parallèles et à leur utilisation (ou à leur non utilisation) dans les codes de simulation pour l'énergie nucléaire. Nous présenterons également quelques voies d'amélioration à la conception des dites bibliothèques pour améliorer leur utilisabilité et leur extensibilité étant donné les contraintes liées aux nouvelles architectures de calcul.

DE LA RECHERCHE À L'INDUSTRIE

LES BIBLIOTHÈQUES SCIENTIFIQUES MASSIVEMENT PARALLÈLES : UTILISATION DANS LES GRANDS CODES DE SIMULATION POUR L'ÉNERGIE NUCLÉAIRE

Christophe CALVIN,
CEA/DEN/DANS/DM2S - CEA Saclay – France
christophe.calvin@cea.fr

SÉMINAIRE ARISTOTE – 15 MAI 2013 – POLYTECHNIQUE

www.cea.fr

INTRODUCTION

LES SYSTÈMES NUCLÉAIRES ÉLECTROGÈNES

1950 1970 1990 2010 2030 2050 2070 2090

Generation I
UNGG
CHOOZ

Generation II
REP 900
REP 1300
N4

Generation III
EPR Flamanville
EPR 2020

ASTRID
Conception
Exploitation d'ASTRID
Retour d'expérience

Construction
Exploitation des RNR

Les Grands Outils pour le Développement du Nucléaire

Le programme Simulation

Conception des installations via l'utilisation d'applicatifs métiers

- Explorer des domaines difficilement accessibles par l'expérimentation
- Réduire les durées d'étude
- Limiter les investissements

Nécessité d'amplifier cette activité

Environnement de simulation
Architecture des calculateurs

SALOME
(couplage de plates-formes)

Plates-formes des grandes disciplines du nucléaire

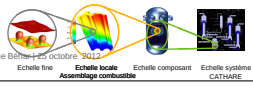
1 - Neutronique
2 - Thermo-hydraulique
3 - Mécanique, thermique
4 - Chimie du cycle
5 - Matériaux

Méthodes numériques
Architecture logicielle

Etudes amont Modélisation Expérimentation

Recherche scientifique et technologique de base

Exemple:
Couplage Multi-échelle en thermo-hydraulique



Christophe Bertier | 25 octobre 2012

LES DOMAINES DE LA PHYSIQUE POUR LA SIMULATION NUMÉRIQUE

La plupart des grands domaines de la physique sont présents :

- Mécanique des structures
- Dynamique rapide
- Thermo-hydraulique
- Neutronique, radioprotection
- Physique du combustible : thermo-mécanique, physico-chimie, matériaux
- Matériaux pour le nucléaire
- Physico-chimie

LES CODES DE LA DEN ET LES OUTILS/BIB EXTERNES

Dans cette présentation, nous nous intéresserons :

- Aux bibliothèques et outils externes utilisés par les codes de simulation
- Aux applications de la DEN qui pourraient utiliser des bibliothèques et outils externes
- Aux applications et outils de la DEN destinés à la communauté Open Source

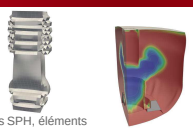
Nous présenterons quelques perspectives :

- Sur l'utilisation ou la non utilisation de bibliothèques et outils externes dans les codes de la DEN

EXEMPLES D'APPLICATIONS UTILISANT DES BIBLIOTHÈQUES

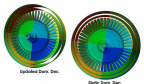
EUROPLEXUS

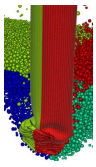
Logiciel de dynamique rapide



- Intégration temporelle explicite (stabilité conditionnelle)
- Grands déplacements et rotations
- Interaction Fluide-Structure
- Grands nombres de matériaux et de formulations : EF, VF, particules SPH, éléments discrets

Spécificités





- Structure de données vectorielles
- Grande importance des performances numériques
- Liaisons cinématiques complexes (conditions de contact et liens IFS)

Copropriété CEA/JRC
Développement ouvert à un **nombre limité de partenaires majeurs (EDF,** ONERA)
Distribution par le CEA à partir de 2013 : **Version de Production sous licence,**
Version Education & Recherche gratuite pour l'académie

EPX - HPC

Projet RepDyn (ANR-COSI-09-011)

ANR Calcul parallèle en dynamique rapide des fluides et des structure

Partenaires : CEA, EDF, LaMSID, ONERA, LaMCoS, INRIA

- Evaluation du parallélisme disponible en environnement industriel
- Développements algorithmiques pour lever les verrous existants
- Exploration de stratégies complémentaires

	Décomposition avec pondération unitaire	Décomposition avec pondération calculée
Temps de résolution pour 8 proc.	315 s	119 s

Test EDF : effet de la pondération de la décomposition de domaine

Simulation de l'essai MARA10 : perspectives favorables pour l'interaction fluide-structure

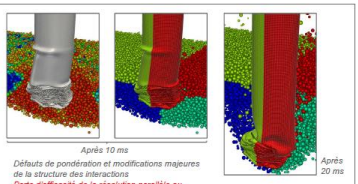
	Temps écoulé pour 2500 cycles	Speed-up / 4 proc	Efficacité
4 processus	13039 s		
8 processus	5703 s	2.44	1.22
16 processus	4035 s	3.45	0.86
32 processus	2963 s	4.70	0.59

Instabilité de Richtmeyer-Meshkov : extensibilité acquise pour les modèles mono-formulation

	Temps écoulé pour 1000 cycles	Speed-up / 4 proc	Efficacité
4 processus	2941 s		
8 processus	1447 s	1.96	0.98
16 processus	669 s	4.25	1.06
32 processus	368 s	7.72	0.97
48 processus	252 s	11.3	0.94
64 processus	211 s	13.5	0.84
80 processus	173 s	16.4	0.82

EPX - HPC

Simulation de l'essai Meppen (CEA/EDF R&D)

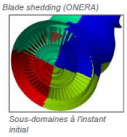


Après 10 ms — Après 20 ms

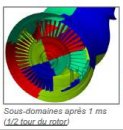
Défauts de pondération et modifications majeures de la structure des interactions

Perte d'efficacité de la résolution parallèle au cours du calcul

Blade shedding (ONERA)



Sous-domaines à l'instant initial



Sous-domaines après 1 ms (1/2 tour du rotor)

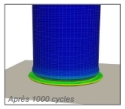
Perte d'efficacité de la procédure d'identification des candidats au contact remote

Prédominance de la phase de calcul des forces de liaison globales

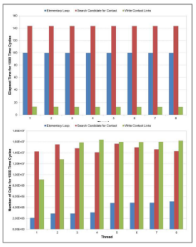
Perte d'efficacité de la résolution parallèle au cours du calcul (encore…)

EPX - HPC

Crash Missile Meppen Instant initial



Après 1000 cycles



Approche par vol de travail avec KAAPI

- Programmation DFG
- Implémentation FORTRAN aisée (boucles parallèles en particulier)
- Performance équivalente à Intel TBB ou Cilk avec un environnement plus simple

Investissement CEA/INRIA en complément du projet RePDyn

2011 : Stage de Master 2 de D. Leone
Interface FORTRAN pour KAAPI dans EPX

2011/2012 : Contrat F. Lementec (suivi T. Gautier)
Consolidation de l'implémentation et définition du cadre générique d'intégration de KAAPI dans EPX

2013/2016 : Thèse de M. Sridi (directeur B. Raffin, suivi T. Gautier)
Auto-paramétrage pour EPX en environnement hybride MPI/KAAPI

83% du travail accéléré par KAAPI
Speed-up 6/8 (memory bound)

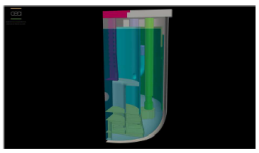
Séminaire Calcul Intensif I 19 FÉVRIER 2013 I PAGE 19

EPX - HPC

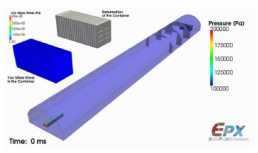
PRACE Preparatory Access 0624 (février-juillet 2012)

- Machine cible : TGCC/Curie *Fat Nodes* (32 cœurs, 128 Go RAM, Intel Westmere)
- Objectif : calcul parallèle sur 1024 pour des simulations fortement couplées en Interaction Fluide-Structure

Test n°1 : Accident de Dimensionnement de Confinement dans un réacteur Gen IV



Test n°2 : Onde de Choc sur un Container Métallique (modèle NTNU/JRC Ispra)



De l'expérience… sans maîtrise

- Un exécutable mal placé et un mois de déboguage inutile, un code modérément au point
- Des modèles de taille relativement petite (~ 2M de mailles) : gestion reportée des modèles de grande taille
- Des boucles *memory bound* et des facteurs NUMA importants : utilisation de KAAPI limitée à 8 *threads*

EPX - HPC

Test n°1 : Accident de Dimensionnement de Confinement dans un réacteur Gen IV

Processus MPI	16	32	64	128
Threads KAAPI	8	8	8	8
Nombre total de CPU	128	256	512	1024
Temps (s) pour 3400 pas de temps	2357	3496	1262	2478

Matériau multi-constituant pour le fluide : nombreuses itérations internes, accès répétés à la mémoire centrale
Performances instables et non-reproductibles

Test n°2 : Onde de Choc sur un Container Métallique (modèle NTNU/JRC Ispra)

Processus MPI	32	64	128	32	64	128
Threads KAAPI	1	1	1	8	8	8
Nombre total de CPU	32	64	128	256	512	1024
Temps (s) pour 2000 pas de temps	1547	885	577	640	440	345

Ça descend… … et c'est plus stable

Conclusion sur le PRACE Preparatory Access

Objectif	Résultat	Commentaires
Exécuter EPX sur de grands nombres de cœurs	OK	Point de passage obligé et attendu dans le cadre du projet RePDyn (*recommandation de l'évaluation à mi-parcours*).
Mettre en œuvre et valider la stratégie parallèle hybride	Partiel	Extensibilité croissante de 32 à 1024 cœurs possible dans les cas favorables. Efficacité à améliorer significativement. Approche distribuée relativement satisfaisante. Limitation due à la gestion de la mémoire dans EPX (redondance et boucles *memory bound*).
Mesurer l'extensibilité obtenue et identifier les verrous	OK (à consolider)	Importante source d'expérience pour des simulations de cette envergure. Besoin de mise en œuvre systématique et d'un effort plus continu à cette échelle.

EPX - HPC

Un parallélisme à mémoire distribué dynamique

- Réactualisation de la décomposition de domaine
- Evaluation en cours de calcul des coefficients de pondération

Un solveur distribué pour le calcul des forces de liaison globales

- Méthode d'évaluation de l'opérateur de condensation
- Solveur parallèle décentralisé

Le calcul avancé à mémoire partagé en complément

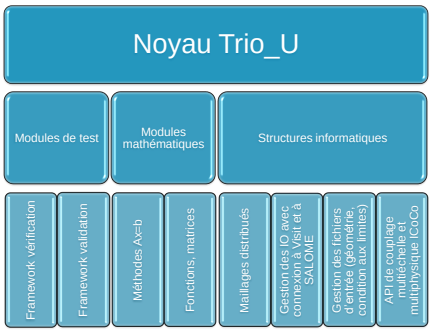
- Réponse à une évolution manifeste du *hardware*
- Limitation du nombre de sous-domaines
- Apport déterminant de l'équipe MOAIS (INRIA/Laboratoire d'Informatique de Grenoble) XKAAPI

Concrétisation et mise en œuvre d'une approche hybride à grande échelle

- Confrontation au problème d'échelle réel (*PRACE Preparatory Access 0624* en 2012)
- Mise en œuvre exclusive de simulations représentatives du contenu algorithmique complet d'EPX

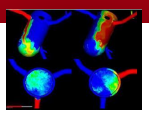
TRIO_U : CFD / DNS 3D PARALLÈLE

Un noyau + des applications – Dédiés à la CFD et DNS

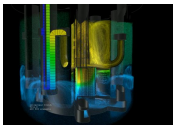


TRIO_U : EXEMPLES D'APPLICATIONS

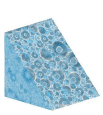
- TRIO_U : Code CFD monophasique + DNS Diphasique



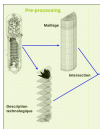
- Code MC2 : Simulation des écoulements dans les cœurs de réacteurs refroidis au sodium liquide – approche milieux-poreux



- MPCUBE – nuclear waste storage



- GENEPI+ : TH pour les générateurs de vapeur



SYSTÈMES ET SOLVEURS LINÉAIRES DANS TRIO_U

- Grande diversité de méthodes et schémas numériques entrainant une grande variété de systèmes linéaires à résoudre

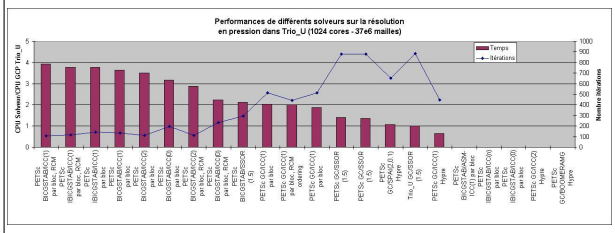
		Sparse	Symmetric	Constant
Pressure solvers	Pressure linear systems for incompressible flow	X	X	X
	Pressure linear systems for quasi compressible flow	X	X	
	Pressure linear systems for diphasic flow	X	X	
Implicit Scheme		X		

- Solveurs linéaires
  - Native solvers : GCP, GMRES, Precond : diag, ssor
- PETSC

PETSc

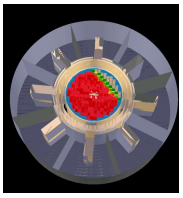
TRIO_U : SOLVEURS PARALLÈLES

- Branchement sur bibliothèque d'algèbre linéaire parallèle PETSc pour la résolution du Σ linéaire en pression



TRIPOLI-4

- Code de transport neutronique par méthode Monte Carlo
- Code de référence pour :
  - Etudes en physique des réacteurs
  - Radioprotection
  - Criticité
  - V&V
- Moteur géométrique – moteur de simulation
- Géométries 3D très complexes
- 2 moteurs géométriques
  - In-house
  - ROOT
- Outils d'exploitation des résultats : ROOT
- IHM SALOME-TRIPOLI



ROOT

EXEMPLES D'APPLICATIONS POTENTIELLEMENT CLIENTES D'OUTILS EXTERNES

CAST3M : CARACTÉRISTIQUES - ELÉMENTS DE CONTEXTE

Plate-forme de R&D en mécanique

- En développement continu depuis 1983
- Donner à l'utilisateur la capacité de traiter quasiment tout problème de mécanique des milieux continus par MEF
- Accès à toutes les variables internes (analyse numérique)

2000 utilisateurs dans 350 organismes (industriels, universités, organismes de recherche)

- ~1500 téléchargements par an : http://www-cast3m.cea.fr
- ~100 sites « industriels » (CEA – DEN/DAM/DSM/DRT, IRSN, EDF, AREVA – NP/NC/TA, IRSN, Bureaux d'études, …)

Principales collaborations

- IRSN (F), ENSIETA (F), JRC ISPRA (Italy), ENEA (Italy), IGCAR (India)

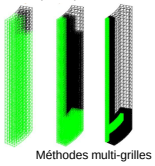
CAST3M - HPC

Efficacité : Toujours un challenge en mécanique du solide non-linéaire implicite

- Compétition entre les tâches de résolution globale et les tâches élémentaires locales :
  - Scalabilité limité pour les applications où la résolution matricielle est dominante (globale)
  - Excellente scalabilité pour les applications à loi de comportement dominante (locale)

Difficulté principale pour l'objectif HPC :

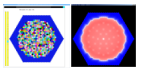
- Augmentation des problèmes de convergence pour les grands maillages
  - Efforts importants sur la modélisation et l'analyse numérique
  - Apports des méthodes
    - de décompositions de domaines
    - Multi-grilles : Gain en temps CPU et espace mémoire par rapport à un maillage unique adapté localement (h-raffinement) : 50 % des nœuds et 75% du temps CPU gagnés ! - Plus d'intérêt à utiliser un solveur parallèle à chaque niveau qu'à paralléliser les niveaux
  - Bibliothèques numériques parallèles pour la résolution des systèmes :
    - Méthodes directes (MUMPS)
    - Méthodes itératives



Méthodes multi-grilles

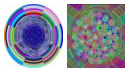
APOLLO3 : CODE DE TRANSPORT DÉTERMINISTE

New generation deterministic 3D transport code developed at CEA.

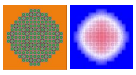


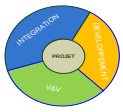
Objectives: provide numerical toolboxes giving the capability to build calculation schemes to:

- model and simulate GENII, GEN III and GEN IV reactors
- carry out R&D studies and validation processes requiring energy and spatial refined meshes.



Main features: advanced lattice and core solvers, parallel computation and functionalities for all kind of reactors.





- Goal oriented project organized around 3 axes: development, integration and V&V
- In partnership with **AREVA** and **EDF**.

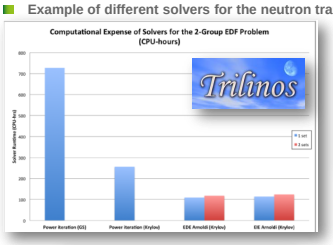
LES SOLVEURS DANS APOLLO3

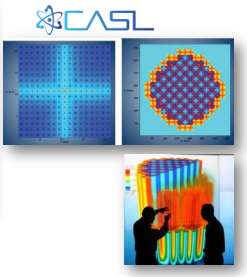
La résolution de l'équation du transport des neutrons

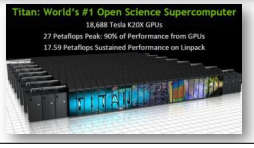
- Différentes approches, méthodes pour résoudre cette équation :
  - Approche simplifiée : diffusion ou transport simplifié : MINOS
  - Transport EF en non structuré : MINARET
  - Transport par méthodes des caractéristiques (courtes, longues) : IDT, TDT, …
- Globalement il s'agit de la résolution d'un problème aux valeurs propres généralisés
- A ce jour, chaque solveur a sa propre méthode de résolution

KRYLOV SOLVER FOR THE NEUTRON TRANSPORT EQUATION

- Example of different solvers for the neutron transport equation with DENOVO/CASL





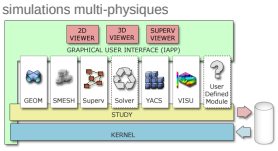


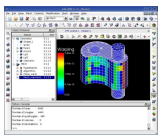
U.S. DEPARTMENT OF ENERGY

EXEMPLES D'APPLICATIONS EN OPEN SOURCE

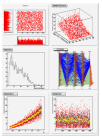
QUELQUES OUTILS/PLATEFORMES DISPONIBLES EN OPEN SOURCE (OU EN COURS)

- **SALOME** : Plateforme logicielle d'intégration pour des simulations multi-physiques

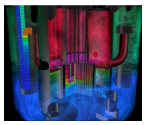




- **URANIE**: Plateforme incertitudes



- **SNORKY3D**: Outil de visualisation d'écoulements complexes 3D instationnaire



CONCLUSIONS ET PERSPECTIVES

UTILISATION DE BIBLIOTHÈQUES DANS LES APPLICATIONS SCIENTIFIQUES DE LA DEN

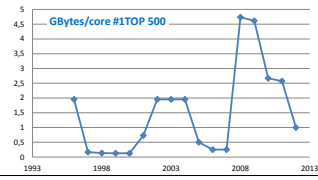
- Les situations sont assez contrastées :
  - Des applications utilisent, généralement avec succès, des bibliothèques externes et pas uniquement des bib. numériques parallèles
  - Des applications n'en utilisent pas du tout
  - La DEN est fournisseur d'outils pour la communauté :
    - Outil à utiliser
    - Plateforme d'intégration
    - Plateforme générique
- Quelques raisons principales à la non utilisation d'outils externes :
  - Nécessaire adaptation de l'application à l'outil : à prendre en termes de contrainte de conception
  - Intégration de développements non maitrisés
  - Très (trop) grande optimisation de certaines parties des applications qui rend très délicat voire quasi impossible l'utilisation de bib. externes :
    - Stockage de matrices
    - Problème à résoudre couplant plusieurs équations
    - Langage/env. de programmation non standard
- Mais l'utilisation de bibliothèques externes va devenir de plus en plus nécessaire

SUPERCOMPUTING TRENDS

Main Challenges for Exascale:

- Power (Energy Efficiency Computing)
  - Low power processor (ARM)
  - Maximize Flops/watt (GPU, MIC)
  - Minimize the memory per core
- Communication
  - Optimize and reduce data transfer
  - Reduce or avoid blocking communications
- Fault tolerance
  - MTBF <1h

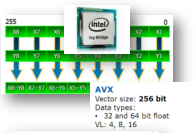
Systems	2009	2011	2015	2018
System Peak Flops/s	2 Peta	20 Peta	100-200 Peta	1 Exa
System Memory	0.3 PB	1 PB	5 PB	10 PB
Node Performance	125 GF	200 GF	400 GF	1-10 TF
Node Memory BW	25 GB/s	40 GB/s	100 GB/s	200-400 GB/s
Node Concurrency	12	32	O(100)	O(1000)
Interconnect BW	1.5 GB/s	10 GB/s	25 GB/s	50 GB/s
System Size (Nodes)	18,700	100,000	500,000	O(Million)
Total Concurrency	225,000	3 Million	50 Million	O(Billion)
Storage	15 PB	30 PB	150 PB	300 PB
I/O	0.2 TB/s	2 TB/s	10 TB/s	20 TB/s
MTTI	Days	Days	Days	O(1Day)
Power	6 MW	~10 MW	~10 MW	~20 MW

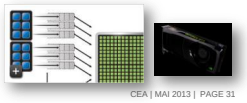


IMPACTS SUR LES APPLICATIONS SCIENTIFIQUES ET LES ALGORITHMES NUMÉRIQUES POUR L'EXTREME SCALE

Nécessité de repenser les algorithmes et les architectures logicielles

- Minimisation :
  - De la consommation énergétique
  - Des communications globales dont les synchronisations
- Maximiser le degré de parallélisme des algorithmes et leur "vectorisation" (SIMD)
  - Multi / hyper threading
    - 1 à 2 threads par cœur de calcul sur SandyBridge, IvyBridge, Hazwell → quelques 10^{aines} de threads par sockets
    - 4 threads par cœur de MIC → 244 threads
    - 1 thread par cœur de GPU → > 1500 threads
  - "vectorisation" (SIMD)
    - AVX1, AVX2, AVX3 …





IMPACTS SUR LES APPLICATIONS SCIENTIFIQUES ET LES ALGORITHMES NUMÉRIQUES POUR L'EXTREME SCALE

- Nécessité de concevoir des nouvelles architectures de codes scientifiques et d'algorithmes numériques pour prendre en compte ces modifications matérielles :
  - Algorithmes polymorphiques (s'adaptant au degré de parallélisme),
  - Utilisation de composants optimisés (comme les bib. numériques)
  - Auto tuning / smart tuning
  - Prise en compte de la tolérance aux pannes au niveau de l'application et au niveau des algorithmes
  - Nécessité d'avoir des runtime performants pour la gestion dynamique du multi-threading

Trilinos PETSc

StarPU Inria XKAAPI

CONCLUSION ET PERSPECTIVES

- Travaux en cours sur des algorithmes d'algèbre linéaire adapté à l'ultra-scale:

- Méthodes de Krylov pour la recherche de valeurs propres et la résolution de systèmes linéaires
- Méthodes d'auto-tuning et de smart-tuning
- Minimisation des communications globales
- Tolérance aux pannes
- Support des accélérateurs (GPU et MIC)
- Implémentation dans un environnement ouvert et réutilisable

2 thèses en cours

- Establish a framework based on Trilinos to enable our eigensolver (ERAM, MERAM) running on the hybrid platform in a flexible way (easy to perform different auto tuning strategies, easy to explore more mathematical possibilities but with a solid support of Trilinos packages, easy to benefit the scalability and maintainability due to the template or other characteristic of Trilinos, etc.)
- Evaluate and validate the capability of the cutting-edge many core systems, especially the interest for MIC and its coordination with GPUs(K20).

CONCLUSION ET PERSPECTIVES

- National and International collaborations

Merci !

Questions ?

Commissariat à l'énergie atomique et aux énergies alternatives
Centre de Saclay | 91191 Gif-sur-Yvette Cedex
T. +33 (0)1 69 08 68 80 | F. +33 (0)1 69 08 66 42

DEN/DANS
DM2S

Etablissement public à caractère industriel et commercial | RCS Paris B 775 685 019

2.13 Patrick Moreau (Inria)

Mais qui est l’éditeur de ces bibliothèques ?

Mais qui est l'éditeur de ces bibliothèques?

Patrick Moreau
Inria

Résumé

De nombreux logiciels issus de la recherche rencontrent de forts succès dans l'industrie et prennent une part importante dans l’activité de ces acteurs économiques. En conséquence, les exigences de la part de la communauté des utilisateurs (qualité, réactivité, pérennité) deviennent croissantes, ce qui conduit à l’apparition de nouvelles tâches hors recherche. Les conditions du transfert technologique et de l'innovation sont alors favorables dans cet écosystème open source.

Nous nous arrêterons plus particulièrement sur la phase intermédiaire que peut être un consortium dans le passage de l'open source de recherche à l'open source commercial.

Mais qui est l'éditeur de ces bibliothèques?

Séminaire Aristote du 15 mai 2013

Bibliothèques pour le calcul scientifique: outils, enjeux et écosystème

Patrick MOREAU
Responsable du Patrimoine Logiciel
Direction du Transfert et de l'Innovation

patrick.moreau@inria.fr

Paris , le 15 mai 2013

Une équation parfois complexe

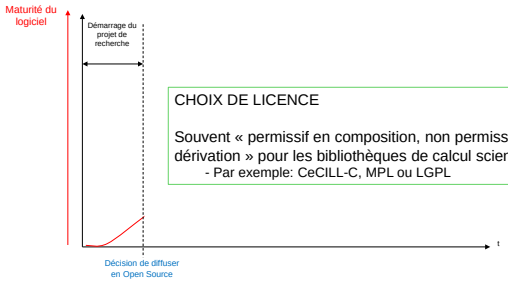
Succès d'un logiciel (ou d'une plateforme logicielle) issu de la recherche
- Logiciel souvent open source mais pas forcément
- Utilisation par un grand nombre d'acteurs externes (industriels / recherche / enseignement)

Exigences croissantes de la part de la communauté des utilisateurs (industriels / recherche / enseignement)
- Qualité, réactivité, pérennité

→ D'où l'apparition de nouvelles tâches **hors recherche**
- Gestion compliquée, surcharge de travail
- Hors mission Institut

Dans le cas de l'open source

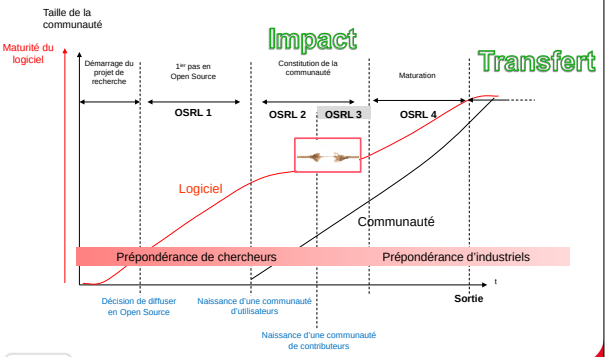
Facilite l'expérimentation, l'utilisation



CHOIX DE LICENCE

Souvent « permissif en composition, non permissif en dérivation » pour les bibliothèques de calcul scientifique
- Par exemple: CeCILL-C, MPL ou LGPL

Cycle de vie



Transfert par l'open source?

La Recherche = temporairement « éditeur »
- Définition de la roadmap, intégration, packaging, maintenance, support, …

Transfert = pérennisation de l'édition du code
- Transfert vers une entreprise
- Création d'entreprise
- Transfert du travail d'édition dans une structure ad-hoc ou préexistante
- Contribution à une communauté cible préexistante (Eclipse, FSF)
- Etc.

La diffusion en open source n'est pas du transfert
- Elle conduira (ou pas) à du transfert
  - Le transfert découlera des conditions économiques créées par l'impact
- Au bout de 5 à 10 ou 15 ans

Consortium: voie de maturation intéressante

C'est quoi un consortium?

Définition

Wikipedia:
- Un **consortium** (du latin signifiant « partenariat » ou « association ») est une collaboration temporaire entre plusieurs acteurs à un projet ou programme dans le but d'obtenir un résultat.

Deux choses à retenir:
- une grande variété de cas
- la qualification de « **temporaire** » est clé

Divers exemples

Initié par Inria (et ses éventuels partenaires)
- Scilab
- CAML
- PREMIA
- MONOLIX
- ObjectWeb (devenu consortium OW2)
- PHARO
- IHE Gazelle

Dont Inria a été moteur
- W3C Europe

Autres
- Seiscope

Qu'est-ce qui va nous faire décider de lancer un consortium?

La coincidence de plusieurs éléments…

La situation "marché" favorable à un consortium

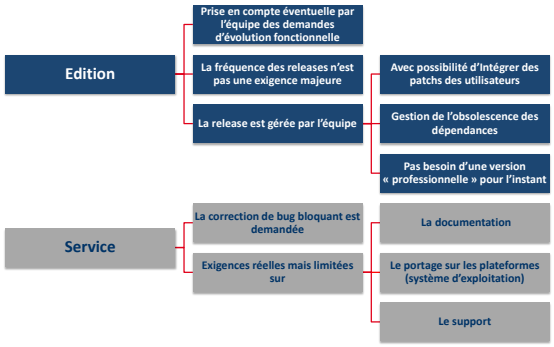
- Il y a des utilisateurs actuels (grand compte / PME / recherche / enseignement)
  - Idéalement, comprenant des acteurs économiques
- Le logiciel prend une part importante dans l'activité de l'utilisateur.
  - *Il est prêt à financer, mais pas assez pour créer une activité économique*
- Le potentiel d'utilisateurs existe.
  - *Mais le nombre d'utilisateurs n'est pas suffisant pour créer une entreprise*
- Il existe des domaines potentiels non explorés

L'analyse

- Décrire ce que font ces utilisateurs
- Quel niveau de leur activité (veille / prototypage/ produit) dépend du logiciel?
- Décrire pourquoi le logiciel est important: décrire les solutions alternatives, le caractère différentiant du logiciel

Consortium = catalyseur du marché

Des attendus modérés de la part des utilisateurs



Le logiciel

Grande généricité

- Les membres, bien que concurrents, peuvent être partenaires dans le consortium

L'analyse

- Décrire pourquoi il y a une forte mutualisation sur le logiciel

Les attendus de l'équipe de recherche

- L'animation d'une communauté d'utilisateurs
- Le recueil des besoins de la communauté cible
- Des nouveaux défis scientifiques émanant des relations avec les utilisateurs
- La mise en place d'une collaboration effective de développement
- L'équipe n'est pas prête à « lâcher » l'édition du logiciel

Les attendus de l'Institut

La création d'un consortium requiert de l'effort de la part de l'Institut

→ L'impact potentiel de l'actif logiciel doit donc être considéré suffisamment important.

L'analyse

- Est-ce que le jeu en vaut la chandelle?
- Quel coût l'Institut est-il prêt à supporter ?
- Sur quelle durée?

Etude d'alternatives

(exercice à faire régulièrement pendant la vie du consortium)

→ Existe-t-il des consortia proches?

→ Peut-on rester en statu quo?

→ Existe-t-il des sociétés (existantes ou à venir) pouvant répondre en partie aux besoins des utilisateurs?

→ A-t-on des contacts avec des sociétés pouvant répondre en partie aux besoins des utilisateurs?

Envisager les grandes lignes de la sortie

(exercice à faire régulièrement pendant la vie du consortium)

Le montage d'un consortium n'est pas du transfert technologique

- Potentiel de création de start-up
- Est-ce qu'un des membres à venir du consortium pourra poursuivre l'édition du code?
- L'arrêt

Viabilité économique hors consortium

Les questions à se poser pour esquisser les grandes lignes du consortium

Est-ce que le consortium doit porter sur tout le logiciel?

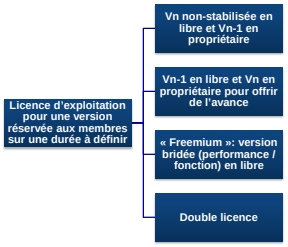
- Peut-on mettre en évidence un tronc commun et des modules métier ?
- Identifier l'intérêt de chaque partie du logiciel:
  - Qualité
  - Utilité

L'analyse

- Etablir l'architecture du code

Les membres payeraient pour quelle offre?*pas de limite à la créativité!*

- Attribution d'un mug
- Participation
  - à la définition des priorités de développement
  - à la définition du contenu des versions
- Participation altruiste à la recherche
- Formation
- Support sur l'outil (intégration / « hot line »)
- Animation scientifique
- Usine à projets collaboratifs
- Code sprint
- Etc.



Exemples d'offre de licence

Montage du consortium

Un consortium se caractérise en général par un **contrat d'adhésion** unique hormis les clauses financières qui peuvent être adaptées.

- Quel est le financement probable des membres?
  - Possibilité d'une segmentation de l'offre aux membres (Gold, Silver, …)?
  - Cotisation selon la taille de l'entreprise?
- Durée de l'engagement initial
- Choix de la gouvernance (Comité de pilotage, comité scientifique, …)
- Choix du montage juridique (par ordre croissant de complexité de mise en œuvre et d'indépendance vis à vis de l'Institut)
  - Sans structure dédiée (hébergé par Inria)
  - GIS
  - Association loi 1901
  - Fondation(s)
- Définition des protections (marque, logo, …) si pas déjà fait

Merci pour votre attention

et

merci à Brigitte Duême, Inria, Responsable transfert « aéronautique défense,spatial et sécurité »

ARISTOTE Association Loi de 1901. Siège social : CEA-DSI CEN Saclay Bât. 474, 91191 Gif-sur-Yvette Cedex.
Secrétariat : Aristote, École Polytechnique, 91128 Palaiseau Cedex.
Tél. : +33(0)1 69 33 99 66 Fax : +33(0)1 69 33 99 67 Courriel : Marie.Tetard@polytechnique.edu
Site internet http://www.association-aristote.fr