Colloque Penser Pétaflops CEA-CNRS
Atelier No 2
Mutualisation des codes et des grands outils logiciels

D. Veynante, CNRS
J.-Ph. Nominé, CEA

Mutualiser renvoie à l'idée de mettre quelque chose en commun, de le répartir, de le partager. Dans le cas présent il s'agit des **outils** que sont les logiciels de calcul, et plus généralement le **savoir-faire** qu'ils nécessitent ou qu'ils résument. On parle ici des **codes** au sens large, incluant les **logiciels de pré-traitement, de post-traitement**, constituant un environnement logiciel complet de calcul, ainsi que, le cas échéant, des **données produites par la simulation ou qui lui sont associées**.

L'idée de **mutualisation** est généralement associée à la **gestion du risque ou de la difficulté** : la visée ou l'espoir peut être :

- de **diminuer les risques, les coûts, les délais** par la mise en commun ;
- **de résoudre des problèmes plus difficiles**, en regroupant les efforts, les compétences, les savoir-faire - problèmes qui resteraient hors de portée d'individus isolés ou d'équipes trop modestes ;
- en ne dupliquant pas certains efforts, de dégager des ressources utiles à autre chose : **optimiser les moyens humains**;
- éventuellement, **être plus forts**, pour influer sur l'émergence de standards ou dialoguer avec des éditeurs de logiciels

Sommaire

1 Démarche adoptée

Après quelques échanges par email, environ la moitié des personnes inscrites à l'atelier se sont réunies au CNRS à Paris le 8 septembre pour quelques présentations et une discussion. Une brève synthèse a circulé et conduit à l'exposé de la journée de restitution des ateliers du 13 novembre. Des contributions complémentaires ont été récupérées en novembre.
Ce court document suit à peu près la trame de l'exposé du 13 novembre et dégage finalement quelques remarques générales, propositions et recommandations.
Nous avons mis en annexe les détails de la partie « Domaines d'application et communautés scientifiques », qui sont soit des contributions directes de spécialistes (nous les en remercions) soit de courtes synthèses provenant de la réunion du 8 septembre, soit encore simplement le fruit de discussions informelles.
Il est évident que ces réflexions ne constituent qu'une première étape, en particulier l'aspect « données produites par la simulation » a été très peu couvert et mériterait en soi un approfondissement.

2 Le « Petaflop/s » : quoi, pourquoi ?

2.1 Puissance de calcul : pour quoi faire ?

Le besoin sans cesse croissant de puissance de calcul est une donnée permanente désormais établie dans la plupart des activités scientifiques et techniques. La puissance de calcul n'est pas une fin en soi, elle peut servir :

- A résoudre des **problèmes de plus grande taille** (« weak scalability »)
- A résoudre plus vite des problèmes de taille identique (« strong scalability », réduction du « **time to solution** », visée souvent plus ambitieuse et semée d'embûches que la précédente)
- A résoudre des **problèmes nouveaux ou plus complexes**, souvent « multi-formes » (multi-échelles, multi-physiques, multi-plateformes, multi-utilisateurs)

2.2 Architectures matérielles et technologies

Aujourd'hui on peut s'accorder sur le fait que la principale difficulté qui nous attend est l'adaptation de certains codes aux machines de très grande puissance que nous ne doutons pas d'obtenir, sans être certains de ce qu'elles seront exactement. Le logiciel a du mal à suivre les tendances matérielles pour le calcul à grande échelle (*capability*) et à s'adapter pour tirer réellement profit de machines très fortement parallèles - une puissance de **1 Petaflops crête en 2010** demandera l'agrégation de l'ordre de **50000 à 300000 processeurs « généralistes »**, autour de 100000 si on retient des processeurs c*ommodity* - de grande diffusion.

La macro-architecture d'une machine de « classe Petaflops » sera sans grande surprise, elle interconnectera des nœuds de calcul en nombre suffisant, avec quelques variantes possibles sur la technologie et la topologie du ou des réseau(x) d'interconnexion. C'est au niveau des processeurs et des nœuds de calcul « élémentaires » que fleurit une grande variété de solutions, avec quelques tendances incontournables et d'autres moins claires.
La multiplication du nombre de cœurs est une tendance incontournable (***multi-core*** , de 4 à 12 cœurs par CPU dans un proche avenir; le ***many-core*** (au-delà de 16 ou 32 cœurs par

CPU) est encore sujet de recherche mais des processeurs de cette nature pourraient rapidement voir le jour (Intel Larrabee p.ex.).
L'hybridation des types de processeurs (processeurs CPU généralistes, vectoriels, et unités spécialisées comme cartes graphiques – GPU – processeurs Cell, processeurs ClearSpeed, FPGA…) est une autre tendance, sujet de beaucoup d'études et déjà réalité dans quelques machines. Cette hybridation peut se faire à différents niveaux de granularité : sous-systèmes homogènes différents mais agrégés (p.ex. vecteur+scalaire, scalaire+GPU) ; lames à base de processeurs différents mais juxtaposées dans les mêmes cabinets ; processeurs différents au sein d'une lame. Et qui sait à terme peut-être des cœurs de types différents fortement intégrés sur la même carte. Cette tendance va sans doute affecter aussi les stations de travail et les serveurs généralistes, c'est déjà le cas pour les CPU multi-cœur qui équipent les PC grand public.

On peut donc dire pour résumer que les types d'architectures à prendre en compte sont en nombre réduit, mais que la tentation ou le besoin d'hybridation (pour tenter de gagner en rapport « performance/coût » ou « performance/consommation électrique ») commence à brouiller la donne et que le **multi-cœur** – voire le ***many-core*** - plus ou moins **hétérogène** nous guette.

2.3 Un problème de programmation et d'outils de développement

Dans tous les cas le constat est cruel, face à ces tendances : il faudra apprendre à **programmer « très en parallèle » à différents niveaux éventuellement imbriqués dans un même code**, sans qu'un modèle de programmation universel s'impose facilement ni que des outils géniaux émergent pour nous assister rapidement dans tout cela (compilateurs mais aussi débogueurs, outils d'analyse de performances, de parallélisation plus ou moins automatique…).
Notons aussi que la programmation parallèle hybride, au sens typiquement de l'usage conjoint de MPI entre nœuds et de OpenMP ou directement de threads au sein de noeuds SMP, est déjà pratiquée par certains ; ce que nous évoquons ici est une généralisation et une accentuation probable de cette tendance.

Nous parlons ici de Petaflops « crête » et de «calculs de grande taille », extrêmes en termes de découpage et de parallélisation pour maîtriser temps de cycle et/ou taille mémoire, en vue d'une extensibilité « faible » ou « forte ». Ces deux aspects ne sont d'ailleurs pas forcément simultanément accessibles. Par exemple, un code de volumes finis parallélisé par domaine nécessite que chaque processeur traite un nombre suffisant de points de maillage pour limiter les communications entre processeurs. Dès lors, l'augmentation du nombre de processeurs bénéficiera surtout au traitement de problèmes plus grands pour un même temps de restitution et non à la diminution de celui-ci pour un problème donné.

2.4 Le « pétaflops » : barrière symbolique - usages

Toutes les recherches, toutes les études, toutes les applications ne nécessitent pas des machines de classe Petaflops et des calculs lourds. Mais de plus en plus exigeront cette échelle ou en bénéficieront.
Il faut noter également qu'une machine Petaflops sera généralement partagée, même par un nombre limité d'applications, qui de fait n'auront pas toujours chacune la totalité de la machine pour elles, mais une fraction plus ou moins importante. On pourrait donc se dire que certaines applications aujourd'hui en service sur les machines récentes des centres de calcul nationaux ont pu approcher cette échelle de fonctionnement, disons de 10 à 25% du Petaflops, et que l'on a peut-être désormais certains codes à peu près préparés (voir les « Grands Challenges » des différents centres).

En fait il faut prendre le Petaflops comme une barrière symbolique qui n'est qu'une étape. La machine Roadrunner de Los Alamos a franchi la première cette barrière de manière soutenue avec le Linpack, suivront des applications plus variées mais en nombre réduit, et toutes ne pourront revendiquer le Petaflops soutenu. Le vrai enjeu est plutôt de se mettre dans le mouvement et de viser au-delà du Petaflops crête : le multi-Petaflops crête, le Petaflops soutenu… Ces horizons sont bien pris en compte dans les programmes HPC américains et japonais. L'évolution matérielle va se poursuivre inexorablement et exigera des changements ou adaptations, pour des codes de plus en plus nombreux, de toute façon.

3 Pourquoi mutualiser ?

Nous l'avons évoqué : en HPC il est désormais admis que la difficulté principale est généralement et de plus en plus **la programmation et le support applicatif**, plus que l'obtention de puissance brute de calcul.
N'oublions pas également que le calcul n'est qu'un moyen et les logiciels les outils : à un niveau plus large, il s'agit aussi de partager et comparer des données (en partie issues de calculs) de manière cohérente, pour mener des études, des recherches.
Il est donc naturel de se poser la question du regroupement de certains efforts, moyens et savoirs pour le développement logiciel et l'exploitation des résultats de simulation. **Logiciels et données** cristallisent l'essentiel du **patrimoine des savoir-faire et des connaissances** : leur durée de vie peut excéder celle des machines d'un ordre de grandeur.

Les finalités ou visées de la mutualisation peuvent être diverses :

- **Qualité** : avoir de meilleurs modèles et de meilleurs codes qui les implantent
- **Efficacité** : être plus efficaces techniquement et « économiquement » ; aller plus vite, avoir plus de confort, dégager des ressources que l'on peut affecter à des développements supplémentaires ou plus ambitieux
- **Performance** : aborder et résoudre des problèmes hors de portée en-dessous d'une certaine « masse critique », tant en modélisation qu'informatiquement (on peut soulever une pierre de 200 kilos à 5 mais pas seul même en y passant 5 fois plus de temps – précisons : sans dispositif technique auxiliaire !)

4 Mutualiser : gain ou frein ?

4.1 Performance vs. portabilité ?

Il y a sans doute des limites ou des contreparties aux bénéficies potentiels de la mutualisation. Evoquons-en quelques unes, le fil directeur étant que **l'élargissement du partage peut avoir des effets réducteurs** – tout comme une intersection se réduit avec l'augmentation du nombre de parties prenantes. Dans certains cas des inconvénients peuvent eux-mêmes avoir des contreparties positives.

Il est clair que plus on est nombreux à partager, plus il est difficile de préserver les intérêts de chacun. Des choix généraux tendent à se faire au détriment de choix spécifiques.

Des adaptations de codes nécessaires au portage efficace sur des machines de puissance crête de l'ordre du pétaflops peuvent ainsi aller à l'encontre d'une **portabilité** plus générale,

très importante pour des codes à **longue durée de vie et large diffusion**. On peut espérer qu'il existe des manières d'aborder quand même ces questions : modularité, localisation dans des parties précises du code des parties sensibles moins portables, sujettes à variantes ou ré-écritures; outils de génération de code spécifique, p.ex. pour tirer parti d'unités de calcul spécialisées où la programmation est « acrobatique ». Peut-être faut-il accepter de ne pas avoir une version strictement équivalente d'un code sur tout type de machine et gérer des configurations différentes ? Une solution pourrait aussi être de développer un code aussi standard que possible mais s'appuyant sur des bibliothèques de fonctions plus spécifiques des machines.

4.2 Différents types de codes

Il faut sans doute distinguer, schématiquement, **différents types de codes : de production, de recherche, d'étude**. Cette pseudo-classification n'a d'autre visée que de souligner des différences légitimes d'organisation.

Un code de production a une structure stabilisée et il est utilisé pour « produire » des résultats de manière systématique, paramétrique éventuellement. « Production » doit s'entendre ici dans un sens très large : des modifications du code, par exemple pour implanter des nouveaux modèles, sont bien sûr possible mais restent limitées (pas d'action sur le « moteur numérique » du code, ni sur sa structure interne).
Un code de recherche est par nature plus évolutif en termes de modèles implantés, d'algorithmes, voire de techniques de programmation, et sépare moins les fonctions « utilisation » et « développement ».
Un code d'étude serait un code axé sur l'exploration ou l'expérimentation d'un modèle plus ciblé.

Il n'y a pas a priori de différence de valeur ou de qualité entre ces types de codes, mais un code de recherche ou d'étude correspond souvent à une configuration plus légère tandis que ce sont les codes de production qui sont sans doute les plus concernés par des approches mutualisées.

Mutualiser ne présente en fait d'intérêt qu'à partir d'une certaine échelle (ou « masse » : lignes de codes, nombre d'utilisateurs, durée de vie…). Rien n'empêche toutefois d'atteindre cette échelle également en regroupant différents codes de recherche partageant des méthodes, des composants. La difficulté est alors souvent de **démontrer le bénéfice de la mutualisation** (convaincre…) - laquelle mutualisation a souvent un prix en rapidité de mise en œuvre, d'apprentissage, d'appropriation.

La mutualisation suppose aussi un gros travail d' « **environnement** », inexistant ou rare pour un code plus exploratoire. En effet, plus les utilisateurs sont nombreux, plus le code doit être convivial à utiliser (fichiers d'entrées, de sortie, structuration facilement compréhensible, etc...). Plus généralement, la mutualisation doit s'accompagner d'un gros travail de **génie logiciel** (suivis des versions, tests systématiques avant mise à disposition d'une nouvelle version, etc...).

4.3 Limitations diverses

Un autre inconvénient de la mutualisation, surtout pour des codes utilisés par des chercheurs « mobiles » et « collaborants », peut être la **perte de maîtrise directe du source**, du contenu du code, la difficulté de détacher le code de sa base locale pour « partir avec » ou le diffuser. Par exemple un code de thèse qui s'appuie sur des composants et un

environnement logiciels spécifiques, dont il a tiré des bénéfices, peut devenir difficile à transplanter (pour partir avec en post-doc p.ex.). En effet ils se peut qu'on doive « tirer » avec le codes toutes les dépendances, et celles-ci ne sont pas toujours des éléments très standard et ouverts. Cela peut-il d'ailleurs militer en faveur de l'usage de composants/environnements ouverts ou de la diffusion ouverte de composants développés localement ?

Un inconvénient perçu susceptible de faire obstacle à la mutualisation peut aussi provenir de la **perception du code, en tant que tel, comme vecteur d'originalité ou d'avantage compétitif.** Il n'est pas forcément évident de verser dans une approche communautaire un développement original ou donnant un avantage dans des domaines à forte compétition scientifique entre équipes. On peut penser à plusieurs manières de nuancer ou gérer ce genre de situation :

- le temps : publier par exemple avant de diffuser le développement logiciel ou de le partager
- les vertus de l'open source « réussi » : un logiciel utilisé par de nombreuses personnes pour des applications différentes a des chances de bénéficier d'un retour de la communauté, que ce soit en termes de nouvelles fonctionnalités ou de bugs identifiés voire corrigés

De manière générale, on perçoit bien que **la mutualisation suppose une adhésion ou un consentement** et ne saurait purement découler de directives imposées. On peut à ce propos citer des projets et initiatives qui émanent des chercheurs, tels les sites et portails de partage et d'information PLUME[1], RELIER[2], ENVOL[3]. Ces actions montrent qu'un terreau favorable existe déjà.

5 Que mutualiser - Comment mutualiser ?

5.1 Chaînes et résultats de calcul

Rappelons que nous nous intéressons à deux niveaux « logiciels » de mutualisation :

- les codes au sens large (logiciels de calcul)
- les données liées à la simulation (résultats de calcul – pas forcément indépendamment de résultats expérimentaux ou des données utilisées en entrée de la simulation)

Le partage des données est naturel à certaines communautés « mondialisées » (climat, physiques des hautes énergies/particules, observation spatiale…). Nous n'avons guère approfondi cet axe lors de cette réflexion, il y aurait matière à. Le Centre de données astronomiques de Strasbourg (CDS, http://cdsweb.u-strasbg.fr) met ainsi à la disposition de la communauté des données qui génèrent aujourd'hui plus de publications que l'exploitation de données nouvelles. Le Center for Turbulence Research de l'université de Stanford (http://www.stanford.edu/group/ctr) a bâti une bonne partie de sa notoriété sur la mise à disposition de base de données issues de simulations numériques directes pionnières, permettant un travail très important de modélisation de la turbulence.

[1] http://www.projet-plume.org/ (Promouvoir les Logiciels Utiles Maîtrisés et Economiques dans l'Enseignement Supérieur et la Recherche)
[2] http://www.projet-plume.org/relier (REférencer les développements Logiciels Internes de l'Enseignement supérieur et de la Recherche)
[3] http://www.projet-plume.org/envol (Projet ENVOL : formation pour le dEveloppemeNt et la ValOrisation des Logiciels en environnement de recherche)

Concernant les logiciels, il faut selon nous avoir une **vision globale de tous les composants des chaînes de calcul, pré-traitements et post-traitements** compris - qui peuvent dans certains domaines être eux-mêmes des logiciels très complexes de maillage, de visualisation. Tant mieux si dans certains cas ces fonctions sont plus simples ou ne posent pas de problème particulier.

5.2 Niveaux de mutualisation

Différents niveaux « logiciels » peuvent se prêter à la mutualisation, correspondant à des choix d'ordre technique ou organisationnel (architecture logicielle, modèle de cycle de vie, gestion de projet…)

.

- Le premier niveau de mutualisation à envisager est sans doute celui de composants logiciels plus élémentaires (**bibliothèque** numérique, graphique, **solveur**, **coupleur**… ceci renvoie à l'**Atelier 3**).
- On peut dégager un **« noyau dur** » de fonctions et services (gestion du parallélisme et des entrées/sorties, typiquement) organisées en « **framework** » et ré-utilisables – exemple d'ARCANE, CEA/IFP
- Les développements peuvent être organisés en **modules/codes reliés par coupleurs** (exemple des codes de climatologie en France, voir en Annexe 2 : code NEMO, coupleur OASIS p.ex.). Modules et codes ainsi que coupleur peuvent être mutualisés plus simplement et un assemblage plus global de chaînes de calcul en résulter.
- Une autre modalité, peut-être pas si différente, est de rassembler des codes et modules de pré/post-traitement dans une **plate-forme** générale et extensible, telle SALOME ; typiquement on a ici un effort fédérateur d'abord autour d'un bus/format de données standardisé puis d'interfaces de gestion d'études assez génériques
- Est-il plus facile de mutualiser les **pré/post-traitements** ? Les techniques de **maillage** et de **visualisation**, aux fondements assez génériques, recèlent souvent et rapidement des particularités liées aux domaines d'applications. Mais il y a dans ces activités des exemples de composants ou d'environnements largement diffusés et accessibles qu'il serait dommage de ne pas chercher à prendre comme bases quand ils répondent aux exigences (p.ex. VTK, Open Cascade).
- Evidemment nous n'oublions pas la question des **données et résultats** de calcul divers, qui peut déjà bénéficier de standardisation de formats de données (hélas encore bien trop nombreux et disparates entre activités – CAO, maillages, calcul, visualisation - et domaines – fluides, structures…).

5.3 Pratiques techniques et modèles économiques

Nous ne nous appesantirons pas sur les pratiques techniques liées à la **gestion de version, de configuration, aux tests et à la validation en général**. Il semble que celles-ci s'installent naturellement quand elles correspondent à un besoin, une nécessité ou un bénéfice. On peut simplement insister sur le fait que des témoignages de partage ou de mutualisation d'approches de validation sont apparues, et de manière générale le **partage de savoir-faire et de méthodes** (pas seulement de code ou de données) est sans doute à encourager beaucoup plus.

Concernant les « modèles économiques », il existe différentes options assez bien connues :

- Open source
- Hybride « à la Eclipse » : briques de base open source pour fédérer un maximum de personnes, plug-ins spécifiques commerciaux
- Propriété partagée
- Propriété limitée aux développeurs
- Doubles licences avec p.ex. une version open source (recherche et enseignement) et une version plus restreinte d'accès
- …

Il n'y a pas de réponse toute faite sur le bon modèle à adopter, cela dépend du contexte, de l'histoire des équipes. Disons qu'il y a, a priori, dans les modèles évoqués ci-dessous, de quoi couvrir toutes les situations qui peuvent se présenter.

6 Domaines d'applications, communautés

Cette section résume quelques réflexions sur certaines communautés (ou domaines d'application). Il n'y a ici nulle prétention à l'exhaustivité. Les sources utilisées sont des témoignages recueillis lors de réunions ou par courrier électronique, quelques recherches sur le web, des discussions informelles, ou encore l'expérience personnelle des contributeurs à ce rapport. En annexe figurent quelques détails et surtout des textes de contributeurs spécialistes de certains domaines.

On constate globalement de grands contrastes.

Des pratiques mutualisées très structurées existent dans certains cas, à des **échelles très larges** comme pour la communauté « **Climat** ». Mais cela reste exceptionnel et lié sans doute à des circonstances historiques particulières (gros outils de calculs, structuration nécessaire vis-à-vis du programme GIEC). Inspiration à coup sûr pour beaucoup d'autres domaines, ce modèle ne se transpose pas complètement partout ailleurs.

La communauté académique de la simulation numérique de la combustion turbulente se retrouve autour du code AVBP (http://www.cerfacs.fr/cfd/avbp_code.php) mis à sa disposition par ses co-propriétaires, le CERFACS et l'Institut Français du Pétrole. Cet exemple de mutualisation réussie (voir annexe) résulte d'une initiative individuelle dans une société de droit privé...

Certains domaines possèdent des **structures fédératives** d'ordre général qui n'ont pas pour objectif premier ou pour vocation de mutualiser les codes mais pourraient le faciliter et y contribuer du fait de leur existence et rayonnement (p.ex. **EFDA** dans la communauté Fusion : http://www.efda.org/, Europea Fusion Development Agreement).

Des initiatives ou projets « thématiques » plus ciblés existent ou démarrent dans certains domaines. Voir les contributions « **Astrophysique** » en annexe par exemple (HORIZON : ; COAST :). En **physique des matériaux** on a vu, avec notamment un fort axe français, l'émergence de projets de grands codes, ayant bénéficié parfois de projets ANR ou pôles de compétitivité (ANR LN3M LN3M http://www-drfmc.cea.fr/sp2m/L_Sim/LN3M/ avec les codes ABINIT www.abinit.org et STAMP ; IOLS https://iols.c-s.fr qui se prolonge en EHPOC dans SYSTEM@TIC). Cette échelle de fonctionnement assez délimitée est sans doute très prometteuse car opérant à des dimensions « raisonnables ». La gestion dans la durée reste toutefois un enjeu, car ni l'ANR ni les pôles de compétitivité ne sauraient garantir cela. Notons qu'ABINIT a précédé les projets subventionnés cités.

D'autres domaines et laboratoires souffrent de difficultés liées à la taille des équipes « codes » ou « support applicatif », surtout si le HPC (calculs lourds) n'est pas le besoin primordial dans leur thématique. En annexe nous résumons les témoignages apportés par le LMT/Cachan et l'Institut Jean Lamour de Nancy dans cet esprit. Dans ces cas-là la mutualisation doit pouvoir apporter beaucoup, mais peut-elle, doit-elle être entreprise « localement » (plus simple mais limité) ou inter-laboratoires éventuellement distants (plus complexe) ?

7 Autres éléments : programmes de R&D d'autres pays

Il a été rappelé lors de nos discussions que les **Etats-Unis** apportaient un fort soutien aux développements mutualisés de logiciels scientifiques, via notamment le programme **SciDAC** du DOE/Office of Science (http://www.scidac.gov/).
Les **équipes mixtes** de spécialistes des applications, de l'analyse numérique et de l'informatique sont encouragées. La **visualisation** et la **gestion des grands volumes de données** sont aussi clairement prises en compte. L'**open source** est largement favorisé.

Au **Japon**, le projet national **NGS** (Next Generation Computer) comporte, en plus du développement des calculateurs de 10 Petaflops soutenus pour 2011/2012, un important volet applicatif : **21 applications** ont été identifiées dans le domaine des nanosciences, des sciences de la terre et de la vie et de l'astrophysique.

En France l'**ANR**, à travers les appels à projets **CIS** (désormais englobés dans **COSINUS** depuis 2008), a encouragé le développement de plate-formes logicielles ou le passage à l'échelle de codes. L'open source est encouragé mais cela reste incitatif, notamment des industriels ou éditeurs de logiciel propriétaire sont bienvenus dans les projets. Les projets ANR sont de relativement courte durée (typiquement 3 ans) et l'Agence n'a pas vocation à assurer la pérennité à moyen et long termes des développements, a priori du ressort des organismes.

8 Moyens et compétences

La question n'est pas tant de défendre encore ici la nécessité de spécialistes du parallélisme et du génie logiciel, en plus grand nombre, pour le développement des codes scientifiques pour machines massivement parallèles, car nous espérons que ce besoin est reconnu.
Il s'agit plutôt de souligner différentes préoccupations relatives à :
- **la localisation et l'organisation des compétences**
- la formation des « informaticiens » en informatique et en sciences, mais inversement, aussi, la formation des scientifiques à l'informatique
- la **valorisation des activités et métiers « informatiques »** dans le contexte de la recherche scientifique

Concernant la localisation et l'organisation des compétences, il s'agit de doser entre constitution d'équipes regroupant les différentes spécialités « verticalement » par projets, et d'équipes de support applicatif et d'expertise qui seraient auprès des moyens (calculateurs) et en relation avec différents groupes d'utilisateurs scientifiques. Les deux approches ont leur place et leur nécessité. La « Maison de la Simulation » que doivent mettre en place CEA, CNRS et INRIA sur le plateau de Saclay devrait jouer un rôle central dans ces processus de mutualisation.

Les deux autres questions ont bien sûr été abordées par l'Atelier 4 « Formation », auquel nous renvoyons. En bref, la spécialisation nécessaire des scientifiques et développeurs dans leurs domaines respectifs ne doit pas cautionner l'absence de solide formation au HPC pour les scientifiques qui ont besoin de cet outil ni de bagage dans différents domaines scientifiques (modélisation, physique, analyse numérique…) pour les informaticiens. Et il faut arriver à valoriser les activités de «développement » qui ne sont pas directement sources de publications et de reconnaissance scientifique.

Peut-on finalement espérer que la mutualisation puisse avoir comme effet vertueux une visibilité accrue, et la mise en valeur, des activités plus techniques (programmation) dans un contexte scientifiques, ainsi qu'une plus grande facilité pour obtenir des postes pour le développement ?

9 Remarques finales et recommandations

Cette section rassemble des remarques générales et recommandations qui nous semblent résumer les sections qui précèdent et en découler.

- **Informer et sensibiliser** encore et encore sur les **enjeux pratiques réels**. Paralléliser efficacement sur plusieurs milliers, voire dizaines de milliers de processeur n'est pas le même travail que pour quelques dizaines ou centaines de processeurs. L'architecture des machines et les algorithmes associés doivent être pris en compte pour développer des nouveaux modèles physiques : certains peuvent être performants d'un point de vue scientifique sans pouvoir être implantés sur machines massivement parallèles.
- **Partager plus d'informations** sur les **évolutions technologiques et les feuilles de route** des constructeurs entre spécialistes du HPC et des applications (vision partagée des enjeux logiciels sur les machines 2010, puis au-delà). Un projet tel que PRACE se construit une telle vision et peut sans doute aider à ce type de démarche, dont le prolongement naturel peut être de participer aux orientations ou même d'influer sur les tendances de conception des supercalculateurs.
- La course à la performance n'est pas tout, il faut d'abord **privilégier la qualité (numérique, modèles), la pérennité des développements**.
- **Analyser des expériences passées ou actuelles** de mutualisation, et dégager les raisons de succès ou échecs éventuels.
- Prendre en compte les **dimensions historiques et culturelles** des différents domaines.
- **Dégager des priorités** (domaines d'applications sélectionnés, composants les plus importants…) pour mieux utiliser les moyens mobilisables.
- **Organiser des équipes pluri-displinaires** (physique, numérique, informatique) ; **former les développeurs** encore plus au génie logiciel et aux nouvelles formes de programmation
- Il faudrait des initiatives plus « politiques » : PRACE p.ex. est un projet « infrastructure » avec une fraction de ses moyens sur le logiciel/les applications mais ce n'est que 1/8 environ donc très peu, saupoudré sur pas mal de codes ; donc des projets centrés sur les logiciels seraient utiles aussi. **Un SciDAC européen** ? En effet l'échelle nationale serait sans doute vite débordée pour de telles ambitions. On peut citer à cet égard un travail important réalisé dans le cadre **HET** et sans doute pas encore suffisamment exploité :
  « The Scientific Case for a European Super Computing Infrastructure », http://www.hpcineuropetaskforce.eu/deliverables
- Les **organismes** se doivent de soutenir les actions de mutualisation des codes sous des formes à définir (p.ex. **labellisations**), ne serait-ce que pour garantir leur pérennité. Néanmoins ce soutien ne pourra fonctionner qu'accompagné d'**opérations « bottom-up » conjointes** : la mutualisation ne pourra être un succès que si les partenaires en sont eux mêmes intimement convaincus. Inciter, convaincre, à des échelles judicieusement choisies est important. La « Maison de la Simulation » pourrait, en particulier, prendre une part active à ces opérations de soutien.

Annexe 1 : Liste des participants au groupe de travail

Berthelot	Christophe	christophe.berthelot@bull.net	BULL
Besbes	Mondher	mondher.besbes@institutoptique.fr	LCFIO – CNRS
Boussa	Hocine	hocine.boussa@cstb.fr	CSTB
Brun	Allan Sacha	sacha.brun@cea.fr	CEA-Saclay
Chauliac	Christian	christian.chauliac@cea.fr	CEA
Cohen	Claude	claude.cohen@bull.net	BULL
Danjean	Vincent	Vincent.Danjean@imag.fr	Université Grenoble
David	Olivier	olivier.david@bull.net	Bull HPC
Daydé	Michel	dayde@enseeiht.fr	IRIT – INPT/ENSEEIHT
Deghilage	Gilles	gilles.deghilage@tridiagonal.com	Tridiagonal
Dussoubs	Bernard	bernard.dussoubs@mines.inpl-nancy.fr	LSG2M – Ecole des Mines
Farge	Marie	farge@lmd.ens.fr	ENS-Ulm
Foujols	Marie-Alice	marie-alice.foujols@ipsl.jussieu.fr	IPSL – Pôle de modélisation
Fraigneau	Yann-Christophe	yann.fraigneau@limsi.fr	LIMSI-CNRS UPR 3251
Kern	Michel	michel.kern@recherche.gouv.fr	Ministère de la Recherche
Lévy	Claire	Claire.Levy@locean-ipsl.upmc.fr	LOCEAN-IPSL-CNRS
Louvet	Violaine	louvet@math.univ-lyon1.fr	CNRS/Université Lyon 1
Mascart	Patrick	Patrick.Mascart@aero.obs-mip.fr	Observatoire Midi-Pyrénées
Miniussi	Alain	alain.miniussi@oca.eu	Obs. de la Cote d'Azur
Morvan	Dominique	dominique.morvan@univmed.fr	Université de la Méditerranée
Nominé	Jean-Philippe	jean-philippe.nomine@cea.fr	CEA/DIF
Poinsot	Thierry	poinsot@imft.fr	Institut de Mécanique des Fluides de Toulouse
Refloch	Alain	alain.refloch@onera.fr	ONERA
Requena	Stephane	stephane.requena@genci.fr	GENCI
Rohmer	Marie-Madeleine	mmrohmer@quantix.u-strasbg.fr	CNRS
Tenaud	Christian	Christian.Tenaud@limsi.fr	LIMSI
Valcke	Sophie	valcke@cerfacs.fr	CERFACS
Vilotte	Jean-Pierre	vilotte@ipgp.jussieu.fr	Institut de Physique du Globe
Leclerc	Hugo	leclerc@lmt.ens-cachan.fr	LMT-Cachan
Dada	Laurent	laurent.dada@cea.fr	CEA/DEN

Annexe 2 : Exemples de domaines d'application et de communautés scientifiques

1. Climat

Cette communauté internationale très organisée ne s'est pas construite en un jour (30 ans de recul). Différents pays montrent différentes pratiques « techniques » : en France, on observe une approche par agrégation/couplage de composants/modèles ayant leur vie propre. Cette méthode modulaire a l'avantage de la souplesse, les USA p.ex. ont des codes plus monolithiques. Exemple : le code NEMO (océan) se couple à l'atmosphère par l'outil OASIS.
Les évolutions progressives des composants/modèles sont coordonnées : modèles N s'appuyant sur des composantes validées N-1. Les codes sont principalement Fortran/Fortran 90, ils ne comportent pas de noyaux de calcul coûteux, ils traduisent et capitalisent les savoir-faire des modèles.
La mutualisation atteint également des outils de couplage, de visualisation et de traitement des données.
Les équipes de développement sont assez importante, mais beaucoup d'intervenants cumulent ou alternent les activités d 'utilisation et de développement. Il existe certaines équipes plus spécialisées pour l'intégration ou le développement de composants (tel le coupleur OASIS, bibliothèque d'échange, transformation et interpolation de données entre pas de temps de codes).
La communauté fait vivre une vision partagée des visées et besoins à 5 ou 10 ans, exprimée en termes de complexité/résolution/nombre de calculs.
Soulignons aussi une longue pratique communautaire des analyses croisées et comparaisons de résultats. Des projets nouveaux de standardisation des métadonnées dont également à signaler (ex : METAFOR).

2. Matériaux et mécanique

Nous avons, lors de la réunion de travail du 8 septembre, bénéficié de deux témoignages de laboratoires ou instituts de taille significative (quelques dizaines à centaines de personnes), dans le domaine de la mécanique et des matériaux (LMT/Cachan et Institut Jean Lamour à Nancy).
A chaque fois, une très large variété d'activités de besoins « métier » a pour corollaire des besoins en calculs et développement de codes très diversifiés et dispersés : échelles physiques et modèles très différents ; calculs de petits à très gros ; codes commerciaux, tiers, ou développés sur place.
Dans les deux cas, il existe un accès possible aux moyens de calcul nationaux. Le besoin d'efforts et de support local « numérique et informatique » pour développer des codes et préparer des grands calculs est bien identifié mais les ressources dédiées sont très limitées (de l'ordre de 2 à 4 personnes).
Le,LMT développe une approche originale de mutualisation par technique à base de langage « introspecté » et de générateurs de code spécialisé sur des cibles variées (dont des machines parallèles avec accélérateurs de calcul). Ceci encourage la production de « codes de recherche » par des non informaticiens. Les tout premiers retours sur cette jeune approche renvoient à des difficultés évoquées section 4.3 : p.ex. l'adoption par des doctorants n'a pas été évidente à cause du supplément de formation et d'appropriation requis, sans garantie de continuation facile au-delà de la thèse (dépendance par rapport à un environnement « local »).
Nous ne sommes pas ici dans des logiques de « communauté », au sens par exemple de celle de la physique des matériaux « computationnelle » évoquée en section 6. Mais ces exemples renvoient à des questions de taille critique (du groupe d'intervenants HPC « local »), de développement durable et d'accès distant à des moyens de haute performance avec un support adapté pour des utilisateurs plus « occasionnels » du HPC extrême.

3. Sciences de la terre

Nous renvoyons à la conbribution « Quelques défis applicatifs en en Sciences de la Terre » communiquée par J.-P. Vilotte , E. Chaljub, E. Dormy, A. Fournier, J. Virieux à l'atelier 3 et déposée sur le wiki de cet atelier à l'adresse :
http://penser-petaflops.cines.fr/lib/exe/fetch.php?id=accueil&cache=cache&media=challengessciencesdelaterre.pdf

4. Astrophysique

Contribution A. S. Brun, SAp, CEA-Saclay, sacha.brun@cea.fr

Le calcul HPC en astrophysique est crucial pour interpréter et comprendre la complexité des processus physiques présents dans les objets cosmiques. L'avènement d'ordinateurs de classe pétaflopique ouvre de nouveaux horizons pour des grands challenges en astrophysique, notamment en cosmologie et formation des grandes structures de l'Univers, sur l'évolution et la collision de galaxies, en physique du milieu interstellaire et formation des étoiles, sur la modélisation des dynamos astrophysiques, sur la convection turbulente et le magnétisme solaire et stellaire, sur les disques d'accrétion ainsi que sur la formation et la migration planétaire et enfin sur le développement de simulations multi-échelles incluant plusieurs classes d'objets astrophysiques imbriqués.
La dynamique des fluides astrophysiques (ou AFD) est souvent le « fil d'Ariane » de tous ses défis numériques. En effet les objets cosmiques possèdent souvent des propriétés et/ou des forces similaires, comme par exemple, l'influence des forces de gravité et de la rotation (Coriolis et centrifuge), la présence de gaz ou plasma plus ou moins stratifiés et/ou turbulents, la présence d'effets liés au transfert de rayonnement et de chaleur, voire de l'influence d'un champ magnétique, d'une chimie complexe ou d'effets relativistes.
L'universalité des lois physiques et leurs applications aux objets cosmiques permet donc aux astrophysiciens de développer des programmes numériques de tous premiers plans faisant appel à des algorithme et méthodes numériques originales et spécifiques au domaine pouvant être utilisé/mutualisé par une communauté large. De plus l'avènement d'ordinateurs massivement parallèles a incité les astrophysiciens à développer des programmes MPI, OpenMP ou hybride performants sur ces plateformes.
Comme programmes et/ou méthodes numériques communes on citera par exemple :
Les méthodes à raffinement de maille (AMR), les méthodes particulaires (SPH), les méthodes spectrales basées sur les FFT, les méthodes aux caractéristiques pour le traitement des ondes ou de conditions aux limites ouvertes, les méthodes multi-groupe pour le transfert de rayonnement etc….
Des solveurs ou des librairies scientifiques (Paramesh, FFTW, BLAS-LAPACK) standardisés ont vu le jour et sont régulièrement utilisés dans les codes astrophysiques.
Plusieurs codes communautaires ont été développé de part le monde, on pourra notamment citer : ZEUS, Ramses, Pencil, Gadget, Parody, Flash Center, etc….
Des efforts de validation de ces codes aux méthodes variées par des comparaisons systématiques sur des cas tests bien définis (ou benchmark) ont vu le jour et permettent une mutualisation de la définition de tels tests cruciaux pour la qualité des résultats scientifiques publiés.
La visualisation de résultats (data cube) multi-dimensionels et dépendant du temps de plusieurs champs scalaires et/ou vectoriels ou de quantités dérivées des champs primaires (soit via des coupes, des iso-surfaces, ou un rendu volumique) est également très importante et un facteur clé de l'analyse des simulations astrophysiques. Mutualiser les efforts de développement des (d'un) logiciels de visualisation est donc prioritaire. De tels efforts ont déjà été entrepris et ne demandent qu'à être renforcés. On citera par exemple la librairie astrophysique d'IDL ou les logiciels modernes de visualisation SDvision (CEA-IRFU) et VAPOR (NCAR-Boulder, USA).
Associés avec la visualisation, la mutualisation de serveurs graphiques pour visualiser concrètement et en temps réels ces jeux de données dépassant maintenant régulièrement les 500^3 est également nécessaire. De même un méso centre de calcul (ou du temps garanti sur les grands centres nationaux (GENCI)) dédié à l'astrophysique est plus que d'actualité.
Comme on vient de le voir l'astrophysique a déjà un pied dans les calculs HPC du XXIème siècle il faut donc renforcer cette approche en mutualisant encore plus nos efforts et en « industrialisant » les processus de développement de ces codes communautaires.

Il reste cependant beaucoup à faire notamment le saut/portage/optimisation des codes astrophysiques aux machines extrêmement parallèles (> 50,000 cœurs). Un effort de mutualisation français, européen et international est donc nécessaire. L'utilisation de GPU et de machines hybrides (CPU-GPU) devra aussi faire l'objet d'efforts concertés que se soit pour le calcul HPC ou pour la visualisation des résultats.

Contribution de Pierre-Yves.Longaretti@obs.ujf-grenoble.fr
Chercheur en astrophysique, LAOG, Grenoble.

Le développement de codes complexes est un travail de longue haleine et qui ne conduit pas nécessairement à des publications rapides. De ce point de vue, cette activité semble similaire au développement d'instruments.

Il faut distinguer au moins deux types d'outils numériques. D'une part, les outils très génériques, comme la résolution de problèmes d'algèbre linéaire ou autres (genre Matlab ou Mupad), de problèmes de statistiques (genre R), ou de problèmes de traitement d'image (genre IDL). Ces codes peuvent être commerciaux ou non, mais correspondent à des cahiers des charges stricts permettant une utilisation autonome avec une documentation fouillée. Leur développement est un travail d'équipes souvent importantes, sur une longue période de temps. Ils s'adressent à communautés d'utilisateurs très vastes et très variées. Quand ils sont développés pour des besoins plus spécifiques pour des communautés plus ciblés, ils le sont en général dans le cadre de programmes internationaux, et représentent un travail reconnu en tant que tel, souvent lié à un développement instrumental (c'est le cas par exemple des outils numériques de dépouillement des résultats d'expériences ou d'observation sur de très grands équipements, comme le CERN, ou le Very Large Telescope au Chili). Il n'y a pas de problème de reconnaissance ni d'évaluation pour ce genre de travail. Les outils numériques en question sont de façon quasi-systématique très connus dans les communautés cible.

Le deuxième type de code correspond à la résolution de problèmes plus spécifiques, comme les codes de résolution des équations fluides (hydrodynamiques et magnéto-hydrodynamiques, par exemple) ou plus généralement de milieux continus (un exemple peu connu peut-être concerne les codes de résolution des équations de la chromodynamique quantique sur réseau), ou de codes de résolutions de structure électronique en physique moléculaire ou en physique du solide etc. Là non plus, le travail d'inventaire est en fait peu utile, pour diverses raisons. S'il est vrai que ces codes demandent quelquefois des années de développement, ils commencent en général à être productifs pendant la phase de développement même, et s'ils sont performants, il sont très vite identifiés par les résultats qu'ils produisent dans les communautés concernées. Les développeurs de ce type de code ont de plus largement réalisé que la façon la plus simple d'augmenter la reconnaissance et la visibilité de leur travail est de mettre leurs codes à la disposition de la communauté. En astrophysique, par exemple, un code comme ZEUS, l'un des premiers conçu pour résoudre des problèmes magnéto-hydrodynamiques, est devenu très célèbre de cette façon, et continue d'être l'un des principaux codes de cette communauté. Les risques d'appropriation du travail réalisés dans le développement du code sont extrêmement faibles, parce que l'utilisation de ces outils très spécialisés réclame souvent une collaboration avec leurs développeurs. La reconnaissance du travail de développement de code dans ce cas se fait non seulement à travers les publications que les développeurs produisent de façon directe avec leur code, mais aussi indirectement à travers ces collaborations. Souvent, les développeurs de code performants sont donc très connus dans les communautés auxquelles ils s'adressent. Ces outils sont de surcroît souvent très adaptés à des problèmes spécifiques, et nécessitent des adaptations (quelquefois importantes) à tout nouveau problème. C'est lié tant à la façon « semi-artisanale » dont ils sont réalisés qu'au besoin de les adapter autant que possible à l'architecture des machines sur lesquels ils tournent, les développeurs travaillant toujours à la limite de faisabilité dans la recherche de résultats nouveaux. Il est important de noter que cette grande adaptation des codes au problème fait que leur versatilité est limitée. Par exemple un code tel que ZEUS mentionné plus, développé en astrophysique, ne sera que peu utile pour résoudre des questions comme celle de la génération du champ magnétique terrestre, qui pourtant est lui aussi un problème de magnéto-hydrodynamique. C'est lié tant à des questions de géométrie, de conditions aux limites que de régimes physiques différents, alors que, d'un regard extérieur, les problématiques astrophysiques et géophysiques en magnéto-hydrodynamique sont très proches. Cet état de fait ne

semble pas destiner à changer rapidement, la performance sur un problème donné étant critique pour les progrès en recherche par rapport à la versatilité.

5. Mécanique des fluides – combustion

La mutualisation des codes en mécanique des fluides au delà d'un seul laboratoire reste aujourd'hui très limitée, les notions de propriétés et de concurrence entre équipes restant très sous-jacentes (certains laboratoires ont vendu, voire vendent encore, directement ou indirectement codes et résultats de simulations numériques). Ce comportement relativement individualiste est probablement accentué par l' équipement en clusters des laboratoires, en partie pour pailler la défaillance passée des moyens de calculs nationaux. On peut toutefois signaler le code AQUILON, développé par le laboratoire TREFLE à Bordeaux (www.trefle.u-bordeaux1.fr/index.html?/aquilon/index.html) qui peut être mis à disposition de la communauté scientifique.

Une notable exception : AVBP développé par le CERFACS et l'Institut Français du Pétrole, tous deux co-propriétaires (http://www.cerfacs.fr/cfd/avbp_code.php). Ce code pour la simulation aux grandes échelles des écoulements compressibles, réactifs ou non, est mis gracieusement, sources comprises, à la disposition de la communauté scientifique. De nouvelles versions sortent régulièrement et peuvent intégrer les apports des laboratoires si ceux-ci le souhaitent et si ces apports sont compatibles avec la politique de développement du code. Ce code a la particularité d'avoir été testé sur la plupart des architectures massivement parallèles (BlueGene, Tera10,...) avec d'excellents facteurs de speed-up. C'est probablement un des codes les plus aptes à exploiter efficacement rapidement des machines de classe pétaflops.

Annexe 3 : Présentations de la journée du 8 septembre 2008