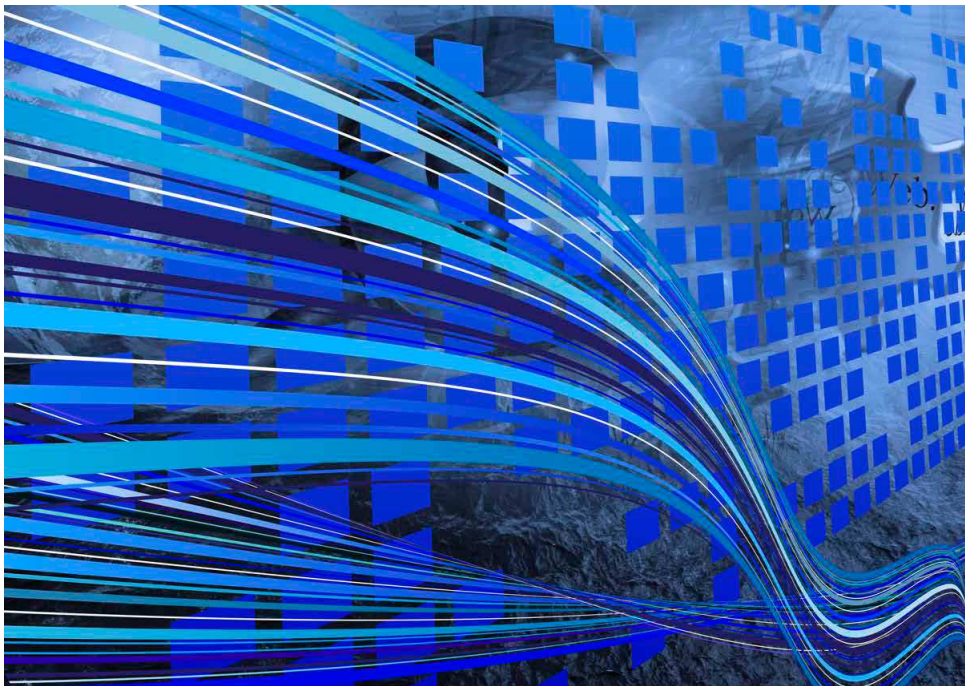
Quelles architectures pour les simulations de demain ?

Amphithéâtre Becquerel, École Polytechnique - Palaiseau

Jeudi 5 février 2015



Coordination Scientifique

Christophe Calvin (CEA)

Christophe Denis (EDF)

Thiên-Hiêp Lê (ONERA)

Editorial Board

Actes du séminaire du 05/02/2015

Quelles architectures pour les simulations de demain?

Séminaire Aristote, 05/02/2015 à l'École Polytechnique - Amphi. Becquerel

Coordination scientifique :

Christophe Calvin (CEA/DEN)

Christophe Denis (EDF)

Thiên-Hiêp Lê (ONERA/DSNA)

Partenaires :

Asma Farjallah (INTEL)

Laurent Grandguillot (HP)

Sommaire

Compte-rendu des interventions :

Introduction

L'association Aristote, société savante, sans but lucratif, regroupe les acteurs des nouvelles technologies de l'information et de la communication. Elle organise des cycles de séminaires, des formations et réunit depuis 25 ans organismes de recherche, grandes écoles, entreprises et PME.

Les séminaires, avant tout destinés à favoriser l'information et les échanges, peuvent naître d'une volonté globale, mais aussi et surtout des propositions des différents membres, comme nous le rappelle **Thiên-Hiêp Lê (ONERA)** en début de journée.

Thématique :

Le thème de cette journée "Quelles architectures pour les simulations de demain?", est né d'une proposition de **Christophe Denis (EDF)**.

Aujourd'hui, c'est un fait, nous avons besoin de toujours plus de puissance de calcul pour simuler la réalité. Les besoins sont par exemple croissants en terme de simulations multi physiques et multi-échelles.

Les limites de la Loi de Moore sont bien présentes ; on ne peut plus compter sur une augmentation forte de la vitesse des processeurs, et les problèmes de dissipation thermique deviennent limitant. Cela rompt avec des habitudes d'évolutions progressives prises avec les calculateurs HPC.

Il faut alors maintenant se tourner vers des architectures multicoeurs, hétérogènes, et hiérarchiques. Cependant les codes de calcul ne sont pas encore adaptés à ces architectures.

Il y a un besoin impératif de définir des nouvelles méthodologies (utiliser des modèles réduits, prévoir la performance, travailler sur la qualité numérique des solutions) et de repenser les algorithmes actuels.

Il faut également réfléchir sur les méthodes pour mailler des énormes domaines ; c'est à dire savoir ce qu'on veut calculer, où, finement.

C'est dans ce contexte, aujourd'hui, que le projet Européen EESI (Europen Exascale Software Initiative) a pour feuille de route de "relever le défi des nouvelles générations de systèmes parallèles, composés de millions de noyaux hétérogènes qui fourniront des performances exaflopiques en 2020."

1. Modélisation de la performance et optimisation d'un algorithme hydrodynamique de type Lagrange-Projection sur processeurs multi-coeurs

Florian De Vuyst (ENS Cachan), Thibault Gasc (Maison de la Simulation), Mathieu Peybernes (CEA) et Raphaël Poncet (CGG)

Les architectures de simulations actuelles sont relativement complexes. On distingue deux niveaux de complexité. La complexité de communications inter-nœuds MPI et la complexité sur un nœud de calcul.

Le modèle de performance Roofline a pour but d'être en mesure d'estimer et de comprendre la performance numérique atteignable sur une machine dont on connaît les caractéristiques. C'est un modèle analytique simplifié.

Cela permet d'identifier l'efficacité de l'implémentation sur une machine actuelle, de comparer l'efficacité de deux méthodes, de prévoir et d'anticiper la performance d'architectures futures (qualitativement et quantitativement). Cela permettra ainsi d'orienter les choix d'achats.

Cela intéresse donc à la fois le développeur du code, l'utilisateur et l'acheteur des machines.

Ce modèle se base sur 4 informations qui caractérisent le système pour prévoir la performance:

Deux paramètres liés à l'architecture : La **bande passante** et le **Peak** (limite de performance théorique de la machine)

Deux paramètres liés à l'algorithme : L'**intensité arithmétique** (rapport nombre d'opérations / quantité de données) et le **peak spécifique** (la vitesse maximale de l'exécution de l'algorithme)

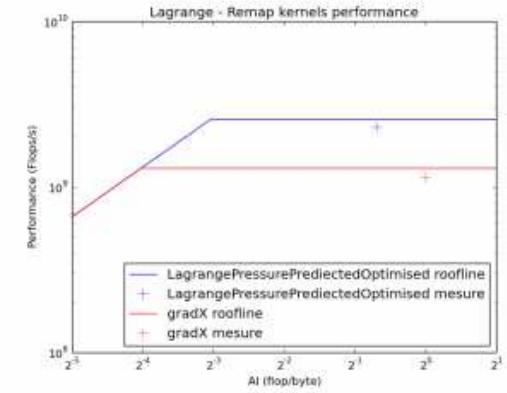


Figure : roofline pour différents kernels

Sur la pente du toit, le facteur limite est la bande passante.
Sur la partie horizontale, c'est le Peak qui est limitant.

Les limites du modèles roofline sont la non prise en compte du comportement hardware (au niveau du cache), et la mauvaise prédiction au niveau des points du "coin du toit". (Le modèle ECM permet de réduire les erreurs.)

En conclusion, un modèle simple de type Roofline donne de bonnes indications qualitatives, pour obtenir une vision plus fine, il faudra travailler sur la définition précise du Hardware.

2. Goal oriented modeling in Computational Mechanics

Ludovic Chamoin (ENS Cachan, INRIA Rocquencourt)

L. Chamoin dans son équipe au LMT Cachan (Laboratoire de Mécanique des Structures travaille en vérification et validation des modèles (passage de l'expérimental au théorique). C'est une approche qui peut coûter très cher si on prend le parti de tout contrôler.

Dans une approche ingénieur, souvent, ce qui intéresse, c'est surtout ce qui se passe sur le local. Par exemple, à proximité de la fissure dans le cas d'une rupture mécanique. D'où l'intérêt d'avoir des modèles bien adaptés à ce qu'on en attend.

Quand on s'intéresse à une quantité d'intérêt, l'erreur sur le problème adjoint et sa solution permette d'avoir une estimation de l'erreur sur tout le calcul. On réduit la zone d'étude grâce à l'utilisation d'un problème adjoint. Cette réduction se fait au prix d'une erreur de discrétisation & de réduction.

L'intérêt des techniques présentées est qu'elles sont non intrusives. Cela permet de calculer une solution plus précise sans changer le maillage, en le raffinant. On peut enrichir localement et on introduit un terme résiduel pour vérifier les conditions limites.

1) Validation de la qualité d'un modèle élément fini

Pour s'assurer de la qualité du modèle, il faut s'assurer de la qualité de la discrétisation.

Classiquement, deux types d'outils d'estimations d'erreur sont disponibles:
- Outils a priori (permettent d'anticiper la convergence ou non, mais pas de quantifier l'erreur)
- Outils a posteriori (méthode des résidus d'équilibres, contrôle globalement l'erreur pour savoir comment raffiner le maillage).

Au LMT Cachan, l'outil utilisé est l'approche duale (nécessite d'avoir accès aux données sur le bord des éléments, mais majore l'erreur vraie sur chaque élément).
L'erreur sur une quantité d'intérêt est le produit de l'erreur sur le problème de référence et de l'erreur sur le problème local. Cela permet d'affiner les zones qui sont pertinentes pour la quantité d'intérêt.

Au LMT, les objectifs sont de trouver une méthode garantie, précise et pratique pour l'ingénieur. Ces dernières années via de nouvelles méthodes, on obtient :

Qint - Quh - Q Corr ≤ Erreur Problème × Erreur Problème adjoint
Qint, quantité d'intérêt exacte
Quh, quantité d'intérêt obtenue par les éléments finis
Q corr, terme de correction

On voit qu'au prix d'assez d'efforts sur le problème adjoint, on peut obtenir une très bonne précision sur l'erreur globale. L'intérêt est de venir de manière non intrusive enrichir les solutions localement (avec des solutions pré-calculées qu'on vient injecter et un terme résiduel). Cela permet d'agir sans changer le maillage, ce qui est très efficace dans une approche ingénieur et permet d'obtenir de très bons résultats.

Perspectives pour ces travaux : très bons résultats pour les problèmes linéaires, pour les problèmes dynamiques le problème est de suivre la propagation des solutions dans la structure, en non linéaire la problématique porte sur la précision. Pour l'instant par contre, on a aucune idée de comment faire pour les problèmes instables du type flambement par exemple.

2) Validation de la réduction de modèle

La première stratégie, le **couplage de modèle** consiste à remplacer un modèle très fin et par conséquent trop cher, par un modèle homogène, sauf dans une zone très précise. On peut obtenir une prédiction de l'erreur de couplage (exemple couplage particulaire/continu) et adapter les paramètres jusqu'a obtenir la précision souhaitée sur la quantité d'intérêt.

La seconde stratégie est la **méthode PGD** (Proper Generalized Decomposition). On va alors chercher la solution sous forme de modes.

Sur des problèmes multi-paramètres, avec une approche standard, on a un nombre d'inconnues qui croît très rapidement avec l'augmentation du nombre de paramètres.

La méthode PGD permet de résoudre ce type de problème sous formes de problèmes uni-variables.Il est à nouveau nécessaire d'avoir une estimation sur la précision. Le LMT a développé des outils dans cette optique.

Ces outils permettent d'avoir une erreur sur les quantités d'intérêt, de séparer si l'erreur vient du maillage ou du nombre de mode pour savoir s'il faut plutôt calculer le mode suivant ou raffiner le maillage. On va ainsi progresser par itération (plus on avance en nombre de mode, plus on va avoir besoin d'adapter le maillage).

Aujourd'hui les ambitions sont d'arriver dans le futur à séparer automatiquement un problème multidimensionnel en problème monodimensionnel.

Une autre idée dans le futur c'est de faire "discuter" un modèle de calcul avec une application expérimentale, pour créer une boucle de contrôle entre un modèle numérique et une application (intérêt double : prévoir l'application avec le modèle et valider le modèle avec l'application)

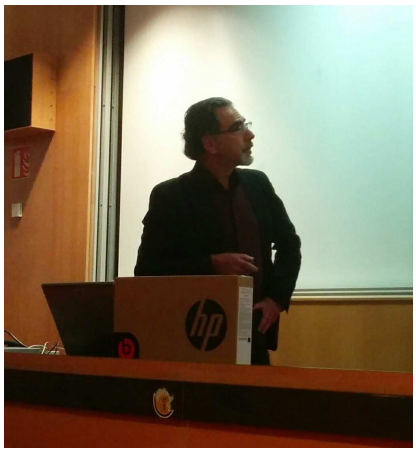
En conclusion la démarche globale est de **"Construire des modèles orientés vers l'application, intelligent, maîtrisés et pratiques aux applications réelles."**

"Essentially all models are wrong, but some are still useful"

George E.P.Box.

3. Développement de maquettes de solveurs d'écoulements compressibles en Volumes Finis non structurés pour des clusters de GPU Tesla

Jean-Marie Le Gouez (ONERA Châtillon, Département Simulation Numérique des écoulements et Aéroacoustique)



Les simulations à l'Onera demandent à la fois des modèles de fermetures résiduelles des schémas et plus d'efficacité et de robustesse. Aujourd'hui les plans de développement portent à la fois sur des travaux pour de nouvelles fonctions et l'interopérabilité, mais aussi la prise en compte et réduction des risques sur la scalabilité vers le calcul petaflopique.

Actuellement, les utilisateurs internes demandent des meilleures performances analytiques et de nouvelles classes de méthodes numériques.

Une des problématiques des grands codes est qu'il faut à la fois les faire tourner et les améliorer. Ils seront cela dit nécessairement l'objet d'une réingénierie dans le futur.

Pour alimenter la réflexion et pour avoir une vision concrète et pratique des problématiques, l'Onera a lancé un certains nombre de prototypes (FUNK, AGORA, NEXTFLOW) sur les stratégies de maillages, nouvelles méthodes, sur l'usage de la LES, et une prise en compte accrue du HPC.

La présentation porte surtout sur NextFlow qui traite des volumes finis d'ordre élevé.

Le but est de proposer des architectures logicielles et des méthodologies, de développer des solveurs modulaires, et d'étendre la base d'outils de couplage et de pilotage d'applications complexes.

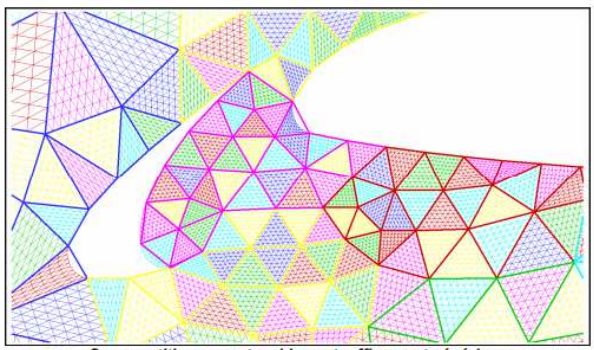
L'idée est d'avoir des utilitaires de maillages sur des topologies simples pour ne jamais manipuler le maillage dense en temps que tel, et que celui-ci soit directement stockés sur les processeurs et les réseaux.

L'objectif : calculer des solutions précises avec des maillages assez grossiers.

La méthode est fondée sur une base Fortran MPI, qu'on dérive en C-Cuda. Une première sur une structure de donnée IJK et une seconde partitionnée en bloc.

En maillage IJK, bonne performances sur un GPU. La difficulté apparaît quand on s'adresse au maillage non structuré. Le problème est qu'on a besoin d'accéder à 30-60 cellules au voisinage et donc cela engendre un stress mémoire important.

Le problème est de trouver un compromis entre optimiser l'utilisation des caches et favoriser la connectivité. On va ensuite créer des blocs de 32-64 éléments. On va connecter les éléments internes via des échanges en mémoire globale de tableaux d'adresses propres.



Sous-partitionnement en blocs et raffinement générique (inspire par les techniques graphiques de "tessellation" dans le rendu surfacique)

4. Quels défis pour la programmation efficace d'architectures manycoeurs?

Eric Petit (UVSQ)

Aller vers l'Exascale, c'est aller vers de plus en plus de nœuds, et de plus en plus de cœurs. Un certain nombre de ressources ne passent pas à l'échelle, telles que la mémoire par cœur, la bande passante, les protocoles de cohérence, les interconnexions réseaux. La pression des niveaux hiérarchiques devient plus importante, on a plus de niveaux et moins d'uniformité. Cela complique la mise au point de l'algorithme.

Il faut trouver des expressions d'algorithmes pour chaque échelle et éviter les évènements globaux.

Il s'agit plus d'une évolution que d'une révolution. On a besoin de plus de parallélisme dans la communication et les calculs, et d'arriver à privatiser la mémoire pour chaque processeurs (pour limiter les besoins de communication).

Conséquence de la Loi d'Amdahl : "Plus on a de cœur, plus grande est la proportion en temps d'exécution du code séquentiel"

On a donc besoin aujourd'hui de plus de concurrence dans les programmes et de localité dans les communications.

Il est difficile de faire des expériences directement sur de grands code industriel, le portage est compliqué d'où l'intérêt est de se limiter à une "proof of concept"

La conception de proto-application permet de reproduire à l'échelle le comportement d'applications HPC, pour supporter le développement d'optimisations ré-implantables ensuite dans les solutions originales.

S'il est difficile pour différentes raisons industrielles d'ouvrir et de partager le code et les use case, ce qui est compréhensible, cela a un réel intérêt de rendre open source le problème.

Cela permet de :
- soutenir l'effort communautaire
- produire des résultats reproductibles et comparables
- avoir une bonne interface avec les développeurs d'applications

Il y a deux voies possibles pour la construction d'une proto-application :

La "mini" application, qui mime l'application globale avec des propriétés simplifiées, qui présente l'avantage de présenter de faibles problématiques de propriété industrielle. Elle est cependant d'une représentativité limitée, éloignée du code réel et ne se focalise pas sur un problème spécifique.

La deuxième solution, la proto application "stripped down" (épurée), qui va se focaliser sur une problématique précise, être représentative à l'échelle, et où on pourra faire facilement un retour à l'utilisateur. Elle est partielle, il faut faire plusieurs proto-applications pour une application donnée. Il faut également vérifier s'il n'y a pas de problème de couplage dans la partie extraite. Dans ce cas, la propriété intellectuelle peut également être plus complexe à gérer.

La proto-application étudiée avec Dassault-Aviation, l'assemblage d'une matrice à partir d'un maillage non structuré pour un code élément fini.

Les approches existantes étaient:

- La décomposition de domaines ; approche efficace pour les architectures actuelles mais sous optimisée avec les architectures futures.

- Le coloriage, qui consiste à mettre en couleurs différentes chaque élément qui partage un arc. C'est simple à mettre en œuvre mais cela a une mauvaise localité. Cette méthode a un fort besoin de bande passante et synchronisation.

L'approche proposée a été de diviser récursivement les domaines en sous domaines. On lance la fonction d'assemblage séquentiel de Dassault mais sur des sous domaines qui peuvent tourner en parallèle.

On obtient bien des tâches parallèles, des données localisées, les synchronisations ne se font qu'entre voisins (localité des communications) et le chemin critique est la hauteur de l'arbre de récursion.

Deux manières de mesurer le gain de localité :
- Mesurer les caches misses, bon indicateur, mais faux.
- Distance à la diagonale : 95% des éléments plus proches dans la nouvelle version.

A propos de la vectorisation ; on peut toujours vectoriser sur des éléments. Le coloring est fait pour les machines vectorielles, et un cœur reste une machine vectorielle. Première approche, affecter à un élément la première couleur dispo pour minimiser le nombre de couleurs; bon ratio de vectorisation mais chute avec la taille des vecteurs car le parallélisme de données dans un sous domaine est faible. Ce que l'on veut c'est un maximum de couleurs qui aient la longueur d'un vecteur. La nouvelle stratégie proposée modifie l'algorithme en bornant le nombre d'élément par couleur. On a beaucoup plus de couleurs mais on gagne 10% sur le ratio de vectorisation.

Cependant, avec la taille des vecteurs qui augmente, on a besoin de plus de données pour extraire du parallélisme, alors que les tailles de caches diminuent dans le même temps. Ce compromis s'illustre dans trois axes :

Pur Divide and Conquer : très bonne localité, supporte les données non-structurées, mais plus de

Pur Coloring : perte de localité, mauvais support des données non-structurée, mais bon parallélisme.

Problème structuré : localité et bon parallélisme de donnée, mais on ne peut plus traiter certains problèmes.

Il faut trouver le bon compromis.

Au final les machines vectorielles sont bien connues, on est donc sur une nouvelle branche qui part sur une base solide.

En question, on souligne que travailler ensemble via les proto-applications permet de créer des ponts entre recherche et industrie. Recrutement de stagiaires ensemble etc.

5. Algorithmes parallèles pour la dynamique rapide adaptés aux architectures modernes : contraintes spécifiques et choix stratégiques

Vincent Faucher (CEA/DEN/DM2S)

La présentation traite des travaux réalisés dans EUROPLEXUS SOFTWARE. V. Faucher nous rappelle qu'au CEA, la population est plus constituée d'algorithmiciens et de physiciens que de développeurs.

Malgré un besoin de repenser les codes, on ne peut pas partir de zéro. On est en face d'algorithme de plusieurs millions de lignes de code, qui sont le fruit de la capitalisation d'une expérience historique.

De plus il y a une certaine inertie sur ces codes, vu qu'ils doivent rester opérationnels pour servir des clients dont les besoins sont immédiats. On est de plus contraint par une structure de données existante et un langage historiquement utilisé, le fortran.

Sur Europlexus : l'application est de traiter des équations d'équilibre.

Spécificité : on est dans le cadre d'une application couplée on ajoute un certains nombre de contraintes cinématiques. La discretisation (elt /vlm fini) permet d'arriver à un système matricielle.Pas de difficultés dans l'expression des vecteurs de forces et de flux. Mais il faut exprimer les forces au niveau des couplages.

Deux approches :

-**Par résidus de contraintes** : on explicite les forces de liens, mais on est dépendant de coefficients arbitraires de pénalisation.

-**L'approche duale du problème** : via multiplicateur de Lagrange ; système qui vérifie exactement les contraintes cinématiques imposées , pas d'impact sur la stabilité, mais par contre le système matriciel est complexe à construire.

C'est la deuxième approche qui est utilisée, et qui comporte trois tâches principales:

- écrire les boucles élementaires du calcul de contrainte
- écrire les contraintes cinématiques
- le calcul des forces de liens (problème d'algèbre linéaire)

La stratégie de parallélisation globale d'Europlexus :

- Distribuer les données par noeud de cluster (décomposition de domaine indépendante de la connectivité par *recursive othoghonal dissection*) ; ce qui localise les contraintes cinématiques.
- Avoir une approche générique des contraintes cinématiques. (via solveur spécifique)

- La décomposition dynamique de domaine. (cela permet de gérer les changements de topologie en cours de simulation)
- Mémoire partagée à l'intérieur des sous domaines.

Difficultés rencontrées :

- Pour exploiter plus de threads que dans un socket.
- Difficile pour un maillage d'optimiser la proximité au noeud et au voisin.

Stratégie de "vol de travail" de l'application :

Macro thread par socket qui peuvent "se voler des éléments" , en étant certain que les vols sont efficaces.

Au niveau de la stratégie future et d'orientation des programmes, là où la Petascale est bien définie (gros clusters , uniforme etc) , l'exascale est variable sur la localisation des tâches, sur l'hétérogenéité des équipements et on maîtrise mal encore les contraintes d'utilisation de puissance.

Demain, on a besoin d'algorithmes qui gèrent mieux l'asynchronisme, pour minimiser les contraintes de flux et remplir au mieux les coeurs.

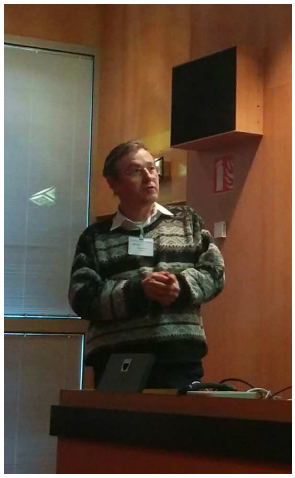
Attention aussi à ce que les solutions restent bien controlées par les physiciens du modèle, et à ne pas dégrader la solution pour faciliter la réalisation du calcul.

Dans les prospections également :

- auto-tuning de l'application
- diagnostic en cours d'exécution
- équilibrage de la charge en cours

6. On the road again

A. Refloch (ONERA)



L'objectif de la présentation est de parler de la route du Teraflop vers l'Exascale.

Aujourd'hui, plus on avance dans la technologie et les modèles, plus on s'ajoute de contraintes.

On veut faire plus de physiques, **plus d'échelles**, traiter **plus de données** d'entrées, et également **plus d'exécutions**. Les géométries sont de plus en plus complexes. Les simulations sont de plus en plus multiphysiques, on utilise des solveurs Lagrangiens avec des nécessités de parallélismes, et on a également des problématiques de couplage.

Les défis de la simulation multiphysique sont supérieurs à la somme des défis de la monophysique. Il faudra du HPC à tous les niveaux, et pas uniquement sur la partie solveur, pour passer à l'Exascale.

La présentation se base sur l'expérience acquise sur CEDRE (code de référence Onera pour l'énergétique et la propulsion). Ce code utilise la décomposition de domaine et est multiphysique. Il a également une approche zonale, avec les défis de gestion des interfaces que cela comprend.

Les applications de ce code sont variées (aérodynamisme, thermique, propulsion liquide, foudre...)

Localité : Ce code utilise des maillages non structurés généralisés : ***"Le secret de la performance, c'est la localité"***

Mémoire : C'est un pourcentage important du prix de la machine (environ un tiers). On introduit dans le code des mesures internes de l'occupation mémoire. La tendance est à la baisse de mémoire par cœur de plus (à voir avec la technologie?). Avec l'Impact du Big Data, on peut espérer d'avoir plus de mémoires disponibles à faible coût dans le futur.

Entrées /Sorties : Orientation vers un fichier unique (un seul fichier pour redémarrer). De plus avoir un grand nombre de fichiers n'est pas optimisé pour les systèmes parallèles (back up). Les entrées/sorties sont difficiles à appréhender, peu de documentation, peu de moyen d'accès coté utilisateur.

Le **couplage** devient un moyen naturel pour utiliser l'Exascale (biblio de couplage développée à l'Onera).***" Il faudra avoir un coupleur dans sa caisse à outil dans le futur"***

Le couplage interne dans CEDRE est possible, la multi physique est facilitée par le bon accès à l'information.

L'environnement de la chaîne de calcul permet de remettre des éléments externalisables (pré & post) au niveau du solveur, et de réaliser de l'analyse in-situ. Par contre, il y a un réel besoin en ressources humaines pour exploiter ces nouvelles données en direct.

Sur les parties mobiles, A. Refloch nous cite l'exemple du moteur Vulcain de Snecma, avec un maillage qui doit entre autre s'adapter au fur et à mesure à la régression du propergol. Le remaillage en cours de calcul, c'est un vrai challenge pour l'Exascale.

Un autre aspect parfois négligé de la scalabilité, c'est qu'elle implique des besoins de gros moyens pour le développement, pas seulement pour le calcul. C'est parfois complexe à faire comprendre au niveau des acheteurs de matériel.

Au niveau des outils; le profiler mémoire et le debugger sont des vrais besoins.

La route vers l'Exascale subit de nombreuses contraintes : dans le cas des codes de recherche pure, au vu de la problématique de la taille des développements, des équipes, la question est complexe. De plus ces codes évoluent très vite, sont utilisés par l'industrie (nécessité de maintenir une portabilité), et présentent une nécessité de validation. Une des contraintes supplémentaire sera ensuite l'ergonomie, pour faciliter l'usage des modèles et codes.

En conclusion, l'Exascale, c'est un challenge hardware et système, mais aussi pour les applications. Pour les applications, le problème, c'est qu'il va falloir s'adapter rapidement dans un contexte qui est flou au niveau des technologies aussi bien que des usages.

7. Quelle adéquation possible entre la puissance des supercalculateurs Tier0 et une formulation éléments finis implicite adaptative?

Hugues Digonnet (Institut du Calcul Intensif-Ecole Centrale de Nantes)

L'idée était d'avoir un retour sur la capacité à faire tourner des calculs sur des calculateurs de type Tier0 (qui fait partie du top européen). Ce sont des machines qui ont un très grand nombre de cœurs, beaucoup de mémoire. Du point de vue applicatif, deux critères permettent de parler de "massivement parallèle" , quand le nombre de voisins n'évolue plus, ou quand le nombre de cœurs est du même ordre de grandeur que le nombre de données local à un cœur.

Pourquoi une formation implicite adaptative? Cela permet de conserver des pas de temps important par rapport à une méthode explicite. Les maillages non structurés sont plus automatiques à générer et plus adaptés aux formes réelles. On parle d'"adaptatif" car le système évolue au cours du calcul. On recherche aussi à utiliser un maillage minimal basé sur un estimateur d'erreur.

Les données d'entrées réelles sont massives, l'approche est de numériser la pièce réelle et de l'importer directement dans les calculs plutôt qu'une modélisation CAO, ce qui peut présenter un intérêt réel de gagner du temps sur le maillage de pièces complexes souvent plus long que les calculs.

Le mailleur peut mailler partout sauf aux interfaces, qu'il pourra traiter après un repositionnement dynamique.

La difficulté est d'obtenir de bonnes performances parallèles. Cela s'est obtenu par des permutations des données reliées au sein d'un vecteur. On obtient une très bonne accélération jusqu'a 4000 cœurs. L'intérêt est de pouvoir calculer des détails très précis en restant global. L'idée c'est que les mailles sont grandes là où il n'y a pas de variations, et fines là ou on a des variations.

Pour des systèmes plus importants, difficulté à faire converger. Nécessité d'une méthode multi-grilles. On obtient un maillage final à 33 milliard de noeuds, 67 milliards d'éléments.

Quelques exemples de simulations : données réelles vers simulation. Cela permet de réaliser des Tomographies 3D de microstructures ou de capturer des détails de paysage urbain.

Conclusion : capacité d'utiliser le Tier-0 (on a été capable d'utiliser 200 To de ram et d'utiliser 100 milliards de ddl)

Il ne s'agit pas seulement d'avoir du gain en calculant en parallèle. Avec le maillage adaptatif on peut réduire le nombre d'inconnus de facteur 1000. L'idée est de combiner les différentes approches d'optimisation.

8. Aeromines - A new cloud computing platform

Elie Hachem (Mines Paris -CEMEF)

Aujourd'hui le Cloud est de plus en plus développé dans tous les domaines. L'idée est de créer une plate forme de calcul scientifique paramétrée pour l'utilisateur et simple à utiliser.

On utilise des méthodes adaptatives dans le temps et dans l'espace, pour faire des applications adaptées aux usages.

L'idée est d'utiliser la **méthode d'immersion de volumes.**

1) Représentation de la pièce et son environnement

2) Le solveur doit gérer des discontinuités fortes ; entre 20°C et 1000°C par exemple. Il doit également comprendre les différentes phases liquide/solide

La géométrie surfacique est représentée dans un maillage. On passe directement du CAD au maillage.

La méthode d'immersion de volumes nécessite de préparer le maillage partout où va passer le solide. Les applications peuvent être diverses. On peut gérer des géométries complexes et mobiles (problématique de pièces qui bouge + fluide)

Applications sur des fluides multiphases : chargement d'un four avec six pièces

Applications extrêmes : simulation avec un F22 Raptor (22 millions de noeuds, 120 millions d'éléments ! Mêmes les mouvements des volets sont pris en compte)

Les grandes problématiques sont d'avoir une interface facile d'accès pour l'utilisateur. L'interface permet le visuel, soit le maillage, soit les streamlines.

L'interface présente l'intérêt qu'une fois que le problème a été paramétré par les équipes d'Aeromines, on peut facilement venir simuler en se connectant depuis n'importe quel périphérique qui a accès au web (même un smartphone suffit), de lancer la simulation, d'en voir l'évolution et de récupérer le résultat final. Il est également possible de régler facilement les paramètres.

Avec l'application pré-paramétrée, cela permet très simplement au client de vérifier l'impact d'une forme de drône sur l'aérodynamisme par exemple.

Cette plateforme sert par exemple dans le cadre d'un MOOC Mécanique des Fluides, où les étudiants peuvent effectuer des "travaux pratiques" sur le portail. Il y a un certain nombre d'exemples classiques qui sont déjà paramétrés sur la plateforme. Les résultats sont consultables en ligne ou téléchargeables.

L'objectif que la plateforme soit un outil d'enseignement, et un outil "pay as you go" pour les entreprises (avec également une prestation d'expertise).

La grande problématique est de travailler le solveur pour absorber des mailles très hétérogènes.

9. Application d'ExpressFabric dans le projet Open Compute, architecture hybride arm/x86: le projet Daap

Jean-Marie Verdun (Splitted-Desktop Systems)

Développement d'ordinateur sur le plateau de Saclay, dans le cadre du projet Open Compute. Principalement dans l'optique "pour les data center" ou on rencontre des problèmes de connexion d'échelles.

Projet : Développer des clusters avec un maximum de processeurs dans un environnement fortement contraint (températures élevées + humidité).

Les contraintes étaient de fonctionner à 35° et de bien gérer l'énergie, et également de bien choisir les entrées sorties car c'est un fort élément de coût à l'heure actuelle.

Objectifs principaux : fonctionner à très haut T, baisser les coûts, et réduire les pannes liées à l'interconnexion.

Le choix technologique a été d'utiliser des processeurs de PC portables pour leurs caractéristiques intéressantes pour l'application en matière de tolérance aux températures (95°-105°) et d'économie d'énergie (via la capacité à être plus modulaire grâce à un nombre plus grand de transistors).

Au niveau des connexions le choix a été d'utiliser des interfaces PCI express qui ont l'avantage de déjà exister sur les CPU, et d'éviter les surcoûts liés à l'achat de connexions.

Une des autres approches "arrêter d'utiliser des serveurs avec trop de socket" pour l'usage. Cela est à la fois cher et source de pannes potentielles. Par l'expérience, la CPU est rarement cause de la panne. Cela permet également un gain de place pour faire des machines denses.

Un autre aspect est de trouver des solutions pour avoir des grappes de processeur avec un système de remote management, pour faciliter la gestion de l'infrastructure, quitte à perdre en finesse.

La combinaison de 12 processeurs de pc portables AMD + fabrique PCI Express présente un réel intérêt pour le calcul scientifique. L'interconnexion locale a pour latence back to back 150 nanosecondes et l'architecture supporte des cartes réseau très haut débit.

Pour revenir plus sur le projet :

Les choix sont principalement pilotés par le coût avec pour objectif d'être à 350$ par PCB (12 coeur, 32 Go, 300 Gb PCI Express)

La démarche d'Opencompute est de fournir des ordinateurs, mais aussi des outils pour en développer. Il y a une plateforme web ou chacun peut participer aux différents éléments du projet. Par exemple aujourd'hui des réflexions se portent sur le boîtier en collaboration. La documentation, les schémas sont accessibles. Seuls les chips ne sont pas accessibles.

Le projet est à la fois OpenHardware, mais aussi openfirmware (bios, fabrique pci express entièrement configurable)

10. Evolution de l'architecture du système d'information scientifique d'EDF R&D

Hugues Prisker (EDF)

Depuis l'année 2000, EDF R&D a lancé son programme d'acquisition de moyens de calcul haute performance.

Ces moyens de calcul ont été dans la plupart des cas des architectures en cluster de PC qui progressivement sont passées du statut de machines de laboratoire vers des supercalculateurs industriels classés au Top 500 et hébergés en datacenter.

Cette politique de mise en place de ces machines a toujours été conduite avec l'étude fine des besoins des utilisateurs d'EDF, des évolutions des grands codes de simulation, des contraintes de l'ingénierie opérationnelle tout en restant en adéquation avec les roadmaps des constructeurs de matériels.

L'infrastructure scientifique d'EDF R&D comporte certes des supercalculateurs, mais également un réseau de 1200 stations de travail ainsi que des péta-octets de données en ligne.

C'est à travers l'expérience acquise au cours de ces années, mais aussi des besoins futurs et des (r)évolutions technologiques annoncées, prévisibles ou espérées que nous nous proposons de dégager les cibles de l'infrastructure et de l'architecture du système de d'information scientifique d'EDF R&D de demain.

11. Impact des architectures matérielles Exascale sur les environnements systèmes de calcul.

Pascale Rossé-Laurent (BULL)

Les besoins des simulations futures impliquent des besoins d'évolution de l'architecture pour les constructeurs. Les nouvelles tailles, l'évolution de la décomposition, etc. Pour l'architecture, les contraintes d'énergie, de coût et de qualité de service (performance & fiabilité) sont importantes.

De plus en plus besoin de faire des vérifications in-situ, les schémas d'accès aux données évoluent, l'implémentation des points de reprise et l'assistance à la mise au point de codes sont également des préoccupations.

Les applications sont de plus en plus variées et le défi est de plus en plus important pour Bull de trouver la bonne architecture à y opposer. Les modèles de données deviennent de plus en plus immenses, un processeur a de plus en plus d'activités à gérer en propre.

La consommation énergétique devient un challenge. Pour y répondre, BULL développe des techno low-power, des mémoires non volatiles, pour limiter la puissance de calcul et d'accès aux données.

Attention il faut regarder l'équation énergétique sur l'ensemble : les noeuds, le réseau le stockage. "Ne pas déplacer le problème hors champ"

Les aspects robustesses et maintenabilité deviennent critiques sur de très grosses fabriques. Il faut savoir précisément où est la panne, en temps réel et savoir s'y adapter.

La scalabilité des architectures demande des optimisations à tous les niveaux (nœuds, échanges de données, I/O) . Il faut également gérer l'hétérogénéité des ressources.

Bull a lancé un programme Exascale avec un volet matériel, mais aussi applicatif. Sur le plan matériel, il faut rester capable d'intégrer assez de technologies. Il faut concevoir un packaging avec une densité pour intégrer beaucoup de nœuds de calcul. L'objectif est d'avoir une armoire qui soit un îlot autonome de calcul, qui détecte automatiquement ses propres pannes et s'adapte aux pertes. Il y a également ajout d'un interconnect BXI.

Autre enjeu, avoir des données proches des calculs, ce qui permet de mieux optimiser la bande passante. On ne peut plus se passer de l'infrastructure de bas niveau ; *"Etre transparent à l'application coûte 2 fois plus cher qu'avoir une bonne entre application/hardware."*

Au niveau de l'environnement d'exécution, des travaux sont lancés pour comprendre comment les nouveaux types de modèles et d'applications vont interagir avec les calculateurs.

Un grand nombre de programme de recherches sont en cours chez BULL :

-COLOC pour la localité des données

-DATASCALE pour les mouvements de données

-MOBUS pour le scheduling

-HDEEM pour la mesure fine de la consommation électrique

-ELCI pour établir une "proof of concept" de pile logicielle optimisée pour les grands solveurs numériques.

12. High Level performance prediction following application characterization

Thierry Philippe (INTEL)

Problématiques actuelles d'Intel et des clients : Comment designer les architectures futures, quelles seront les performances? Pour Intel il faut comprendre à quoi vont ressembler les applications dans le futur.

Les réponses viendront principalement de l'application. "Est ce qu'on peut prévoir la performance assez simplement?"

On veut déterminer la performance actuel ("characterization") et en formaliser un modèle d'extrapolation ou des simulateurs. L'objectif est également d'avoir une idée de la performance des matériels futurs. Tout cela pour dimensionner au mieux les machines qui correspondront aux applications et aussi influencer la conception interne.

Il faut prendre en compte tous les niveaux que ce soit à grande échelle au niveau du simulateur, au niveau core, socket et nœud, mais aussi au niveau des clusters pour déterminer les enjeux de communications et de topologie.

Les méthodes ont des précisions et des vitesses d'exécution très différentes. Il faut essayer d'être pragmatique selon les usages, et avoir une vision multi-approche pour faire des prédictions cohérentes.

Les simulateurs cycle-accurate peuvent donner le nombre de cycle en simulant précisément l'application (c'est par contre long à mettre en place).

L'approximation du premier ordre : temps total = temps de bande passante + temps CPU. C'est une façon d'estimer qui néglige l'impact de la latence sur la bande passante, qui néglige l'utilisation mémoire et les communications entrées sorties, mais qui permet une première estimation.

L'autre approche est de s'inspirer du benchmark pour prévoir la performance en quelques minutes.

Il existe différents outils :

-Speed Of light : avec les spécifications théoriques on peut déterminer les bornes maximales de performances en terme de bande passante et de flops. Cela donne déjà un bon indicateur des performances qu'on ne dépassera pas.

-Le roofline tel que vu en première présentation (page 4).

En conclusion : il faut toujours utiliser des simulateurs, l'extrapolation haut niveau donne une bonne estimation à 2 ou 3 ans. Le formalisme est le même pour les 3 hauts niveaux. Il faut par contre améliorer la prise en compte des caches et de l'interconnexion I/O pour avoir de meilleures estimations.

13. Architecture for extreme scale simulation

Patrick Demichel (HP)

Les années 2020 vont être les années des données extrèmes dans un contexte Big Data : on va avoir besoin de puissance de calcul massive pour en faire l'analyse. Cela est dû à l'explosition du nombre de périphériques connectés. L'enjeu va être de trouver les ressources pour exploiter les quantités de données collectées. On est de plus dans un contexte où on sait que la Loi de Moore s'essoufle et on ne pourra pas compter sur l'évolution linéaire des technologies.

Face à ces problématiques, HP développe plusieurs technologies.

HP Moonshots : Serveur avec un système de stockage intégré. Un serveur plus spécialisé pour sa charge de travail, qui permet un gain d'espace, de puissance, et de coût.

La photonique permet de transmettre à une bande passante multipliée par 30, pour une énergie réduite d'un facteur dix, et un coût équivalent. La photonique en s'affranchissant des distances permet de placer différemment la mémoire dans les architectures, on peut avoir des architectures de cartes plus simples.

La mémoire non volatile : les memresistors changent la manière dont on stocke les données. Cela permet d'utiliser des rack mémoire avec une occupation "au besoin" et de stocker à grande vitesse.

L'augmentation des volumes de données va créer une nécessité de plus en plus importante de traiter les données sur place. Même avec la photonique, cela va devenir impossible de déplacer les quantités engendrés par le big data.

La philosophie générale est d'utiliser ses nouvelles technologies non pour se substituer aux anciennes, mais pour reconcevoir de manière globale l'architecture.

Ces nouvelles architectures vont demander des efforts importants en termes d'applications, car il faudra repenser les algorithmes, les couches logicielles pour en tirer pleinement parti. Un des enjeux importants de ces nouveaux systèmes est également la sécurité, l'objectif d'HP est de les concevoir comme des systèmes capables de se défendre eux mêmes contre les attaques. C'est absolument indispensable pour que l'internet des objets devienne une réalité rassurante pour le client.

Informations :

Vous pouvez retrouver les présentations sur :

http://www.association-aristote.fr/doku.php/public:seminaires:seminaire-2015-02-05