RENPAR'15 / CFSE'3 / SympAAA'2003
La Colle sur Loup, France, 15 au 17 octobre 2003

Calcul de rais en tomographie sismique : exploitation sur la grille

Marc Grunberg
Stéphane Genaud

LSIIT-ICPS, UMR 7005,CNRS-ULP,
Bd S. Brant, 67400 Illkirch
grunberg@renass.u-strasbg.fr, genaud@icps.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 dans cet article, l'application parallèle de tomographie que nous avons développée, ainsi que son exploitation sur une grille de calcul expérimentale utilisant le réseau Renater. L'application présente deux phases distinctes : une première 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 globales de cette application par rapport à l'évolution du réseau sous-jacent, et nous ramenons ces chiffres à ceux d'une exploitaiton sur une machine parallèle et sur un cluster. Notamment, le gain de performances 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.

Mots-clés : Application parallèle MPI, Grille de calcul, Tracé de rai sismique, Maillage de la Terre.

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 grille de calcul, par rapport à des architectures plus homogènes (cluster et machine parallèle). Nous présenterons tout d'abord la méthode de tracé de rais sismiques utilisée, et nous exposerons ensuite notre stratégie de parallélisation. Les résultats de nos évaluations sur grille de calcul, machine parallèle, ainsi que cluster, seront présentés dans la dernière partie.

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 : $p = r.\frac{\sin(i)}{v(r)}$, 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.
Pour tracer le rai sismique, nous utilisons un modèle de vitesse couramment utilisé en géophysique : le modèle IASPEI91 [9] 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. Typiquement un rai sismique est souvent discrétisé par plusieurs centaines, voire plusieurs milliers de points en fonction de sa longueur.

2.2. Le maillage

Le maillage initial 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.
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 dé-

composition 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.
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, en ignorant les liens entres cellules, et en allouant de la mémoire pour une cellule seulement lorsque nécessaire. L'économie mémoire est d'environ 130 Mo par processeur[1] pour un maillage global comme celui que nous utilisons.
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[2].
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 *all-to-all* 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**	*one to all comm.*	le processeur maître distribue la description du maillage aux autres processeurs.
2. **distribution des blocs de rais**	*one to one comm.*	chaque processeur demande au processeur maître un bloc de rais à traiter.
3. **le tracé de rai**	*calcul*	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.
4. **échange de données**	*all to all comm.*	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.

[1] L'encombrement mémoire restant dans ce cas, est environ de 190 Mo par processus sur 16 processeurs, et de 320 Mo sur 8 processeurs.
[2] Nous décrivons ici 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 [5].

5. **fusion des informations**	*calcul*	les informations concernant une cellule sont fusionnées.
6. **calcul de score**	*calcul*	pour chaque cellule, on calcule un score à partir des informations des rais.
7. **concaténation**	*all to one comm.*	le score de chaque cellule est envoyé au processeur maître.

4. Exploitation sur une grille

4.1. Préambule

Dans cette partie nous allons présenter les performances de notre application obtenues sur une grille de calcul, construite dans le cadre du projet TAG[3]. 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, Clermond-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 [7] (versions 1.1.4 et 2.0) et nous utilisons MPICH-G2 [6] comme bibliothèque de communication.

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 [8],
– 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.

4.2. Les plate-formes de test

Les expériences réalisées sur les deux configurations de grille sont comparées aux mêmes exécutions sur des plate-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.

– **Machine parallèle** : la machine utilisée, leda dans la table 1, est une SGI Origin 3800 située au CINES à Monptellier. 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.

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

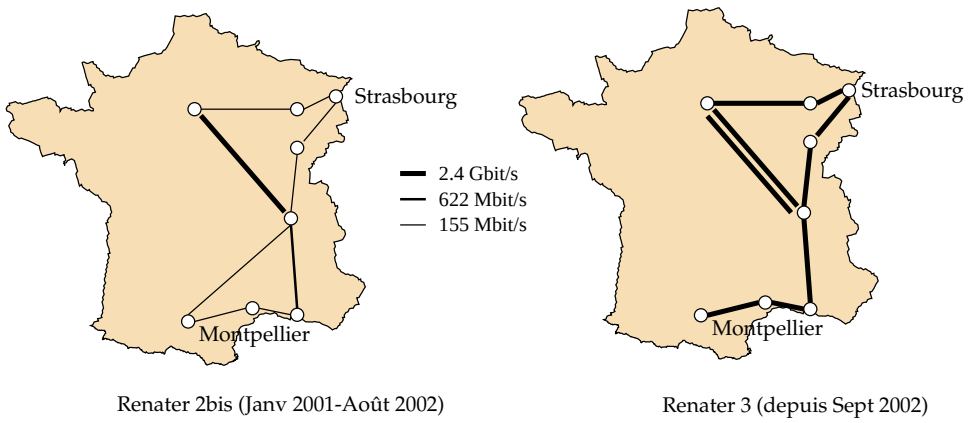


FIG. 1 – Partie de la dorsale nationale Renater empruntée et évolution des débits entre Renater 2bis et 3.

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

TAB. 1 – Ressources de calcul utilisées dans les expériences.

- **Cluster** : le cluster de PC est composé de 6 noeuds interconnectés par un switch Gigabit Ethernet. Chaque noeud 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 [10].
- **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.
- **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.

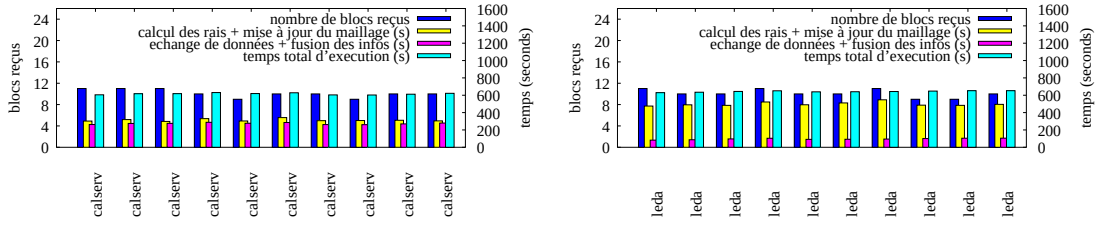
4.3. Expériences

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 2 présente, pour 10 processeurs, les temps mesurés sur le cluster (2a), la machine parallèle (2b). Ce sont là des mesures de référence qui permettent d'évaluer l'exécution sur la configuration de type grille métopolitaine présentée sur la figure 3. On retrouve dans les résultats du cluster et de la



(a) Cluster de PC (b) Origin 3800

FIG. 2 – Performances sur un cluster de PC et une SGI Origin 3800 (10 processeurs).

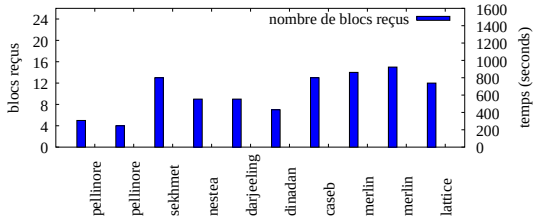
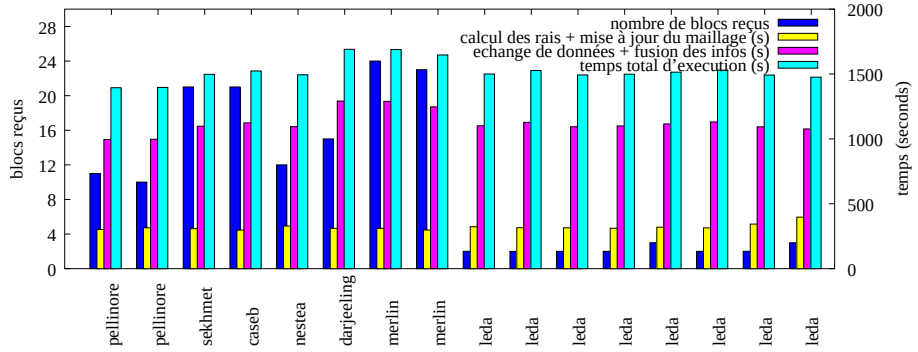


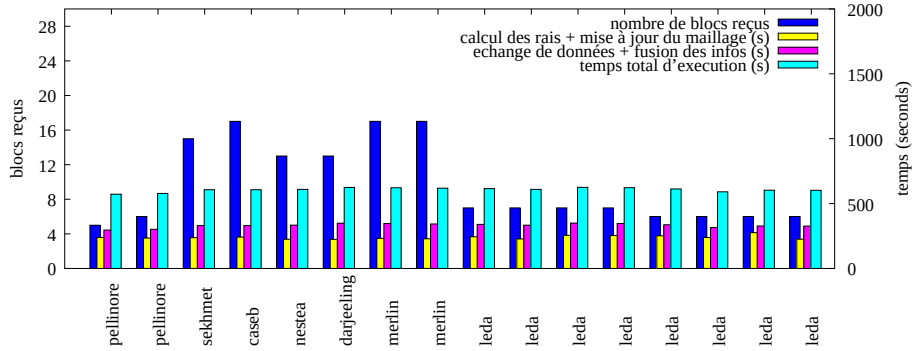
FIG. 3 – Performances de la configuration type grille métropolitaine (10 processeurs).

machine parallèle, les spécificités de ces architectures. Les processeurs étant homogènes, le nombre de blocs de rais distribués est équitablement reparti sur chaque processeur. Les puissances de calcul aggrégées (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.

L'exécution sur la grille métropolitaine ne met pas en défaut l'équilibrage de charge de l'application malgré l'hétérogénéité des processeurs et du réseau. La répartition de la charge est hétérogène (15 blocs pour le processeur le plus rapide, 4 pour le plus lent) mais les temps de fin sont quasi-égaux, après 900 s. Le temps de calcul des rais, sur la grille, est légèrement meilleur que celui du cluster, ce qui s'explique par la puissance aggrégée 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.



(a) Grille nationale Renater 2



(b) Grille nationale Renater 3

FIG. 4 – Performances avec 16 processeurs sur une grille avec Renater 2 et Renater 3

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 2002, et nous l'avons recommencée en mai 2003. Entre-temps, l'infrastructure reliant les deux sites a progressé, comme le montre la figure 1. 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 4a pour Renater 2, et figure 4b 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 aggrégée de 20,94) est approximativement le même qu'avec une configuration de 16 processeurs sur la machine parallèle du CINES présenté en figure 2b (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 ce dernier 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 durée de l'échange des données (étapes 5 et 6) qui représentait alors 75% du temps total de l'application.

5. 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 super-calculateur 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 ([2], [4] 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. Un cas plus réaliste consisterait en un plus grand nombre de processeurs en chaque site.
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 ([3]). Les recherches doivent aussi inventer et tester des techniques de programmation (telles celles proposées dans [1] qui visent à augmenter la granularité) sur des grilles à grande échelle. 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.

Bibliographie

1. Gabrielle Allen, Thomas Dramlitsch, Iand Foster, Matei Ripeanu Nicholas T. Karonis, Edward Seidel, and Brian Toonen. Supporting efficient execution in heterogeneous distributed computing environment with cactus and globus. In *Proceedings of SuperComputing 2001*. IEEE, November 2001.
2. D. Bansal, H. Balakrisham, and S. Floyd. Dynamic behavior of slowly-responsive congestion control algorithms. In *Proceedings of SIGCOMM 2001*, volume 31. ACM Computer Communication Review, August 2001.
3. Olivier Beaumont, Larry Carter, Jeanne Ferrante, Arnaud Legrand, and Yves Robert. Bandwidth-centric allocation of independent tasks on heterogeneous platforms. In *International Parallel and Distributed Processing Symposium IPDPS'2002*. IEEE Computer Society Press, 2002.
4. Wu chun Feng, Mike Fisk, Mark Gardner, and Eric Weigle. Dynamic right-sizing : An automated lightweight, and scalable technique for enhancing grid performance. In *Proceedings of the 7th International Workshop on Protocols for High-Speed Networks*, April 2002.
5. Romaric David, Stéphane Genaud, Arnaud Giersch, Benjamin Schwarz, and Éric Violard. Source code transformations strategies to load-balance Grid applications. In Manish Parashar, editor, *Proceedings of Grid Computing (GRID 2002)*, volume 2536 of *Lecture Notes in Computer Science*, pages 82–87, Baltimore, USA, November 2002. Springer-Verlag.
6. Ian Foster and Nicholas T. Karonis. A grid-enabled MPI : Message passing in heterogeneous distributed computing systems. In *Proc. of the 1998 ACM/IEEE conference on Supercomputing*, Nov. 1998.
7. Ian Foster and Carl Kesselman. Globus : A metacomputing infrastructure toolkit. *The International Journal of Supercomputer Applications and High Performance Computing*, 11(2) :115–128, 1997.
8. Marc Grunberg, Stéphane Genaud, and Catherine Mongenet. Parallel seismic ray-tracing in a global earth mesh. In *Proceedings of PDPTA'02*, pages 1151–1157, June 2002.
9. Brian L.N. Kennett. Iaspei 1991 seismological tables. *Research School of Earth Sciences Australian National University*, 1991.
10. Jeffrey M. Squyres, Andrew Lumsdaine, William L. George, John G. Hagedorn, and Judith E. Devaney. The interoperable message passing interface (IMPI) extensions to LAM/MPI. In *Proceedings, MPIDC'2000*, March 2000.