Calcul de rais en tomographie sismique

Exploitation sur la grille

Stéphane Genaud* — **Marc Grunberg*,****

* *LSIIT-ICPS, UMR 7005 CNRS-ULP*
Bd S. Brant, F-67400 Illkirch

** *IPGS, UMR 7516 CNRS-ULP*
5, rue R. Descartes, F-67084 Strasbourg

genaud@icps.u-strasbg.fr
grunberg@renass.u-strasbg.fr

RÉSUMÉ. *La tomographie sismique permet de modéliser la structure interne de la Terre. L'utilisation de très grandes quantités de données permet d'affiner la qualité du modèle mais requiert une puissance de calcul considérable. Nous présentons une application parallèle de calcul de rais sismiques, ainsi que son exploitation sur une grille de calcul expérimentale utilisant le réseau Renater. L'application présente une phase massivement parallèle de tracé de rais sismiques dans un maillage du globe, suivie d'une phase d'échange global d'information entre les processeurs. Nous montrons l'évolution des performances de l'application par rapport à l'évolution du réseau sous-jacent, et nous ramenons ces chiffres à ceux obtenus sur une machine parallèle et un cluster. Le gain obtenu en utilisant Renater 3 au lieu de Renater 2 suggère que l'exploitation d'applications parallèles de classe similaire est envisageable sur de telles grilles.*

ABSTRACT. *Seismic tomography enables to model the internal structure of the Earth. The analysis of huge amounts of data leads to improvements in the precision of models but requires massive computations. We present a parallel application for seismic ray-tracing and its exploitation on an experimental computational grid built over the Renater network. The application first phase is a massively parallel ray-tracing computation in an Earth mesh, followed by an all-to-all exchange of information between participating processors. We show how the application performance evolves when the underlying network changes and we compare this performance with results obtained on a parallel computer and on a cluster. The gain when using Renater 3 instead of Renater 2 suggests that exploiting similar parallel applications on such grids is conceivable.*

MOTS-CLÉS : *application parallèle MPI, grille de calcul, tracé de rai sismique.*

KEYWORDS: *MPI parallel application, computational grids, seismic ray-tracing.*

1. Introduction

L'enjeu essentiel de la tomographie sismique est de pouvoir modéliser de façon réaliste la structure interne du globe. La tomographie utilise des méthodes ayant pour objectif de décrire la géométrie et les caractéristiques physiques des hétérogénéités, soit de température, soit de composition, de l'intérieur de la Terre. Les paramètres qui nous permettent de déduire ces caractéristiques physiques sont les vitesses de propagation des ondes sismiques qui sont de deux types : compression ou cisaillement.

Les données fondamentales pour aborder cette modélisation 3D sont les sismogrammes collectés, lors de séismes, par différentes stations sismologiques couvrant la surface de la planète. Ces sismogrammes sont analysés pour déterminer les temps d'arrivée des différentes ondes sismiques, ainsi que pour calculer la localisation du foyer du séisme. Des organisations internationales, comme l'ISC (*International Seismic Center*) conservent l'ensemble de ces informations dans des bases de données qui peuvent contenir plusieurs millions de ces temps d'arrivée.

La première phase de notre application consiste en une modélisation de la propagation d'un front d'onde sismique (ou rai) du foyer du séisme à une station sismologique. Cette modélisation nous permet, en utilisant les banques de données, de tracer de très grandes quantités de rais dans un maillage de la Terre. Aujourd'hui, les méthodes de tomographie utilisées s'appuient principalement sur un maillage constitué, pour une couche géologique donnée, de cellules de dimensions constantes en latitude et longitude. La qualité de la solution tomographique (*i.e.* le modèle de Terre) va dépendre principalement de la géométrie des rais sismiques. On dira qu'une cellule du maillage est bien illuminée si elle est caractérisée par un bon échantillonnage 3D des rais en azimut et en incidence. Compte tenu d'une distribution non uniforme des séismes et des stations sismologiques, certaines régions à l'intérieur de la Terre sont richement *illuminées*, d'autres très pauvrement. Il en résulte, de ce fait, un biais du modèle tomographique. Notre objectif final est de construire un maillage adaptatif de la Terre, où la quantité d'informations apportée par les rais est distribuée de façon homogène sur l'ensemble des cellules, c'est-à-dire qu'une région avec peu de rais sera modélisée par de grandes cellules alors qu'inversement une région richement illuminée sera décrite par de nombreuses petites cellules.

Le travail présenté ici concerne le tracé de rais, et la construction du maillage uniforme de départ. L'ensemble des algorithmes a été parallélisé et implémenté sur différentes plates-formes : machine parallèle, cluster et grille de calcul. L'objectif de cet article est d'évaluer les performances de notre implémentation sur une grille de calcul, par rapport à des architectures plus homogènes (cluster et machine parallèle). Nous présentons tout d'abord la méthode de tracé de rais sismiques dans la section 2, et nous exposons ensuite notre stratégie de parallélisation dans la section 3. La section 4 est consacrée à l'évaluation des résultats obtenus sur une grille de calcul. Ces résultats sont aussi comparés à ceux obtenus sur la machine parallèle et le cluster. Enfin, dans la section 5, nous replaçons ce travail dans le contexte de la recherche actuelle en présentant des expériences similaires.

2. Le tracé de rai sismique dans un maillage

2.1. *Le tracé de rai sismique*

Une onde sismique est modélisée par un ensemble de rais représentant la propagation de son front d'onde du foyer du séisme vers les stations sismologiques. Le trajet d'un rai est constamment perpendiculaire au front d'onde et obéit aux lois de la réflexion/réfraction lorsqu'il atteint une interface géologique. En étant réfléchie ou transmise, l'onde sismique peut éventuellement passer d'un mode de propagation en compression (onde P) à un mode en cisaillement (onde S), et réciproquement. Tous ces changements qui se produisent lors de la propagation du rai constituent *la signature du rai*. Cette signature est formée de symboles qui permettent de reconstituer son histoire. Par exemple, un rai ayant une signature PcS signifie que l'onde de compression se propage jusqu'à l'interface manteau-noyau où elle est convertie en onde de cisaillement et est réfléchie vers la surface. Les sismogrammes, enregistrés par les stations sismologiques lors d'un tremblement de terre, sont analysés pour détecter les différents temps d'arrivée des ondes sismiques et déterminer leur signature.

Le calcul du trajet d'un rai nécessite la connaissance des coordonnées du foyer du séisme et de la station sismologique sur laquelle il a été détecté, de sa signature, ainsi que d'un modèle de vitesse. L'algorithme du tracé de rai repose sur la loi de Descartes en géométrie sphérique, qui décrit les variations de vitesses sismiques :

$$p = r.\frac{\sin(i)}{v(r)} \qquad [1]$$

où p est le paramètre de rai constant en tout point du rai, $v(r)$ est la vitesse de propagation de l'onde à une profondeur r, et i l'angle d'incidence du rai à cette même profondeur comme illustré figure 1.

Pour tracer le rai sismique, nous utilisons un modèle de vitesse couramment utilisé en géophysique : le modèle IASPEI91 (Kennett, 1991) qui ne dépend que de la profondeur. Le calcul ne dépendant pas de l'azimut, le rai est par conséquent dans le plan défini par le foyer, la station sismologique et le centre de la Terre. Le tracé de rai est tout d'abord calculé en 2D dans un plan, puis replacé en 3D par une succession de trois rotations.

La construction du rai en 2D est un processus itératif qui consiste à propager le rai pas à pas, en procédant soit par segments linéaires élémentaires, soit par décalages angulaires élémentaires, en fonction de l'angle d'incidence du rai. Quand celui-ci est proche de 90° l'algorithme basé sur l'équation de Descartes passe d'un mode de propagation par segments linéaires à une propagation par décalages angulaires, pour garantir un tracé correct. Typiquement un rai sismique est souvent discrétisé par plusieurs centaines, voire plusieurs milliers de points en fonction de sa longueur.

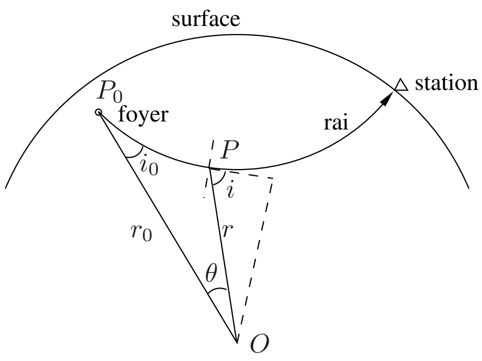


Figure 1. *Chaque point P du rai est défini par sa profondeur r, son angle d'incidence i, et θ = ($\widehat{P_0OP}$), où O est le centre de la Terre. Le premier point P_0, situé au foyer, est défini par (r_0, i_0 et $\theta_0 = 0$)*

2.2. *Le maillage*

Le maillage initial d'une partie ou de la totalité de la Terre est obtenu en décomposant une sphère en un certain nombre de couches concentriques, de la surface jusqu'au centre. Chaque couche est ensuite décomposée, à partir du centre de la Terre, en secteurs angulaires à la fois en latitude et en longitude. Les volumes élémentaires ainsi créés peuvent être approximés par des hexaèdres et constituent les cellules du maillage. La figure 2 montre une portion du maillage faisant apparaître cinq couches (il y a une couche très fine en surface). Les cellules sont représentées par le quadrillage apparaissant en filigrane.

L'objectif final étant de construire un maillage adaptatif, le programme de tracé de rais calcule non seulement le trajet du rai, mais aussi certaines informations relatives à la distribution des rais dans les cellules du maillage initial (l'*illumination*). La construction de ce nouveau maillage sera réalisée en fusionnant les cellules du maillage initial, dont l'illumination n'est pas suffisante.

3. Parallélisation

La difficulté pour paralléliser notre application réside dans le fait d'avoir à stocker simultanément en mémoire, pour chaque processeur, les rais et le maillage.

Une possibilité serait d'allouer à chaque processeur une partie distincte du maillage basée sur une décomposition géographique (par exemple nous pouvons attribuer les deux hémisphères à deux processeurs). Chaque rai serait alors tracé par le processeur responsable de la région géographique dans lequel il se propage.

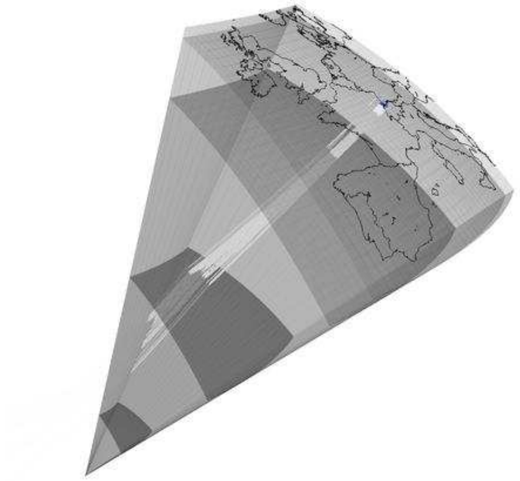


Figure 2. *Tracé de rais arrivant sur l'Alsace. Le tracé est opéré dans un maillage*

Cependant cette approche pose deux problèmes. Premièrement, le calcul des rais sur les différents processeurs serait largement déséquilibré puisque les capteurs sont irrégulièrement placés à la surface du globe (il y a très peu de capteurs au niveau des océans par exemple) et par conséquent certains processeurs ne verraient que très peu de rais passer dans leur zone. Deuxièmement, à chaque fois qu'un rai passerait d'une zone à une autre, cela conduirait à des communications inter-processeurs. En outre, augmenter le nombre de processeurs ferait croître le nombre de communications, et conduirait à des performances catastrophiques.

Notre stratégie de parallélisation est basée sur la réplication du maillage sur chaque processeur. Pour contourner les problèmes liés à l'occupation volumineuse en mémoire d'une telle réplication, chaque processeur calcule seulement un minimum d'informations sur le maillage : les liens entre cellules ainsi que les coordonnées des sommets des cellules ne sont pas stockés, et la mémoire nécessaire pour stocker les informations d'une cellule est allouée seulement lorsque nécessaire. L'économie mémoire réalisée en utilisant un tel maillage *allégé* est conséquente : si on utilisait un maillage complet dans notre expérience (nous avons 712800 cellules) la part de mémoire occupée par les liens et les coordonnées serait d'environ 130 Mo. Quand nous mesurons l'espace mémoire occupé par un processus (code et totalité des données) de l'application avec maillage allégé, chaque processus prend 190 Mo sur 16 processeurs, et 320 Mo sur 8 processeurs. Sans le maillage allégé, cette occupation mémoire serait donc de 320 Mo et 450 Mo respectivement, soit un gain de 41 % avec 16 processeurs, et 29 % avec 8 processeurs. De plus, cette économie a l'énorme avantage de permettre l'exécution du programme sur des machines ayant moins de 512 Mo de mémoire.

Un nombre quelconque de processeurs peut être utilisé lors d'une exécution parallèle, et nous plaçons habituellement un processus par processeur. Chaque processus reçoit du maître la description du maillage correspondant à la zone d'étude ainsi qu'un premier bloc de rais à tracer. Chaque processus commence alors à calculer ses rais segment par segment. Pour chaque segment, le processus teste si le rai vient d'entrer dans une nouvelle cellule, auquel cas une zone mémoire est allouée et les propriétés géométriques et géophysiques du rai y sont stockées. Dans le cas contraire les informations de la cellule traversée sont mises à jour. Dès qu'un processeur a terminé les calculs de rais du bloc, il en demande un nouveau au processeur maître. A titre indicatif, les tailles des blocs dans les expériences décrites dans la suite sont respectivement de 8170 et 5106 rais pour 10 et 16 processeurs. Notons que nous ne décrivons ici que l'implémentation *maître/esclave* de l'application ; c'est la version utilisée dans cet article car elle garantit un équilibrage de la charge de calcul satisfaisant en environnement hétérogène (David *et al.*, 2002).

Une fois que tous les rais ont été calculés par l'ensemble des processus, l'information contenue dans une cellule donnée est potentiellement répartie sur plusieurs processeurs. Nous devons alors fusionner, pour chaque cellule, l'ensemble de ces données distribuées. A chaque processus est affectée une partie distincte du maillage appelée *sous-domaine*, issue d'un découpage régulier du maillage. Il s'ensuit que chaque processus est capable de déterminer calculatoirement à qui est attribué un sous-domaine donné. Notons qu'il peut exister un déséquilibre de la répartition des données dans de tels sous-domaines, mais que le découpage en sous-domaines équilibrés nécessite une phase supplémentaire d'échange globale d'information qui n'a pas encore été étudiée. Chaque processus est responsable de la récupération auprès des autres des informations pour les cellules de son sous-domaine. S'en suit d'abord une phase de communication de type *tous à tous* où chaque processus envoie aux processus appropriés les données qu'il a calculées, puis reçoit ensuite les données des autres processus concernant son *sous-domaine*. Notons que cette phase est extrêmement coûteuse en temps : pour n processus il y a n^2 échanges de *sous-domaines* nécessaires, et pour des exécutions comme celle présentée section 4.1, environ 1,5 Go de données au total est échangé. Chaque processus ayant reçu toutes les données le concernant, fusionne les informations des *sous-domaines* et calcule un *score* unique, par cellule, correspondant à son illumination.

Le schéma général de l'application est décrit ci-dessous :

1. Description du maillage	Le processeur maître distribue la description du maillage aux autres processeurs *(communication un à tous)*.
2. Distribution des blocs de rais	Chaque processeur demande au processeur maître un bloc de rais à traiter *(communication un à un)*.
3. Le tracé de rai	Chaque processeur calcule les trajets des rais et met à jour les cellules traversées. Retour à l'étape (2) jusqu'à épuisement des blocs de rais *(calcul)*.

4. Echange de données	Chaque processeur envoie aux autres les données des cellules dont ils ont la charge, et récupère les données des cellules dont il a la charge *(communication tous à tous).*
5. Fusion des informations	Les informations concernant une cellule sont fusionnées *(calcul).*
6. Calcul de score	Pour chaque cellule, on calcule un score à partir des informations des rais *(calcul).*
7. Concaténation	Le score de chaque cellule est envoyé au processeur maître *(communication tous à un).*

4. Exploitation sur une grille

Nous utilisons le terme *grille* dans cet article avec le sens que lui ont donné (Foster *et al.*, 1998b). Notre grille est donc une fédération de machines et de services qui se reconnaissent dans une autorité de certification commune. De manière plus précise, nous ne nous intéressons ici qu'au service permettant de faire du calcul réparti, et c'est pourquoi nous parlons de *grille de calcul*. Un tel service dans une grille consiste à l'heure actuelle en la possibilité de vérifier les identités des utilisateurs du service, de placer automatiquement des exécutables sur les ressources de calcul distantes (*executable staging*), d'unifier les différents flots d'entrées/sorties, et de synchroniser le déclenchement des exécutables sur les ressources de calcul. Dans le futur, on disposera probablement en standard d'ordonnanceurs à l'échelle d'une grille, capables d'utiliser le service d'annuaire pour localiser les ressources les plus adéquates et de planifier l'exécution parmi d'autres travaux. Notre objectif dans cet article est de relater une expérience : peut on utiliser efficacement à l'heure actuelle une grille de calcul pour faire le travail habituellement réalisé sur une machine parallèle ou un cluster.

4.1. *Préambule*

Dans cette partie nous présentons les performances de l'application décrite précédemment, obtenues sur la grille de calcul construite dans le cadre du projet TAG[1]. Cette grille de calcul est constituée de machines très diverses (PC, Sun 450 et 6800, SGI Origin 2000 et 3800) distribuées sur plusieurs sites (Strasbourg, Clermont-Ferrand, Montpellier) dont les réseaux locaux sont fédérés par le réseau Renater. D'un point de vue logiciel, toutes les machines sont installées avec Globus (Foster *et al.*, 1997) (versions 1.1.4 et 2.0) et nous utilisons MPICH-G2 (Foster *et al.*, 1998a) comme bibliothèque de communications.

1. Projet *Transformations et Adaptations pour la Grille* (TAG), http ://grid.u-strasbg.fr

Il est capital de remarquer que les expériences conduites dans ce type d'environnement très hétérogène n'ont que des valeurs indicatives. Pour l'exécution d'une application donnée, le choix des différents paramètres a une importance cruciale et peut amener des comportements radicalement différents. En particulier, nous devons choisir un sous-ensemble des processeurs de cette grille et une période d'exécution. Durant cette période, le débit des liens réseaux reliant ces processeurs ainsi que la disponibilité des processeurs peuvent varier de manière très importante. Pour complexifier encore l'étendue des tests possibles, le comportement de l'application peut changer énormément avec la taille, voire la nature du jeu de données en entrée. Ainsi, nous avons été guidés dans le choix de nos expérimentations par les éléments suivants. Nous travaillons sur un jeu de données constant, qui est l'ensemble des évènements sismiques enregistrés en 1999. Quantitativement, l'expérience consiste à calculer les trajets de 346052 rais sismiques à l'intérieur d'un maillage de la Terre, ainsi qu'à calculer les scores des cellules impactées. Le maillage est constitué de 712800 cellules réparties sur onze couches. L'utilisation de ce jeu de données constant nous permet d'étudier :

– les performances de l'exécution sur une telle grille comparativement à des architectures homogènes sur la base des mesures de performance initiées dans (Grunberg *et al.*, 2002),

– l'évolution des performances à neuf mois d'intervalle alors que l'infrastructure réseau a progressé.

Concernant le choix des processeurs, nous avons sélectionné deux sous-ensembles de notre grille, dont nous pensons qu'ils reflètent des exploitations possibles des grilles de calcul dans le futur, à savoir :

1) l'utilisation de ressources de calcul au sein d'une même université : en l'occurence, dix PC distribués sur deux campus de Strasbourg reliés par le réseau métropolitain,

2) la distribution des calculs à l'échelle nationale mais en utilisant un réseau à haut débit : ici huit processeurs du centre de calcul national universitaire (CINES) sont « appuyés » par huit autres processeurs sur des campus strasbourgeois. La figure 3 sur laquelle nous reviendrons plus en détails par la suite montre la liaison entre les sites de Strasbourg et Montpellier.

4.2. *Les plates-formes de test*

Les expériences réalisées sur les deux configurations de grille sont comparées aux mêmes exécutions sur des plates-formes homogènes (machine parallèle, cluster) dont la puissance de calcul est du même ordre de grandeur. Les ressources de calcul de notre grille sont prises parmi les processeurs listés dans le tableau 1. La colonne ratio indique les puissances relatives des processeurs, établies à partir d'une série de benchmarks utilisant la version séquentielle de l'application, et normalisées par rapport à la puissance de la machine pellinore.

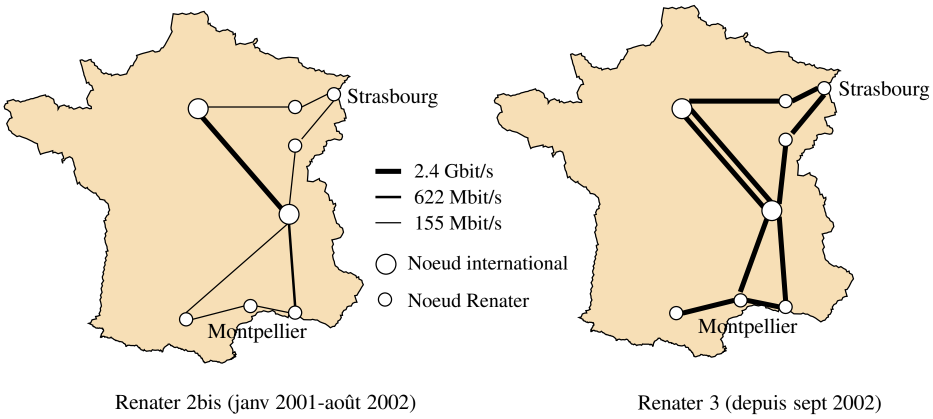


Figure 3. *Partie de la dorsale nationale Renater empruntée et évolution des débits entre Renater 2bis et Renater 3*

Machine	Nb CPUs	Type processeur	Ratio
pellinore	2	PIII/800	1
sekhmet	1	XP1800	1,61
caseb	1	XP1800	1,61
lattice	1	XP1800	1,61
nestea	1	P4/1700	1,4
darjeeling	1	P4/1700	1,4
dinadan	1	PIII/933	1,14
merlin	2	XP2000	2,26
leda	512	R14K/500	1,05
calserv	12	P4/1700 Xeon	1,4

Tableau 1. *Caractéristiques des ressources de calcul utilisées dans les expériences*

– **Machine parallèle :** la machine utilisée, leda dans le tableau 1, est une SGI Origin 3800 située au CINES à Montpellier. La puissance de calcul de chaque processeur n'est pas très élevée mais le réseau d'interconnexion entre les processeurs ainsi que les opérations entrées/sorties sont extrêmement performants. La bibliothèque de communication MPI utilisée est celle fournie par le constructeur.

– **Cluster :** le cluster de PC est composé de 6 nœuds interconnectés par un switch Gigabit Ethernet. Chaque nœud est un bi-Pentium Xeon 1,7 Ghz avec 1 Go de RAM, du type de calserv dans la table 1. La bibliothèque de communication utilisée est LAM/MPI (Squyres *et al.*, 2000).

– **Grille métropolitaine :** la première configuration, avec 10 processeurs, ne comporte que des PC. Les machines sont réparties à Strasbourg sur deux campus universitaires géographiquement distants. Le réseau d'interconnexion reliant les différentes machines sur les deux campus supporte un débit de 622 Mb/s mais les communications sont limitées par les routeurs et cartes réseaux de type fast-ethernet 100 Mb/s. Typiquement, le débit va de 3 Mo/s entre les deux campus jusqu'à 8 Mo/s pour des machines connectées au même routeur.

– **Grille nationale :** la deuxième configuration, à 16 processeurs, regroupe les deux sites strasbourgeois, ainsi que la machine parallèle du CINES. La particularité de cette configuration est qu'elle a bénéficié, à la fin de l'année 2002, d'une amélioration du réseau d'interconnexion entre le site de Strasbourg et Montpellier, dans le cadre du passage de Renater 2 à Renater 3.

4.3. *Expériences*

Les expériences présentées ci-dessous illustrent les deux cas de figure d'utilisation de grille cités en préambule. La première est celle d'une utilisation inter-campus, et la deuxième entre sites nationaux. Les comportements des exécutions sont examinés au regard des temps pris par l'application pour réaliser :

– le calcul des rais ainsi que la mise à jour du maillage (étapes 2 et 3),

– l'échange de données avec les autres processeurs et fusion des informations (étapes 4 et 5),

– la totalité des traitements (étapes 1 à 7),

– le nombre total de blocs de rais reçu par processeur.

4.3.1. *Expérience à l'échelle métropolitaine*

La figure 4 présente, pour 10 processeurs, les temps mesurés sur le cluster (4a) et la machine parallèle (4b). Ce sont là des mesures de référence qui permettent d'évaluer l'exécution sur la configuration de type grille métopolitaine présentée sur la figure 5. On retrouve dans les résultats du cluster et de la machine parallèle, les spécificités de ces architectures. Les processeurs étant homogènes dans les deux cas, le nombre de blocs de rais distribués est équitablement reparti sur chaque processeur. Les puissances de calcul agrégées (sommes des ratios) utilisées sur le cluster et la machine parallèle sont respectivement 14 et 10,5. Les performances relevées pour le calcul des rais corroborent ces mesures : le cluster a une puissance de calcul environ 50 % supérieure à celle de la machine parallèle. En revanche, le réseau d'interconnexion de la machine parallèle est extrêmement efficace par rapport à celui du cluster, et la phase d'échange des données entre processeurs est environ trois fois plus rapide sur la machine parallèle. Globalement, les temps totaux d'exécution sur le cluster et la machine parallèle sont équivalents, de l'ordre de 600 secondes.

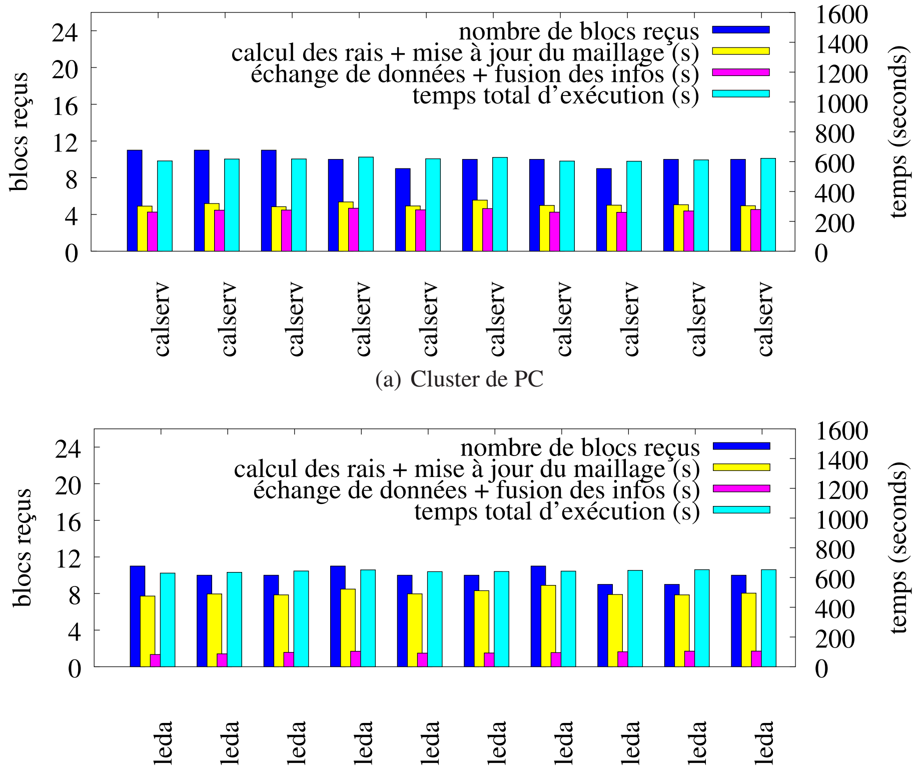


Figure 4. *Performances sur un cluster de PC et une SGI Origin 3800 (10 processeurs)*

L'exécution sur la grille métropolitaine (figure 5) ne met pas en défaut l'équilibrage de charge de l'application malgré l'hétérogénéité des processeurs et du réseau. Le nombre de blocs distribués à chacun des processeurs varie beaucoup (15 blocs pour le processeur le plus rapide, 4 pour le plus lent) mais les temps de fin sont quasi-égaux, après 900 secondes. Le temps de calcul des rais, sur la grille, est légèrement meilleur que celui du cluster, ce qui s'explique par la puissance agrégée de 15,3. Par contre, de manière prévisible, la phase d'échange de données, pénalisée par le réseau, est deux fois plus lente que celle du cluster. Il en résulte que le temps global de l'application est 50 % plus lent sur la grille que sur le cluster ou la machine parallèle.

4.3.2. *Expérience à l'échelle nationale*

L'expérience suivante a été menée sur une même configuration de grille reliant Strasbourg et Montpellier à deux reprises : la première expérience a eu lieu en août

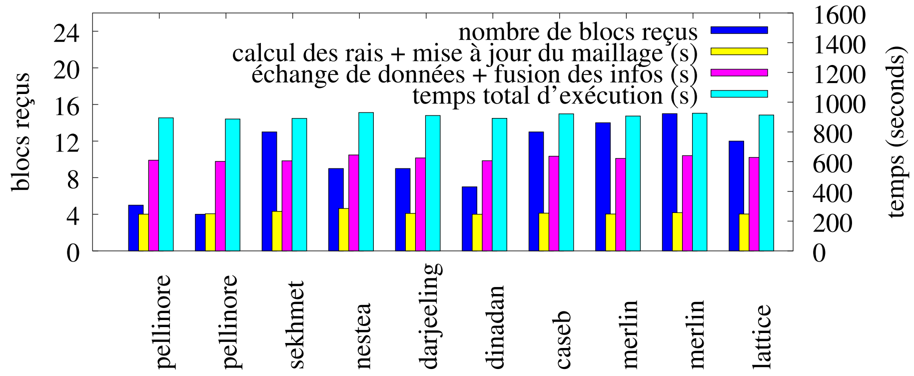
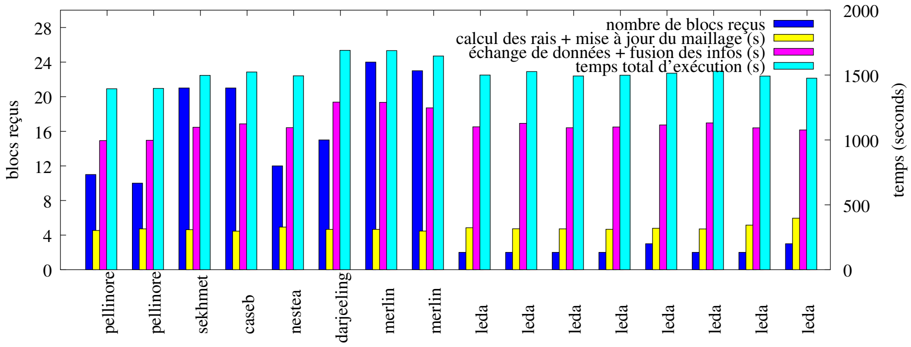


Figure 5. *Performances de la configuration type grille métropolitaine (10 processeurs)*

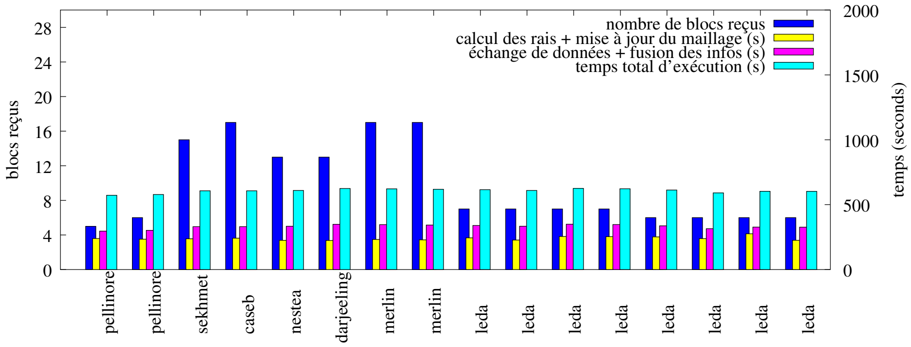
2002, et nous l'avons recommencée en mai 2003. Entre-temps, l'infrastructure reliant les deux sites a progressé, comme le montre la figure 3. Des tests de type ping-pong nous ont permis d'évaluer qu'une application MPI pouvait disposer d'un débit moyen de 0,2 Mo/s et 0,5 Mo/s des campus strasbourgeois vers le CINES avec Renater 2. Après la connexion à Renater 3, le débit moyen des deux campus vers le CINES approche 1 Mo/s. Les résultats sont présentés en figure 6a pour Renater 2, et figure 6b pour Renater 3.

Nous obtenons avec la nouvelle infrastructure réseau des performances inattendues. Le temps total d'exécution sur cette plate-forme (avec une puissance de calcul agrégée de 20,94) est approximativement le même qu'avec une configuration de 16 processeurs sur la machine parallèle du CINES présenté en figure 7 (puissance cumulée de 16,8). Bien que la puissance de calcul de la grille soit supérieure à celle des 16 processeurs du CINES, nous avons été surpris que la supériorité du réseau d'interconnexion de la machine parallèle ne donne pas un net avantage à ce type d'architecture dans ce cas.

En comparant les temps d'exécution de l'application globale, on constate que les performances ont quasiment triplé dans la nouvelle version de Renater. Les processeurs situés au CINES ont reçu, au maximum, trois blocs de rais par processeur avec Renater 2 et sept avec Renater 3 (les données initiales sont à Strasbourg). On retrouve pour l'exécution avec Renater 3, comme pour la configuration métropolitaine, la corrélation entre la puissance relative des processeurs et le nombre de blocs reçus par processeur. En revanche, la faible qualité de l'interconnexion entre Strasbourg et Montpellier, sous Renater 2, avait complètement déséquilibré la répartition de charge. La faible participation en calcul des processeurs du CINES ne s'explique que par la forte latence pénalisant ces communications nationales (demande de travail des processeurs, envoi des blocs à traiter). Cette observation est encore plus flagrante sur la



(a) Grille nationale Renater 2



(b) Grille nationale Renater 3

Figure 6. *Performances avec 16 processeurs sur la grille nationale avec Renater 2 puis Renater 3*

durée de l'échange des données (étapes 5 et 6) qui représentait alors 75 % du temps total de l'application (contre 50 % avec Renater 3).

5. Travaux similaires

Il est difficile d'établir une liste de travaux directement comparables à celui présenté dans cet article. En effet, dans le domaine des grilles de calcul, la majorité des travaux actuels proposent de nouvelles architectures logicielles adaptées à cet environnement hétérogene. Parmi ces projets d'infrastructure, on peut citer Globus (Foster *et al.*, 1997) (que nous avons utilisé dans ce travail), ou XtremWeb (Cappello *et al.*, 2002). Chacun dans des contextes différents, fournissent des mécanismes de déclenchement de processus à distance permettant de déployer du calcul parallèle.

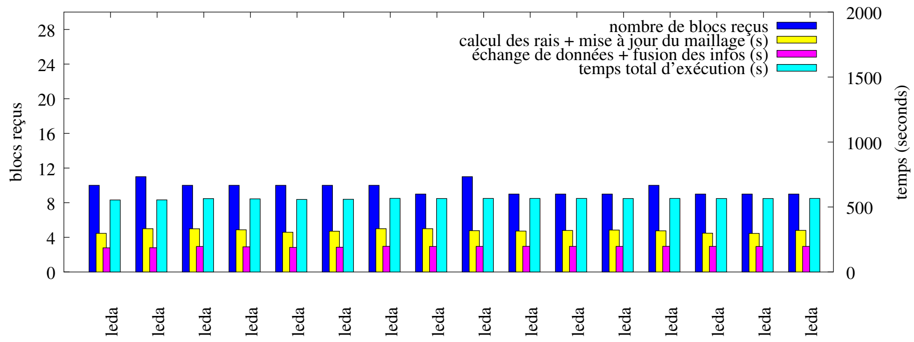


Figure 7. *Performances sur une Origin 3800 (16 processeurs)*

Certains projets tentent d'étudier l'ensemble des besoins liés à la programmation sur la grille. Ainsi, le projet Albatross (Kielmann *et al.*, 2002) aux Pays-Bas, couvre de nombreux thèmes de recherche : les modèles de programmation, avec une approche alternative basée sur le paradigme *diviser pour régner* proposé dans le langage Satin (van Nieuwpoort *et al.*, 2000), les bibliothèques de communication de niveau utilisateur, avec une version de MPI pour la Grille baptisée MagPie (Kielmann *et al.*, 1999), et enfin une couche de communication de bas niveau nommée Panda, conçue pour les communications entre clusters distants, permettant également de simuler des communications longue distance sur un cluster unique. Les expérimentations menées dans ce projet utilisent la grille de test DAS reliant quatre clusters répartis aux Pays-bas (à Amsterdam, Leyde et Delft). Des résultats concernant les performances d'un ensemble d'applications exécutées sur la grille DAS, et l'influence de l'équilibrage de charge sont présentés dans (van Nieuwpoort *et al.*, 2001).

Cependant, les études quantitatives sur les performances des applications parallèles sur ces grilles sont pour l'instant peu nombreuses. Il faut remarquer que ce sont essentiellement les projets de type *Internet Computing* comme SETI@home (Sullivan *et al.*, 1997) déployant des cas très particuliers d'applications parallèles sans dépendances entre les tâches de calcul, qui ont publié le plus de résultats à propos de l'exploitation effective d'applications.

Concernant l'exploitation d'applications parallèles générales, des travaux intéressants sont menés au sein du projet DAMIEN (Gabriel *et al.*, 2003). Dans ce projet, les expérimentations sont effectuées sur une grille constituée de ressources prises sur trois machines parallèles sur trois sites européens (Rouen, Barcelone et Stuttgart) reliées par le réseau Géant. Les applications sont des codes MPI existants (par exemple RNAfold (Hofacker *et al.*, 2002)) utilisant pour la grille la bibliothèque de communication PACX-MPI (Gabriel *et al.*, 1998). Ce travail décrit une approche possible pour le portage d'applications parallèles sur la grille. Les auteurs préconisent deux types d'analyse à l'aide d'outils adéquats pour faciliter ces portages. La première analyse est l'observation du comportement de l'application à l'aide d'un outil de trace (par exemple la librairie MetaVampir utilisée ici) qui enregistre durant l'exécution le chronogramme des messages échangés entre les processeurs. Cette analyse sert à repérer d'éventuels goulots d'étranglement ou des schémas de communications inadaptés à une exécution sur la grille. Les auteurs font remarquer que pour cette première analyse, qui est une phase de test, les exécutions ne portent généralement que sur des jeux de données de taille petite ou moyenne afin de raccourcir la durée d'exécution par rapport au temps normal en production. Avant de lancer l'exécution sur le jeu de données réel, il est proposé une deuxième analyse qui est la prédiction du temps d'exécution en fonction des caractéristiques de la grille utilisée (réseaux, processeurs). Pour cette analyse, Dimemas (Badia *et al.*, 2003) peut être utilisé pour prédire le comportement de l'application, sur la base des traces de MetaVampir. Ce logiciel possède un modèle de coûts des communications tenant compte de l'hétérogénité, de l'architecture réseau ainsi que d'une fonction du trafic réseau que l'utilisateur fixe dans un fichier de configuration. Ce projet nous semble intéressant car il essaye d'étendre l'utilisation d'outils déjà connus pour la programmation parallèle sur machines homogènes à la grille, et expérimente cette pratique sur des applications réelles. On peut cependant regretter que les logiciels utilisés, hormis PACX-MPI, ne soient pas publiquement disponibles.

Une autre expérience d'envergure, décrite dans (Allen *et al.*, 2001), est celle d'une simulation numérique dans le domaine astrophysique utilisant un total de 1500 processeurs. Les processeurs sont pris sur deux sites (SDSC à San-Diego et NCSA à Champaign-Urbana) et quatre machines parallèles (une IBM Power-SP et trois Origin 2000). Les sites sont reliés par un lien à 622 Mb/s donnant un débit moyen de 3 Mo/s. Les trois Origin 2000 sont connectées par un lien gigabit-ethernet offrant un débit de 100 Mo/s. L'intérêt du projet réside dans les adaptations nécessaires de l'application pour atteindre une efficacité acceptable. L'application testée ici fait des opérations de type *stencil*, c'est-à-dire qu'on doit calculer, à chaque itération, les valeurs de points organisés en tableau, en fonction des valeurs des points voisins à la même itération. La parallélisation de l'application repose sur la décomposition du domaine (le tableau). L'utilisation d'une telle configuration matérielle permet de calculer une simulation cinq fois plus importante que ce qui avait réalisé jusqu'alors : le tableau contient environ $8 \cdot 10^8$ points. Chaque processeur s'occupe ici du calcul d'environ $6 \cdot 10^4$ points. Dans ce cas, le nombre de communications entre machines est assez faible proportionnellement au nombre de points : étant donné la forme du tableau choisie, il n'y a que 120 processeurs qui communiquent entre les deux sites à chaque itération. Vis-à-vis de l'unique lien entre les deux sites, ce nombre reste toutefois élevé. Une des adaptations apportée à l'application pour accroître son efficacité est la mise en place d'une *zone de réplication* : des portions du tableau de points sont dupliquées sur des processeurs distants, ce qui engendre des calculs redondants. En contre-partie, cela permet pour une zone de recouvrement de taille g de n'envoyer des messages (plus longs) que toutes les g itérations, économisant ainsi de la latence. Les auteurs ont montré dans (Ripeanu *et al.*, 2001) que cette technique n'apportait pas de gain de performances sur une machine parallèle classique en raison de la faible latence du réseau d'interconnexion. Dans cette expérience en revanche, ils ont utilisé avec succès un mécanisme auto-adaptatif ajustant la taille de la zone de recouvrement en cours d'exécution, selon l'évolution de la performance. Avec ces adaptations, l'efficacité obtenue est de 88 % sur 1140 processeurs, et 63 % sur 1500 processeurs.

6. Conclusion

Savoir à quel horizon le concept de grille sera une réalité permettant de fédérer un ensemble hétérogène et dynamique de ressources de calculs pour en faire un supercalculateur est une question ouverte. Certains auteurs pensent que cette utilisation est utopique si on ne remplace pas certaines technologies actuelles. Ainsi, il est souvent dit que les algorithmes du protocole TCP ne sont pas adaptés au calcul sur des grilles à large échelle ((Bansal *et al.*, 2001), (Feng *et al.*, 2002) par exemple). Les expériences présentées dans cet article sur deux configurations successives de Renater donnent un ordre d'idée des gains que l'on peut espérer sur ce type d'application avec les technologies actuelles. Les performances de cette application, utilisant 16 processeurs, sur l'infrastructure réseau de Renater 3, sont comparables en temps d'exécution à celles obtenues sur une machine parallèle Origin 3800 avec 16 processeurs. Ces résultats nous encouragent à poursuivre des expérimentations, afin d'étudier notamment l'extensibilité des performances quand le nombre de processeurs et de sites impliqués augmente. Une continuation logique de cette expérience consisterait à utiliser un plus grand nombre de processeurs en chaque site. Notons que cela ne comporte aucune difficulté technique supplémentaire quand il s'agit de prendre plus de processeurs sur une machine parallèle, mais que la mobilisation (et l'administration qui s'ensuit) de nombreuses machines individuelles est un problème réel.

Parallèlement, de vastes travaux doivent être menés concernant l'algorithmique des applications destinées à la grille. Les recherches doivent se porter sur des aspects théoriques pour, notamment, optimiser les algorithmes de distribution des données en tenant compte de l'hétérogénité inhérente aux grilles (Beaumont *et al.*, 2002). Les recherches doivent aussi inventer et tester des techniques de programmation pour les grilles à grande échelle. Dans cette optique, nous avons mentionné celles proposées dans (Allen *et al.*, 2001) qui visent à augmenter le grain du parallélisme. Nous pensons que de nombreuses expériences doivent encore valider les idées et les théories, afin de constituer une expertise des méthodes de programmation sur la grille.

7. Bibliographie

Allen G., Dramlitsch T., Foster I., Karonis N. T., Ripeanu M., Seidel E., Toonen B., « Supporting Efficient Execution in Heterogeneous Distributed Computing Environment with Cactus and Globus », *Proceedings of SuperComputing 2001*, ACM/IEEE, p. 52, November, 2001.

Badia R. M., Labarta J., Gimenez J., Escale F., « DIMEMAS : Predicting MPI applications behavior in Grid environments », *Workshop on Grid Applications and Programming Tools (In conjunction with GGF8)*, p. 50-60, June, 2003.

Bansal D., Balakrisham H., Floyd S., « Dynamic Behavior of Slowly-Responsive Congestion Control Algorithms », *Proceedings of SIGCOMM 2001*, vol. 31, ACM Computer Communication Review, August, 2001.

Beaumont O., Carter L., Ferrante J., Legrand A., Robert Y., « Bandwidth-centric allocation of independent tasks on heterogeneous platforms », *International Parallel and Distributed Processing Symposium IPDPS'2002*, IEEE Computer Society Press, 2002.

Cappello F., Djilali A., Fedak G., Germain C., Lodygensky O., Néri V., *XtremWeb, une plateforme de recherche sur le Calcul Global et Pair à Pair*, Hermes Sciences - Lavoisier, chapter 6 - Metacomputing, p. 153-188, 2002.

David R., Genaud S., Giersch A., Schwarz B., Violard É., « Source Code Transformations Strategies to Load-balance Grid Applications », *in* , M. Parashar (ed.), *Proceedings of Grid Computing (GRID 2002)*, vol. 2536 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 82-87, November, 2002.

Feng W., Fisk M., Gardner M. K., Weigle E., « Dynamic Right-Sizing : An Automated Lightweight, and Scalable Technique for Enhancing Grid Performance », *Proceedings of the 7th International Workshop on Protocols for High-Speed Networks*, April, 2002.

Foster I., Karonis N. T., « A Grid-Enabled MPI : Message Passing in Heterogeneous Distributed Computing Systems », *Proc. of the 1998 ACM/IEEE conference on Supercomputing*, Nov., 1998a.

Foster I., Kesselman C., « Globus : A Metacomputing Infrastructure Toolkit », *The International Journal of Supercomputer Applications and High Performance Computing*, vol. 11, n° 2, p. 115-128, 1997.

Foster I., Kesselman C., *The Grid, Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, Inc., 1998b.

Gabriel E., Keller R., Lindner P., Müller M. S., Resch M., « Software Development in the Grid : The DAMIEN tool-set », *Proceedings of Computational Science - ICCS 2003*, vol. 2659 of *Lecture Notes in Computer Science*, Springer-Verlag, p. 235-244, 2003.

Gabriel E., Resch M., Beisel T., Keller R., « Distributed Computing in an Heterogeneous Computing Environment », *Proceedings of PVM/MPI*, p. 180-187, 1998.

Grunberg M., Genaud S., Mongenet C., « Parallel Seismic Ray-tracing in a Global Earth Mesh », *Proceedings of PDPTA'02*, p. 1151-1157, June, 2002.

Hofacker I. L., Fontana W., Bonhœffer L. S., Tacker M., Schuster P., « Vienna RNA Package », October, 2002. `http://www.tbi.univie.ac.at/~ivo/RNA`.

Kennett B. L., IASPEI 1991 Seismological Tables, Technical report, 1991.

Kielmann T., Bal H. E., Maassen J., van Nieuwpoort R., Eyraud L., Hofman R., Verstoep K., « Programming Environments for High-Performance Grid Computing : the Albatross Project », *Future Generation Computer Systems*, vol. 18, n° 8, p. 1113-1125, 2002.

Kielmann T., Hofman R. F. H., Bal H. E., Plaat A., Bhoedjang R. A. F., « MagPIe : MPI's collective communication operations for clustered wide area systems », *ACM SIGPLAN Notices*, vol. 34, n° 8, p. 131-140, 1999.

Ripeanu M., Iamnitchi A., Foster I., « Cactus Application : Performance Predictions in Grid Environments », *Europar 2001*, vol. 2150 of *Lecture Notes in computer Science*, Springer-Verlag, p. 807-816, 2001.

Squyres J. M., Lumsdaine A., George W. L., Hagedorn J. G., Devaney J. E., « The Interoperable Message Passing Interface (IMPI) Extensions to LAM/MPI », *Proceedings of MPIDC'2000*, March, 2000.

Sullivan W. T., Werthimer D., Bowyer S., Cobb J., Gedye D., Anderson D., « A new major SETI project based on Project Serendip Data and 100,000 Personal Computers », *Proceedings of 5th Intl. Conf. on Bioastronomy*, vol. IAU Colloq. No. 161, Editrice Compositori, 1997.

van Nieuwpoort R. V., Kielmann T., Bal H. E., « Satin : Efficient Parallel Divide-and-Conquer in Java », *Euro-Par 2000*, Munich, Germany, p. 690-699, August, 2000.

van Nieuwpoort R. V., Kielmann T., Bal H. E., « Efficient Load Balancing for Wide-area Divide-and-Conquer Applications », *PPoPP '01 : ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*, p. 34-43, June, 2001.

Article reçu le 19 janvier 2004
Version révisée le 10 décembre 2004

Stéphane Genaud *a obtenu un doctorat en informatique de l'Université Louis Pasteur en 1997. Depuis 1998, il est maître de conférences à l'Université Robert Schuman, et mène ses recherches au sein du laboratoire LSIIT, dans l'équipe ICPS. Depuis fin 2001, il est le coordinateur du projet TAG qui étudie les performances des applications parallèles sur les grilles de calculs.*

Marc Grunberg *est ingénieur au Réseau National de Surveillance Sismique (ReNaSS) dont le siège central est à Strasbourg. Il est responsable du développement logiciel ainsi que l'administration des systèmes. Doctorant en informatique à l'Université Louis Pasteur, ses thèmes de recherche concernent le développement d'algorithmes parallèles appliqués à la tomographie sismique.*