Parallel Adaptive Mesh Coarsening for Seismic Tomography

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

Stéphane Genaud
Catherine Mongenet
LSIIT-ICPS, UMR 7005,CNRS-ULP,
Bd S. Brant, F-67400 Illkirch

Abstract

Seismic tomography enables to model the internal structure of the Earth. In order to improve the precision of existing models, a huge amount of acquired seismic data must be analyzed. The analysis of such massive data require a considerable computing power which can only be delivered by parallel computational equipments. Yet, parallel computation is not sufficient for the task: we also need algorithms to automatically concentrate the computations on the most relevant data parts.

The objective of the paper is to present such an algorithm. From an initial regular mesh in which cells carry data with varying relevance, we present a method to aggregate elementary cells so as to homogenize the relevance of data. The result is an irregular mesh which has the advantage over the initial mesh of having orders of magnitude less cells while preserving the geophysical meaning of data. We present both a sequential and a parallel algorithm to solve this problem under the hypotheses and constraints inherited from the geophysical context.

1. Introduction

Seismic tomography is an important field in geophysics. Its objective is to build a global seismic wave velocity model of the Earth, that is to determine the physical characteristics and heterogeneities of the various parts of the Earth interior. The data used to build such a model are obtained from the seismic events recorded by the many seismic stations (or captors) covering the planet.

When a seismic event occurs, the network of stations record the corresponding seismograms. These seismograms are analyzed to localize the source (e.g. the earthquake hypocenter) and determine the wave type and the wave travel time. The data are then stored in databases by international institutions such as the ISC (International Seismic Center). Obviously, the number of such records is huge. For instance, our case study process the seismic data set acquired by the ISC for year 1999 which represents 817000 records. From each record, the wavefront propagation from the source to the captor, called the *seismic ray*, can be computed. Computing the ray paths from the records is called the *ray-tracing* process.

Seismic tomography is a three-phases process :

- The space of study (either the whole globe or just a region) is meshed with elementary cells.
- Then the set of rays is traced using an initial velocity model. During the ray-tracing phase, each ray segment crossing a cell brings some information stored into that cell. Because of the uneven repartition of both the seismic stations and the earthquake centers, some regions are crossed by very few rays. We call *illumination* the property reflecting the wealth of information for a cell, which depends on the number and variety of the rays (in terms of incidence angle) crossing it.
- Eventually, the data collected enable to setup an enormous overdetermined system of equations called the *inverse problem*. In this third phase, the objective is to find the velocities that best fit the computed ray paths, yielding a new velocity model. The parametrization of the inverse problem is crucial for its resolution. With a regular mesh, the parametrization generates as many unknowns for regions with badly illuminated cells than for zones with a good ray sampling, leading to an ill-conditioned system. Hence, the construction of an adaptive mesh in which only cells with enough information remain small, while poorly informative cells are gathered to increase the local illumination, is necessary to improve the model accuracy. Furthermore, the reduction in the number of cells (two orders of magnitude in this paper) allows to use a fine resolution for well-sampled zones in the inverse problem without exceeding the current computing capabilities.

Various methods have already been studied to tackle this problem. They can be roughly classified in two categories.

In the first category, the mesh is build according to some static information: for instance the method in [10] proposes a grid optimization which uses a very dense model gridding in localized areas where sources are densest. Another method enables some optimizations by interactively changing the grid boundaries [6]. In the second category are methods which automatically adapt the mesh depending on the cells illumination (as we do in this work). Different approaches have been studied to construct such irregular meshes: Delaunay or Voronoi decompositions have been used to build tetrahedral meshes in [8] and [7], while non uniform sized rectangular blocks are used in [9]. However, these studies mainly comment on the geophysical results obtained through the method used, and give few details on the adaptive algorithms used though it obviously has an impact on the parameterization problem.

The objective of our work is thus to design a method to build such an adaptive mesh. Because of the huge quantities of data to be processed, the 3D mesh construction algorithm has been parallelized. The whole approach is decomposed into 3 successive steps :

- the construction of an initial regular 3D mesh,
- the seismic ray tracing process in the regular mesh,
- the construction of the adaptive mesh obtained by merging adjacent cells of the original mesh to properly fit the illumination criteria.

The paper is organized as follows. Section 2 shows how to compute the ray tracing and to build the initial mesh where each cell contains the information on its crossing rays. Section 3 explains how the adaptive mesh is constructed, whereas section 4 focuses on the parallelization of the algorithm. Section 5 presents experimental results. Concluding remarks and future works are given in section 6.

2. Initial mesh construction

The first step of the method, which consists in tracing the seismic rays in a regular Earth mesh, has been fully presented in [3]. The main ideas underlining this process are briefly presented in this section.

The ray path modelization represents the wavefront propagation from one seismic hypocenter to one station. Each time a ray reaches a geological interface (for instance the lower mantle / outer core interface) it can be either transmitted or reflected, depending on the incidence angle and the geological characteristics of the interface. It can also change its propagation mode, from compressional mode to shear mode or vice-versa. These informations define the *signature* of the ray. In order to trace the ray, we use an initial velocity model, the IASPEI91 model [4], based on a decomposition of the Earth interior into 11 layers in which the ray tracing computation depends only on the depth. The tracing algorithm uses the Snell-Descartes law in spherical geometry to compute the discretization. The ray is computed in 2D and then positioned in the 3D space with appropriate rotations.

The 2D ray path computation is done using an iterative process that builds the ray path segment by segment using either elementary equal-length segments or equal-angle segments depending on the incidence angle. The signature of the ray is used to decide how to propagate the ray each time an interface is reached.

The ray information is computed and stored into the initial regular 3D mesh. This mesh is obtained from the decomposition of the sphere into a given number of layers from the surface to the center. Each layer is then decomposed into regular angular sectors (both in latitude and longitude) issued at the center of the Earth. Each elementary volume thus obtained defines a cell that can be approximated by an hexahedron.

Each cell carries the information brought by all ray fragments that cross it:

- the number of rays,
- for each ray : the length of the ray fragment in the cell, the input and output impact points in the cell, the input and output incidence angles,
- encoded information to measure the way the rays are distributed in the cell. A cell is not very well illuminated if most of its crossing rays are roughly parallel. In the other hand, the cell is efficiently illuminated if the azimuth of its crossing rays are widely distributed in the 3D space.

From all these informations we compute the *score* associated with a given cell. The score computation can be parameterized. For instance, a very simple score would only take into account the number of rays crossing the cell, whereas a more sophisticated one would take into account the distribution of the rays in the cell.

In a global Earth model, a ray path is usually discretized using several hundred to several thousands of points depending on the ray length and its nature. Since the millions of rays contained is the databases have to be discretized, billions of information have to be computed. Therefore the ray tracing and initial mesh construction processes require a parallel computation.

The parallelization lies in the concurrent computations of ray paths using independent copies of the mesh. The set of rays to be traced is divided into subsets according to the number of computing processors. Each of these processors receives a description of the mesh and one or more subsets of rays to trace[1]. During the first step of the algorithm,

[1] Load-balancing issues have been addressed in [2].

each processor computes the ray paths of its subset independently of any other processor and updates its local copy of the mesh with information brought by the computed rays.

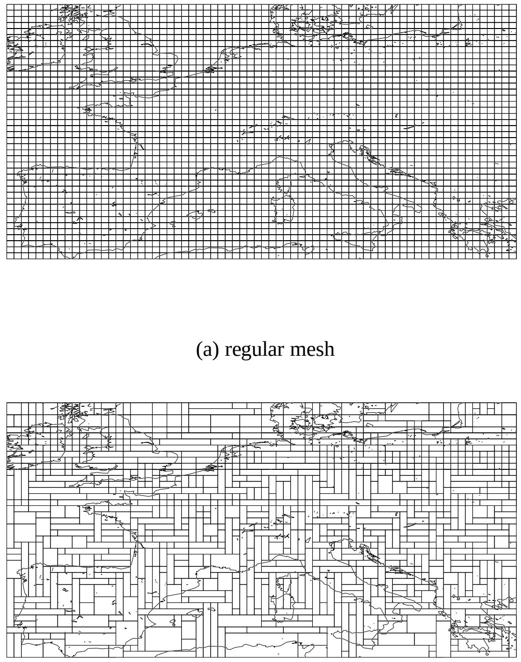
Once all the rays have been traced, each processor contains a representation of the whole mesh where only the information related to the rays it has traced are collected. The second phase of the algorithm requires therefore to merge the informations of all the rays in all the cells into a unique regular 3D mesh. This step is also computed in parallel as follows. The global mesh is decomposed into sub-domains and each processor is responsible for one sub-domain. An all-to-all communication phase occurs : each processor sends the partial informations it carries in each sub-domain to the processor responsible for the sub-domain. Once the communication phase is over, each processor merges the whole rays information in all the cells of its sub-domain. The final step of the algorithm consists in a collective communication to gather all the sub-domains on one master processor.

This parallel algorithm has been programmed in C (with specific routines in Fortran) and implemented using the MPI library [5]. It has been tested on various parallel platforms: an Origin 2000 parallel machine, a cluster of PCs and a grid with machines distributed all over France.

3. Adaptive mesh algorithm

The main contribution of this paper is the construction of an irregular mesh from the initial regular one. Figure 1 gives a partial representation of a regular mesh of 259200 cells covering the globe coarsened to 21383 cells. The construction is obtained by merging cells of this initial mesh in order to get well-illuminated cells. How well a cell is illuminated depends on the score related to that cell. As mentioned in section 2 the score computation relies on a specific parameterized function that uses the sets of information collected at each cell. The higher the score is, the better the illumination is. The merged cells in the irregular mesh are called *meta-cells*.

One peculiarity of the meshes we construct is their multi-layers aspect that comes from geophysical properties. This allows to run independently our algorithm on each layer of the 3D-mesh. We start from a one-layer regular mesh Ω of cells noted $C_{x,y}$, where x, y are the cell's longitude and latitude coordinates in Ω (the depth coordinate is given by the considered layer). The objective is to construct an irregular mesh Ω' made of juxtaposed, non-overlapping meta-cells. A meta-cell $M^{p,q}_{x,y}$ is a rectangular area composed of $p \times q$ juxtaposed cells whose upper-left corner is originated at coordinates x, y. The surface of a meta-cell is limited by parameters: $0 < p \leq P$ and $0 < q \leq Q$, so that the solution does not produce very big meta-cells that would not make sense from a geophysical point of view[2].


(a) regular mesh

(b) irregular mesh

Figure 1. $0.5^\circ\times\ 0.5^\circ$**regular mesh and its corresponding irregular mesh at 660-760 km depth.**

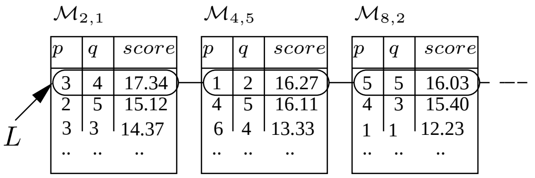
We can give an overview of the algorithm by decomposing it in two phases:

1) Configurations computation and ordering. For each cell $C_{x,y}$ we evaluate the score $S^{i,j}_{x,y}$ of each meta-cell $M^{i,j}_{x,y}$ that can be constructed from $C_{x,y}$, i.e. we compute the scores : $\{S^{i,j}_{x,y} \mid i \in [1, P] \ \wedge\ j \in [1, Q]\ \}$. We sort these values in decreasing order in a list called $\mathcal{M}_{x,y}$. This step is repeated for all the cells, resulting in one list $\mathcal{M}$ per cell.

Because the head element of an $\mathcal{M}$ list is the best candidate meta-cell, we build a list L containing all these head elements. This list is also sorted by decreasing scores: $L = \texttt{sort}\ (\{\cup\ \texttt{head}\ (\mathcal{M}_{x,y}), \forall x, y \mid C_{x,y} \in \Omega\ \})$.

[2] These parameters are set by the geophysicist.

At this point the situation corresponds to the one exemplified on figure 2. In this example the first three best scores are respectively the ones of meta-cells $M^{3,4}_{2,1}, M^{1,2}_{4,5}$ and $M^{5,5}_{8,2}$, being by definition the leading elements of the meta-cells lists $\mathcal{M}_{2,1}, \mathcal{M}_{4,5}$ and $\mathcal{M}_{8,2}$. These best scores are therefore the first elements of list L.



p	q	$score$
3	4	17.34
2	5	15.12
3	3	14.37
..	..	..

p	q	$score$
1	2	16.27
4	5	16.11
6	4	13.33
..	..	..

p	q	$score$
5	5	16.03
4	3	15.40
1	1	12.23
..	..	..

Figure 2. A configurations ordering example

2) Configurations selection. The algorithm must now select the meta-cells with the best scores in such a way that the remaining meta-cells define a paving of the domain (i.e. the final set of meta-cells covers the whole domain with neither overlap nor hole). It therefore consists in scanning the initial list L and ”cleaning” it by removing all the meta-cells that do not belong to the final irregular mesh. Obviously, the head element of L ($M^{3,4}_{2,1}$ on figure 2) represents the meta-cell with the best score amongst all possible meta-cells, and as such is selected. As a consequence, all other potential meta-cells that have a common cell with the selected meta-cell will never be realized since they would overlap. They must therefore be removed from the set of candidates.

The removing phase process, described by Algorithm 1, is the core of our method. We use classical functions on lists, `head` and `tail` to return the first element and the rest of the list respectively. Function `remove` which returns a list without a given element has $O(1)$ complexity due to the pointers used in the implementation. Function `insertsorted` which inserts an element at the right place in a sorted list of length l has a $O(l + \log_2 l)$ complexity.

The removing process is decomposed into two phases, as sketched on figure 3.

Let us suppose $M^{p,q}_{x,y}$ has been selected for realization. The *forward removing* removes all meta-cells $M^{p',q'}_{u,v}$ intersecting with the selected one, such that their heading cell $C_{u,v}$ is contained in $M^{p,q}_{x,y}$ ($x \leq u < x+p$ and $y \leq v < y+q$). The *backward removing* successively analyses each elementary cell $C_{u,v}$ of the selected meta-cell. For each of them, it scans the rectangular area of dimension $P \times Q$ focused on $C_{u,v}$ (as shown on the figure 3 (b)) and removes from L all meta-cells $M^{p'',q''}_{a,b}$ which contain $C_{u,v}$. After that, we must find the next valid meta-cell $\mathcal{M}_{a,b}$ to insert at the right place in L. Notice that in both cases, the meta-cell to be removed has a lower score than the selected one. It is therefore located after the selected meta-cell in list L as shown on figure 3.

```
Data    : meta-cell $M^{p,q}_{x,y}$ to realize, list $L$
Result  : partial cleaned list $L$
foreach $\{(u,v) \mid C_{u,v} \in M^{p,q}_{x,y}$ and $C_{u,v} \neq C_{x,y}\}$ do
    // Forward removing
    $M^{p',q'}_{u,v} \leftarrow$ head ($\mathcal{M}_{u,v}$)
    $L \leftarrow$ remove ($M^{p',q'}_{u,v}, L$)
    free ($\mathcal{M}_{u,v}$)
    // Backward removing
    $Area \leftarrow \{C_{u-i,v-j} \mid i \in [0,P[$ and $j \in [0,Q[\ \}$
    foreach $\{(a,b) \mid C_{a,b} \in Area$ and $C_{a,b} \neq C_{x,y}\ \}$ do
        $M^{p'',q''}_{a,b} \leftarrow$ head ($\mathcal{M}_{a,b}$)
        if $C_{u,v} \in M^{p'',q''}_{a,b}$ then
            $L \leftarrow$ remove ($M^{p'',q''}_{a,b}, L$)
            $\mathcal{M}_{a,b} \leftarrow$ tail ($\mathcal{M}_{a,b}$)
            //Find the next valid meta-cell $M_{a,b}$ to
             be inserted in $L$
            $found$ = False
            while not $found$ and $\mathcal{M}_{a,b} \neq \emptyset$ do
                $M^{p'',q''}_{a,b} \leftarrow$ head ($\mathcal{M}_{a,b}$)
                if $C_{u,v} \notin M^{p'',q''}_{a,b}$ then
                    $L \leftarrow$ insertsorted ($M^{p'',q''}_{a,b}, L$)
                    $found$ = True
                else
                    $\mathcal{M}_{a,b} \leftarrow$ tail ($\mathcal{M}_{a,b}$)
                endif
            endw
        endif
    endfch
endfch
```

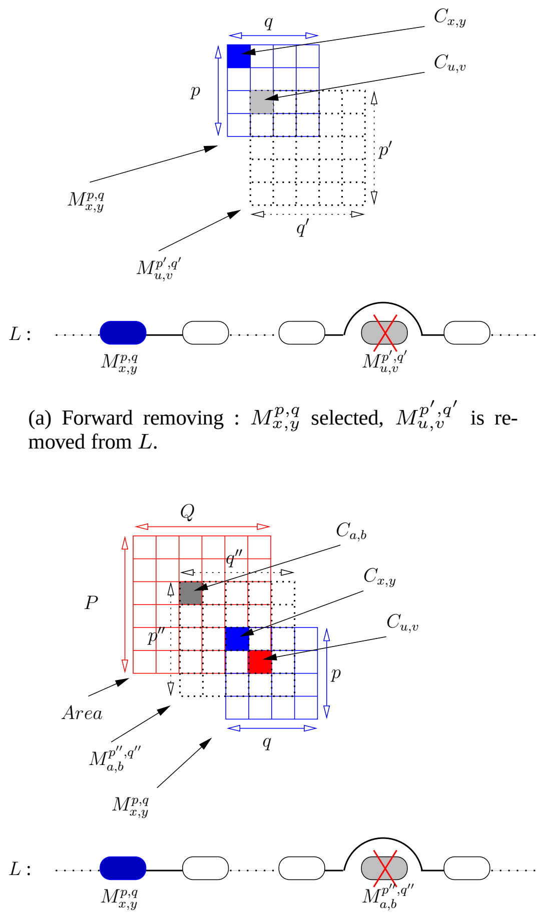
Algorithm 1: Removing phase process (call to `SelectClean` ($M^{p,q}_{x,y}$, L))

We encapsulate Algorithm 1 in a procedure called `SelectClean` ($M^{p,q}_{x,y}$, L). The rest of the sequential algorithm consists in scanning L, and calling `SelectClean` for each element not yet selected. This is described by the main procedure (Algorithm 2).

At each iteration the algorithm removes many meta-cells due to the backward and forward removing process. At the first iteration the number of removed meta-cell is in $O(P^2Q^2)$. After a certain number of steps, most of the cells that should be removed have already been erased in previous iterations, and therefore the remaining iterations are in $O(1)$. This behavior is illustrated on figure 4: for $P = Q = 10$, with $Card(L) = 259200$ we call 16420 times the `SelectClean` procedure but almost all meta-cells have been removed before the 4000th call.

4. Parallelization of the algorithm

The parallelization of the algorithm consists in dividing the domain of the initial mesh Ω into adjacent non-overlapping sub-meshes $\Omega_{i,j}$ such that $\cup\Omega_{i,j} = \Omega$ and $\cap\Omega_{i,j} = \emptyset$. Each sub-mesh $\Omega_{i,j}$ is treated by a given pro-



(a) Forward removing : $M_{x,y}^{p,q}$ selected, $M_{u,v}^{p',q'}$ is removed from L.

(b) Backward removing : $M_{x,y}^{p,q}$ selected, $M_{a,b}^{p'',q''}$ is removed from L. The next valid element of $\mathcal{M}_{a,b}$ is then inserted into L.

Figure 3. Examples of meta-cell removing

cessor $Proc_{i,j}$ which computes the initial list of meta-cells $L_{i,j}$ and then cleans it up by selecting the ones that define the paving.

In order to build such a list, the processor must contain enough information to properly activate the `SelectClean` procedure. This is why the data available on each processor correspond to overlapping sub-domains, as shown on figure 5 (b). The size of the overlap is $2(P-1) \times 2(Q-1)$ and is correlated to the cleaning process as shown on figure 5(a). Hence the extreme cells of $\Omega_{i,j}$ (in black on figure 5(a)) require, when selected both a forward cleaning and a backward cleaning that can reach cells whose distance from $\Omega_{i,j}$ is $2(P-1)$ or $2(Q-1)$ at the farthest. The overlapping sub-domain related to sub-mesh $\Omega_{i,j}$ is denoted by $D_{i,j}$.

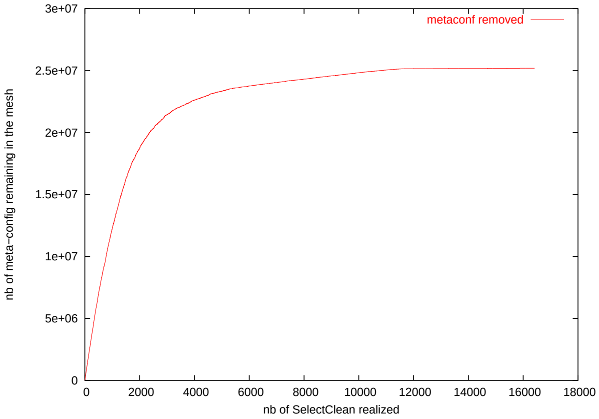


Figure 4. Cumulated number of meta-cells configurations removed as a function of the number of `SelectClean` calls.

```
Data    : Initial mesh $\Omega$, $\mathcal{M}$ list and initial list $L$
Result  : Cleaned list $L$ containing the meta-cells of irregular mesh $\Omega'$
$M_{x,y}^{p,q} \leftarrow$ GetFirst($L$)
while $M_{x,y}^{p,q} \neq \emptyset$ do
    $L \leftarrow$ SelectClean ($M_{x,y}^{p,q}$, $L$)
    $M_{x,y}^{p,q} \leftarrow$ GetNext($L$)
endw
```

Algorithm 2: Main procedure

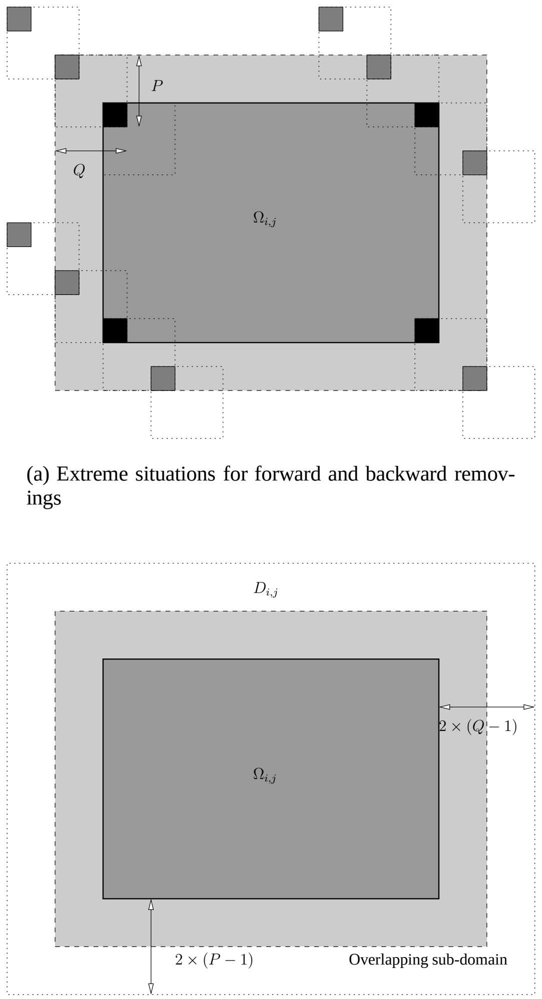
The first overlap zone of size $(P-1) \times (Q-1)$ (in light grey on figure 5) is the one on which initial list $L_{i,j}$ of best-scores meta-cells is computed. It contains cells whose corresponding meta-cells could further on interfere with a meta-cell of $\Omega_{i,j}$. Initial list $L_{i,j}$ is computed in the same way as in the sequential algorithm and is sorted on decreasing scores.

Once the cleaning process starts on list $L_{i,j}$, processor $Proc_{i,j}$ must compute both forward and backward removing as defined in the sequential mode. It must therefore have the information related to the second overlap zone (in white on figure 5), in order to remove all invalid meta-cells. The set of all configurations related to possible meta-cells is computed on this whole overlapping sub-domain $D_{i,j}$ at the beginning of the computation.

Each time processor $Proc_{i,j}$ selects a meta-cell whose related cleaning process interferes with a sub-mesh adjacent to $\Omega_{i,j}$, it must communicate the corresponding information to the appropriate processor.

On the other side, when the current meta-cell to be treated by processor $Proc_{i,j}$ intersects its overlap zone, it is not allowed to treat it, because the processor which is responsible for this meta-cell is its neighbor. Therefore it has to wait till it receives information from this neighbor.

Every time processor $Proc_{i,j}$ receives from one of its



(a) Extreme situations for forward and backward removings

(b) Overlapping zone around $\Omega_{i,j}$ defining the domain $D_{i,j}$.

Figure 5. Overlap related to a given sub-mesh

neighbor the reference of the meta-cell the neighbor has selected, it first inserts it into its private list $L_{i,j}$, with respect to decreasing scores, and then treats it, i.e. runs the `SelectClean` procedure in order to remove all the invalid meta-cells it induces.

Because the various meta-cells lists are always kept ordered, the whole process will run without any deadlock. At the end, the various $L_{i,j}$ lists have to be merged to get the final irregular mesh. Some meta-cells can belong to two adjacent sub-mesh (when they intersect with an overlap zone) and the merging step must of course add them only once in the resulting mesh.

This whole process is described by Algorithm 3.

```
Data    : $D_{i,j}$ with cells $C_{x,y}$, meta-cells lists $\mathcal{M}_{x,y}$ and initial list
          $L_{i,j}$.
Result  : Cleaned list $L_{i,j}$ containing the meta-cells of irregular mesh
          $\Omega'$.
$MsgList \leftarrow \emptyset$
$M^{p,q}_{x,y} \leftarrow$ GetFirst($L_{i,j}$)
while $M^{p,q}_{x,y} \neq \emptyset$ do
    // meta-cell $M^{p,q}_{x,y}$ is selected
    if $C_{x,y} \in \Omega_{i,j}$ then
        // process $P_{i,j}$ is in charge of the creation
         of meta-cell $M^{p,q}_{x,y}$.
        foreach $\{(k,l) \mid k \in [i-1,i+1] \;\wedge\; l \in$
        $[j-1,j+1] \;\wedge\; (k,l) \neq (i,j)\}$ do
            if $M^{p,q}_{x,y} \cap D_{k,l} \neq \emptyset$ then
                $msg \leftarrow (x,y,p,q,S^{p,q}_{x,y})$
                // non blocking send
                SendMsg($msg, P_{k,l}$)
            endif
        endfch
        // elements of $L_{i,j}$ that share cells with
         $M^{p,q}_{x,y}$ must be removed
        SelectClean($M^{p,q}_{x,y}$, $L_{i,j}$)
    else
        // proc. $P_{i,j}$ has received or will receive
         a message from the proc. in charge of the
         creation of $M^{p,q}_{x,y}$.
        $reached \leftarrow$ False
        while $reached$ = False do
            // messages from neighbor processors
             received
            foreach $\{(k,l) \mid k \in [i-1,i+1] \;\wedge\; l \in$
            $[j-1,j+1] \;\wedge\; (k,l) \neq (i,j)\}$ do
                while $\exists$ messages of $P_{k,l}$ do
                    $msg \leftarrow$ RcvMsg($P_{k,l}$)
                    AddMsg($msg$, $MsgList$);
                endw
            endfch
            // message list $MsgList$ is sorted by
             decreasing scores
            sort($MsgList$)

            // synchronization phase
            $msg \leftarrow$ GetFirst($MsgList$)
            while $msg \neq \emptyset$ and $S^{p,q}_{x,y} \leq$ GetScore($msg$)
            do
                // forward and backward cleanings in
                 domain $D_{i,j}$, according to $msg$
                $M^{p',q'}_{x',y'} \leftarrow$ GetMetaCell($msg$)
                SelectClean($M^{p',q'}_{x',y'}$, $L_{i,j}$)
                RemoveHead($MsgList$)
                $LastMsg \leftarrow msg$
                $msg \leftarrow$ GetFirst($MsgList$)
            endw

            // checking whether or not there are
             information on $C_{x,y}$ ?
            if $S^{p,q}_{x,y} >$ GetScore($msg$) then
                $reached \leftarrow$ False
            else
                // $msg = \emptyset$ ie. $MsgList$ is empty
                if $S^{p,q}_{x,y} <$ GetScore($LastMsg$) then
                    // $M^{p,q}_{x,y}$ may have already been
                     selected
                    if isInvalidated($M^{p,q}_{x,y}$) then
                        $reached \leftarrow$ True
                    else $reached \leftarrow$ False
                else
                    // $S^{p,q}_{x,y} =$ GetScore($LastMsg$)
                    if $LastMsg \equiv M^{p,q}_{x,y}$ then
                        $reached \leftarrow$ True
                    else $reached \leftarrow$ False
                endif
            endif
        endw
    endif
    $M^{p,q}_{x,y} \leftarrow$ GetNext($L_{i,j}$)
endw
```

Algorithm 3: parallel algorithm for process $P_{i,j}$

5. Experimental results

We have run a series of benchmarks to assess the application performances on two kinds of cluster. The first is a commodity cluster with six SMP bi-processors PC/Xeon 1.7Ghz, each bi-processor sharing 1 GO RAM and linked with a gigabit ethernet network. The second cluster has 30 SMP nodes with two Itanium2 1.3Ghz and 8 GB RAM on each node. It has a double internal interconnection network: a gigabit ethernet network is used for NFS links while messages of parallel applications transit through an independent Myrinet network.

As one of our objectives is to develop a code portable to a wide range of parallel architectures we have chosen the MPI standard [5] to develop the parallel version. The program has been linked against the LAM-MPI implementation [1] on the PC/Xeon cluster, and compiled with gcc 3.3.3. On the Itanium cluster, the results are drawn from the program using the MPI-GM library from Myricom and compiled with the Intel icc 8.0 compiler. In all cases, compilers were invoked with an O3 optimization level.

The data set we use comes from the ray-tracing of 817000 rays in an initial 11 layers mesh (as defined in the IASPEI91 model). For all tests we work on the 7th layer which contains more data than other layers, and therefore implies a longer configuration ordering because the score computation time increase with the data quantity.

The initial regular mesh is build from a decomposition of a given layer into cells $0.5°\times 0.5°$wide, resulting into 259200 cells by layer. We have run the application with 10×10 as maximum sizes for the meta-cells and the experimental results are presented in tables 1 and 2. They show for various numbers of processors, the total application time, the time for the first phase of the algorithm, that is the configurations *computation and ordering*, and the time for the second phase, *configuration removing*. As explained in the previous section, this last phase requires cooperation between processors for decision making and the time spent waiting is reported between parentheses as idle time. Times are in seconds, and for runs with several processors the average time is reported. The last column indicates the speedup obtained.

np	whole app.	comp. and ordering	selection (idle)	speedup
1	160	116	43 (0)	1
4	75.5	36	37.2 (19.7)	2.11
8	42.7	19.25	22.9 (16.1)	3.74
12	38	14	22.2 (17.4)	4.21

Table 1. PC/Xeon 1.7Ghz cluster benchmarks with $P \times Q = 10 \times 10$

np	whole app.	comp. and ordering	selection (idle)	speedup
1	100	78	21 (0)	1
4	41	23.7	17 (9.2)	2.43
8	23.5	12.4	10.75 (7.6)	4.25
12	19.8	9.2	10.3 (8)	5.05
16	17	7.25	9 (7.2)	5.88

Table 2. Itanium2/1.3Ghz cluster benchmarks with $P \times Q = 10 \times 10$

Quite obviously, the speedup for the parallel version of the application show a limited scalability in the number of processors.

Indeed, the algorithm design involves a heavy communicating scheme for the parallelized application in the second phase since a lot of small and asynchronous messages must be exchanged for processors to make their selection/removing decisions. In order to guarantee that the final irregular mesh does contain the best illuminated cells, each processor has to wait for information from one of its neighbor when it is working on a meta-cell that has interaction with its own overlapping zone. Hence it cannot decide to select this meta-cell without knowing whether it has been invalidated or not by its neighbor. Therefore, a processor may have to wait till it gets this information. However, the first phase requires only independent computations whose complexity depends on the data quantity in cells and on the cells sizes. This leads to the idea that the choice for the parameters has a great influence on the performances of the parallel version. Choosing a finer mesh resolution would induce more parallel computations (with the only overhead of the overlap zone to compute) in the first phase.

To estimate the overhead inherent to the second phase, we might consider the ratio between the number of cells in a sub-mesh $\Omega_{i,j}$ and the number of cells in its overlap zone which are included in the $L_{i,j}$ list. This number depends on the size $P \times Q$ of the meta-cells.

As we can see in table 3, this ratio is roughly equal to 3 for 16 processors, i.e. there are only three cells in $\Omega_{i,j}$ for one in the overlap zone which have populated list $L_{i,j}$. This implies a low computation/communication ratio that explains the performances curves on figure 6.

np	ratio
16	3.25
12	3.79
8	4.71
4	6.77

Table 3. Ratio between the number of cells in $\Omega_{i,j}$ and in its overlapped zone.

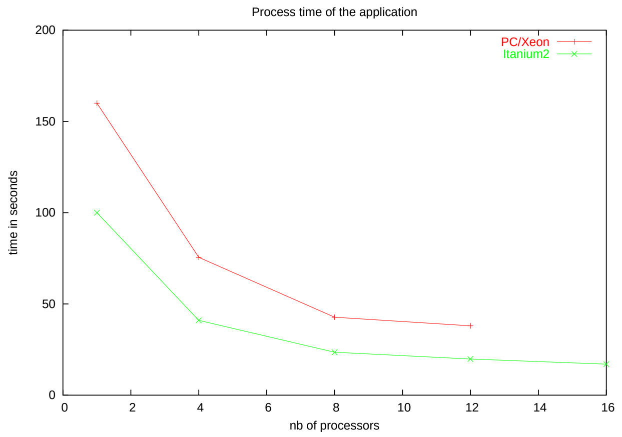


Figure 6. Performances

In order to reduce the idle time, extra-work is currently under way to optimize the parallel algorithm. The idea currently studied is to make processors manage several sub-domains. By assigning several non-adjacent sub-domains to a given processor, we minimize the probability for the processor to enter an idle state waiting on all sub-domains. Each processor will compute the selection phase for its assigned sub-domains in a preemptive way, by switching from one sub-domain to the next after a certain number of either `SelectClean` calls or sent messages.

6. Conclusion

In this paper we have proposed a method to construct an adaptive mesh for seismic tomography. Because of the nature of the data and of the geophysical problem, the resulting irregular mesh is build by layers and is characterized by well-illuminated cells. We first discuss a sequential algorithm to solve the problem. Given the size of the seismic data to be processed, it is argued that parallel computers are required and a parallel version of the algorithm is also proposed. We have experimented the application with real data by running it on two parallel platforms. Experimental results show that the speed-up is limited due to the idle time during the second phase. Further work is currently under way to improve the parallel algorithm and to reduce this idle time. The idea currently developed is to make processors manage several sub-domains so that a processor entering in an idle wait for a cell, would be able to switch to another sub-domain requiring computations, thus overlapping communications.

References

[1] G. Burns, R. Daoud, and J. Vaigl. LAM: An Open Cluster Environment for MPI. In *Proceedings of Supercomputing Symposium*, pages 379–386, 1994.

[2] S. Genaud, A. Giersch, and F. Vivien. Load-balancing scatter operations for grid computing. In *Proceedings of 12th Heterogeneous Computing Workshop (HCW ’03)*. IEEE Society Press, April 2003.

[3] M. Grunberg, S. Genaud, and C. Mongenet. Seismic ray-tracing and Earth mesh modeling on various parallel architectures. *The Journal of Supercomputing*, 29(1):27–44, July 2004.

[4] B. L. Kennett. IASPEI 1991 seismological tables. *Research School of Earth Sciences Australian National University*, 1991.

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

[6] A. Michelini. An adaptative-grid formalism for travel-time tomography. *Geophysical Journal International*, (121):489–510, 1995.

[7] M. Sambridge and R. Faletic. Adaptive whole earth tomography. *Geochem., Geophys., Geosyst*, 4(3), March 2003.

[8] M. Sambridge and O. Gudmundsson. Tomography systems of equations with irregular cells. *J. of Geophys. Res.*, 103(No. B1):773–781, 1998.

[9] W. Spakman and H. Bijwaard. Optimization of cell parametrizations for tomographic inverse problems. *Pure and Applied Geophysics*, (158):1401–1423, 2001.

[10] C. Thurber and D. Eberhart-Phillips. Local earthquake tomography with flexible gridding. *Computers & Geosciences*, (25):809–818, 1999.