# Parallel Seismic Ray Tracing in a Global Earth Model

Marc Grunberg[*], Stéphane Genaud[†] and Catherine Mongenet[‡]
[*]marc.grunberg@eost.u-strasbg.fr, tel (+33) 390240033 / fax : (+33)390240101
EOST, 5 rue R. Descartes, F-67084 Strasbourg
[†]genaud@icps.u-strasbg.fr, tel (+33)390244542 / fax : (+33) 390244547
[‡]mongenet@icps.u-strasbg.fr, tel (+33)390244536 / fax : (+33) 390244547
LSIIT-ICPS, Bd S. Brant, F-67400 Illkirch

*Abstract*— **A major research topic in geophysics deals with the modelization of the Earth interior, and seismic tomography is a mean to improve knowledge in this field. In order to improve the accuracy of existing methods however, huge quantities of information must be computed. We present in this paper the design of a software program implementing a fast seismic ray-tracing in a Earth mesh. We show that massively parallel computational resource may be used to process huge quantity of data.**

*Keywords*— **Parallel MPI application, geophysics, tomography, ray-tracing, mesh.**

## I. Introduction

One of the main domains in geophysics concerns seismic tomography. Its objective is to build a global seismic velocity model. In such a model any point in the Earth interior (defined by its latitude, longitude and depth) is characterized by its velocities ($V_p$, $V_s$) which give information about the physical rock properties at that point.

In order to build such a model we use seismic events information recorded in databases by international organizations such as ISC (*International Seismic Center*). Seismic events are recorded by networks of stations (or seismic captors) located all around the world. After such event, the seismogram data are analyzed in order to localize the earthquake hypocenter. Each database record corresponds to one event and lists all the waves arrival times at the different stations. These databases contain a huge amount of information: for instance, the ISC catalog contains several millions of such arrival times.

A seismic wave is modelized by a set of *rays*. Each ray represents the wavefront propagation from the hypocenter (source) to one station.

The final objective of the seismic tomography process is to build an accurate velocity model from an initial one, given ray travel times computations. The first step is therefore to trace these seismic ray paths using information extracted from the databases and then to compute their travel times. These computed times are then compared with the measured ones (from the databases). In the second step, the initial velocities model is refined in order to get a better fit between the computed and the measured travel times.

The ray tracing algorithm is an iterative process that builds the ray path step by step using linear segment increments. Thus the ray path is a set of discretized points with appropriate information (incidence angle, phase). In a global Earth model a ray path is usually discretized using several hundred to several thousands points depending on the ray length and its nature. Since the ISC database contains millions of such rays and because of these discretizations, we have to compute billions of information items. Therefore parallel computation is highly required.

The final objective of our project is to build an adaptive mesh of the Earth given a set of seismic rays. The size of a cell in the mesh will be adapted depending on the "illumination" quality, that is a region with few crossing rays will be modelized with large cells whereas a region sampled by many rays in various directions will yield small cells.

In this paper we only focus on ray-tracing and propose an efficient solution to implement this problem on parallel systems. Section II explains the requirements of the ray tracing process, while section III shows how parallelization of the computation can be implemented. Experimental results as well as comparisons to related work are given in section IV, concluding remarks and future work are discussed in the last section.

## II. The seismic ray tracing process

The ray path modelization represents seismic wave front propagation from one source (seismic hypocenter) to one receiver (station). Each time a ray reaches an interface (lower mantle-outer core, outer core-inner core, ...) it can be either transmitted or reflected. Moreover its phase can change from $P$-phase to $S$-phase (converted phases) and vice-versa. A $P$ wave is a compression wave while a $S$ one is a shear wave.

The seismograms recorded at stations are analyzed to pick the different arrival times of the waves and to identify the corresponding ray by a precise label (or signature) such as $P$, $S$, $PcP$, $PKP$, $SKS$ , *etc.* (see table I).

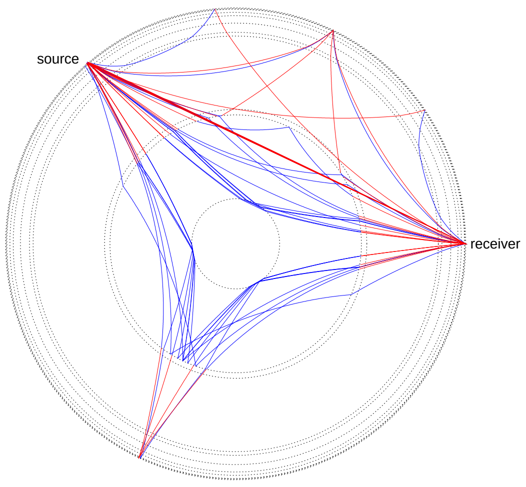

Fig. 1. Visualization of 27 seismic ray paths. Each ray path is computed given an initial angular incidence $i_0$ and ray signature from table I (diffracted rays $P$dif and $S$diff were not computed). Blue paths symbolize compression waves, while the red ones are shear waves.

The Earth velocity model retained as the initial one is based on eleven layers given by the 1D IASPEI91 model [1] which depends only on the depth. The ray-tracing algorithm is based on the Snell-Descartes law in a spherical geometry and describes the variation of seismic wave velocities:

$$p = r.\frac{\sin(i)}{v(r)} \tag{1}$$

where $p$ is the ray parameter which is constant on any point of the ray, $v(r)$ is the wave propagation velocity at depth $r$ and $i$ is the incidence angle of the ray at depth $r$ (figure 2).

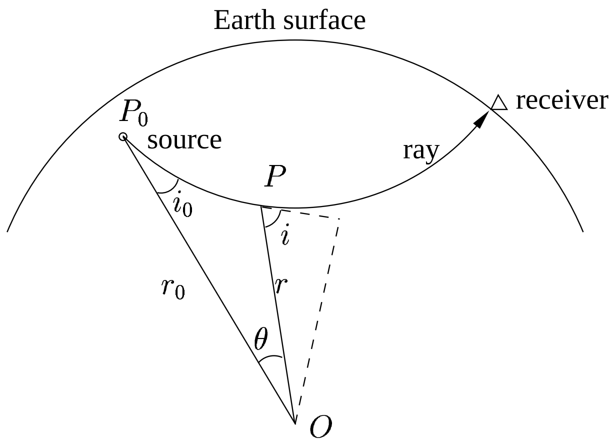

Fig. 2. Each ray point $P$ is defined by its depth $r$, angular incidence $i$, and $\theta$ ($\widehat{P_0OP}$). First point $P_0$ (source) is defined by ($r_0$, $i_0$, $\theta_0 = 0$). $O$ is the Earth center.

The ray path computation relies on the initial incidence angle ($i_0$, given by IASP library), and on the ray signature which is used to guide the ray-tracing process. It does not depend on the azimuth. The ray is therefore two-dimensional in a plane defined by the source, the receiver and the Earth center. The ray path computation is done in this plane and then three successive rotations are applied to set the rays in the 3D space.

The 2D ray path computation is done using an iterative method which propagates the ray segment by segment using either elementary equal-length segments or equal-angle segments depending on the incidence angle. When the incidence angle is close to 90° the algorithm based on equation (1) switches from equal-length segments to equal-angle ones. During this process when a segment reaches an interface, the in-

Table I

29 seismic signatures given by IASP library for an angular distance (source,receiver) of 130°, and 0 km source depth.

| | | |
|---|---|---|
| $P$diff | $PKPdf$ | $PKiKP$ |
| $PP$ | $SKPbc$ | $PKSbc$ |
| $SKPab$ | $PKSab$ | $PKSdf$ |
| $SKPdf$ | $SKiKP$ | $SKSac$ |
| $SKSdf$ | $SKKSac$ | $PKKPdf$ |
| $S$diff | $PS$ | $SP$ |
| $PKKSbc$ | $SKKPbc$ | $PKKSdf$ |
| $SKKPdf$ | $SKKSac$ | $SKKSdf$ |
| $P'P'df$ | $SS$ | $S'S'ac$ |
| $S'S'ac$ | $S'S'df$ | |

formation extracted from the ray signature is used to guide the ray: it can be transmitted or reflected, or changes the nature of the seismic phase (see figure 1).

Since our final goal is to build an adaptive mesh, the ray-tracing process not only computes the discretized rays paths and travel times, but also stores all the required information associated with a ray in the cells of an initial regular mesh. The cell geometry of this mesh results from a decomposition of the sphere into regular angular sectors from the center to the surface. Each sector is decomposed into a given number of layers (in table IV, there are 8 layers). Each elementary volume thus obtained defines a cell which can be approximated by an hexahedron. This information will be used in a second step to refine the mesh by merging cells whose "illumination" is not good enough. The mesh adaptation is realized by cells merging and hence the mesh data structure must manage neighborhood information through a set of links in all six directions. The ray information is mainly composed by the length of the ray in the cell, the coordinates of its first and last point in the cell and its incidence angle. Moreover other informations are computed on each cell to guide the mesh adaptation such as the ray density (*i.e.* the number of rays crossing the cell with respect to the cell size), the average ray length, the number of ray hits by cell face, and the cell fill rate, which measures the quality of the spatial repartition of rays in the cell.

## III. Parallelization

### A. Parallelization strategy

The difficulty of parallelizing the application lies in keeping information related to both rays and mesh in each processor local memory.

One possibility is to make each processor own a distinct part of the mesh based on a geographic decomposition (for example we could attribute the two hemispheres to two processors). Each ray would then be traced by the processor owning the geographic area the ray is traveling through.

The pitfall of this approach is two-fold: first, the ray tracing computations would be largely unbalanced as the captors are (geographically) dispatched in a disparate manner at the globe surface (oceans have very few captors for example) and hence some processors would see very few rays passing in their zone. Secondly, inter-processors communications would have to take place every time the ray tracing pass from one zone to another. Furthermore, increasing the processors number would also increase the communications number and lead to catastrophic performances.

Our strategy of parallelization is based on the replication of the mesh information on each processor. To overcome memory constraints due to such a replication, we only compute some minimal mesh information on each processor, ignoring the links between cells and allocating memory for cells only when needed. In the examples treated in section IV we thus spare 13.5 Mbytes of such administrative information. For a global Earth mesh it would be larger than 200 Mbytes.

Each processor receives from the master process a description of the mesh (such as the description presented in table IV) covering the area under investigation and a set of rays to trace. Each process then starts to compute the rays of its set in turn. At the beginning of each ray segment, the process tests if the ray path has entered a new mesh cell and tests to which cell the segment endpoint belongs to. If the ray is the first to enter the cell, memory is allocated for this cell in order to store information related to the ray traversal, otherwise the cell existing data is updated with information brought by the ray.

### B. Implementation

It is important to us that the design and the implementation of the code be modular and portable. It is modular as each functionnality is written as a library: the mesh feeding uses the library that implements the ray-tracing algorithm based on the Snell-Descartes law, which in turns uses a library implementing the IASP91 model and another one that provides phases parsing (to guide the ray-tracing process). Thus, other ray-tracing algorithms could be easily plugged in the application in the future.

Concerning portability, the code is written in C and contains Fortran routines borrowed from Kennett [1]. The library used to parallelize the code is MPI [2] (note however that the build of a sequential or parallel version can be chosen at compile time). All these technical choices come from portability requirements since the application must build and run on any Unix System: we discuss in section V our future work plans whose objective is to make the application run on a *computational grid* [3], where portability requirements are strong. Currently, the application runs on Linux (various distributions), IRIX, SunOS and HPUX systems. Our application also generates VTK [4] files to visualize the results.

## IV. Experimental results

We carried out two experiments to validate the scalability of our application. In the first experiment, concerning the study of a regional area (the Euro-Mediterranean area), we have used the data set from Granet and Trampert [5] of about 17000 rays with a $P$ signature, which reduces to 10751 rays after removing those which get out the geographic boundaries of the studied area described in table IV. We also restricted our benchmark to $P$-type rays to be able to compare our results to other experiments described in the literature.

In the second experiment, we traced all the seismic events of year 1999, as recorded in the world-wide ISC databases. The data sets is composed of about 827000 rays, of which 325749 can be traced.

For the two experiments, we completed the *ray-tracing process* with a phase feeding an appropriate mesh to store the various information elements described in the end of section II, called the *mesh update process*. We first describe the behavior of the two experiments concerning the ray-tracing process only (section IV-A), and we compare afterwards the behavior of the mesh information computation in both cases (section IV-B).

## A. Ray-tracing speed

In the following, we try to assess and compare to related experiments two metrics:
– the number of rays computed by our ray-tracing algorithm in a given time, called the *ray-tracing speed*,
– and the number of rays treated after having extracted and put their information in the light mesh structure.

Table II and III presents the ray-tracing speed obtained on an SGI Origin 3800 parallel computer with 500 MHz Mips R14K processors. We have chosen a 2 km long discretization step all along the ray-tracing to get a very good precision. Column $np$ indicates the number of processors involved. All times are in seconds, $min$ and $max$ refer to the shortest and longer (resp.) computation times observed on the different processors. These times measure both the data transmission and ray computation (I/O operations are excluded from the measurements). The absolute speed $speed_a$ is the number of rays computed divided by $max$, while the relative speed is $speed_r = speed_a/np$. Column $e$ indicates the efficiency of the parallel runs depending on $np$. All times are average times based on measures from two to four runs.

### A.1 Regional area

In the first experiment, the table shows that we achieved a ray-tracing speed of 3821 rays.$s^{-1}$ on 64 processors. Using only one processor, the ray-tracing speed is about 168 rays.$s^{-1}$. We can compare this result to the ray-tracing algorithm of Bijwaard and Spakman [6] whose speed is claimed to be 15 rays.$s^{-1}$ on a Mips R8K. Other recent results with a ray-tracing method based on minimization of travel times come from Cores *et al.* [7] who obtain a maximum speed of about 12.5 rays.$s^{-1}$ on a Sparc-Ultra 1. However, they only tested their algorithm on small data sets. Sambridge and Gudmunsson [8] obtained a ray-trace speed at about 12.5 rays.$s^{-1}$ on a Sun Sparcstation 10/40.

Table II
Ray-tracing speed of 10751 $P$-type rays, with a 2 kms ray discretization in a regional mesh.

| $np$ | $min$ | $max$ | $speed_a$ | $speed_r$ | $e$ |
|---|---|---|---|---|---|
| 1 | 65.36 | 65.36 | 164.49 | 164.49 | 100% |
| 2 | 33.16 | 33.18 | 324.07 | 162.03 | 98% |
| 8 | 8.25 | 8.67 | 1240.50 | 155.06 | 94% |
| 16 | 4.27 | 4.79 | 2245.64 | 140.35 | 85% |
| 32 | 2.36 | 3.11 | 3462.48 | 108.20 | 66% |
| 64 | 1.75 | 2.81 | 3821.45 | 59.71 | 36% |

Obviously, these speeds would be much faster on a MIPS R14K processor. However notice that it is difficult to compare these results more precisely because the algorithms are differents and mainly because we select a 2 kms step whereas Bijwaard and Spakman [6] used a step varying from 10 kms to 5 kms depending on depth.

In terms of speed-up, we can see from results in table II that the algorithm is scalable up to 16 processors (Bijwaard and Spakman [6] used a maximum of 3 processors for the computation, hence we cannot compare this aspect). Our parallel algorithm shows an efficiency of 85% and 65% with 16 and 32 processors respectively, and decreases dramatically to 36% with 64 processors.

We must also note that the processor load imbalance (let us define it by $(max - min)/max$) is correlated with the efficiency. It stays acceptable up to 16 processors (about 10%) but grows up to 37% with 64 processors.

This behavior can be explained by the relatively small size of the data set: with 64 processors, each processor only has 168 rays to compute. In this case, the overhead due to data transmission and the imbalance in ray length distribution to processors becomes predominant. The experiment at the Earth scale with many more rays, described in the following paragraph, confirms this explanation.

### A.2 Global area

The second experiment deals with the computation of rays issued from seismic events which occured during the year 1999. Unlike the Euro-Mediterranean experiment, we have a very large set of ray signatures as we treat rays with $P$, $S$, $Pn$, $Pg$, $pP$, $PKP$, $PcP$, $pPKP$, $ScP$, *etc*, signatures. Table III gives the ray-tracing speed for more than $3.10^5$ rays (about 30 times more than in the previous experiment).

Table III
Ray-tracing speed of 325749 rays (large set of ray signature), with a 2 kms ray discretization in a global mesh.

| $np$ | $min$ | $max$ | $speed_a$ | $speed_r$ | $e$ |
|---|---|---|---|---|---|
| 2 | 960.70 | 1158.58 | 281.14 | 140.56 | 97% |
| 8 | 244.99 | 308.43 | 1056.15 | 132.01 | 81% |
| 16 | 132.47 | 171.02 | 1904.74 | 119.04 | 73% |
| 32 | 86.32 | 108.63 | 2998.70 | 93.70 | 57% |
| 64 | 49.90 | 69.35 | 4697.17 | 73.39 | 45% |

The overall ray-tracing speed is a bit slower in this experiment, excepted for 64 processors. It is not suprising as the dataset contains rays with higher lengths, like $PKP$ rays, which take more time to compute. However, we can notice that the efficiency decreases less quickly: from 57% to 45% with 32 and 64 processors, correlated with a load imbalance that stays more stable in this experiment, between 20% and 28% on 8 and 64 processors respectively. This can be explained by the fact that each processor receives a bigger set of rays (more than 5000 rays per processor for 64 processors) and hence, the data transmission overhead is not significant as compared to the computation time of rays.

## B. Ray-tracing in a mesh

### B.1 Regional area

For the regional area study, we now focus on the ray-tracing performed in a 8-layers mesh described by the configuration file of table IV, containing 196992 cells, all cells being 0.5°× 0.5°wide in longitude and latitude. The studied area is the same as the one in the study of Granet and Trampert [5] excepted that our cell size is six times thinner in both longitude and latitude. The objective here is not to enhance the resolution (which would make little sense from a geophysical point of view due to the source/receiver configuration) but rather to test our application with a big enough number of cells.

Table IV
The mesh description of the regional Euro-Mediterranean area

```
<?xml version="1.0"?>
<mesh
 lat-unit-size="0.5" lon-unit-size="0.5"
 lat-min="30"  lat-max="87"
 lon-min="-42" lon-max="66">
 <layer name="11" zstart="0"    zend="40"   />
 <layer name="12" zstart="40"   zend="100"  />
 <layer name="13" zstart="100"  zend="250"  />
 <layer name="14" zstart="250"  zend="400"  />
 <layer name="15" zstart="400"  zend="650"  />
 <layer name="16" zstart="650"  zend="900"  />
 <layer name="17" zstart="900"  zend="1200" />
 <layer name="18" zstart="1200" zend="1500" />
</mesh>
```

Each processor manages its own ray data set. In the first step it computes the ray paths using the iterative discretization process described above. In the second step it scans the discretized points and updates cell information in the mesh as explained in section II. Figure 3 and 4 show the visual results of the whole process. Table V presents performance results on the same data set and in the same format as table II. The extra-columns *ray* and *mesh* express the relative durations of the ray-tracing process and mesh update with rays computed data respectively. The results show that the mesh update takes roughly as long as the ray-tracing process. This ratio decreases as the number of processors involved increases since each processor has less rays to trace and hence, less information has to be added to cells.

### B.2 Global area (with an Earth mesh)

The biggest run concern the ray-tracing of the data set for the whole year 1999 in a global Earth

Table V

Ray-tracing speed with mesh cells information computation (10751 $P$ rays)

| $np$ | $min$ | $max$ | $speed_a$ | $speed_r$ | $ray$ | $mesh$ |
|---|---|---|---|---|---|---|
| 2 | 77.19 | 77.32 | 139.05 | 69.52 | 43% | 57% |
| 8 | 18.66 | 19.62 | 547.96 | 68.50 | 44% | 56% |
| 16 | 9.40 | 10.22 | 1052.47 | 65.78 | 47% | 53% |
| 32 | 4.5 | 5.69 | 1889.46 | 59.05 | 51% | 49% |
| 64 | 2.34 | 4.12 | 2609.47 | 40.77 | 68% | 32% |

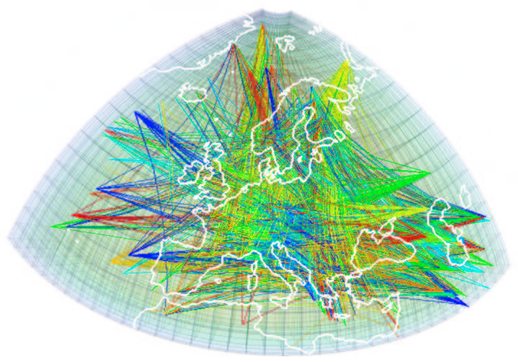

Fig. 3. VTK image of 2500 rays in a mesh of the Euro-Mediterranean area.

mesh. The mesh used here (described in table VI) is made of 11 layers, and all its cells are $1° \times 1°$ wide so that there are a total of 712800 cells in the mesh. Note that, though we use cells of identical size, the cells could be of arbitrary sizes in latitude and longitude in a given layer.

Table VI

The mesh description of the global area

```
<?xml version="1.0"?>
<mesh
lat-unit-size="1" lon-unit-size="1"
lat-min="-90" lat-max="90"
lon-min="0"   lon-max="360">
 <layer name="l0"  zstart="0"    zend="20"   />
 <layer name="l1"  zstart="20"   zend="35"   />
 <layer name="l2"  zstart="35"   zend="120"  />
 <layer name="l3"  zstart="120"  zend="210"  />
 <layer name="l4"  zstart="210"  zend="410"  />
 <layer name="l5"  zstart="410"  zend="660"  />
 <layer name="l6"  zstart="660"  zend="760"  />
 <layer name="l7"  zstart="760"  zend="2740" />
 <layer name="l8"  zstart="2740" zend="2889" />
 <layer name="l9"  zstart="2889" zend="5153" />
 <layer name="l10" zstart="5153" zend="6371" />
</mesh>
```

The performance results are presented in table VII. The performance is not affected by the big amount of memory needed. The number of rays computed per second appears to be still acceptable and, though the mesh has many more

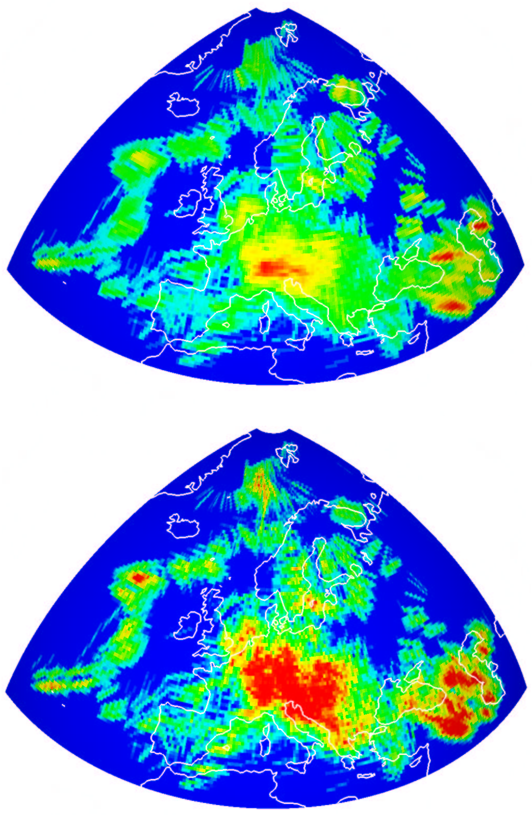

Fig. 4. Slice of the of the Euro-Mediterranean mesh (layer 5) after cells information update: the upper figure represents the ray density and lower one shows the cell fill rate. The color scale goes from blue for poorly informative cells, to red for highly exposed cells.

cells, the ratio of the mesh update duration as compared to the ray-tracing process stays comparable to what was observed with the regional area mesh.

Table VII

Ray-tracing speed with mesh cells information computation (325749 rays, various signature)

| $np$ | $min$ | $max$ | $speed_a$ | $speed_r$ | $ray$ | $mesh$ |
|---|---|---|---|---|---|---|
| 8 | 434.93 | 633.68 | 514.06 | 64.25 | 48% | 52% |
| 16 | 213.15 | 332.66 | 977.96 | 61.12 | 51% | 49% |
| 32 | 115.32 | 186.04 | 1748.14 | 54.62 | 58% | 42% |
| 64 | 68.39 | 114.88 | 2830.26 | 44.22 | 60% | 40% |

The overall results exhibited in this last experiment are quite satisfactory: the information that can be extracted from the seismic events of a whole year have been structured into a geographical mesh in less than 2 minutes. This can not be done on standard computing equipement mainly because of RAM requirements: our attempts to run this experiment on a PC equipped with 2 GBytes rapidly failed because of memory exhaustion.

## V. Conclusion

We have described in this paper the design and the performances of the core part of a seismic tomography tool under developement. This work is the preliminary step of an ongoing project whose aim is to build an adaptive mesh to modelize the Earth interior. We put forward the parallel design of the program, since the huge quantity of data to be computed will require massively parallel computations. At this step of the project, we have evaluated and compared the performances obtained to related work.

The results and the good scalability show that the extension to a wider set of data, that is millions of rays traced in a full Earth mesh are reachable. The use of massively parallel computational resources will be in this case unavoidable. This is the reason the project is part of a larger initiative called TAG[1] whose objectives are to port scientific applications on the grid [3] and to develop tools for efficient execution of programs in this framework.

## Acknowledgements

All experiments have been carried out on the SGI Origin 3800 with 512 processors of the CINES (`http://www.cines.fr`). We want to thank them for letting us access to their computational resources. We are grateful to Michel Granet who provided valuable comments, and to the RENASS, the French National Seismic Network (`http://renass.u-strasbg.fr`) for providing ISC data.

## References

[1] Brian L.N. Kennett, “Iaspei 1991 seismological tables,” *Research School of Earth Sciences Australian National University*, 1991.

[2] Message Passing Interface Forum, *MPI : A message-passing Interface Standard*, June 1995.

[3] Ian Foster and Carl Kesselman, *The Grid, Blueprint for a New Computing Infrastructure*, Morgan Kaufmann Publishers, Inc., 1998.

[1]TAG stands for Transformations and Adaptations for the Grid, `http://grid.u-strasbg.fr`

[4] Will Schroeder, Ken Martin, and Bill Lorensen, *The Visualization Toolkit, An Object-Oriented Approach To 3D Graphics*, Prentice Hall, December 1997.

[5] Michel Granet and Jeannot Trampert, “Large-scale p-velocity structures in the euro-mediterranean area,” *Geophys. J. Int.*, vol. 99, pp. 583–594, 1989.

[6] Harmen Bijwaard and Wim Spakman, “Fast kinematics ray tracing of first and later arriving global seismic phases,” *Geophys. J. Int.*, vol. 139, pp. 359–369, 1999.

[7] Debora Cores, Glenn M. Fung, and Reinaldo J. Michelena, “A fast and global two point low storage optimization technique for tracing rays in 2D and 3D isotropic media,” *Journal of Applied Geophysics*, vol. 56, no. 45, pp. 273–287, September 2000.

[8] Malcom Sambridge and Olafur Gudmundsson, “Tomography systems of equations with irregular cells.,” *J. of Geophys. Res.*, vol. 103, no. No. B1, pp. 773–781, 1998.