aristote

À la croisée des révolutions numériques

École Polytechnique, Palaiseau
Amphithéâtre Becquerel

En route vers l'Exascale !

Jeudi 23 mai 2019

Coordination scientifique :

- **Thierry GOLDMANN** (CNRS/IDRIS)
- **Christophe CALVIN** (CEA)

Renseignements, programme…
www.association-aristote.fr

En route vers l'Exascale !

« En route vers l'exascale ! »

Conférence Aristote du 23 mai 2019

1 INTRODUCTION

Thierry Goldmann (CNRS - IDRIS) Christophe Calvin (CEA)

L'échéance est proche. Le monde basculera bientôt dans l'ère de l'exascale. Exa quoi ? Un monde « exa », c'est un monde en milliard de milliard. « Exa » est le préfixe du système international d'unités qui représente 10^18. Donc un trillion, en échelle longue. Il provient du grec « six » car il vaut 1000^6.

Que l'on parle des projets scientifiques européens, chinois, américains ou japonais, l'échéance est toute proche ! Les supercalculateurs vont voir le jour, approchant ou dépassant les 1000 pétaflops, donc l'exaflop. Soit plus d'un milliard de milliards de calculs par seconde… Et si sur le papier c'est envoutant, dans la réalité, cela va changer énormément de choses pour le monde de la recherche, et de l'informatique de manière générale. *« Il est plus que temps de faire le point sur les enjeux du monde exascale dans le milieu de l'informatique »*, estime Christophe Calvin, président d'Aristote et responsable du secteur simulation numérique et Calcul Intensif au CEA. C'est ainsi qu'est né l'idée de ce séminaire, coorganisé avec Thierry Goldmann du CNRS et en charge de l'assistance à la visualisation des données à l'IDRIS.

L'arrivé de l'échelle « exascale » va remettre en cause de nombreux paradigme informatique, dans la manière de coder, de construire les algorithmes et de penser les univers applicatifs (les projets pour lesquels on se sert des supercalculateurs) et de penser les projets de recherche. *« Ces avancées vont poser plusieurs défis : des défis technologiques, des défis économiques, organisationnels et scientifiques. Le but est de les passer en revue dans cette journée de conférences. Nous avons notamment, et je tiens à le remercier, la chance d'avoir avec nous Laurent Crouzet, du ministère de la recherche, qui travaille sur EuroHPC, et nous dira les dernières avancées du projet. »*

Comme à chaque fois la conférence Aristote sera entrecoupée de questions et d'échanges. Le but est de produire un discours vivant. D'interagir avec l'auditoire, et de ne surtout pas se limiter à de simple présentation.

Editorial Board

Table des matières

2 POLITIQUE NATIONALE ET EUROPEENNE DU HPC

Laurent Crouzet (MESRI)

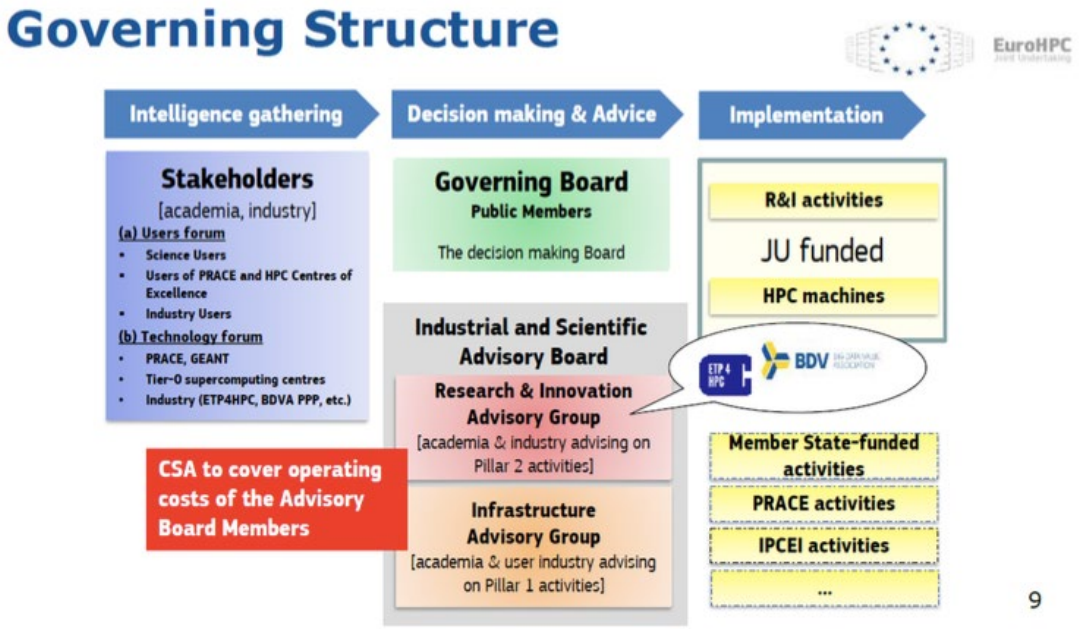
L'Union Européenne est à l'aube de gigantesques projets pour le monde de l'exascale. Laurent Crouzet, Deputy Director of the Mathematics, Physics, Nanotechnology and ICT Department Scientific Advisor for HPC and Digital Infrastructures et membre du service pour la stratégie de recherche et de l'innovation au sein du Ministère de l'éducation supérieure, de la recherche et de l'innovation, revient sur l'état des lieux du projet.

La vidéo de la présentation est disponible ici :

EuroHPC

En 2016, les pays européens ont lancé un vaste chantier : le Digital Single Market in Europe. Le but est de doter le vieux continent de toutes les structures adéquates dans le monde du digital, afin de ne pas dépendre des marchés américains et asiatiques. Le projet vise à créer une infrastructure de dimension mondiale pour stocker et gérer la donnée, d'établir des connexions à haut débit pour transporter toutes ces data, et des structures informatiques de haute performance pour travailler sur ces données. C'est dans cette troisième directive que s'inscrit le projet EuroHPC. Son objectif ? Monter au moins deux superstructures qui atteindront l'exascale, et plusieurs petites, de l'ordre du pétaflop.
Pour construire ces superalculateurs sur le vieux continent, la Commission Européenne, les Etats Membres et des entreprises privées ont constitués une gigantesque *Joint Venture*, avec des modes d'organisation complexes entre les pays pour voter (excepté Malte, Chypre et le Royaume-Uni, simples observateurs).

Si le fonctionnement est détaillé dans le graphique plus bas, il faut retenir que le total des fonds levés dépasse le milliard d'euros investis. Comment ? Pour tous les projets, chaque fois que les Etats membres investissent, la Commission Européenne s'est engagée à mettre au pot la même somme, dans une limite de 486 millions d'euros, financé dans le plan Horizon2020. Ainsi, le budget total public s'élève à 486*2 soit 976 millions d'euros, auquel il faut ajouter 422 millions d'euros issus des entreprises privées. Question planning, le but est de mettre sur pied un supercalculateur fonctionnel d'ici 2020. Plusieurs millions d'euros seront alors alloués ensuite pour développer l'univers applicatifs (ce qui tournera sur ce calculateur) et la Recherche et Innovation. Car le projet se décompose en deux parties distinctes : une sur les infrastructures, où il faut coordonner 28 pays, et une sur le développement de la technologie et de l'univers applicatif qui viendra utiliser ces infrastructures. Aujourd'hui, cette deuxième partie rassemble moins de 200 millions d'euros *« mais nous allons voir pour l'augmenter un peu. Car si nous voulons un programme ambitieux, il nous faut des développements technologiques ambitieux »*, annonce Laurent Crouzet.



Qui pour les héberger ?

Parmi les grandes décisions actuelles, la Joint Venture est en train de choisir quels sont les pays qui accueilleront les supercalculateurs. A ce jour, on peut retenir plusieurs choses : la France et l'Allemagne ne se sont pas présentées car ces deux pays espèrent héberger un supercalculateur encore plus puissant, en cours de projet, lui, dans les années qui viennent. On trouve donc trois candidatures pour accueillir des structures pre-exascale. Un consortium porté par la Finlande et qui regroupe la Suède, la Norvège, la Pologne, la République Tchèque et la Suisse ; et un consortium porté par l'Espagne avec le Portugal, la Turquie et la Croatie. Un troisième consortium italien, avec la Slovénie, a également porté sa candidature. *« Le choix sera compliqué car tout le monde a un dossier solide »*, prévient Laurent Crouzet.
Pour suivre tous les programmes d'investissements, tous les documents et les argumentaires sont publiques. Chacun peut suivre l'évolution des projets. *« C'est un projet d'ampleur, jamais vu en Europe. La Chine, ces dernières années, a bouleversé tous les classements mondiaux, et tous les pays ont vu leur rang remis en cause, à cause de la Chine. On espère qu'avec EuroHPC on restera dans la course, mais surtout, que les machines serviront »*, conclut le spécialiste.

3 UN PANORAMA INTERNATIONAL DES INITATIVES EXASCALES

Par Stéphane Réquéna, Genci et Christophe Calvin, CEA

Que se passe-t-il dans le monde du point de vue des HPC ? Christophe Calvin dresse le bilan et fait l'état des forces en présence. « *A ce jour, ce n'est pas compliqué, quatre zones sont à observer : la Chine, les USA, le Japon et l'Europe* », explique le chercheur.

La vidéo de la présentation est disponible ici

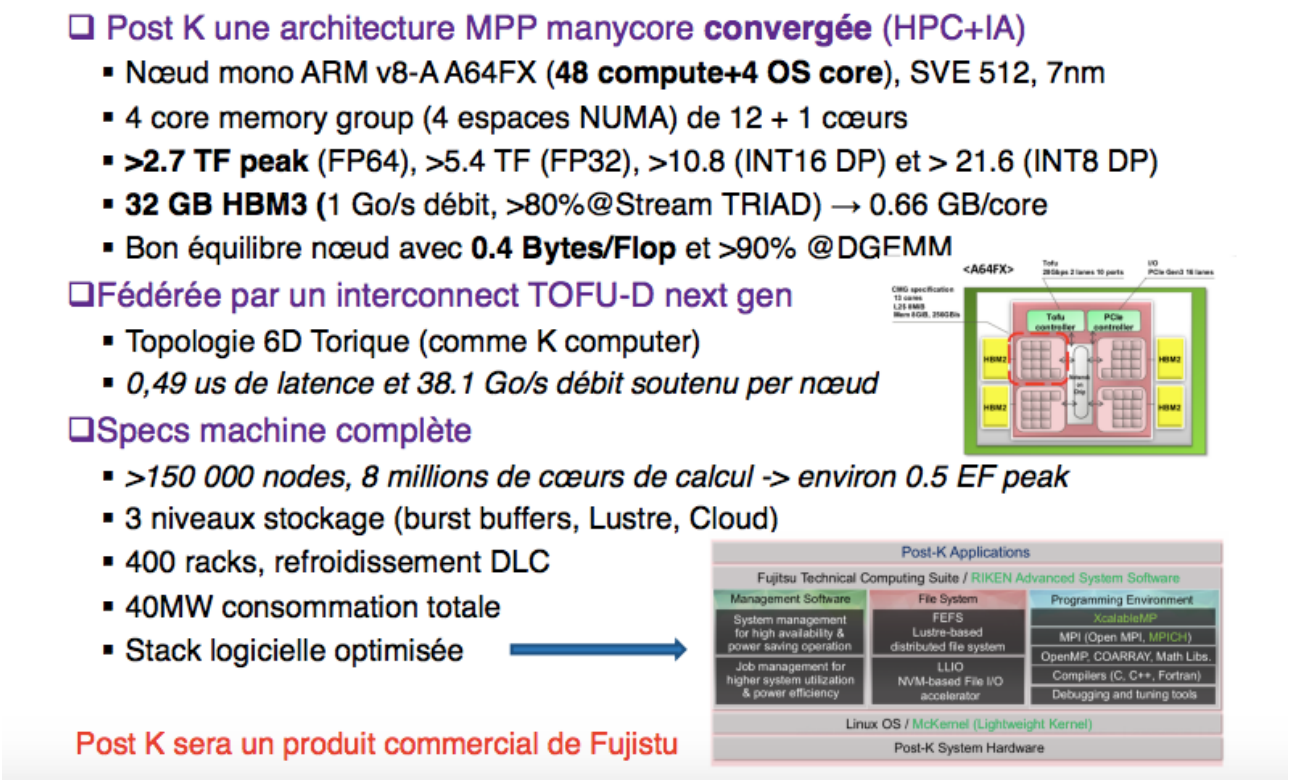
Le Japon : en attente de la suite

La situation au Japon est dans un entre deux, puisque le supercalculateur K Computer s'est arrêté de fonctionner en août 2019. Il montait jusqu'à 11,3 pétaflop/s crête et était installé au Riken, à Kobe. C'était un pure MPP, c'est à dire qu'il possédait une suite de processeurs identiques, des Starck, redesigné par Fujitsu. Il se présente en réseau torique, c'est d'ailleurs une des particularités des machines japonaises. Une autre particularité du monde informatique nippon, réside dans le fait qu'ils conçoivent tout, de A à Z. Ce sont eux qui font le hardware et les logiciels, « *ils fabriquent tout, jusqu'au clous et aux vis* », ironise Christophe Calvin. K Computer délivrait une puissance de 12.6MW. Il a été premier du top500 de 2011 à 2012. Il est maintenant classé 18e. En revanche, il se classe premier du graph500 depuis 2014, deuxième du HPCG depuis 2016.

Mais en parallèle, depuis plusieurs années, se prépare le projet Post-K. Les études ont démarré en 2012, pour lancer le projet en 2014. Depuis, seules quelques infos sont parvenues, au coup par coup. Qu'en sait-on ?

Post-K sera doté d'un budget proche du milliard de dollars. Il aura un processeur ARM (ARM64fx) et un réseau Tofu2 réalisé par Fujitsu. Il devrait être mis en place entre fin 2019 et début 2020 pour entrer en phase de pré-production en 2020, et sera en production pleine d'ici 2021.

Le détail des considérations techniques de Post-K :



Sur quoi travaillera-t-il ? La santé, la prévention des catastrophes naturelles, les questions énergétiques et la compétitivité industrielles, notamment…

Les États-Unis : la grande course

Le contexte économique est là aussi particulier, puisque Barack Obama, en 2015, a signé l'Executive Order Exascale, afin de débloquer des fonds pour mettre en place la National Strategic Computing Initiative (NSCI). En outre, Donald Trump a lancé le 11 février 2019 un plan global de développement de l'intelligence artificielle : l'initiative « *Maintaining/Accelerating American Leadership in Artificial Intelligence* ». En conséquence, les investissements publics en IA vont être augmentés, et surtout, les ressources fédérales HPC devront être mis à la disposition des projets d'IA. Le but est aussi de développer de nouvelles approches de design, HPC et IA, pour des outils davantage opérationnels. Depuis 2016, un total de 1,7 milliard de dollars ont été débloqués. Au total, les travaux sont répartis sur six laboratoires : Argonne, Lauwrence Berkeley, Lauwrence Livermore, Oak Ridge, Sandia, et Los Alamos. Mais toutes ces structures regroupent les équipes de près de 17 centres d'études. Les projets se concentreront dans trois domaines : Structure et intégration, Technologies Logiciels, et application et développement.

	A21	A22	Frontier (OLCF5)	El Capitan (ATS-4)	NERSC-10	NSF Frontera Follow-on
Location	ANL	ANL	ORNL	LLNL	LBNL/NERSC	TACC
Planned Delivery Date/ Estimated	2022 Q1	2022	2022 Q1	2022	2024	2024
Early Operation	2022, Q2	2023	2022, Q3	2023	2025	2025
Planned/Realized Performance (Pflops)	~1,000	1,300 or higher	1500-3000	4000-5000	8000-12000	500
Linpack Performance (PFlops)	800-900	780-1040	900-2100	2000-3000	2000-3000	
Linpack/Peak Performance Ratio (%)	80-90 (est.)	60-70 (est.)	60-70 (est.)	50-60	50-60	50-55
High Performance Conjugate Gradient (PFlops/s)	20.0-22.5	19.5-26.0	18-36	48-72	52-78	74
GF/Watt	40		60-100	134-200	266-480	

Source : Hyperion HPC User Forum, mars 2019

Voici les détails techniques de toutes les installations attendues :
Les installations sur lesquels nous disposons le plus d'informations sont les structures A21 d'Argonne Labs (ANL) et Frontier à Oak Ridge National Laboratory (ORNL).

A21 sera la première structure exascale des États-Unis, fournie par Intel et Cray. C'est un système MPP hybride. Il sera doté d'un budget de 500 millions de dollars et possèdera au minimum 200 racks Shasta Cray ainsi que des next gen Intel Xeon CPU de 10 nm (3D Foveros), couplés à des Intel Xe discrete GPU. Le supercalculateur pourrait atteindre les 1300 pétaflops, voire les dépasser.

Frontier, lui, sera doté de 1,8 milliard de dollars de budgets. Les 1300 pétaflops seront son minimum. Ce sera aussi une structure hybride avec des processeurs AMD. Il possèdera 1 Gigabit

par MPI Task et au moins 8 PB de mémoire totale. Tout cela pour une puissance maximale de 40 Mégawatt (mais de préférence entre 20 et 30 MW).

Notons que la rivalité entre la Chine et les États-Unis fait que les plannings varient en fonction des annonces des uns et des autres…

En Chine : trois projets en compétition

La Chine reste secrète sur ses projets en matière de supercalculateur. Et il est compliqué d'avoir des informations fiables, et donc, de se faire une idée de l'état d'avancée des travaux. Même si on est certains qu'elle a beaucoup d'avance, a massivement investi dans le domaine et prévoit d'investir encore…

Dans le cadre du plan de développement, trois projets de structures sont en compétition, issues de trois équipes différentes : Tianhe-3 (basée sur du processeur Arm), Sunway, dont les équipes ont développé leur propre processeur, et Sugon, qui grâce au rachat de la licence AMD, peut produire ses propres processeurs et du x86.

Voici les détails techniques annoncés des projets en lice :

	Sunway 2020	Sugon Exascale	NUDT 2020
Key User/Developer	Sunway/NRCPC	Sugon/AMD	NUDT
Planned Delivery Date/ Estimated	2020, 4Q (could slip 1-1.5 years)	2020, 4Q (could slip 1-1.5 years)	2020, 4Q (could slip 1-1.5 years)
Planned/Realized Performance (Pflops)	1000	1024	1000
Linpack Performance (PFlops)	600-700	627-732	700-800
Linpack/Peak Performance Ratio (%)	60-70	60-70 (est.)	70-80
High Performance Conjugate Gradient (Pflops/s)	6-7	9.4-10.1	14-16
GF/Watt	30	34.13	20-30
Linpack GF/Watt	20-23	20.9	23.3-32.0

Notons qu'un des enjeux principal de l'exascale se situe au niveau de la consommation énergétique, « *pour éviter d'avoir un réacteur nucléaire à côté du calculateur* », métaphore Christophe Calvin. La grande bataille consiste donc à avoir la structure la plus puissante pour un coût énergétique le moins élevé possible.

En Europe : A quand l'indépendance ?

De nombreux points de l'état des projets européens ont été présentés dans la première présentation sur le projet HPC (cf. plus haut). Mais Christophe Calvin soulève une problématique majeure en Europe : l'indépendance technologique. C'est pour éviter d'avoir à acheter des

processeurs aux autres pays comme les États-Unis qu'a été lancé le European Processeur Initiative, et réduire la dépendance de l'Europe aux technologies étrangères. *« L'enjeu consiste donc à relancer une dynamique industrielle autour du processeur »,* explique Christophe Calvin.

Enfin, dans le monde, de nombreux mouvement ont eu lieu sur le marché. Par exemple, Nvidia a racheté Mellanox, le premier constructeur mondial de réseau à fines bandes. Hewlett Packard a racheté Cray, pour 1,3 milliard de dollars, et enfin IBM s'est offert Redhat, donc la tendance est à la concentration du marché. Pourquoi ? Car ces entreprises rentrent en concurrence directe avec les Gafa, très présents dans le Cloud et donc, des entreprises qui tirent vers le bas le prix de base des composants. En grossissant par acquisition, ils peuvent rester compétitifs sur le prix des processeurs et des mémoires.

En conclusion, Christophe Calvin explique que l'exascale induit des changements dans la manière de procéder, qu'il faut désormais penser les projets de manières cohérentes, en incluant tout le monde, et en travaillant en boucle entre les concepteurs d'architectures et les cibles applicatives. *« On ne peut plus concevoir un code sans savoir sur quelle architecture il va tourner »* explique-t-il. . Ainsi le portage des applications prendra sûrement plus de temps que d'habitude. Peut-être même plus de temps que la fabrication des machines. *« Et il prendra d'autant plus de temps qu'à l'heure actuelle, on ne sait toujours pas sur quelle machine ils tourneront réellement… »* conclut le chercheur.

4 PANORAMA DES PARADIGMES DE PARALLELISATION POUR LES ARCHITECTURES EXASCALE

Par Pierre François Lavallée, CNRS

Exécuter un code informatique sur une structure de supercalculateur pose plusieurs problèmes. La répartition des flux physiques afin de commander les calculateurs et d'exécuter les calculs posent notamment des questions de simultanéité des commandes et de traitement de la mémoire, pour aller chercher et utiliser les informations. Ainsi, lorsqu'on code les applications, il peut se produire des problèmes d'exécution si on ne prend pas en compte cette conception de l'informatique de haut niveau. Et c'est un total changement de paradigme ! Le parallélisme n'est pas un problème que l'on traite à la fin, une fois le code établi, ou en plus, quand on a le temps, à la fin du projet. Il faut avoir, dès le départ, une nouvelle approche du code, et penser son application avec ça en tête, dès le design de la structure de l'application.

L'essentiel de la présentation est disponible ici.

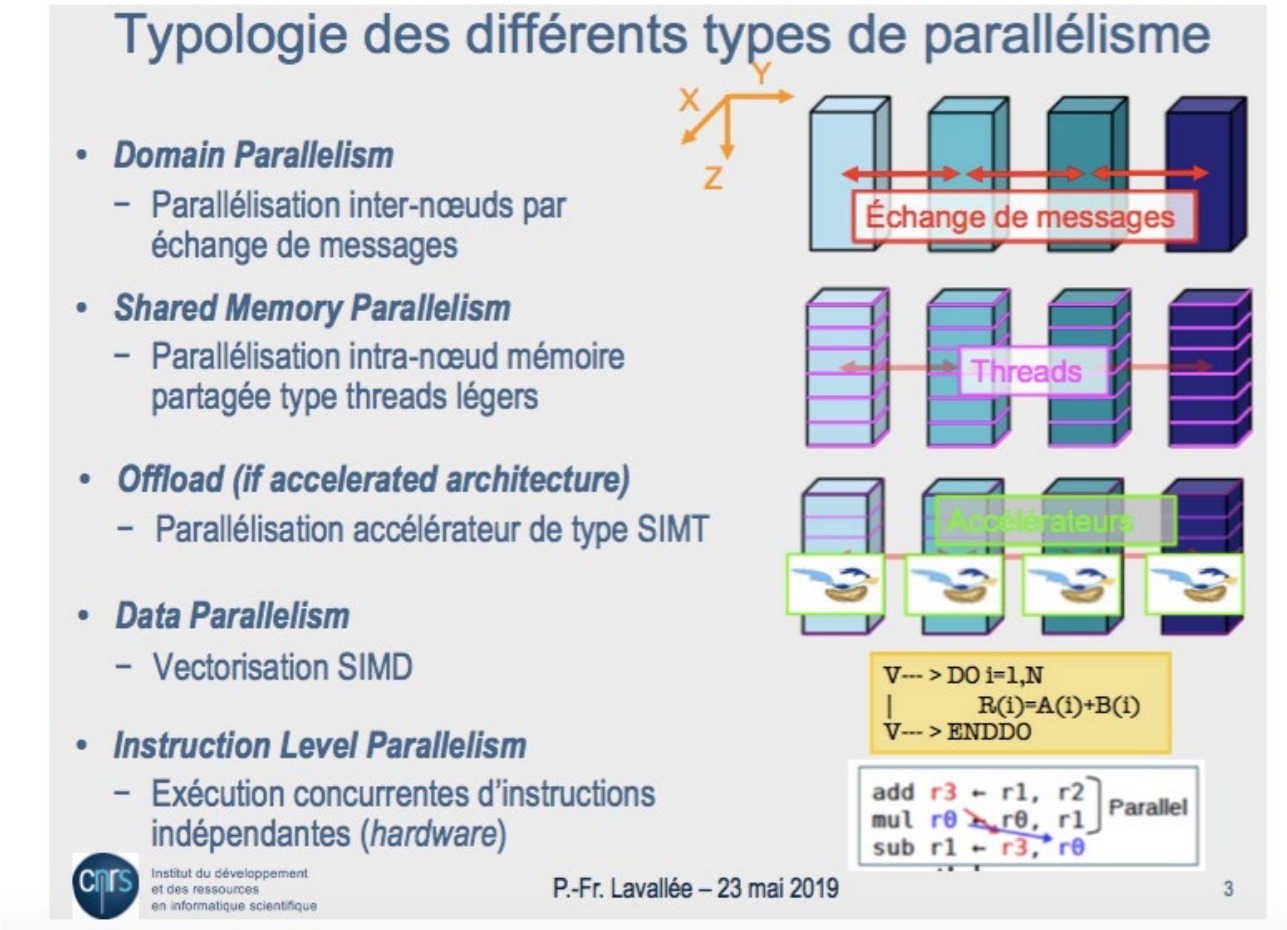
Selon la loi d'Amdahl, une petite partie du programme qui ne peut être parallélisé limite la vitesse globale du programme. Et gagner 1% de vitesse d'exécution sur un supercalculateur qui utilise une quantité astronomique d'énergie (et d'argent) ce n'est pas négligeable ! Le problème n'est donc pas seulement « esthétique » ou « de confort », c'est bel et bien une question de fond, pragmatique, écologique et financière.

Pierre-François Lavallée, chercheur au CNRS vient présenter les principales problématiques liées au parallélisme informatique, et prodiguer des conseils, tant au niveau des protocoles que des langages ou librairies à utiliser pour s'attaquer à cette question.
« *Le parallélisme est une problématique très complexe, car il en existe plusieurs niveaux, qui ne peuvent pas toute être traitée de la même manière* », assure-t-il. Dans un premier temps il dresse un bilan de différentes machines que l'on peut trouver dans les supercalculateurs. Les machines « hybrides accélérées » (Avec aussi bien des CPU que des GPU) comme Aurora A21 aux USA, Argonne, Frontier, à Oak Ridge (USA) et peut être les futurs calculateurs du EuroHPC, ou les machines « homogènes manycores », qui sont soit CPU, soit GPU, mais avec énormément de cœurs de calculs, comme Post-K à Riken au Japon. Dans tous les cas, les deux types de machines ont leurs contraintes communes : le parallélisme « massif » à plusieurs niveaux, et la hiérarchie ou l'organisation de la mémoire.

Différents niveaux de parallélisme

Dans un supercalculateur, le parallélisme intervient à différent niveaux.

1) **Le parallélisme de domaines :** entre les différents domaines de calculs, qui nécessite des échanges d'informations entre les différents nœuds.



2) **Le parallélisme dû au partage de mémoire :** A l'intérieur des nœuds, la mémoire est partagée sur différents « threads » verticaux, et nécessite des échanges de mémoire entre les différents domaines.

3) **L'offload - (sur structure accélérée)** Dans une structure avec accélérateur, il faut les prendre en compte pour le codage et gérer le parallélisme. Une partie du code doit être exécuter sur l'accélérateur, c'est une accélération de type SIMT : Single Instruction Multiple thread.

4) **La vectorisation :** Si vous êtes sur une machine homogène manycores, il faut prendre en compte la gestion de la vectorisation. Notamment via une architecture SIMD (Single Instruction Multiple Data : la même instruction est appliquée simultanément à plusieurs données pour produire plusieurs résultats)

5) **Le parallélisme d'exécutions concurrentes indépendantes** (Instruction Level Parallelism) : ce niveau de parallélisme ne dépend pas du développeur d'application.

La parallélisation idéale

En prenant en compte tous ces niveaux, on peut définir un mode de parallélisation idéal, qu'il faut avoir en tête, sachant qu'il est purement théorique, et donc, inatteignable. Le parallélisme idéal, c'est lorsque tout est parfait !

« Si vous développez un code, vous allez le faire évoluer. Il aura une durée de vie de plusieurs dizaines d'années, devra fonctionner sur plusieurs machines », cadre Pierre-François Lavallée. C'est pourquoi le

paradigme de parallélisation doit être indépendant de la machine sur laquelle évoluera le code. C'est toute la notion de portabilité (qui sera vue dans la conférence d'après).

Ainsi, l'idéal de parallélisation est

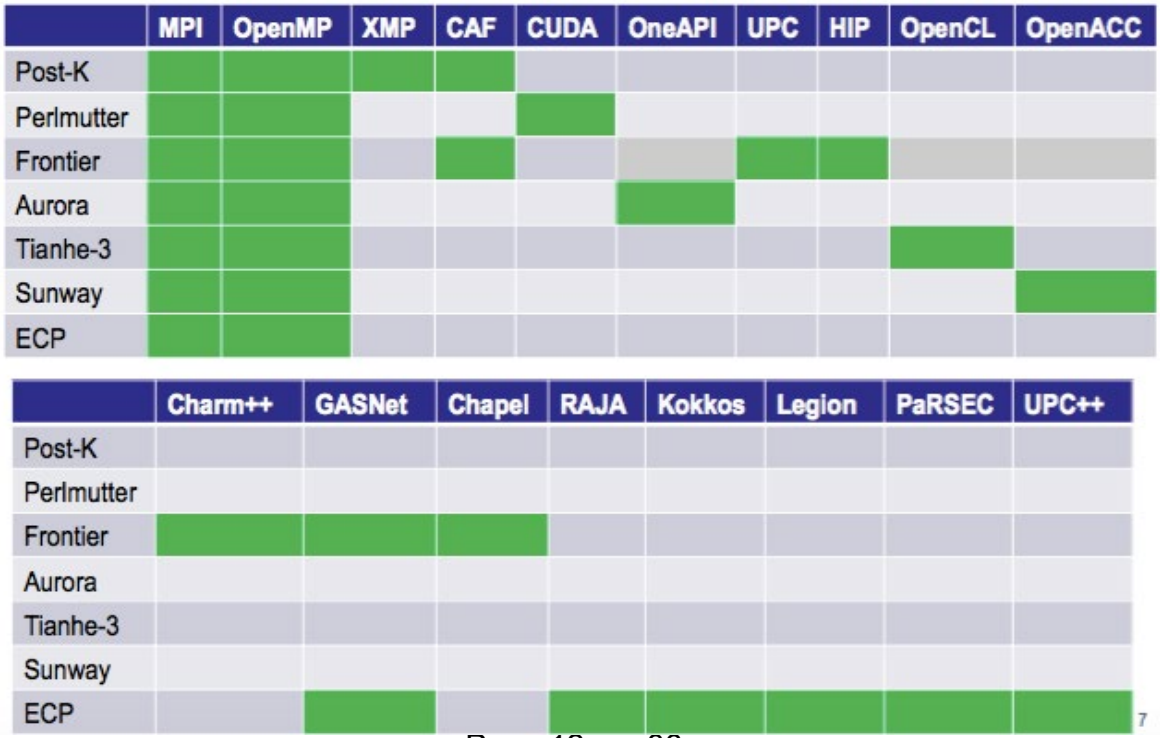
- Portable : il s'exécute sur toutes les machines.
- Performant : Il gère tous les niveaux de parallélisme intra-nœuds définis plus haut, et inter-nœuds (asynchrone, extensible sur un ou des millions de cœurs) et souple.
- Productif : Il optimise la productivité de développement. C'est à dire le rapport entre la performance de l'algorithme et le temps de développement nécessaire. Développez un code parfait sur 100 000 ans ne sert à rien… Ce qui doit lui conférer un caractère accessible, lui permettre de s'introduire dans un écosystème associé (comme les outils de déboggage ou d'analyse de performance).
- Pérenne : Il doit passer d'une machine à une autre, donc être basé sur un modèle normalisé, et normé de facto.
- Compatible avec tous les langages qui seront pris en compte par le supercalculateur (HPC)
- Robuste et stable

« *Bien évidement, aujourd'hui, aucun paradigme de parallélisation ne répond à ces critères,* tempère le spécialiste. *Et aucun n'y arrivera avant les arrivées des premières machines exascales.* »

Mais si on dissocie les niveaux le parallélisme, on peut-on trouver des paradigmes qui répondent à toutes ces contraintes. Et on peut donc, en mélangeant les modèles, atteindre la parallélisation idéale. « *Ainsi une seule stratégie est viable : mêler plusieurs modèles de parallélisation*, conclut Pierre-François Lavallée. *Et s'attaquer à chaque critère de parallélisation différemment.* »

Seul hic dans tout ça : « *l'écosystème de programmation et de parallélisation est complexe. Car les protocoles sont riches. Très riches, voire beaucoup trop riches !* », s'exclame le chercheur. Alors ? Comment choisir et faire le bon choix de protocole sur les machines ?

Quand on classe toutes les machines, et les modèles de programmation compatible avec les architectures (voir tableau ci-dessous), la réponse est assez évidente.



« On se rend compte que les protocoles qui ressortent le plus souvent sont MPI et OpenMP. « A l'inverse, toutes celles et ceux qui ont codé sur Cuda vont devoir recoder leur application… », estime le chercheur. Pas de chance !
Un site performanceportabilty.org permet de mieux faire ses choix, en étudiant et en comparant les niveaux de portabilité. Par exemple, pour le parallélisme de domaine, MPI est le meilleur. Pour la mémoire partagée, c'est OpenMP le plus efficace. Pour l'Offload, mieux vaut passer par OpenMP, OpenACC ou Kokkos, et pour le parallélisme de données (vectorisation SIMD) mieux vaut OpenMP. Globalement, en fonction des priorités de son application, de la longueur des programmes, chacun peut choisir et définir l'architecture la plus adéquate.

De l'intérêt des bibliothèques !

Attention : pour aller plus vite, il ne faut pas oublier que chaque protocole possède sa bibliothèque de programmation. Mieux vaut s'en servir ! *« MPI, par exemple, est très standardisé, avec 879 pages de normes, qui permettent d'exécuter plus facilement des actions »* estime M. Lavallée. Les bibliothèques permettent des optimisations de bas niveau et de meilleures performances impossibles à obtenir à la main, une gestion implicite du parallélisme et des accélérateurs, mais offrent aussi, dès que la bibliothèque n'est pas propriétaire, une portabilité extrêmement simple d'une machine à une autre, sans oublie la robustesse et la stabilité numérique, et, très important: une communauté pour poser des questions sur des forums… Bref, dès que c'est possible, il faut opter pour les bibliothèques.

Les avantages de la bibliothèques MPI, par exemple.

Le chercheur passe alors en revue plusieurs exemples de commandes, et de cas où le plantage peut survenir sans qu'on y prenne garde. (La vidéo est ici)

En conclusion, il faut retenir que le parallélisme doit se gérer dès le départ. Il ne vient pas du nombre de nœuds, que l'on sait gérer depuis longtemps. Le plus complexe se situe en niveau intra-nœuds : pour gérer le nombre de cœur, la vectorisation, l'accélération, l'optimisation de la mémoire en cherchant à accéder à de la mémoire rapide.

En outre, il est illusoire de penser n'avoir qu'un seul niveau de parallélisme. Vous serez obligé de gérer le parallélisme dans votre appli, avec des niveaux différents.

5 MODELE DE PROGRMMATION ASSOCIES A LA PORTABILITE DES PERFORMANCES - C++/Kokkos

Pierre Kestener, CEA Saclay, DRF et de la Maison de la Simulation.

En 2011, tout le monde pensait que l'exascale était déjà finie. Que personne ne parviendrait à aller plus loin. *« Mais rappelons-nous que les prédictions ne sont pas faciles, surtout pour le futur »*, ironise Pierre Kestener. Car le développement de ce que l'on appelle « l'intelligence artificielle », même si c'est un peu un abus de langage, a permis de développer tout le secteur et de donner un nouveau souffle au développement de machine Exascale. Car, ce qu'il faut comprendre, c'est que le but n'est pas seulement de développer une machine performante, mais tout un système économique.

L'essentiel de sa présentation est disponible en vidéo ici.

Ce qu'il faut savoir : Quand on choisit un modèle de programmation, on veut choisir le modèle qui va nous ouvrir des portes. Qui va nous permettre d'aller plus loin. Or, une même implémentation ne peut pas fonctionner de manière optimale sur toutes les architectures. Elle peut perdre en performance d'une machine à l'autre. Ainsi, étudier la portabilité, revient à chercher les points d'entrée, c'est à dire les niveaux d'abstraction, pour écrire un algorithme parallèle, et qui permettront tous d'aller faire fonctionner son application sur une autre architecture, de l'adapter.
La portabilité est un thème en grande réflexion, avec beaucoup de projets de recherche sur le thème, depuis la multiplication des machines.

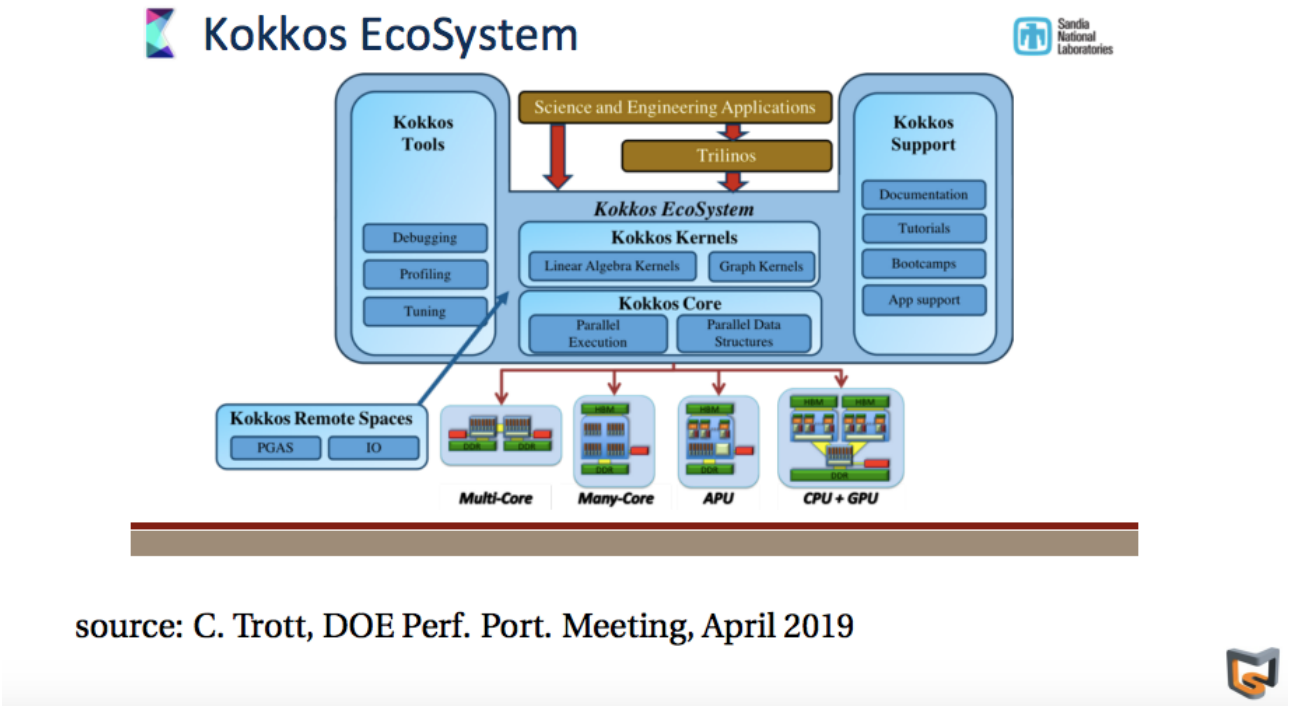
La question : Quelles sont les solutions qui permettent d'obtenir de la portabilité pour les mêmes performances ?

On trouve principalement deux solutions : les approches par parties directives, Open MP, Open ACC. Mais au-delà de ces solutions nominales il y en a d'autres : les librairies. Elles vont fournir des solutions qui ne se situent pas au niveau du compilateur, mais au-dessus, et vont permettre de décrire du parallélisme. Beaucoup de librairies existent depuis plusieurs années, comme Kokkos, Raja, Alpaka, Dash-project, Agency… Toutes les librairies sont écrites en C++, pour un détail simple, c'est qu'elles permettent de se situer à un niveau de métaprogrammation pour masquer certains détails hardware.
« Aujourd'hui, la complexité, c'est que si on veut lancer une application en exascale, il va falloir mixer des approches par directives et des approches par librairies, pour s'attaquer à différents types de parallélismes », estime Pierre Kestener. Il ne faut pas les opposer, elles permettent juste d'obtenir des points d'entrée différents. Mais parfois, avec Kokkos, notamment, tout ce qui est fait en approche par directives, va bénéficier à l'approche par librairies. Car Kokkos permet de générer du code OpenMP ou du code OpenACC. Tout ce qui sera fait en amont, bénéficiera alors de ce qui est fait en dessous.
En outre, il existe aussi des librairies qui permettent de mélanger à la fois le parallélisme en mémoire partagée et le parallélisme en mémoire distribuées. Mais toutes les librairies présentées dans le slide se focalisent principalement sur le parallélisme en mémoire partagée.

La librairie Kokkos

Conçue aux États-Unis, elle s'intéresse uniquement au parallélisme en mémoire partagée. On sait que les machines exascales vont avoir des nœuds de plus en plus gros, et c'est là toute la difficulté : comment gérer efficacement ces nœuds ? Donc Kokkos prend le partie de se focaliser sur ce parallélisme, et laisse le parallélisme en mémoire distribué à MPI. Si la maison de la simulation a investi beaucoup de temps sur cette librairie, c'est parce qu'elle fournit les implémentations pour des pattern algorithmiques parallèles, mais elle fournit aussi des outils pour les conteneurs de données, un atout qui n'existe pas naturellement dans OpenMP/OpenACC. En outre, les

développeurs de Kokkos, font des propositions, qui sont ensuite proposées dans le langage C++. Donc il y a une dynamique qui n'est pas seulement de créer une librairie, mais aussi d'avoir une vision long terme de développement du langage. Cette librairie, n'est pas née d'un projet ex nihilo, mais est issue d'un travail d'algèbre linéaire, nommé Trilinos. Kokkos, c'est aussi tout un écosystème : des outils pédagogiques, des aides aux débogages etc.



Pierre Kestener tente ensuite de démontrer l'utilité de Kokkos pour les conteneurs de mémoires, en prenant l'exemple d'un tableau de données multidimensionnels, avec une boucle *for*, qui va tourner sur GPU ou sur CPU. « *En Fortran, un tableau 2D, c'est quelque chose qui n'existe pas, c'est une abstraction vers une zone mémoire, fondamentalement indépendante. Des langages comme Fortran ou C++ ont fait des choix différents, et Kokkos prend partie de dire que ce n'est pas un choix qui doit être imposé, mais qui doit être décidé uniquement à la compilation, de façon à s'adapter à l'architecture sur laquelle on exécute le programme* », résume le chercheur. Le détail de son exemple et de sa démonstration est ici.

Ce qu'il faut retenir, c'est que Kokkos va permettre d'imbriquer différents niveaux niveaux de parallélisme, qui vont s'adapter à la hiérarchie des architectures. Ainsi, Kokkos propose dès le départ de mettre le parallélisme au cœur de la réflexion. Le mettre en avant, et à l'origine de la réflexion. Et en programmation, si on a la possibilité de démarrer de zéro en prenant en compte le parallélisme, c'est mieux. C'est un gain de temps, et cela offre une meilleure approche, surtout pour le passage à l'exascale.

En faisant un test sur l'exemple du noyau de la chaleur, Kokkos permet de faire différents tests, en jouant avec les boucles *for,* par exemple, et d'adresser beaucoup de questions, tout en étant très productif. En jouant sur les différentes hiérarchies, on peut optimiser la bande-passante. Et ces mêmes codes peuvent être recompilés sans aucune adaptation, pour un GPU, et ainsi permettre d'obtenir d'autres courbes.
A la maison de la simulation, Kokkos est très utilisé en dynamique des fluides (pour modéliser l'atmosphère des exoplanètes, ou en physique solaire…) Pierre Kestener revient alors sur un exemple de simulation pour un écoulement à très haut nombre de Reynolds, et pointe les différences de performances obtenues en CPU et GPU. A voir sur sa présentation.

6 PROGRAMMATION A BASE DE TACHES : QUELS DEFIS POUR LE PASSAGE A L'EXASCALE

Raymond Namyst, de l'université de Bordeaux, Inria STORM

Dans le monde du code, il existe une manière spécifique de penser les algorithmes : la programmation à base de tâche. C'est une technique de codage qui consiste à découper les actions pour théoriser l'application, qui va offrir, parfois, de nombreux avantages.
Quand on parle de partage des tâches, on parle d'un découplage qui n'est pas uniforme selon les applications. L'algorithme de « tâches » peut être écrit de manière récursive, et non forcément à plat. Si la programmation à base de tâche existe depuis longtemps, c'était davantage en tant qu'objet d'étude. Mais de nos jours, elle est de plus en plus adoptée dans le monde des supercalculateurs. Car elle y est particulièrement adaptée.

L'essentiel de la présentation est disponible ici.

Le principe ? On soumet des tâches à un support d'exécution, à qui on doit faire confiance. C'est lui qui pense l'agencement des actions. On va lui déléguer l'ordonnancement des tâches. Cela veut dire qu'on ne va pas contrôler ce qui se passe en dessous, c'est lui qui va savoir comment placer les tâches sur la machine. *« Ce qui n'est pas facile dans l'esprit des programmeurs ! »*, explique Raymond Namyst. On soumet donc les tâches, qui vont former un graphe, car elles ne sont pas toutes indépendantes, et certaines ne peuvent pas s'exécuter tant que d'autres ne sont pas finies.
Il y a des supports, par exemple, dans OpenMP, qui permettent de pouvoir connaître les relations de précédences entre les tâches, en fonction des données auxquelles vous faites référence.
Cette manière de procéder s'appelle le Sequential Task Flow (STF). Mais ce n'est pas la seule : on peut le faire soi-même de manière canonique, ou algébrique.

Le but pour le support d'exécution (SE), c'est de placer les tâches de manière optimale sur une machine, ainsi, il va surtout mettre en oeuvre une politique d'ordonnancement. Plusieurs modes de fonctionnement existent : les SE vont placer les tâches et ensuite faire du « vol de tâches » lorsqu'il y a des pénuries de calculs sur certaines unités de calcul. Des ordonnanceurs essayent de prédire les durées des tâches à l'avance, un peu comme un diagramme de Gant, afin de remplir la machine comme un tétris. D'autres font confiance à la manière avec laquelle les tâches ont été distribuées, et vont essayer de placer les tâches par affinités.

Pierre Namyst montre ensuite un exemple datant de 2008, portant sur la factorisation de Cholesky, pour expliciter comment on arrive à déduire de relation de dépendances entre les tâches et à construire un graphe. L'exemple est ici.

Quel est l'intérêt de la programmation par tâche ?

La portabilité : les programmeurs n'ont plu à prévoir la gestion des blocs de calculs à l'avance.

L'efficacité : on a accès à une dépendance plus fine entre les tâches quand on était par le passé bloqué par des barrières de synchronisation.

L'expressivité : avec la récursivité du processus.

Seul bémol : la compréhension des performances. Quand un programme ralentit, ou affiche de mauvaise performance, comprendre le problème nécessite de rentrer dans un vrai monde de douleur. Et à ce jour, on manque d'outils pour explorer et connaître d'où vient le problème.

On peut ainsi observer l'exemple sur une modélisation de choc moléculaire, où la programmation par tâches a été très utile en découpant l'espace en petite boîte indexées différemment, et en exécutant le programme boîte après boîte.

L'essor des tâches

Pourquoi les tâches sont en plein essor ? Car elles se prêtent bien aux architectures hétérogènes (CPU + GPU). Comme les tâches ont écrit les données en entrée et en sortie, et si elles ont besoin des données en entrée et en sortie, on sait exactement de quoi elles ont besoin avant de s'exécuter soit sur un GPU, soit sur un CPU. Elles permettent donc de gérer les transferts CPU/GPU convenablement. On n'a pas besoin de faire de l'offloads, ou du transfert de mémoire, on peut donner l'information à l'avance. Ce qui est énorme atout.
On peut également spécifier les implémentations à l'avance, ce qui n'est pas possible dans le domaine de l'algèbre linéaire. Et surtout, il est possible de laisser le support d'implémentations choisir l'unité de calcul cible.

Raymond Namyst revient alors sur un exemple sur lequel il a lui-même travaillé : StarPU. Lancée en 2007 il a eu une première version en 2008 : l'idée c'est de faire un suppor où les tâches possèdent plusieurs implémentations. Le système utilise un cache logiciel pour minimiser les transferts CPU et accélérateur, ce qui est plutôt important dans ce type d'application. Pour voir la vidéo de l'exemple, c'est ici. Ce qu'il faut retenir : Placer les tâches sur l'architecture reste assez simple. Le tout se complique lorsqu'il faut prendre en compte le temps de transfert de données. Beaucoup plus difficile à prédire…

Mais, avec l'université du Tenessee, il a été possible de tester StarPU. Résultat, sur 12 CPU, à peu près 200 gigaflops ont été obtenu en plus grâce à une meilleure répartition des tâches. Et chose surprenante : on peut obtenir plus de flops que la somme des flops que chaque unité !
« Ce n'est pas de la magie, tempère Raymond Namyst, *c'est une factorisation. Cela vient du fait que la machine est hétérogène, et que les tâches ont été réparties la ou elles s'exécutent le mieux. »*

A retenir, également, les tâches gagnent du terrain. Car elles sont applicables dans de nombreux domaines (algèbre linéaire creuse, Calcul in Situ)… Mais parmi les avantages, le développement à base de tâche permet de très bien simuler sur une machine artificielle, qui n'existe pas le bruit que l'on obtiendra, par exemple, sur une vraie machine. (SimGrid, par exemple) Ou encore de réfléchir à l'architecture des machines, car l'ordonnanceur peut travailler et répartir les tâches, sans les lancer réellement, mais sur une machine théorique.

Tout n'est pas rose. A l'intérieur des nœuds il se passe des choses qui compliquent la vie des gens, davantage qu'à l'extérieur, où des MPI arrivent bien à suivre les performances du matériel. Dans les nœuds, c'est une autre histoire, et les unités de calculs vectorielles sont larges. La question se passe de savoir ce qu'il se passe sur une machine aux dimensions plus grandes. *« La répartition des tâches est-elle encore optimale sur une machine de 5000 cœurs ? »,* s'interroge le spécialiste. Des problèmes pourront peut-être survenir au niveau du transfert de données. Et est-ce bien raisonnable de vouloir calculer la date de fin de chacune des tâches sur chacun des cœurs, quand on a beaucoup de cœurs ? « Quand on gouverne un pays, on délègue mais on n'ordonnance plus les choses comme on le faisait avant », métaphore Raymond Namyst. Cela pourrait être trop couteux que de regarder ce qu'il se passe dans chacun des cœurs.
Quels défis et quels conseils pour s'y mettre ?

Pour se mettre à la programmation par tache, il faut accepter de ne pas avoir une vision globale de l'activité des cœurs.
De plus, si on veut lancer sur une même machine plusieurs applications (ou bibliothèques), qui n'ont pas été programmées par les mêmes personnes, il va falloir co-ordonnancer plusieurs

bibliothèques en parallèle (par exemple, une factorisation de matrice, avec une transformée de Fourier…) Et exécuter du code, pour définir une frontière, même mouvante, entre les deux applications.

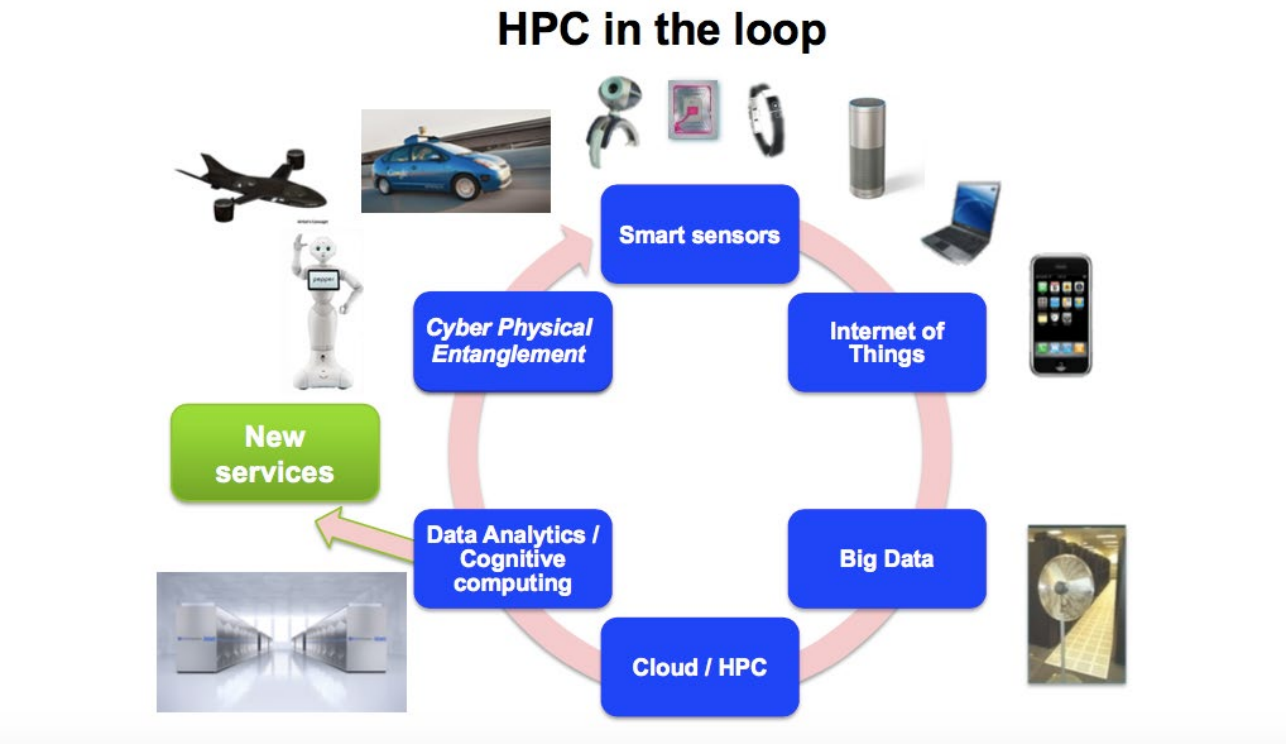
L'autre principal défi : gérer la granularité des tâches. Le déséquilibre de puissance de calcul entre les machines peut coûter cher, si vous exécutez une tâche longue sur un CPU, ça peut vite dériver si votre ordonnanceur se trompe… L'erreur coûte cher dans ce système. Pour cela il faut aller fouiller la granularité des tâches, et à ce jour, les développeurs ont développés leurs propres techniques.

Mais pour aller plus loin, et mieux appréhender la granularité des tâches, il est possible de créer des clusters de CPU, sur lesquels on va effectuer certaines tâches. Cela demande d'effectuer des exécutions parallèles, les tâches deviennent alors malléables, et l'écart se réduit avec les accélérateurs. Seul hic, détermine la taille et le nombre de clusters optimal n'est pas chose aisée, du tout… Quelques travaux ont été effectués sur ce sujet, avec des algorithmes d'optimisation des clusters, mais on n'a que peu de lisibilité sur le résultat.
L'autre technique peut consister à effectuer un diagramme de tâche avec des tâches qui se déclinent en sous graphe. Raymond Namyst termine alors avec un exemple précis sur les équations de maxwell, disponible ici.

7 VERS UNE SOCIETE POST EXASCALE

Marc Duranton et Denis Dutoit, du Commissariat à l'Énergie Atomique et aux énergies alternatives

Le chercheur Denis Dutoit présente des réflexions sur l'avenir. Que deviendra le secteur quand l'exascale sera atteint ? *« Ce ne sont pas des réflexions personnelles, mais des réflexions que l'on a au sein de l'ETP4HPC, et qui sont le résumé d'un document réalisé conjointement avec BDVA et HiPEAC et présenté en Pologne, lors de la semaine HPC ».*



Le document est trouvable en détail ici.
L'essentiel de la vidéo est disponible ici

Il est également inspiré du rapport de vision de l'Hipeac 2019, disponible et téléchargeable ici.

Vers un continuum

La première des choses, c'est que le monde exascale ira vers un *continuum*. Un cycle continu. C'est à dire que les applications commencent maintenant dès le départ, au niveau des capteurs, elles génèrent des données collectées par des systèmes IOT, qui crée des ensembles de données, du Big Data, traitées ensuite sur le cloud ou sur des architectures HPC, avec des approches d'intelligence artificielle, et c'est cela qui génère de nouveaux services ou de nouvelles applications.
La différence, c'est qu'aujourd'hui, on utilise ces approches pour contrôler des nouveaux systèmes physiques (usines, systèmes, voitures).. C'est là que le HPC devient intéressant car on utilise des systèmes informatiques pour modéliser et prédire le comportement de phénomènes réels, et avec des contraintes dedans.

Autre point important dans le domaine, le traitement informatique n'est pas fait que sur HPC mais est réparti sur toute la chaîne. On fait donc une analyse en temps réel de la donnée, dès qu'elle est collectée. Cela permet de la traiter lorsqu'elle est le plus efficace. Cela réduit le temps de latence, augmente la sécurité, réduit les besoins en bande passante (pas besoin de gros tuyaux), réduit

l'accès et le partage des données personnelles ou confidentielles, et cela permet aussi de réduire l'énergie utilisée…

Tout cela demandera une chose : l'interopérabilité des systèmes. Ainsi le post exascale, ce sera toujours de la simulation, mais avec en plus le traitement des données, et des systèmes d'apprentissage par machine learning. Et il faudra savoir tout gérer.

La courbe du Hype et les assistants personnels

Les assistants personnels, sont, au niveau du consommateur, un aspect visible de l'intelligence artificielle. Pourquoi ? Car il c'est une des facettes visibles partout (un milliard de dispositifs Android, 500 millions chez Apple, sur les *device* avec haut-parleur comme Amazon Echo, Google Home, ou encore Baidu, qui a vendu autant de dispositifs qu'Amazon).
Tous ces avancées ont été rendues possibles car en 2005, les progrès en deep learning ont permis de réduire l'erreur en termes de reconnaissance de parole. C'est ce qui a donné l'idée à Google, à l'époque, de faire leur propre système. Pourtant, au même moment, Google mettait en garde : « *Si chacun de nos utilisateurs utilisent cet assistant personnel, pendant 3 minutes par jour, il faudra qu'on double le nombre de nos serveurs* ». Et on imagine bien qu'au niveau des infrastructures, c'est hors de prix, qu'au niveau de l'alimentation électrique aussi… Donc il faut faire quelque chose… C'est depuis ce jour que Google est sorti du logiciel pour faire du hardware. Par exemple des structures de 180 Teraflops. (Google's TPU2). Ils ont obtenu ainsi une structure de 11,5 pétaflops pour près de 400 KW. Et chaque année, ils ont de nouvelles structures, avec une puissance de calculs qui double. Des structures plus complexes, aussi, avec une hybridation, et parfois du watercooling. De même, ces machines ne supportent pas l'I3E (le floating point 16), mais ce que Google appelle le bFloat 16, qui veut dire Brain Floating Point Format. C'est format sur lequel l'exposant repose sur 8bits, ce qui permet d'avoir une plus grande dynamique car c'est ça qui est important pour le deep learning.

C'est ce type de machine qui a permis de mettre en place Alphago, l'intelligence de DeepMind, qui a pu apprendre le jeu de go, et battre le champion du monde, mais sans aucune base de données, simplement en lui explicitant les règles du jeu de go. Après 21 jours, Alphago battait le maître et au bout de 40 jours ils l'ont débranché… Alphago dépassait un « hélo » de 5000, soit le meilleur classement de joueurs jamais atteints par les humains jusqu'à présent. Donc a priori, on peut affirmer que cette machine est imbattable au jeu de go. Mais il y a un petit effet de bord : Alphago tournait justement sur Google TPU V2. Alors, si pour la partie entraînement, il a fallu 16 TPU de 180 TFlops, soit 2,8 Pétaflops, pour la partie jeu, où l'intelligence se bat contre un adversaire, cela nécessite 5000 TPU à 90 Tflops soit 450 Pétaflops. Et là, on doit considérer les 200 Kilowatts de puissance, juste pour les structures, sans même évoquer le refroidissement nécessaire, et pendant 40 jours… Devant toutes ces capacités technologiques, On peut affirmer que oui, Google fait du HPC.

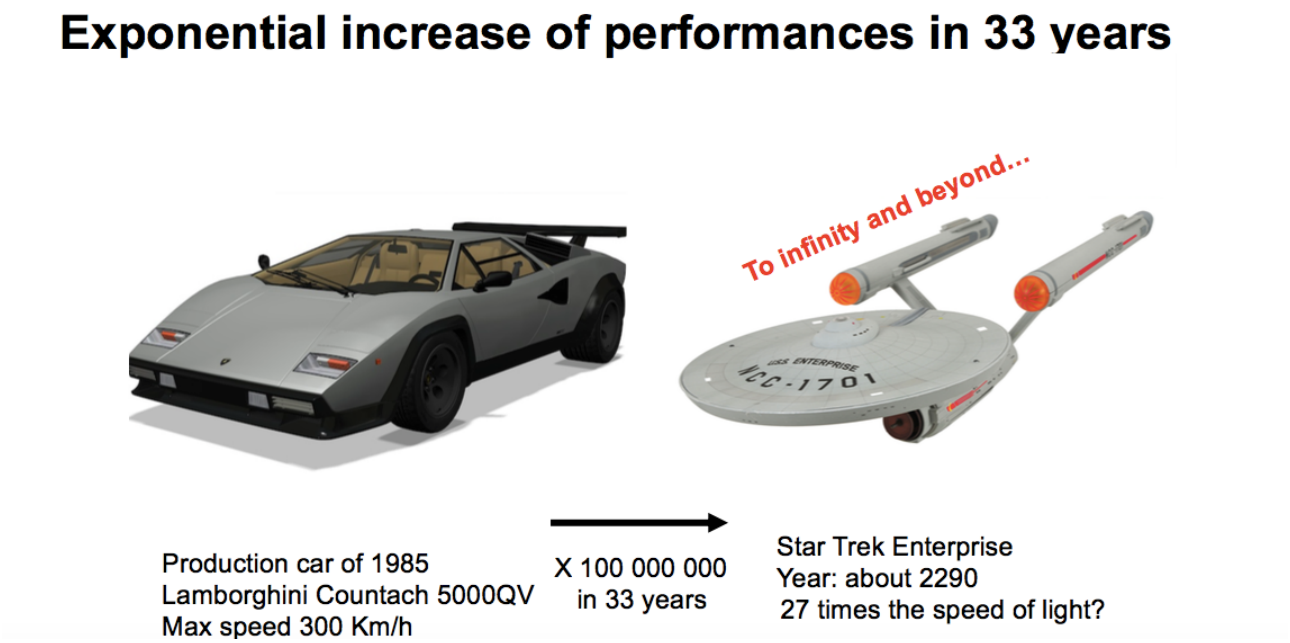
HPC et IA

Le chercheur revient ensuite sur les différents modèles d'apprentissage pour une intelligence artificielle, entre données de diagnostic médical, ou alphago. La présentation est disponible ici.

Ce qu'il faut retenir : On voit de plus en plus de pont entre le monde HPC et l'IA, car on utilise de plus en plus de simulations pour améliorer l'IA et inversement, on utilise des approches d'IA pour améliorer les simulations. Par exemple, l'IA peut aider à déterminer les meilleures conditions initiales, et améliorer la rapidité de simulation.

Des machines hétérogènes

Quand on pose la loi de Moore, on se rend compte que l'accélération de la micro-électronique a été très très rapide. Entre le calculateur Cray en 1985 et Summit en 2017, la puissance de calcul a été multipliée par 100 millions. Si on devait faire une analogie avec le monde du déplacement, en prenant comme référence de départ la Lamborghini de 1985, on devrait avoir en 2019, le vaisseau de Star Trek, qui va à 27 fois la vitesse de la lumière.



Le problème ? C'est que la loi de Moore n'est pas valable. On ne peut pas doubler éternellement toutes les performances en informatique par des avancées technologiques. Car si vous réduisez d'un facteur *a* les performances, la physique, dans les détails est plus compliquée que la théorie… Par exemple, le voltage reste sensiblement le même, la puissance du circuit ne diminue plus au carré etc. Pourquoi ? Car il y a au niveau des transistors, des fuites capacitaires.

Alors pour améliorer le tout, en expliquant d'ou viennent les fuites capacitaires dans les transistors, Denis Dutoit explique les différentes avancées qui pourront permettre d'améliorer encore l'informatique. Au total, ce sont ces modifications qui permettront d'augmenter la densité de transistors par mètre carrés.

Des machines hybrides

Pour réduire la consommation énergétique des machines, les architectes produisent des machines de plus en plus spécialisées. Donc avec davantage de GPU. La plupart des machines du haut du top 500, sont ainsi des machines hybrides.
Ainsi, la machine de demain sera un orchestrateur entre différents accélérateurs. Même si cela pose des questions d'un point de vue logiciel…

L'autre évolution des machines se situera aussi au niveau des mémoires, et notamment des mémoires non volatiles (magnétique, thermiques…). Le but de ces mémoires : aller aussi vite que des Ram ou des mémoires flashs, tout en consommant très peu d'énergie. Le gain sur la structure est substantiel, car pour économiser de l'énergie, il faut éviter de déplacer les données.

En revanche, cela aura des conséquences importantes dans l'univers applicatif, car d'applications prennent en compte le fait que d'accéder à la mémoire prend plus de temps que de faire le calcul. On pourrait ainsi avoir à tout recoder pour gérer ces problématiques de synchronisation…

De manière générale, que ce soit au niveau des accélérateurs, ou de mémoire, pour économiser de l'énergie, c'est toujours la même idée, il faut réduire la distance.

Le chercheur revient ensuite sur le projet EPI, pour European Processor Initiative (présentation ici) et sur les approches optiques. La différence : créer un photon n'est pas cher du tout, mais le transporter, oui, car avec la loi d'ohm, il y a beaucoup de dissipation. A l'inverse, les électrons coûtent cher à produire, mais ne coûtent rien une fois qu'ils sont lancés… Mais les dernières avancées consistent à faire des cartes en Off-Chip, avec des photons. Il faudra surveiller les avancées dans ces deux disciplines car elles pourront à termes avoir divers impacts.

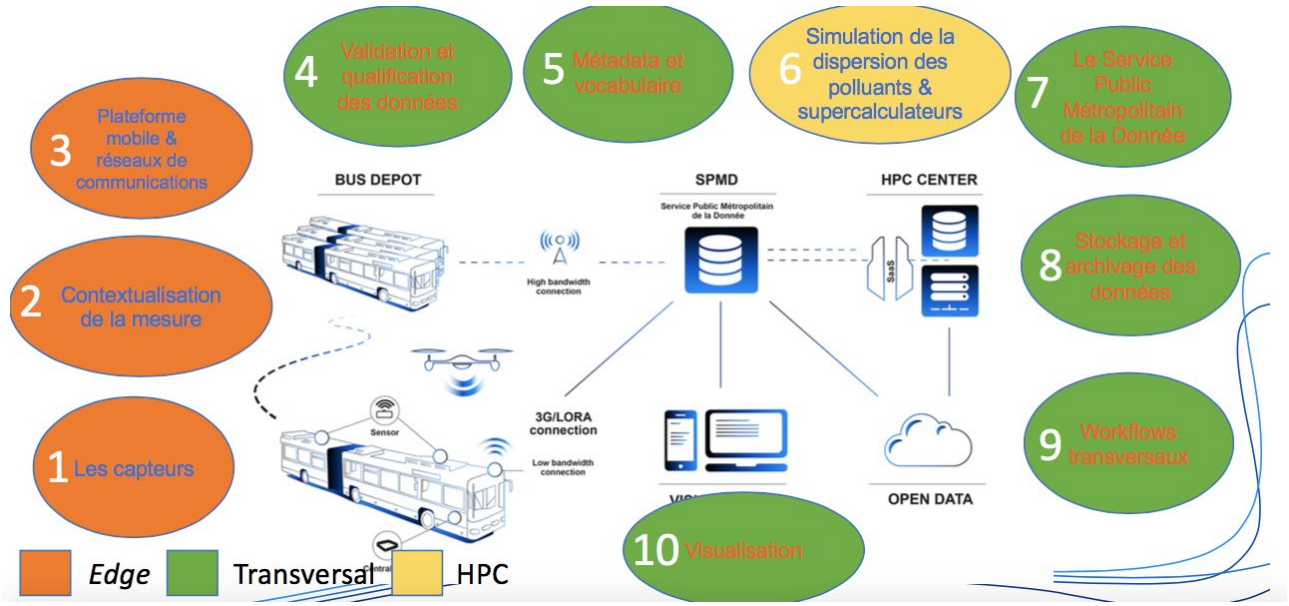
8 Du CAPTEUR AU SUPERCALCULATEUR, POUR UNE MEILLEURE COMPREHENSION DE LA QUALITE DE L'AIR

François Bodins, Rennes1

François Bodin, de l'université de Rennes1 a lancé un projet de start-up, avec la métropole de Rennes pour trouver une solution afin de faire un suivi de la qualité de l'air. La problématique est simple : outre le fait de mesurer la qualité de l'air dans le temps et l'espace, sur l'agglomération rennaise, les équipes cherchent à analyser la dispersion des polluants par simulation numérique, et, dans une optique d'Open data, de mettre toutes les données à dispositions du grand public. C'est ainsi qu'a été lancé le projet AQMO (Air Quality & MObility).

La vidéo de la présentation est visible ici.

Dans une première phase, le projet se concentre, pour des raisons de budget, sur la pollution aux particules fines. AQMO consiste donc à mettre en place un système de captation et de gestion des données observées. « *Il faut faire une différence entre capter une donnée et mesurer* », explique le chercheur. Et notamment lorsqu'on va travailler sur un très grand nombre de data, pour lancer des grandes simulations numériques. Le projet réunit un grand nombre d'acteurs différents, tous spécialisés sur un domaine précis.
Deux solutions se posaient alors aux équipes : soit mettre en place de nombreux capteurs peu couteux sur une vaste étendue, et ensuite faire des statistiques pour corriger les biais des résultats, soit mettre en place peu de capteurs, mais plus précis et plus fiables, sur des unités mobiles. Mais il faudra alors gérer la question de la mobilité du capteur et les problèmes afférents. En l'occurence, les équipes d'AQMO ont opté pour la deuxième solution, en plaçant les capteurs sur des bus, et se sont donc alliés avec la société de transport rennaise : Keolis.



Si l'on schématise le projet, la partie HPC n'intervient que dans une petite portion des étapes à suivre. Mais tout ce qui se passe avant, a forcément une influence.

François Bodin insiste sur la différence entre capteur et mesure. Une valeur captée ne devient une mesure qu'à partir du moment où elle a été calibrée et vérifiée. Cette question est primordiale, pour gérer l'exactitude des résultats, surtout face à un enjeu politique, comme la pollution. « *Si*

vous indiquez à la métropole que les enfants dans une école respirent trop de particules, vous avez intérêt à être sûrs que ce soit vrai », explique le chercheur.
Ainsi, le choix des capteurs est très important. Le budget rentre forcément en compte, mais aussi les conditions d'utilisation des systèmes. Elles vont avoir un impact sur les choix à faire. Autre point d'illustration des différences entre capteurs et mesures : un pic, qui apparaît dans les résultats. Que vaut-il ? Pourquoi y aurait-il un pic de pollution à un moment donné, à un endroit précis du parcours du bus ? *« Il est fort probable qu'il soit bloqué derrière un autre bus, donc à un endroit où la pollution est la plus forte »,* explique le chercheur. Et pour savoir si c'est une erreur de mesure, ou un paramètre contextuel exceptionnel, les chercheurs font appel à de la reconnaissance d'image pour savoir quels véhiculent entourent le bus à chaque moment. Cette analyse est faite dans le bus, pour économiser de la bande passante et ne pas avoir à envoyer l'image à chaque fois. Le chercheur indique tout cela ici. De manière générale, chaque donnée doit être qualifiée avant d'être stockée, pour ne pas soit utiliser de l'espace de stockage, soit de la bande passante. Les données sont ainsi notées selon un référentiel précis.

Vient ensuite le moment tant attendu de la simulation. Là, à partir du stock de données qualifiées et en fonctionnant sur des CPU, on peut obtenir différents modèles de pollution, du plus fin (quelques heures par CPU) au plus précis (30 000 heures de calcul par CPU par jour simulé), avec une grille fine de l'ordre de 3 mètres. Les calculs ont lieu sur un supercalculateur (exemple la machine Turing de l'IDRIS) mais via un mode de fonctionnement différent. Le projet ne peut plus fonctionner par Batch pour être accessible aux métropoles, et surtout de permettre de pouvoir lancer les calculs en cas de catastrophe, par exemple, un gigantesque incendie. AQMO tente donc de fonctionner comme un software as a service (SAAS). Car il ne faut pas oublier que pour fonctionner correctement, le temps de réponse est essentiel. Dans le process, dès qu'un paramètre peut permettre de gagner du temps, il doit être optimisé : définir la réquisition des moyens au préalable, adapter les profils d'exécution, précalculer ce qui peut l'être, intégrer les données dans l'infrastructure de calcul, lancer des tests régulièrement…

Le projet nécessite également de considérer le traitement en flux de données, réactualisées régulièrement, pour pouvoir combiner la simulation à de l'assimilation de données, pour calculer, par exemple, l'évolution de la pollution, et la dispersion des particules.

Au-delà du suivi de la qualité de l'air, AQMO pose un grand nombre de questions sur l'organisation autour des données et du calcul. Le projet travaille par exemple avec le service métropolitain de la donnée. C'est lui qui a la charge de la diffusion des données ouverte (open data) et qui doit définir une forme de gouvernance, mais c'est lui aussi qui permet la consolidation avec les sources citoyennes comme Ambassad'Air à Rennes, et le projet de Smart Citizen de Barcelone. En outre, François Bodin explique comment sont stockées, et où sont archivées les données et revient sur les différents modèles de visualisation. Ici.

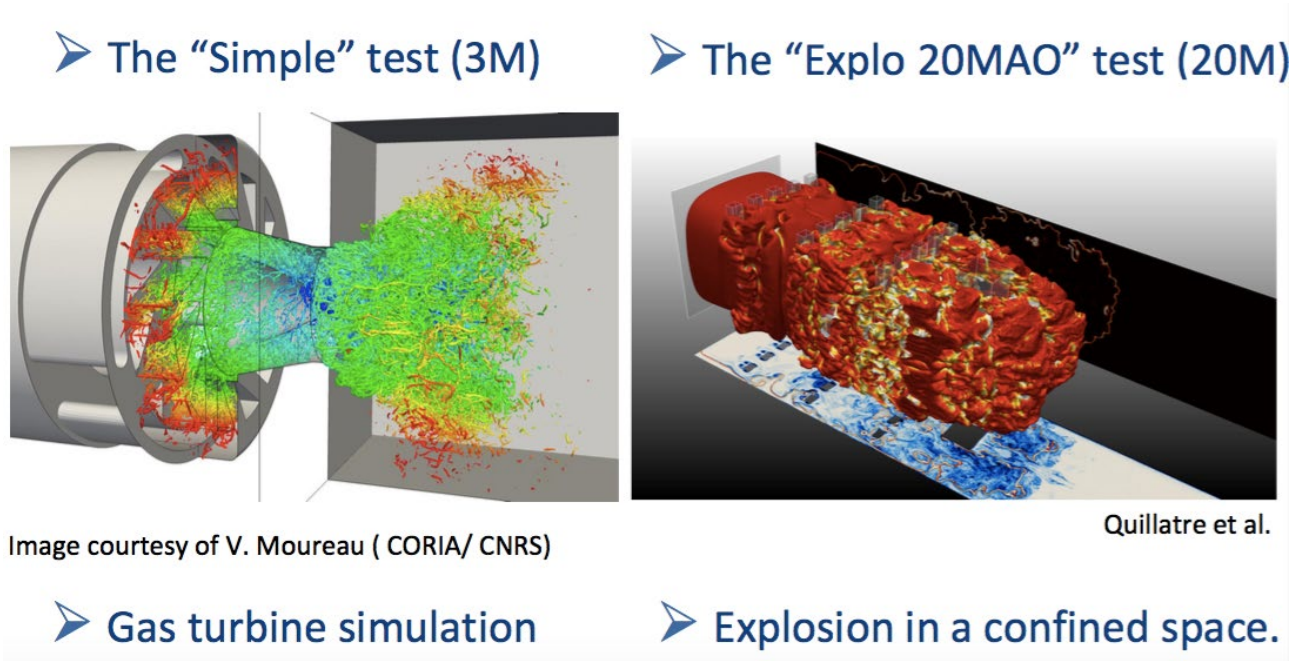
En conclusion, à ce jour si un seul bus est équipé de capteurs, 20 sont attendus d'ici 2020. L'intérêt de ce projet est de questionner la notion de mesures lorsqu'on doit faire des simulations avec un grand nombres de données issues de capteurs physiques. Il faut alors gérer l'erreur dans tout l'écosystème. Cela pose une question de gouvernance de la donnée, de toute sa logistique, et de la pertinence des modèles avec le temps. A terme, et étant donnée le caractère publique, politique et ouvert du projet, il faudra permettre un cycle d'échange des données entre de nombreux acteurs garantissant des données de qualité.

9 PORTAGE SUR GPU D'UN CODE FORTRAN HPC GRACE A OPENACC

Par Jeoffrey Legaux, du Cerfacs.

Le but est de voir comment porter sur un GPU un code écrit en Fortran, en passant par le protocole OpenACC.
La présentation est disponible en vidéo ici

Face au développement des GPU et aux gains de performance qu'ils apportent à l'informatique, il va falloir peu à peu coder sur de tels systèmes, car ils ne sont pas utilisables tels quels. Pour ce faire, soit on code spécifiquement pour eux, soit on utilise des codes historiques qu'on va porter ensuite vers des GPU. Et la tendance veut qu'on préfère utiliser les codes historiques. *« Ils offrent une communauté bien établie, des ressources disponibles, des environnements, des outils… Ce qui est préférable à utiliser des codes totalement nouveaux »*, indique Jeoffrey Legaux. Mais les porter vers un GPU, ce n'est pas forcément évident.
C'est ainsi que le Cerfacs a développer une application, AVBP, qui traite des questions de mécanique des fluides, notamment des phénomènes de combustion. Le chercheur se base pour sa présentation, sur des cas tests : une turbine à gaz, et une simulation d'explosion dans un espace confiné.



ABVP qu'est-ce que c'est ?

Un logiciel développé depuis 20 ans, utilisé par 300 personnes, avec environ 20 nouveaux utilisateurs par an. Mais le problème, c'est que la plupart des utilisateurs viennent de la mécanique des fluides, et non pas du domaine de l'informatique. Donc le logiciel doit être accessible.
A ce jour, AVBP a été développé en full MPI pour travailler sur des CPU. Au total, on trouve 200 000 cœurs qui ont un recouvrement d'environ 90%. La question est simple : comment porter cela sur GPU ?

Les contraintes sont nombreuses : les utilisateurs sont novices en code (des thésards débutants en Fortran) et surtout, très peu de profils formés en informatique, notamment en supercalculateurs. Le code doit donc rester simple, et portable. Le but est de le transférer sur GPU.

En outre, la base déjà existante de Fortran doit rester. Il ne faut pas avoir à demander à tous les utilisateurs de recoder les applications de départs. C'est pour cela qu'il est impossible de passer par CUDA, mais de se tourner vers OpenACC. Ajoutons qu'il vaut mieux rester sur un modèle de programmation par directive. Pourquoi avoir préféré OpenACC à OpenMP ? Car OpenACC est plus simple pour écrire sur GPU, qu'il bénéficie d'une communauté active « et très enthousiaste », estime Jeoffrey Legaux, et d'un soutient très intensif de la part de Nvidia et de PGI. En outre, OpenMP n'ayant démarré qu'en 2017, il reste trop limité à ce jour, et demande encore d'évoluer.

Ainsi ABVP se base sur un code Fortran, gère le parallélisme via MPI et les entrées et sorties parallèles via HDF5.
Lorsqu'on regarde l'étendue des calculs, on voit que c'est la partie de la simulation numérique la plus gourmande en calculs. C'est donc elle qui doit être portée sur GPU.

Etant donnée la structure des calculs (divisés en sous-calculs en local, pour optimiser la gestion des mémoires) comment doit se faire le portage ? Au niveau des fonctions de calculs de haut niveau, (coarse grain) en haut des étapes, ou au contraire, au niveau des calculs locaux (fine grain) ?

Deux tests ont été effectués. Sur une approche coarse grain, les plus grandes étapes des calculs, ont été réduites de moitié, mais d'autres parties qui travaillaient sur d'autres données ont été un peu ralenties. Si les résultats étaient encourageants, ils ont fini par poser plusieurs problèmes : modification du code pour l'accessibilité au donnée car en passant d'OpenMP (mémoire partagée) à OpenACC sur GPU, il n'y a plus de mémoire partagée ; en outre un gros morceau a été transféré sur GPU en mode « boîte noire », sans vision précise, ou encore des limitations dans la faisabilité de certains calculs.
Dans l'autre approche (fine grain), les chercheurs ont repéré les boucles de calculs les plus intenses et consommatrices en calculs pour les porter sur le GPU. (Repérage des boucles, portage des tableaux associés vers le GPU). Le travail de repérage et d'optimisation des boucles se fait très bien, mais est très long…

Conclusion : pour un GPU, mieux vaut avoir de grosses boucles de calculs internes, et de petites boucles externes (et l'inverse sur un CPU).
Sur CPU il vaut mieux des petits groupes pour rester dans le cache. Plus on augmente la taille du groupe, plus le temps de calcul augmente. Sur le GPU, le comportement est inverse, plus on augmente la taille des groupes, plus on est efficace.

Après validation des résultats, les nœuds de calculs ont été portés. Mais le fait de passer par MPI au départ (pour gérer la parallélisation), le protocole demande de recopier des nœuds de données sur le GPI. Mais ces recopies font perdre beaucoup de temps…
« Mais il est possible d'utiliser des directives MPI dites « Cuda-aware ». En mettant des directives OpenACC qui vont dire qu'on va utiliser seulement certains tableaux, on peut faire des échanges MPI d'un GPU vers un autre GPU, sans devoir passer par le CPU », décrit le chercheur. Ils ont dû alors créer ces buffers d'échange. De nombreux problèmes se sont posés, et il a alors fallu récrire un minimum le code… ce qui n'était pas dans le contrat de départ, mais indispensable. Jeoffrey Legaux expose cela ici.

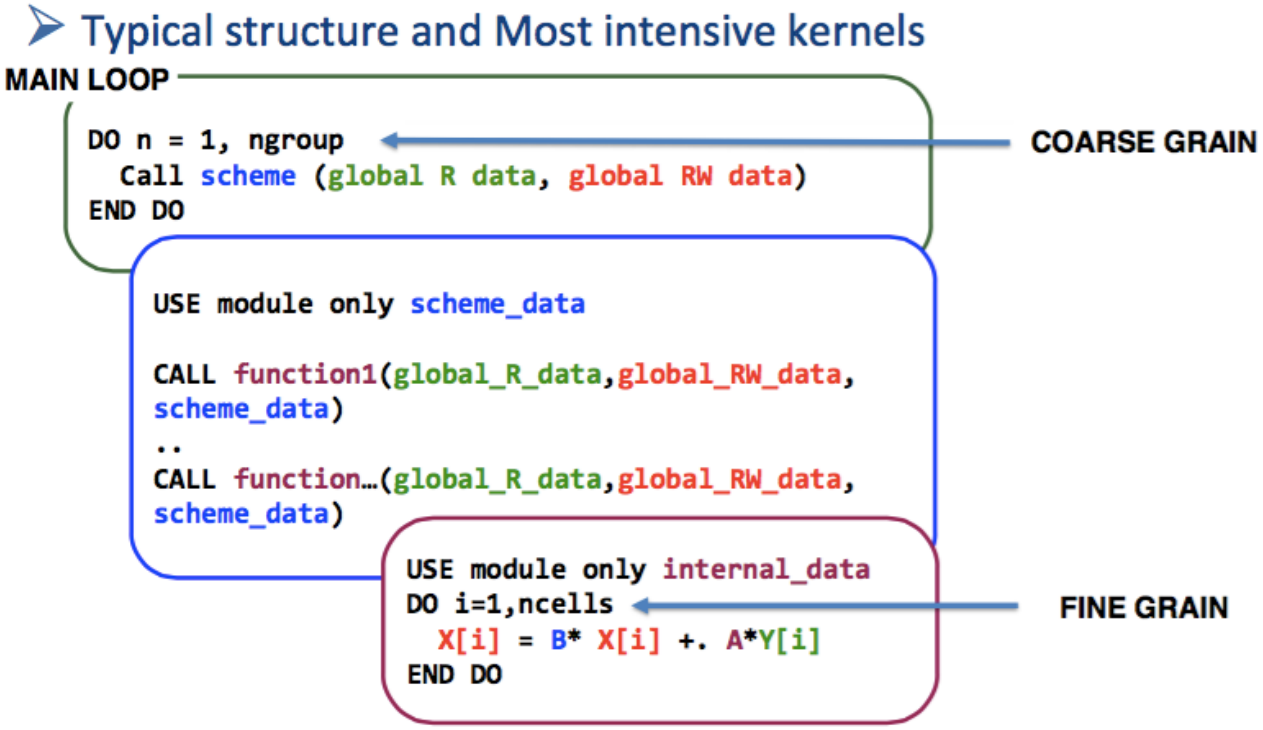
In fine, dans les résultats, il y a bien un gain de performance, mais avec une très faible utilisation du GPU. Le souci ? Comme c'est un portage partiel, il y a beaucoup de copie mémoire…

Il faudrait subdiviser le travail. Et l'outil MPS de Nvidia permet de lancer plusieurs processus sur un seul GPU. Ces processus vont alors se partager les ressources d'un seul GPU. Et le projet ne

souffre pas de multiplier les processus MPI sur une seule carte. Et là, la mise à l'échelle est très importante, et les gains de productivité sont énormes.

In fine, les équipes ont obtenus pour la simulation de soufflerie, un rapport de 18x entre le GPU et le CPU, même s'il reste toujours certains problèmes de copie mémoire.

Conclusion : OpenACC est un bon protocole pour porter un code Fortran sur GPU. Ca fonctionne bien, même si ce n'est pas encore optimisé, les équipes ont obtenu un processus qui fonctionne. Le seul souci réside au niveau de la copie des données qui peuvent prendre beaucoup de temps.
A termes, les équipes espèrent pouvoir réaliser un portage complet des calculs sur GPU, pour obtenir les performances complètes du portage. Là, seuls les plus gros calculs sont portés, il reste énormément de petits calculs à porter, et c'est un travail très conséquent. Mais qui peuvent aussi encore être optimisé (par exemple en calcul asynchrone…) et pourquoi pas travailler avec OpenMP 5, qui offrira les mêmes outils qu'OpenACC.



10 MODELISATION NUMERIQUE DU CLIMAT SUR ARCHITECTURES HAUTE PERFORMANCE : QUELQUES SUCCES ET DEFIS A L'ISPL

Par Thomas Dubos, de l'Institut Pierre Simon Laplace. Enseignant chercheur à Polytechnique, au LMD (Laboratoire de Météo Dynamique), un des deux plus gros centres français de modélisation du climat.

La présentation vidéo n'est pas disponible.

Dans le monde de la simulation climatique, l'exascale reste un horizon à long terme, plutôt qu'un projet tangible. Si certains membres de la communauté estiment que les avancées en informatique pourront améliorer les modèles en poussant vers des résolutions extrêmement élevées (de l'ordre du kilomètre), les plus réalistes pensent toutefois que ces perspectives restent très lointaines. En revanche, tout le monde s'accorde sur le fait que l'exascale pourra donner une direction, une impulsion afin de tirer vers le haut les modèles, et bénéficier à toutes les applications dans le domaine du climat ou de la météo. Avant d'aller plus loin, Thomas Dubos prévient qu'il est atmosphéricien et que sa présentation possède de fait, un biais d'études, et notamment dans le choix des exemples davantage tournés vers des modèles atmosphériques.

Le chercheur commence par quelques généralités sur les simulations climatiques. A commencer par les différences entres météo et climat : différences sur l'objectif (simuler vs modéliser), et les échelles de temps (quelques jours vs plusieurs milliers d'années). En outre, Thomas Dubos précise que dans le domaine de la modélisation climatique, si tout est régi par des équations physiques, il restera toujours des phénomènes nombreux, et non résolus par les équations utilisées dans le cadre des modèles (des turbulences, des nuages, des gouttes d'eau…). Donc il ne faut pas imaginer qu'on pourra faire du climat sans beaucoup de physique sous-maille.

Dans ce contexte, les modèles climatiques ont une résolution de l'ordre de la centaine de kilomètre tandis que les modèles météo, qui vont beaucoup plus vite, peuvent avoir des résolutions bien plus petites (10km). Ces précisions sont importantes, car ce qu'il faut comprendre, c'est qu'un algorithme de modèle climatique doit tourner 1000 fois plus vite que la réalité pour calculer la modélisation sur des centaines d'années, et 10 000 fois plus vite, même, pour un climat ancien. Tandis qu'un modèle météo ira 100 fois plus vite que la réalité.

Comment fonctionne une modélisation climatique ?

Une modélisation climatique est un solveur hydrodynamique. Mais 90% du modèle se concentrera sur la représentation des processus sous maille (les phénomènes physiques non pris en compte dans les grandes équations du modèle). En termes de calculs, « *c'est plus complexe qu'un calcul de viscosité turbulente dans un modèle LES…* » affirme Thomas Dubos, en précisant qu'il ne veut en aucun dénigrer les experts qui travaillent dans ce domaine…

Mais ce n'est pas tout, à ce solveur hydrodynamique, vient se coupler un solveur océanique (un solveur fluide également, mais calé sur le modèle des océans) auquel il faut rajouter d'autres compartiments de modélisation, comme les surfaces continentales (végétations, hydrologie de surface etc. la glace de mer). Tous ces facteurs doivent être pris en compte car ils jouent un rôle dans l'évolution du modèle.

Vie d'un modèle climatique

De manière générale, un modèle climatique passe son baptême du feu lors d'un exercice de simulation internationale, où les paramètres sont définis pour tout le monde après coordination

entre laboratoires. Par exemple, CMIP5, qui participait à un exercice de 2012, a produit 200 Teraoctet pour l'IPSL, et pour CMIP6, au total, une quinzaine de pétaoctet ont été générés en faisant tourner le modèle. Toutes ces données seront ensuite distribuées dans le monde, pour que chaque laboratoire puisse travailler dessus. La résolution des modèles s'améliorant, la quantité de données produites va grandissante.

Voici un exemple des évolutions attendues : (elles représentent davantage des tendances que de chiffres réels)

10-year perspective for CMIP

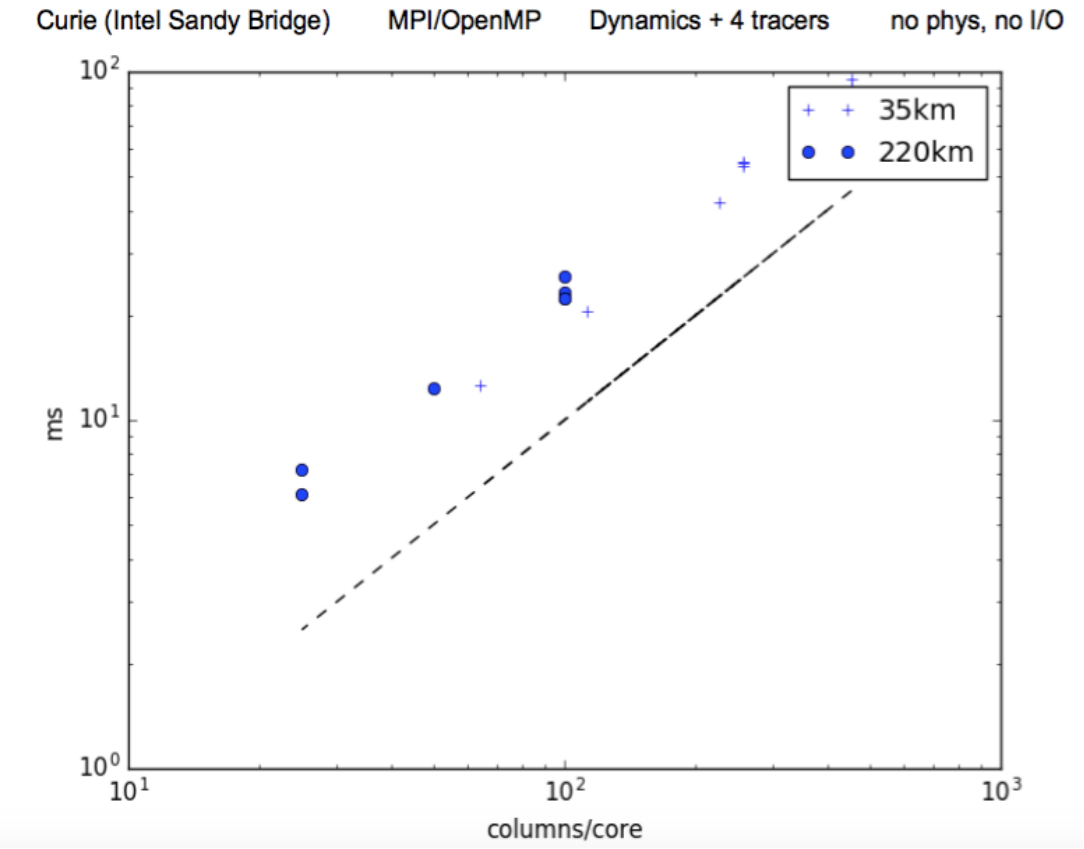
	CMIP5	CMIP6	CMIP7
Year	2012	2017	2022
Power factor	1	30	1000
Npp	200	357	647
Resolution [km]	100	56	31
Number of mesh points [millions]	3,2	18,1	108,4
Ensemble size	120	214	388
Number of variables	800	1068	1439
Interval of 3-dimensional output (hours)	6	4	3
Years simulated	90000	120170	161898
Storage density	0,00002	0,00002	0,00002
Distributed Archive Size (Pb)	**3,19**	**86,05**	**2260,20**

Du vectoriel au parallèle

Au départ, les simulations se basaient sur SX8, soit une trentaine de CPU, et un peu de MPI pour la parallélisation. Les rapports de performance étaient bien meilleurs avec SX8 (jusqu'à 8 fois plus élevée par rapport à Itanium ou Optéron)

Le maillage joue une importance capitale sur les capacités de calculs, augmentant considérablement le nombre de données (à paralléliser avec MPI) et donc la performance globale. Par exemple, un maillage de 25km2 nécessite près de 1 million de cellules soit environ 10 000 processus MPI.
Thomas Dubos revient ensuite sur les différents impacts du maillage, en détaillant le fonctionnement général des calculs par itérations successives de calcul différentiel et les modèles de « découpages » de l'atmosphère. Ce dont on s'aperçoit c'est que la performance dépend surtout de l'implémentation, autrement dit de l'échelle de temps choisie pour le modèle. Ainsi, si l'on compare les performances de calculs en fonction des capacités des machines pour deux maillages différents, on obtient une courbe linéaire. (CF page 29).

La capacité à faire tourner le modèle à une certaine vitesse dépend donc du pas de temps stable maximal que l'on peut obtenir et du temps d'exécution pour avancer d'un pas de temps.
Réduire le temps d'exécution d'une tâche pose donc un gros problème de mise à l'échelle : plus on augmente la résolution, moins on peut faire d'années par jour de calcul.



L'autre nœud du problème consiste en la capacité à entrer et sortir des données. Car un modèle climatique, on l'a vu plus haut, produit un nombre colossal de données. Les synchroniser pose de grandes questions informatiques. Pour ce faire, Yves Meurdesoif a élaboré un outil spécifique de traitement des données en entrée/sortie, qui fonctionne par traitement asynchrone entre la partie client et serveur du réseau.

Les défis

Le premier défi, dans le passage à l'exascale consiste à adapter les codes à de nouvelles architectures (comme dans beaucoup de domaines) soit en portage manuel, soit par insertion de directive OpenACC, mais également par adaptation de la gestion de mémoire…

A l'Idriss, la communauté a établi certaines règles pour le passage au GPU. Ainsi, *« Pour être validée, la version GPU d'un code devra … être en moyenne sur l'ensemble des cas tests au moins 4 fois plus performante que les versions non-accélérées en comparaison nœud à nœud »*, explique le contrat de progrès qui a été établi au sein de la communauté.
De même, la communauté de physiciens doit s'interroger et faire différents choix : que ce soit des questions de rétrocompatiblité (certaines parties de codes ont plus de 30 ans), ce sont des codes très touffus, avec de nombreuses ramifications, certaines parties « maisons », évoluent en

permanence… Autrement dit, la communauté de climatologue aura-t-elle vraiment le besoin d'investir dans une architecture ? Sera-ce une dépense ou un investissement à long terme ?
L'Institut à ce jour, n'a pas encore fait le choix. Mais elle doit y réfléchir.
D'autres pistes sont également ouvertes, notamment en se tournant vers le machine learning, et les récents progrès qui ont pu être fait dans ce domaine. *« On pourrait remplacer une partie du modèle par un émulateur, le faire s'entraîner sur CPU, et ensuite passer en exascale »*, théorise Thomas Dubos, mais pour le moment, rien n'est encore fait.

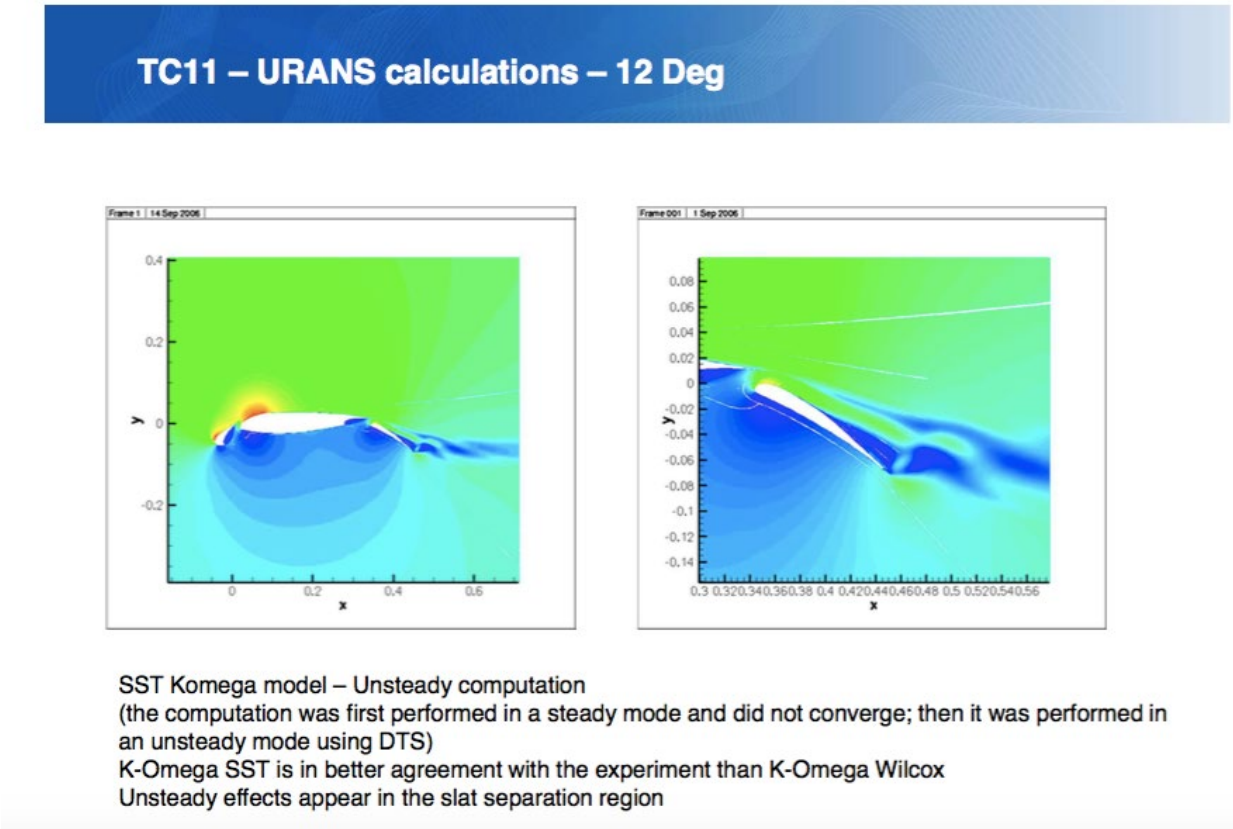
11 MODELISATION DE LA TURBULENCE ET INTELLIGENCE ARTIFICIELLE

Par Vincent Couaillier, de l'Onera.

Objectif : présenter un type d'application particulier sur HPC, la modélisation de la turbulence par IA. Un procédé qui n'a jamais été réalisé.

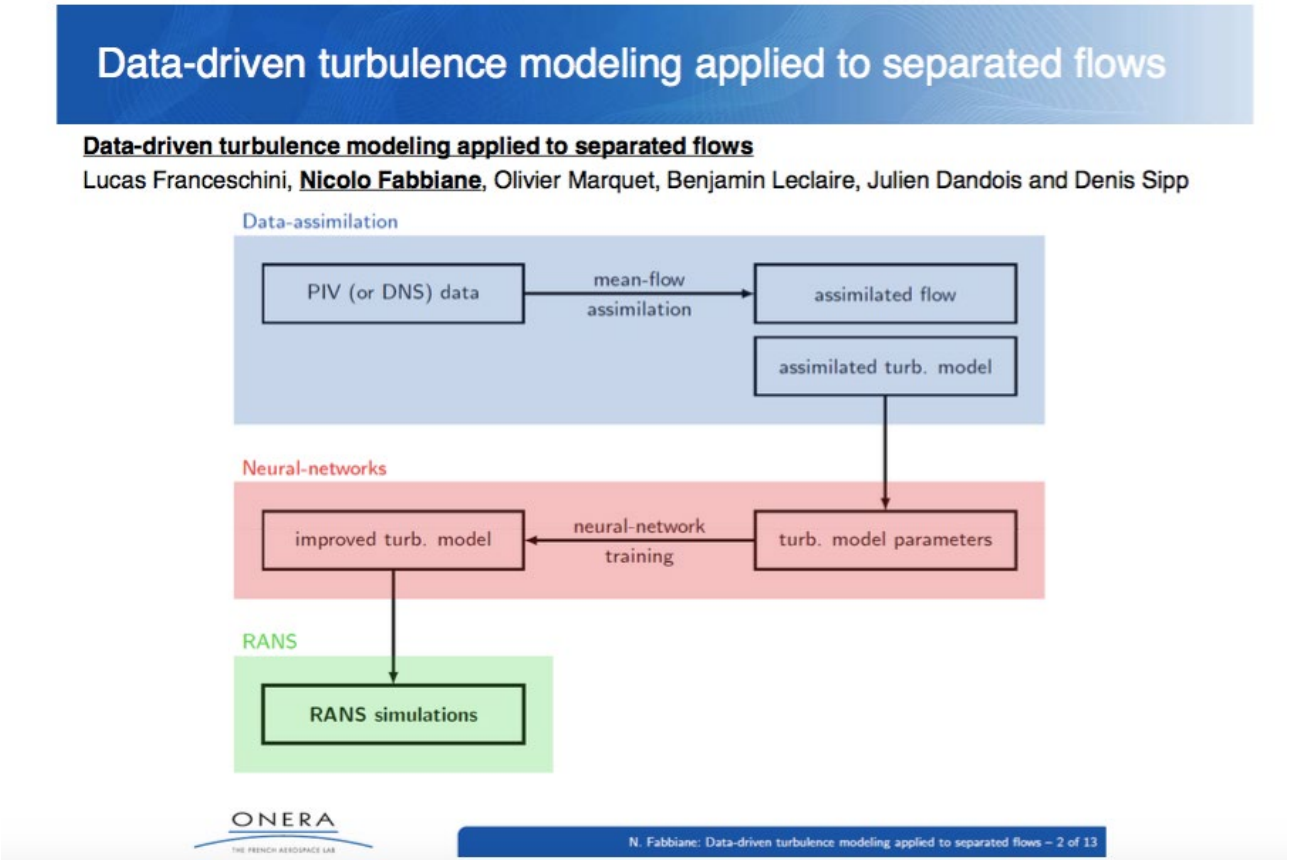
A ce jour, la turbulence reste une des choses les plus complexes à modéliser en mécanique des fluides. *« Ceci entraîne une mauvaise prédiction des caractéristiques dans les situations hors adaptation »*, explique Vincent Couaillier. Mais l'Onera a eu une idée. En se basant sur les progrès des structures de calculs, et sur les progrès liés au machine learning, l'institut ainsi qu'une dizaine de partenaires scientifiques ont créé HifiTurb, qui consiste à générer une grande base de données LES / DNS de turbulence pour des écoulements complexes à grand nombre de Reynolds, du moins pour des configurations géométriques fondamentales, et à faire travailler sur ces bases de données un algorithme de machine learning, qui permettrait de fournir *« des moyens entièrement nouveaux d'extraction d'information caractérisant les grandeurs physiques et leurs interactions à partir de données massives générées par ces calculs LES / DNS »*, continue-t-il. Ainsi, espèrent-ils améliorer les modèles de turbulences dans l'industrie.

Il existe plusieurs modèles de simulation de turbulence : DNS (Direct Numerical Simulation), LES (Large Eddy Simulation), avec des équations de Navier-Stockes filtrées, et RANS (Reynold Average Navier Stockes), mais aussi des modèles hybrides LES/RANS.



L'objectif du projet consistes à simuler numériquement pour des modélisations de hautes fidélité et d'ordre élevé, (LES ou DNS) ainsi que de cas tests représentatifs de problèmes CFD raides. Le problème c'est qu'il existe des incertitudes opérationnelles liées à la résolution de ces modèles, ainsi que des questions de sensibilités numériques. « *aujourd'hui on travaille sur des modèles de base, le but est de s'en servir pour passer à des modèles où on maîtrise les erreurs*, explique Vincent Couaillier. *Si on ne fait pas de progrès là-dedans on aura toujours des problèmes dans les calculs à avoir.* » Car une modélisation de turbulence peut amener à modéliser jusque 3000 milliards de points. Et en fonction de la finesse du maillage et des types de modélisation, le modèle de turbulence peut diverger.

Ainsi l'exascale pourrait permettre de travailler sur de nombreux modèles DNS, ainsi que sur des modèles hybrides LES/RANS, et d'entraîner un modèle à partir d'un autre. En prenant garde, évidemment, aux modèles divergents qui viendrait troubler les données. Cette méthode, par un algorithme de machine learning, en utilisant des réseaux de neurones, pourrait permettre ensuite, d'appliquer un modèle sur des géométries différentes.



En résumé, HifiTurb cherche à combiner différents types de modélisation de turbulence, pour construire une base de données convergentes réalisées à partir de capacité de calcul HPC, pour entraîner ensuite un méta-modèle de modélisation, applicable à de nouveaux paramètres (comme la géométrie), pour mieux rendre compte des turbulences.

12 CONCLUSION

Par Christophe Calvin

Suite à ces différentes présentations, le président d'Aristote est ravi d'avoir couvert tous les aspects des enjeux de l'exascale, des futurs machines aux enjeux politiques internationaux, des défis techniques, et des questions posées pour l'univers applicatifs… Et il souligne la qualité des présentations. *« On a bien compris que peu importe les technologies, l'important c'est de savoir ce que vous devez faire dans le noeud »*, résume-t-il. *« Le principe, c'est qu'il va falloir refaire de la vectorisation. Avant on pensait avec des machines de 200 000 cœurs, il va falloir penser l'hybride CPU GPU, désormais. Heureusement MPI est toujours là, il reste une valeur sûre »*. Pourquoi ? Car le nombre de nœuds va croître de manière exponentielle.

« Pendant les 3 ans qui arrivent, il faudra tester vos programmations sur les trois machines disponibles, afin de voir si les algorithmes tiennent le coup. Et nous, on se retrouve dans quelques mois, pour savoir comment cela aura évolué. »