ANNALS OF GEOPHYSICS, VOL. 52, N. 5 October 2009

Parallelisation technique for serial 3D seismic codes: SMS approach

Arrigo Caserta ([1])([2])**, Vittorio Ruggiero** ([3])**, Maria Pia Busico** ([3]) **and Ivo Opršal** ([2])
([1]) *Istituto Nazionale di Geofisica e Vulcanologia, Roma, Italy*
([2]) *Charles University, Prague, Czech Republic*
([3]) *C.A.S.P.U.R., Roma, Italy*

Abstract
We investigate a fast and easy way to parallelise seismological serial codes mainly oriented for simulating the seismic wave propagation through anelastic dissipative media. Having an efficient modelling tool is important in both assessing strong ground motion and mitigation of seismic hazard when the site effects are considered, and in crustal propagation when the crustal geological structures are of interest. Our chosen case study is representative of a set of such seismological 3D problems. The Scalable Modelling System (SMS) tool for parallelization is considered. The IBM SP5 native compiler has been used. Results such as Speed-Up and Efficiency are shown and discussed. SMS can run both in shared and distributed memory environments. The greater advantages of using SMS in such environments become apparent with the utilisation of a higher number of multiprocessor machines arranged in a cluster. We also demonstrate how successful porting from serial to parallel codes is realised by way of minimal instructions (6% of the serial original code only) provided that an ad hoc profiling analysis of the serial code is first performed.

Key words *parallel computing – SMS – seismic wave propagation – numerical simulations – site effects*

1. Introduction

An important contribution to assessing interaction between near-surface geology and seismic radiation arises from studies involving the so called *site effects* P.-Y. Bard, 1995; 1998; Panza *et al.,* 2001; S. Bonnefoy-Claudet *et al.,* 2006, amongst others). These latter are related to seismic wave propagation phenomena (Aki, 1988; 1993) with important implications for seismic hazard mitigation.

Site effects can be studied using an instrumental approach during earthquakes recording soil shaking on specific stations located on suitable chosen sites (Caserta *et al.,* 2000; Di Giulio *et al.,* 2002; Cultrera *et al.,* 2003; Cara *et al.*, 2005). Nevertheless, it would be more useful to know the site characteristics of the ground motion before an earthquake occurs, in this respect numerical modelling is a powerful approach (Caserta and Lanucara, 2000; Moczo *et al.,* 2001) also for estimating parameters commonly used for earthquake engineering purposes (Fäh *et al.,* 1993; Rovelli *et al.,* 1994). This is particularly true in urban areas where the instrumental approach may not always be used because of the high level of anthropogenic noise (Olsen *et al.,* 2006).

The great drawback to modelling more realistic site dynamics is the impressive computational requirements needed for numerical simulations such as: gigabytes of memory, performances at gigaflops rate, days of computational time to simulate a minute's soil shaking, etc. (Sáncez-Sesma and Luzon, 1995; Graves, 1998). Seismologists have been addressing these issues through the use of parallel comput-

Mailing address: Dr. Arrigo Caserta, Istituto Nazionale di Geofisica e Vulcanologia, Via di Vigna Murata 605, 00143 Roma, Italy; e-mail: arrigo.caserta@ingv.it

ers (Olsen and Archuleta, 1996) coupled with optimization techniques such as the use of unstructured grids to reduce the number of computational nodes (Bao *et al.,* 1996).

During the last 20 years a considerable amount of work has been done in numerically modelling the interaction between both near-surface and inner geological heterogeneities and seismic radiation (see the review of Bard, 1998; Bielak *et al.,* 1998, Panza *et al.,* 2001; Moczo *et al.,* 2007, and references therein). The results are huge serial codes able to numerically simulate the process. It would be useful to reconvert such serial codes into parallel ones, to have more powerful simulations that realistically represent the dynamics of such interaction.

On the other hand, the conversion of a serial code to a parallel one requires typically rewriting only choice available is to rewrite the serial codes from the beginning, and this is not an easy task as well as being energy demanding. Not to mention that writing a 3D code from the scratch, on the basis of our experience, is more time demanding than paralleling an existing serial 3D code. This paper is aimed at investigating a more simple and efficient way to parallelise a 3D serial code, more specifically codes for simulating the seismic wave propagation through viscoelastic dissipative 3D media.

For the purposes of our study, we adopted the serial code of Opršal and Zahradnik (2002), which contains general aspects that make it representative of the dynamics that we want to simulate. Based on a finite-difference technique on a Cartesian mesh (Opršal and Zahradnik, 2002; Opršal *et al.,* 2004, 2005), it computes the full 3D seismic wave-field. Null normal stress boundary condition on the topographic surface is realized by the so-called *vacuum formalism* (Opršal and Zahardnik, 2002). Its stability conditions are represented by the classical C.F.L. criterion for explicit methods (Mitchel and Griffits, 1980).

These features are similarly adopted in many of the serial codes developed in the last years dealing with the seismic wave propagation problem, used to assess strong ground motion (Zahradnik *et al.,* 1993; Graves, 1998; Opršal and Zahradnik, 1999; Pitarka, 1999; Moczo *et al*., 2001; amongst others).

2. Study case: serial 3D wave propagation code

The main guidelines for the previously-referenced program by Opršal and Zahradnik (2002) are based on the formulation of an initial boundary value problem that represents the viscoelastic dynamics driving the full 3D wave-field propagation. The medium is dissipative with arbitrary 3D shape and topography.

Source energy nucleation and its crustal propagation can be numerically modelled through a discrete wave-number method (DW), as well as ray theory or even an analytical solution; the resulting wave-field represents the seismic radiation ready to light up the site under study. The finite-difference (FD) technique is adopted to model the interaction between the site and the seismic radiation previously computed yielding the site response. In such a way we are able to study the combined source – path – site response of the medium under the action of input radiation as it arises from the seismic source and propagated through the crust beneath the site.

The DW-FD coupling used here is realized by extending hybrid technique developed first for the 2D case by Zahradnik and Moczo (1996) and generalized by Opršal *et al*. (2009) to the 3D case.

The main advantage of such a hybrid technique is fast evaluation of site effects and seismic hazard because it needs less computer memory and time than all-in-one source-path-site computational methods. Computations are made in the time domain. The results are synthetic seismograms on the free surface and snapshots of the wave amplitude inside a 3D profile (for further details see Opršal and Zahradnik, 2002). Furthermore, this hybrid technique allows us to study the source-crustal propagation in case we are interested in the crustal propagation only.

Our test case deals with a real geological structure (Fäh *et al.,* 2006), representing the stratigraphy underlying the Roman city Augusta Raurica (fig. 1), located east of Basel, Switzerland. The 3D structure of the area was retrieved through array investigations (H/V, ambient vibrations) and borehole data (Fäh *et al.,*

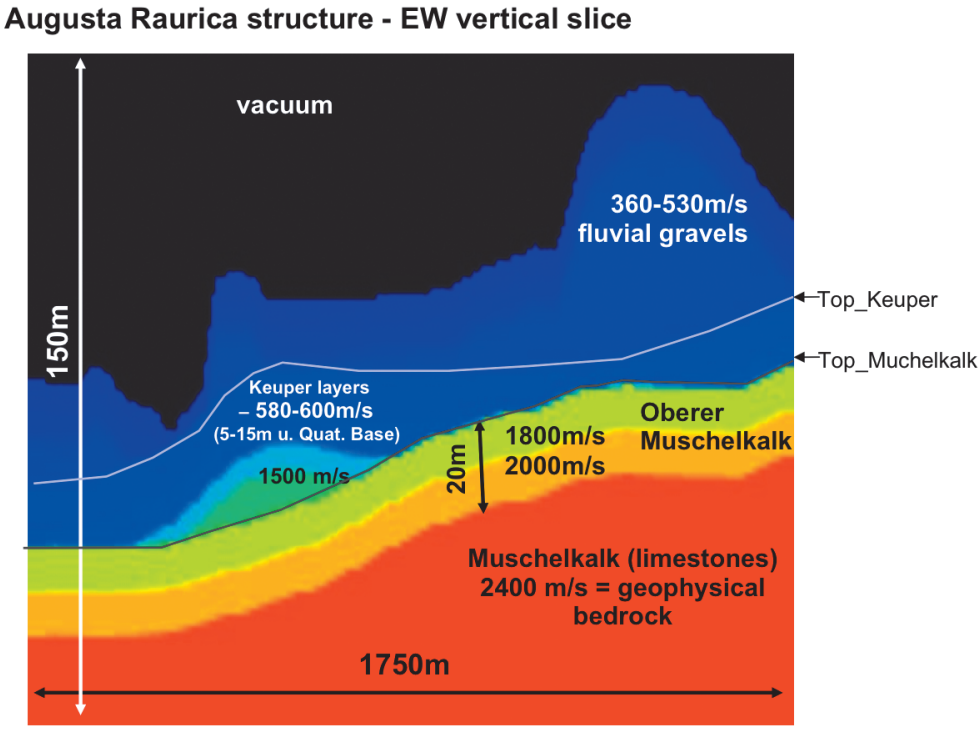

Fig. 1. Geological model at Augusta Raurica. The EW vertical slice is a 3D FD computational model extraction. See also fig. 2.

2006; Opršal and Fäh, 2007), and the topmost structure (namely so-called Top_Muschelkalk interface), was refined in terms of fundamental frequencies fit by modifying the shape of the interface limiting the upper geology from below (see fig. 2). The geological model was converted into a computational one according to the stability conditions of the numerical scheme (fig. 1).

According to Fäh *et al.* (2006), the section of interest is 1552 m (WE) x 1696 m (SN) large rectangular area with constant spatial sampling of 4 m. The computational domain covers the wave dumping area around, resulting in 442 x 480 grid points corresponding to 1764 m x 1916 m of computational model, respectively. The vertical extent of the computational model is 115 grid points corresponding to ~793 m depth while the geological structure differing from bedrock up to topography peaks is less then 150 m. The vertical grid step is not constant, being scaled with respect to the vertical velocity profile.

This study also used the same time-step, 0.00023 s and these parameters guarantee the accuracy of the FD computation up to a frequency of 12 Hz with a usable range up to 10 Hz. We simulate 5 s ground shaking for a total of 21740 time-steps. A problem of near 73 million unknowns is solved at each time-step. Although the Augusta Raurica site was chosen for the purpose of this test case, our study code can be applied to any 3D stratigraphy, as it depends on neither the shape nor the geological details of the site. Indeed, other recent real case studies (Opršal and Fäh, 2007; Sorensen *et al.,* 2006; Opršal *et al*., 2004; 2005), have used this study code to simulate wave-field time evolution and its interaction with the geological structure.

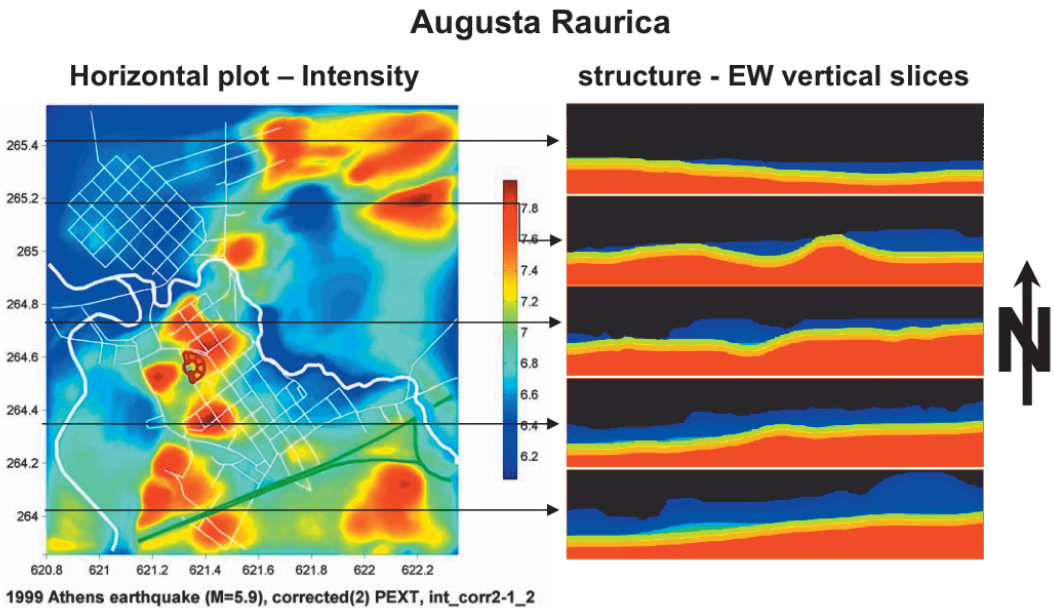

Fig. 2. Intensities for Mw=5.9 1999 Athens-like earthquake simulation in Augusta Raurica (left panel); for details see Fäh *et al*. (2006) and Opršal and Fäh (2007). Thick white lines are rivers Ergolz (left) and Violenbach (right), red semi-ring denotes the amphitheatre, and the green lines in the southern part are for present-time highway. The thin white lines are historically re-mapped streets of the upper and lower city of Augusta Raurica. The right panel depicts vertical slices of computational model realization of the geology structure; vacuum shown as black color. Black arrows connect the surface traces of the vertical geology slices.

3. Parallelisation

Tools to parallelise a code can be divided into high-level and low-level programming. Parallel virtual machine (PVM) and message passing interface (MPI) belong to low-level programming, where the parallelisation scheme is planned, designed and realized by the user. In more detail, the management of information exchanged among processors (*e.g*., message passing, synchronisations, etc.) is realised by the user adding code segments. Although efficient on shared and distributed memory architectures, low-level techniques will not be considered in this study as, on the basis of our experience, too much effort (and time) must be spent to convert a serial code into an MPI (or PVM) parallel one (Caserta *et al.,* 2002). To parallelise our serial code we chose tools based on a high level technique where all low level parallel machinery (creation, communications and maintenance of processes) is realised by directives to the compiler in a user-hiding mode.

The best candidates for such an approach are High Performance Fortran (HPF), OpenMP and Scalable Modelling System (SMS). It must be noted that while OpenMP is only available on shared memory architectures, both HPF and SMS can be used in shared as well as distributed memory environments. Contrary to Caserta *et al*. (2002), we have avoided adopting OpenMP for several reasons. First of all, our main target is to simulate numerical problems with tens of millions of variables, nearly 20 times the average size problem of Caserta *et al*. Therefore, we prefer to adapt our code to distributed memory machines that allow to handle much larger amount of memory. We choose a high-level technique based on distributed memory architecture as it is more suited for our purposes.

Because HPF is maintained by third-party and private companies (see the HPF official website http://hpff.rice.edu), we have chosen the SMS tool which is freely available. See Foster (1995) for a complete survey of parallel programming design and tools on Distributed Memory architectures, while refer to Nichols *et al*. (1996) and the official OpenMP web site for Shared Memory references. As far as SMS is concerned, see Govett *et al*. (2003) and the official SMS website (http://wwwad.fsl.noaa.gov/ac/sms.html). The detection of code segments that must be modified to optimize the serial code and to reduce the memory requirement is performed by the user via the application of a profiling analysis. In such a way, we can ensure a uniformly optimized serial code suitable for parallelisation. Below, we detail how this operation is carried out.

3.1. *SMS technique*

SMS has been developed by the Advanced Computing Branch of the Forecast Systems Laboratory at NOAA (National Oceanic and Atmospheric Administration) (Govett *et al.*, 2003). The user inserts directives in the form of comments (indetified by `!sms$`) into the existing serial Fortran code. SMS translates the code and directives into a parallel version that runs efficiently on shared and distributed memory high-performance computing platforms. The translation is realized by the Parallelising Pre-Processor (PPP), a component of SMS. In other words, SMS translates user directives into MPI-like code. The main advantage of using SMS is that such translation is hidden to the user. This is because SMS is built *on top* of MPI. As a consequence no complicated compiler-generated communication statements have to be included in the code. Moreover, SMS contains a number of features to speed up the debugging process and to support incremental parallelization. In order to obtain constructs ready to be automatically parallelised by the SMS, the first step is to upgrade some commands of the original Fortran77 code to Fortran90. In case the code is already written in Fortran90, the first step is to modify the code in such a way that it can directly understand SMS directives. So, according to SMS parallelization philosophy, we have first to define the decomposition type through the `decomposition declaration` as in the following example

```
module decomp
!sms$distribute (DECOMP_XY. 1. 2)
end module decomp                    (3.1)
```

With the previous declaration we have created a new Fortran90 module to define decomposition structure, `DECOMP_XY`, needed to support data exchange, local and global address translation, do-loop transformations and data decomposition. The number that follows `DECOMP_XY` in the decomposition declaration is the number of dimensions involved in the decomposition declaration. Even though we deal with a 3D problem, the maximum number of dimensions allowed by SMS for parallelizing in the `decomposition declaration` is 2. The next step is to distribute the array representing the solution of our equations of motion, more specifically the array `u (:,:,:,:)` representing soil displacement, as well as all other auxiliary arrays. According to the previous `decomposition declaration` we have:

```
!sms$distribute (DECOMP_XY. 1. 2) begin
   real. allocatable : : u (:,:,:,:,:)
!sms$distribute  end                  (3.2)
```

This directive links the data decomposition structures, defined in eq. (3.1), with arrays targeted for decomposition. With the previous SMS directive we have chosen to distribute the first two indices of array `u`.

Declarations (3.1) and (3.2) allow SMS to understand how the chosen arrays must be distributed among processors. This is because all information necessary to access, communicate, input and output, decomposed and non-decomposed arrays are specified in the previous two declarations. To go on with the parallelization, we insert SMS directives in the original Fortran90 code, after the declaration section. The first instruction is the `decomposition creation`

```
!sms$create_decomp(DECOMP_XY, <nx, ny>, <1, 1>)                    (3.3)
```

where the decomposition name, DECOMP_XY, the global size of the 1st and 2nd decomposed dimensions, <nx, ny> and width of the halo region, <1, 1>, are specified. More in details, <nx, ny> represents the maximum number of grid points of the 1st and 2nd dimension that must be decomposed, whereas <1, 1> represents the number of grid points involved in the halo region; we use a nearest neighbours algorithm in which we have specified the halo region made by just one grid point. Of course, such a halo region is in common between two neighbour processors. SMS requires the indices that have to be parallelized must be both, the first two in the array and they must refer to the grid's nodes. Such constrains might require modifications of the serial code to have the indices in correct order for the SMS parallelisation. The following old Fortran90 instruction in the serial code ALLOCATE (u(ndim,nx,nz,nt)) has the first index not linked with the grid's nodes, ndim is an index referring to the component, actually. Moreover, the indices of the grid's nodes, *i.e.*, nx, ny, nz are in the wrong position to be parallelized. So, according to the aforementioned SMS rules, we have been forced to change the index order as follows

```
ALLOCATE (u(nx,ny,nz,nt,ndim)                                      (3.4)
```

Once we have detected segments of the program that should be parallelized analyzing the code with a profiling software, we insert SMS directives in the serial code. From such profiling analysis it turned out that more than 84% of the time is spent by the serial code in executing the following section for a single time step, that's why we have focused our attention on it:

```
!sms$parallel(DECOMP_XY, <kx>, <ky>) begin
!sms$exchange (u)
.....
      .....
do kz=2,nz-1
      do ky=2,ny-1
         do kx=2, nx-1
        call stencil_full(kx,ky,kx)    ! full form stencil
       enddo
     ennddo
enddo
 .....
 .....

!sms$parallel end
 .....
 .....
subroutine stencil_full(kx,ky,kz)  ! the 3-D stencil:
  .....
  .....
unew=2.*u(kx,ky,kz,2,kul)-(u(kx,ky,kz,1,kul)*dtdt_rho* (&
& u(kx+1,ky-1,2,ku2)-u(kx,ky-1,kz,2,ku2)) -                  &
&(u(kx,ky,kz,2,ku1)-u(kx-1.ky,kz,2,ku1))))+                  &
& ((muf*r1_dyky*(u(k,ky+1,kz,2,ku1)-u(kx,ky,kz,2ku1)))- &

  .....
  .....
u(kx,ky,kz,kt3,ku1)=unew
end soubroutine stencil_full                                       (3.5)
```

The parallel section starts with the SMS instruction `!sms$parallel(DECOMP_XY, <kx>, <ky>) begin`. This instruction defines a region over which parallel computations will be done on each processor's local data, as defined by the given data decomposition `DECOMP_XY`. All do-loops inside a parallel region that reference the specified loop variables (`<kx>, <ky>`) will be translated.

It is worth noting that the order of the indices in the `u` array coupled with the order of the nested loops in (3.5), is the most efficient combination in respect to the so called *easy memory access patterns* problem (Dowd and Severance, 1998; Hennessy and Patterson, 2006). This holds for both serial and parallel codes provided they are written in Fortran language. For different programming languages the coupling between the order of indices in the array and the order of nested loops might be different.

The SMS directive `!sms$exchange (u)` has been introduced to communicate with neighbouring processors to update halo regions and to maintain consistency with the serial code in the updating. Such a communication is necessary because the array `u` shows a index dependence on the first neighbouring grid points, *e.g.* `u(kx+1,ky 1,kz,2,ku2)`. Information provided by (3.2) is used to generate the correct communication code for our exchanged variable. In other words, `!sms$exchange (u)` is in charge of the message passing. Time updating procedure has not been changed from serial to parallel code, details about such a procedure can be found in the Opršal and Zahradnik (2002).

Probable serial parts of the code, that cannot be parallelized, are nested into a parallel one by `!sms$serial` directive. The parallel region is closed by adding the directive `!sms$parallel end`.

It is worth noting that we have obtained the parallel version inserting directives that represent only 6% of the whole serial code.

3.2. *SMS performances*

For this study, numerical tests are run on an IBM SP5 cluster, with 48 processors arranged in six symmetric multi processor (SMP) machines (8 processors each). Communication between processors is realised via shared memory within each 8-way SMP machine, while a high performance switch (HPS) is used to realize communication between the six 8-way SMP machines in a cluster. Each processor has 1.9 Ghz Power5, the integrated cluster has a peak performance rate of 364 Gflops and a global RAM of 192 Gbytes. We used the XLF IBM native Fortran third party compiler optimized according to the target architecture, moreover, we have optimized the SMS tool to the IBM SP5 cluster. In order to achieve suitable performances the first step is to balance the computational work among processors. By default, SMS uses a pre-determined set of rules to decide how grid points are assigned to each process and how many processes are allocated to each decomposed dimension. Roughly speaking, the grid-points are distributed evenly among the processes along a given decomposed dimension.

In order to adopt its domain decomposition we have to run our SMS executable code as follows

```
smsRun np 3dcpp_SMS
```

where `3dcpp_SMS` is our executable whereas `np` is the number of processors.

Ideally, each processor will have the same number of computational grid-points. In practice, most models have computations that vary spatially so some processes may have more work than others. This is commonly known as *load imbalance*. Load imbalances slow down a parallel program because processes with less work are forced to wait for processes with more work to do.

The criterion used by SMS to divide the computational workload is to minimize information that must be exchanged among processors. However, attention should be paid in realizing the domain decomposition process.

This can be easily done by the user because SMS allows to specify the assignment of processors using a process configuration file in the form of a Fortran name list.

In details, we run our executable SMS code as follows

```
smsRun -cf my_config 3dcpp_SMS
```

where `my_config` is our configuration file

```
&decomp
decomp1_name = 'decomp_xy',
decomp1_nps = 2 23/
```

in which we specify, in the last line, the number of processors along x-direction and y-direction, respectively. In this case we have 46 processors available.

Figure 3 shows the time needed for a single time-step versus number of processors for both the domain decomposition suggested by SMS (Default in fig. 3), and the one preferred by us (User in fig. 3). Oscillations in time are clearly recognized as the number of processors increase. Such an effect is strictly linked to the load balancing problem. Indeed, in our case SMS has to divide 480 nodes along the x axis and 442 nodes along the y-axis. In the worst case with 46 processors (see fig. 3), SMS realized such division by default assigning 23 processors along x-axis and 2 processors along y-axis. This causes a big load imbalance well represented in the time needed for a single computational step.

To fix the load balancing problem, we have inverted the default SMS decomposition assigning 2 processors along x-axis and 23 processors along y-axis. Our choice, of 46 processors, improved the time needed for a single iteration by almost 19% (see fig. 3). Such a procedure has been adopted to improve the domain decomposition for other numbers of processors.

A useful index to check any gain in performance with respect to the serial code is the Speed-up, $S(n)$. It is defined as the ratio between the real time employed by one processor to run the code, $T(1)$, and the real time needed by n processors, $T(n)$:

$$S(n) = \frac{T(1)}{T(n)}.$$

Figure 4 left panel compares the Speed-up of both SMS default domain decomposition and our specified decomposition with ideal performance. It emerges that for our modeling the Speed-up scaling is satisfactory up to 24 processors. Beyond this threshold the onset of saturation for $S(n)$ starts. Saturation is an important marker, to illustrate its meaning we have to consider that the user-hiding message passing (among processors) is a time consuming process. Saturation starts when the message-passing time is comparable with the computation time needed by each processor to run its own part of the workload. The communication time increases with the number of processors while the computational time of each processor decreases. Therefore, there is no advantage in increasing the number of processors within a Speed-up saturated regime.

In looking at fig. 4 left panel, we see that the saturation is approached through oscillations in the Speed-up values and this is not surprising taking into account the load balancing analysis discussed above.

Figure 4 right panel shows the performance of the parallel code in terms of the Efficiency, which is defined as the ratio between the Speed-up and the number of processors:

$$E(n) = \frac{S(n)}{n}.$$

By definition, Efficiency is a measure of the relevance of user-hiding message passing for a single process. From fig. 4 right panel, the order of magnitude of the improvement achieved can be seen comparing the values of the Efficiency for the same number of processors in the case of the User and Default domain decomposition. As an example, looking at fig. 4 right panel, at 36 processors we see that the Efficiency corresponding to the User decomposition is almost 10% higher that the Default one.

Moreover, in looking at the Efficiency plot we see an improvement in performances each time a new 8 processors SP5 machine is added to the cluster. This effect is better seen and recognizable in the Efficiency curve corresponding to our domain decomposition choice. In fact, the Efficiency shows a decrease followed by a flat curve almost after every 8 processors. The same behavior has been observed in Caserta *et al*. (2002) for HPF; in our case this trend is more difficult to recognize because load balancing effects are superimposed. It follows from this that advantages in using SMS are ob-

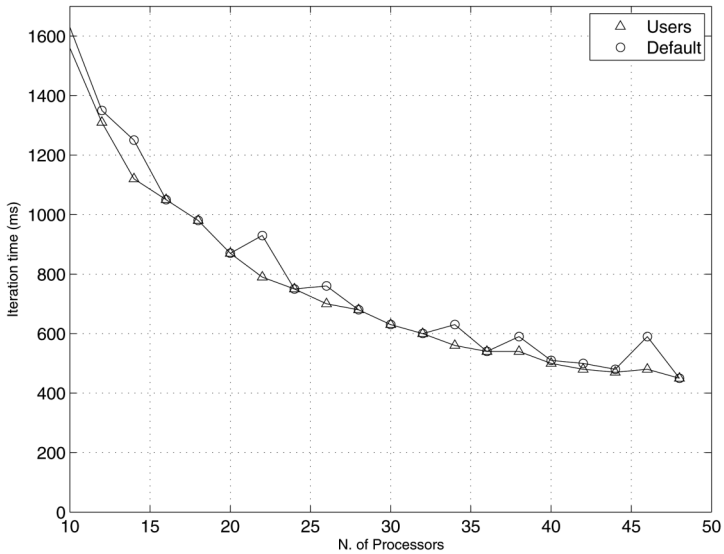

Fig. 3. Iteration time per time step (in milliseconds) vs. number of processors for the domain decomposition suggested by SMS (Default) and that one suggested by us (User). The plot starts from 10 processors in order to better magnify the differences between the two decompositions. Under 10 processors no difference takes place.

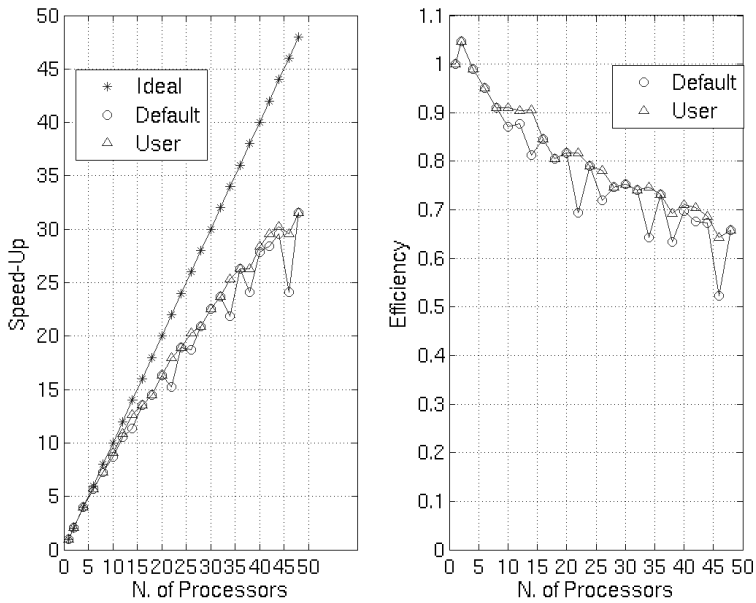

Fig. 4. Plots of Speed-up (left panel) and Efficiency (right panel) vs. number of processors for both the domain decompositions as planned and realized by SMS (Default) and as planned and realized by us (User). Comparison shows up to 48 processors, *i.e.*, using six 8-processor SP5 machines in a cluster. Straight line with asterisks represents the ideal Speed

tained when a cluster of multiprocessor machines are used in a distributed memory environment.

4. Conclusions

We investigated a simple and efficient way to parallelise a serial code, namely 3D seismological code able to simulate the interaction between seismic radiation and near-surface geological structures. The chosen case study code (Opršal and Zahradnik, 2002) and the case test geological site (Fäh *et al.,* 2006) are representative of a wide set of seismological problems dealing with the aforementioned dynamic interaction.

Our parallel approach is based on the SMS high-level tool. It allows the user to avoid all the low-level machinery needed to plan and realise the management of information exchanged among processors, apart from taking care of the domain decomposition problem as discussed in the previous paragraph. This does not mean algorithmic modifications in the serial code were not performed by the user to increase both the Speed-up and the Efficiency. On the contrary, these latter must be included in the profiling analysis applied to the serial code.

In the parallelisation of our case study, numerical tests indicate care must be taken in the domain decomposition process to reduce oscillations in both Speed-up and Efficiency, improving parallel performances. Generally speaking, in a shared memory environment communication is much faster because it takes place in the same cluster node. Nevertheless, we have seen less degradation of performances in adding multiprocessor machines. This phenomenon is clearer in the Efficiency plot (fig. 3 right panel) where the Efficiency remains almost constant when we add a new 8 processor machine, then in the Speed-up plot (fig. 3 left panel); this is mainly due to the load balance problem that is more evident in the Speed-up plot.

Such an effect was also observed in Caserta *et al*. (2002) with smaller size numerical problems, when a new multiprocessor machine is added under HPF compiler.

Moreover, porting from serial to parallel versions can be realized relatively simply for both environments. In fact, the porting is realized through the following simple steps:

– definition of the decomposition structure (see eq. 3.1),

– distribution of variables among processors (see eq. 3.2),

– domain decomposition creation (see eq. 3.3),

– creation of parallel region and exchange of variables among processors, *i.e*., message passing (see eq. 3.5).

Such SMS directives represent around 6% of serial code instructions.

On the other hand, SMS limitations are mainly:

– it can be used for Fortran codes only,

– it is intended for applications using structured Cartesian grids,

– maximum number of array indices that can be distributed is two. We point out that they must be the first two indices in the array. This constrain might require modifications of the serial code to have the indices in the required order. This is what we have been forced to do as in eq. 3.4.

No further limitations are arranged concerning, in particular, the numerical implementation, *i.e*., boundary elements, finite elements, spectral elements, finite differences, and so on. Under the above limitations, the simplicity and efficiency of the SMS tool are evident as it may allow broader and/or detailed physical models to be studied due to the use of parallel machines.

It is worth noting that the majority of 3D seismic codes that represent our target are written in Fortran and are based on the finite differences method on a Cartesian grid. On the other hand, no limitations so, we can conclude that for our purposes the previous limitations are not so severe.

Acknowledgements

We thank Piero Lanucara for useful discussions on SMS and HPF tools and for help in using the sophisticated cluster of parallel machines

at CASPUR computer centre. We also wish to thank Antonio Rovelli for help and criticism. Part of the research activities were carried out by A. Caserta, for his PhD thesis, in cooperation with the Charles University in Prague (Czech Republic) and in the frame of the projects GACR 205/07/0502 and MSM 0021620860. Partially supported by Japanese Society for Promotion of Science award FY2006/P06320.

REFERENCES

AKI, K. (1988): Local site effect on ground motion, in *Earthquake engineering and soil dynamics*, II: *Recent advances in ground-motion evaluation*, edited by J. LAWRENCE VON THUN, ASCE, 103-155.

AKI, K. (1993): Local site effects on weak and strong ground motions, *Tectono-physics*, **218**, 93-111.

BAO, H., J. BIELAK, O. GHATTAS, D.R. O'HALLARON, L.F. KALLIVOKAS, J.R. SHEWCHUK and J. XU (1996): *Earthquake Ground Motion Modeling on Parallel Computers*, Supercomputing '96, (Pittsburgh, Pennsylvania, November 1996).

BARD, P.-Y. (1995): Effects of surface geology on ground motion: recent results and remaining issues, in *Proceedings of the 10th European Conference on Earthquake Engineering*, (Vienna, Austria), 305-324.

BARD, P.-Y. (1998): Microtremor measurements: a tool for site effect estimation?, in *Proceeding of the Second International Symposium on the Effects of Surface Geology on Seismic Motion*, (Yokohama, Japan), pp. 1251-1279.

BIELAK, J., O. GHATTAS and H. BAO (1998): Ground motion modeling using 3D finite element methods, in *Proceedings of the 2nd International Symposium on the Effects of Surface Geology on the Seismic Motion*, (Yokohama, Japan), 121-133.

BONNEFOY-CLAUDET, S., F. COTTON and P.-Y. BARD (2006): The nature of noise wavefield and its applications for site effects studies. A literature review, *Earth Science Review,* **79**, 205-227.

CARA, F., A. ROVELLI, G. DI GIULIO, F. MARRA, T. BRAUN, G. CULTRERA, R. AZZARA and E. BOSCHI (2005): The role of site effects on the intensity anomaly of S. Giuliano di Puglia inferred from aftershocks of the Molise, Central Southern Italy, sequence, November 2002, *Bulletin of the Seismological Society of America*, **95** (4), 1457-1468.

CASERTA, A. and P. LANUCARA (2000): Computer animation as a tool to visualize effects of seismic wave propagation inside heterogeneous media, *Annali di Geofisica* **43** (1), 119-134.

CASERTA, A., F. BELLUCCI, G. CULTRERA, S. DONATI, F. MARRA, G. MELE, B. PALOMBO and A. ROVELLI (2000): Study of site effects in the area of Nocera Umbra (Central Italy) during the 1997 Umbria-Marche seismic sequence, *Journal of Seismology*, **4**, 555-565.

CASERTA, A., V. RUGGIERO and P. LANUCARA (2002): Numerical Modelling of dynamical interaction between seismic radiation and near-surface geological structures: a parallel approach, *Computer & Geoscience*, **28**, 1069-1077.

CULTRERA, G., A. ROVELLI, G. MELE, R. AZZARA, A. CASERTA and F. MARRA (2003): Azimuth dependent amplification of the weak and strong ground motions within a fault zone (Nocera Umbra, central Italy), *Journal of Geophysical Research*, **108**, 2156-2170.

DOWD, K. and C. SEVERANCE (1998): *High performance computing*, (O'Reilly, Cambridge- Kölon-Paris-Sebastopol-Tokyo), pp. 446.

DI GIULIO G., A. ROVELLI, F. CARA, R.M. AZZARA, F. MARRA, R. BASILI and A. CASERTA (2002): Long duration asynchronous ground motions in the Colforito plain, central Italy, observed on a two dimensional dense array, *Journal of Geophysical Research*, **108**, 2486-2498.

FÄH, D., C. IODICE and P. SUHADOLC (1993): A new method for realistic estimation of seismic ground motion in megacities: the case of Rome, *Earthquake Spectra,* **4,** 643-668.

FÄH, D., S. STEIMEN, I. OPRŠAL, J. RIPPERGER, J. WÖSSNER, R. SCHATZMANN, P. KÄSTLI, I. SPOTTKE and P. HUGGENBERGER (2006): The earthquake of 250 A.D. in Augusta Raurica, A real event with a 3D site effect?, *Journal of Seismology*, **10** (4), 459-477.

FOSTER, I. (1995): Designing and Building Parallel Programs: Concept and Tools for Parallel Software Engineering. *Addison-Wesley, Reading*, MA, pp. 381.

GOVETT, M., L. HART, T. HENDERSON, J. MIDDLECOFF and D. SCHAFFER (2003): The scalable modeling system: directive-based code parallelization for distributed and shared memory computers, *Parallel Computing*, **29** (8), 995-1020.

GRAVES, R.W. (1998): Three-dimensional computer simulations of realistic earthquake ground motions in regions of deep sedimentary basin. in *Proceedings 2nd International Symposium on the effects of surface geology on seismic motion, Yokohama, Japan*, pp. 103-120.

HENNESSY, J. and D. PATTERSON (2006): *Computer Architecture: A Quantitative Approach* [With CD-ROM], (Morgan Kaufmann Publishers, San Francisco, CA), pp. 672.

MITCHEL, A.R. and D.F. GRIFFITHS (1980): *The Finite Difference Method in Partial Differential Equations*, (Wiley, New York), pp. 272.

MOCZO, P., J. KRISTEK and E. BYSTRICKY (2001): Efficiency and optimization of the 3-D finite-difference modeling of seismic ground motion, *Journal of Computational Acoustics,* **9** (2), 593-609.

MOCZO, P., J. KRISTEK, M. GALIS, P. PAZAK and M. BALAZOVJECH (2007): The Finite-Difference and Finite-Element Modeling of Seismic Wave Propagation and Earthquake Motion, *Acta Physica Slovaca*, **57** (2), 177-406.

NICHOLS, B., D. BUTTLAR and J. FARRELL (1996): *Pthreads Programming*, (1st ed. O'Reilly, Sebastopol, CA), pp. 283.

OLSEN, K.B. and R.J. ARCHULETA (1996): Three-dimensional simulations of earthquakes on the Los Angeles fault system, *Bulletin of the Seismological Society of America*, **86**, 575-596.

OLSEN, K.B., A. AKINCI, A. ROVELLI, F. MARRA and L. MALAGNINI (2006): 3D ground-motion estimation in Rome, Italy, *Bulletin of the Seisological Society of America*, **96**, 133- 146.

OPRŠAL, I. and J. ZAHRADNIK (1999) Elastic finite-difference method for irregular grids. *Geophysics,* **64** (1), 240-250.

OPRŠAL, I. and J. ZAHRADNIK (2002): Three-dimensional finite difference method and hybrid modeling of earthquake ground motion, *Journal Of Gephysisical Research*, **107**, B8, 2161.

OPRŠAL, I., J. ZAHRADNÍK, A, SERPETSIDAKI and G.-A. TSELENITS (2004): 3D hybrid simulation of the source and site effects during the 1999 Athens earthquake, in *Proc. Of 13th World Conference on Earthquake Engineering,* (Vancouver, B.C., Canada, August 1-6, 2004), Paper No. 3337, pp. 15.

OPRŠAL, I., D. FÄH, M. MAI and D. GIARDINI (2005): Deterministic earthquake scenario for the Basel area: Simulating strong motions and site effects for Basel, Switzerland, *J. Geophys. Res.,* **110** (B4), 19, B04305, doi:10.1029/2004JB003188.

OPRŠAL, I. and D. FÄH (2007): 1D vs. 3D strong ground motion hybrid modelling of site, and pronounced topography effects at Augusta Raurica, Switzerland - earthquakes or battles?, in *Proceedings of 4th International Conference on Earthquake Geotechnical Engineering June 25-28, 2007, Greece*, Paper No. 1416, pp. 12.

OPRŠAL, I., C. MATYSKA and K. IRIKURA (2009): The source-box wave propagation hybrid methods: general formulation and implementation, *Geophysical Journal International*, **176** (2), 555-564, doi: 10.1111/j.1365-246X.2008.03986x.

OPRŠAL, G.F., F. ROMANELLI and F. VACCARI (2001): Seismic wave propagation in laterally heterogeneous anelastic media: theory and applications to seismic zonation, *Advances in Geophysics*, **43**, 1-95.

PITARKA, A. (1999): 3D elastic finite difference modeling of seismic wave propagation using staggered grid non-uniform spacing, *Bulletin of the Seismological Society of America,* **89**, 54-68.

ROVELLI, A., A. CASERTA, L. MALAGNINI and F. MARRA (1994): Assessment of potential strong ground motions in the city of Rome, *Annali di Geofisica,* **37** (6), 1745-1769.

SÁNCEZ-SESMA, F.J. and F. LUZON (1995): Seismic response of three-dimensional valleys for incident P, S, and Rayleigh waves, *Bulletin. of the Seismological Society of America*, **85**, 269-284.

SORENSEN, M.B., I. OPRŠAL, S. BONNEFOY-CLAUDET, K. ATAKAN, P.M. MAI, N. PULIDO and C. YALCINER (2006): Local site effects in Ataköy, Istanbul, Turkey, due to a future large earthquake in the Marmara Sea, *Geophysical Journal International*, **167**, 1413-1424, doi: 10.1111/j.1365-246X.2006.03204.x.

ZAHRADNIK, J., P. MOCZO and F. HRON (1993): Testing four elastic finite-difference schemes for behaviour at discontinuities, *Bulletin of the Seismological Society of America,* **83**, 107-129.

ZAHRADNIK, J. and P. MOCZO (1996): Hybrid seismic modeling based on discretewave number and finite difference methods, *Pure and Applied Geophysics*, **148**, 21-38.

(received, February 25, 2009;
accepted, July 14, 2009)