Acta Universitaria
ISSN: 0188-6266
actauniversitaria@ugto.mx
Universidad de Guanajuato
México

Constantin Manea, Vlad; Manea, Marina; Pomeran, Mihai; Besutiu, Lucian; Zlagnean, Luminita

A parallelized particle tracing code for CFD simulations in Earth Sciences

Acta Universitaria, vol. 22, núm. 5, julio-agosto, 2012, pp. 19-26
Universidad de Guanajuato
Guanajuato, México

A parallelized particle tracing code for CFD simulations in Earth Sciences

Un código paralelizado de trazadores para las simulaciones CFD en Ciencias de la Tierra

Vlad Constantin Manea*, **, Marina Manea*, **, Mihai Pomeran**, Lucian Besutiu**, Luminita Zlagnean**

ABSTRACT

The problem of convective flows in a highly viscous fluid represents a common research direction in Earth Sciences. In order to trace the convective motion of the fluid material, a source of passive particles (or tracers) that flow at a local convection velocity and do not affect the pattern of flow is commonly used. It is presented a parallelized tracer code that uses passive and weightless particles with their position computed from their displacement during a small time interval at the velocity of flow -previously calculated for a given point in space and time. The tracer code is integrated in the open source package CitcomS, which is widely used in the solid Earth community (www.geodynamics.org). It is benchmarked the tracer code on the state-of-the-art CyberDyn parallel machine, a High Performance Computing (HPC) Cluster with 1 344 computing cores available at the Institute of Geodynamics of the Romanian Academy.

RESUMEN

El problema de los flujos convectivos en un fluido altamente viscoso representa una dirección común de investigación en Ciencias de la Tierra. Para rastrear el movimiento convectivo de un material fluido, se utiliza una fuente pasiva de partículas (o trazadores) que fluyen en el campo local de velocidades y no afectan el flujo. Es presentado un código paralelizado de trazadores que utiliza partículas pasivas e ingrávidas cuyas posiciones están calculadas según su desplazamiento durante un intervalo de tiempo pequeño -para una velocidad de flujo previamente calculada. El código de trazadores está integrado en el paquete de código abierto CitcomS, que se utiliza ampliamente en la comunidad de las Ciencias de la Tierra (www.geodynamics.org). Es comparado el código de trazadores en una computadora de alto rendimiento HPCC Cyberdyn, un cluster de alto rendimiento con 1 344 nudos de cálculo, disponible en el Instituto de Geodinámica de la Academia Rumana.

INTRODUCTION

In recent years, the power of numeric flow simulation as a tool for exploring science in general -and Earth Sciences in particular- has reached new levels [1-3]. This research direction represents a promising future intended to provide useful alternatives to laboratory experiments (where realistic input parameters into Earth Sciences simulations are difficult, if not impossible, to implement [4-7]). The new development in High Performance Computing (HPC), in terms of hardware and networking with sufficient bandwidth and low latency [8-11], as well as in term of advanced algorithms capable of simulating complex problems in Earth Sciences [12-16], makes possible today to accurately simulate rock flow inside the Earth. In this paper, it is presented a parallelized code for tracking material flow in specific and different numeric simulations in Earth Sciences. Also, the purpose of this research paper is to benchmark a parallelized tracer-particle code, which is used extensively within material advection models specific for Earth Sciences. Material advection inside the Earth -the motion of high viscous fluid material- is a fundamental physical process and it is the core of many simulation codes: ConMan [17, 18], CitcomS [16, 19, 20], I2Elvis [21, 22]

Recibido: 27 de marzo de 2012
Aceptado: 1 de junio de 2012

Keywords:
Fluid simulation; tracers; parallelization; HPC; Earth Sciences.

Palabras clave:
Simulación de fluidos; trazadores; paralelización; HPC; Ciencias de la Tierra.

*Computational Geodynamics Laboratory. Centro de Geociencias. Universidad Nacional Autónoma de México. Mexico. Phone. (442) 238-1104, ext. 133; fax (442) 5623-4129. E-mail: vlad@geociencias.unam.mx
**Solid Earth Dynamics Department. Institute of Geodynamics of the Romanian Academy. Romania.

and Gale [15]. It is considered a couple of 3D FEM numeric simulations involving incompressible flows with time-dependent behavior inside the Earth's mantle. The parallelized tracer code presented in this paper is integrated in the open source package CitcomS [16, 19, 20], which is widely used in the solid Earth community for simulating large-scale thermo-chemical processes inside the Earth (www.geodynamics.org).

Currently, the parallel computations are carried out on the CyberDyn HPCC, a 1 344 computing cores machine at the Institute of Geodynamics of the Romanian Academy.

MATERIALS AND METHODS

Particle tracing in the context of earth sciences

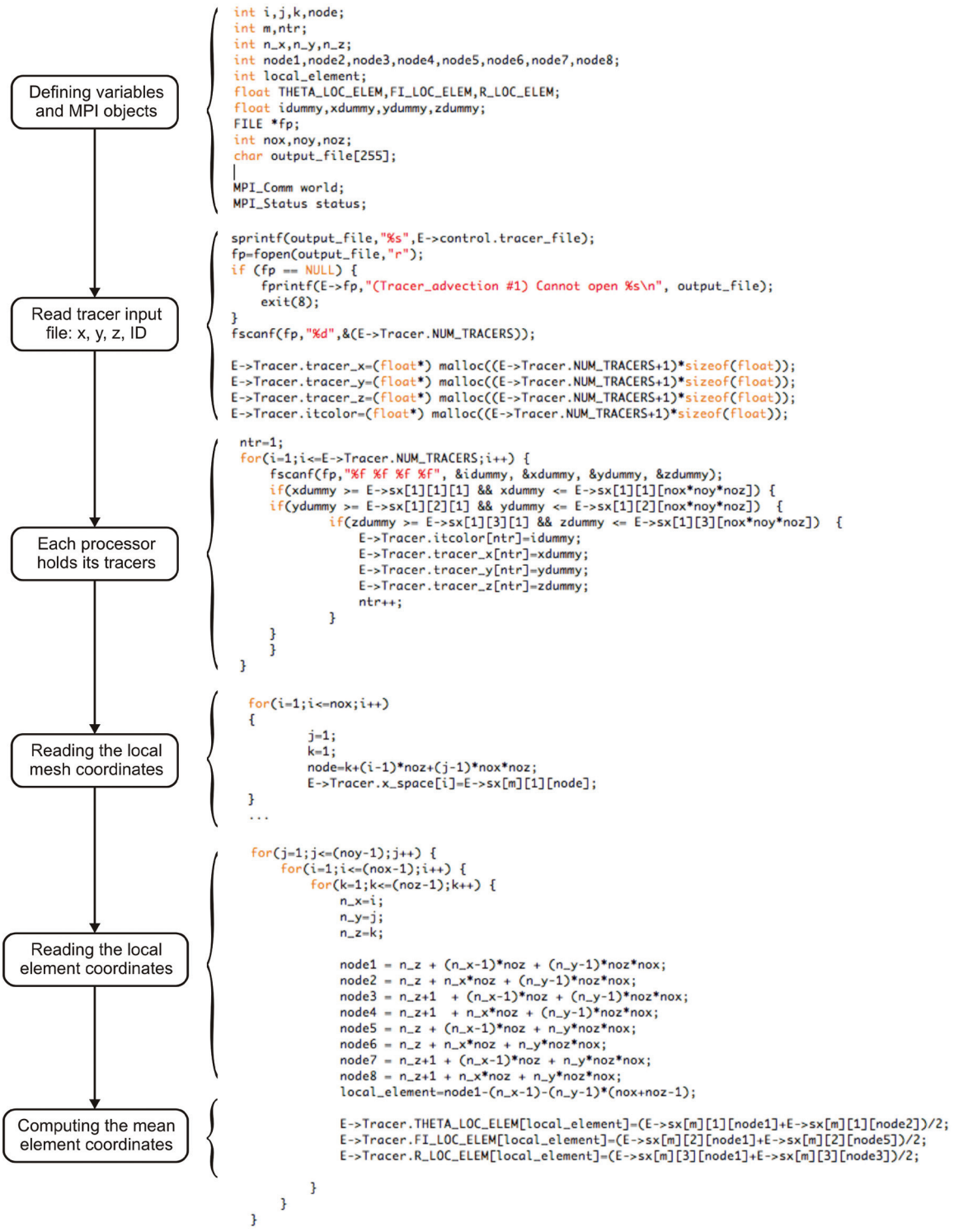
Massless particle tracing is an important step towards acquiring a better insight of the complex phenomena that take place inside the Earth [2, 23]. This makes possible to analyze different factors that are believed to play a key role in the generation of volcanism or in the origin of seismicity deep inside the Earth [24-27]. These massless particles, or tracers, are advected throughout the computational domain simultaneous with the solution evolution; also, they are widely used to sample relevant data and to visualize the material movement through time and space [2]. An example of a particular interest in Earth Sciences simulations is the residence time, a quantity that is attached to particles involved in studying the mantle flow. It basically measures the time that particles remain in their traveling throughout a certain region of analysis, and help to characterize stagnation regions or fast convecting regions [23]. Sometimes, these particles have attached different properties as chemical composition, or they might represent different materials being advected inside the Earth (for example, continental crust or oceanic crust). They also help to understand complex patterns of mantle flow inside the Earth in specific regions, such as beneath an active volcanic arc -a place commonly assumed as magma generator [2, 23, 26]. This is actually the context that motivates the development of a particle-tracing tool: a research instrument which is not frequently encountered in open source parallelized software. It was developed a tracer code that is coupled with a mature and robust FEM code for CFD simulation inside the Earth, CitcomS [16, 19, 20]. In Earth Sciences, it is common to deal with 4D models (3D + time) with large amount of data resulting from the numeric simulations, and in order to accurately track flow changes; the number of these particles can vary from several tens of thousand to several hundreds of million or even billions. Working with such large numbers tracer codes should be efficient, which implies developing parallel implementations [24, 26].

Particle tracing algorithm

In this paper, it is tested a tracer code in a couple of 3D simulations of incompressible fluid flow in a time-varying domain. The work presents the development of an efficient computational implementation to perform particle tracing, where the input data is the velocity field coming from numerical simulations of mantle flow performed with CitcomS. In our numeric implementation, each point position will be passively advected by the velocity field using the mid-point method. The tracer code is written in C and, in order take advantage of the large number of computing cores available for testing on HPCC CyberDyn and Horus, we use MPI (Message Passing Interface) for parallelization [16, 19, 20]. MPI is a standard message passing system that offers the possibility to write and run applications on parallel computers. The MPI standard was completed in 1994 (subsequently MPI1.2 and MPI2 standards were defined; see http://www.mcs.anl.gov/mpi). Nowadays, many vendors are supporting the standard and there are several public domain implementations of the MPI. In this paper, it is used the MPICH implementation from Argonne National Laboratory [28].

The logical scheme of our tracer code, with sample code lines, is presented and discussed in details below. The code is divided in three main parts. The first important code block is used to read the tracers initial conditions and to split the data among processors (figure 1). Here it is used only two MPI structures, namely MPI_Comm and MPI_Status. MPI_Status structure is used by the message receiving functions to return data about a message, and MPI_Comm is the default communicator. Basically, two systems of coordinates are considered: one global -for the whole computational domain- and several local coordinates systems -according to the number of processors. In the present version, the input datasets of this system included information about the initial position of tracers (x, y, z), the tracer group identification (or particle-id), both specified in the global coordinate system. Domain decomposition was used in order to assign processors in each computational domain direction. In other words, to each geometrical direction corresponds a certain number of processors, or computing cores.

The second step in the first part of the code is to split the tracers according with the processor map (each core holding a certain number of tracers). CitcomS uses brick elements and, therefore, each FEM holds eight nodes. This information is applied to compute the center of each element in the local coordinate system.




```
int i,j,k,node;
int m,ntr;
int n_x,n_y,n_z;
int node1,node2,node3,node4,node5,node6,node7,node8;
int local_element;
float THETA_LOC_ELEM,FI_LOC_ELEM,R_LOC_ELEM;
float idummy,xdummy,ydummy,zdummy;
FILE *fp;
int nox,noy,noz;
char output_file[255];

MPI_Comm world;
MPI_Status status;
```

```
sprintf(output_file,"%s",E->control.tracer_file);
fp=fopen(output_file,"r");
if (fp == NULL) {
    fprintf(E->fp,"(Tracer_advection #1) Cannot open %s\n", output_file);
    exit(8);
}
fscanf(fp,"%d",&(E->Tracer.NUM_TRACERS));

E->Tracer.tracer_x=(float*) malloc((E->Tracer.NUM_TRACERS+1)*sizeof(float));
E->Tracer.tracer_y=(float*) malloc((E->Tracer.NUM_TRACERS+1)*sizeof(float));
E->Tracer.tracer_z=(float*) malloc((E->Tracer.NUM_TRACERS+1)*sizeof(float));
E->Tracer.itcolor=(float*) malloc((E->Tracer.NUM_TRACERS+1)*sizeof(float));
```

```
ntr=1;
for(i=1;i<=E->Tracer.NUM_TRACERS;i++) {
    fscanf(fp,"%f %f %f %f", &idummy, &xdummy, &ydummy, &zdummy);
    if(xdummy >= E->sx[1][1][1] && xdummy <= E->sx[1][1][nox*noy*noz]) {
    if(ydummy >= E->sx[1][2][1] && ydummy <= E->sx[1][2][nox*noy*noz])  {
            if(zdummy >= E->sx[1][3][1] && zdummy <= E->sx[1][3][nox*noy*noz])  {
                E->Tracer.itcolor[ntr]=idummy;
                E->Tracer.tracer_x[ntr]=xdummy;
                E->Tracer.tracer_y[ntr]=ydummy;
                E->Tracer.tracer_z[ntr]=zdummy;
                ntr++;
            }
    }
    }
}
```

```
for(i=1;i<=nox;i++)
{
        j=1;
        k=1;
        node=k+(i-1)*noz+(j-1)*nox*noz;
        E->Tracer.x_space[i]=E->sx[m][1][node];
}
...
```

```
for(j=1;j<=(noy-1);j++) {
    for(i=1;i<=(nox-1);i++) {
        for(k=1;k<=(noz-1);k++) {
            n_x=i;
            n_y=j;
            n_z=k;

            node1 = n_z + (n_x-1)*noz + (n_y-1)*noz*nox;
            node2 = n_z + n_x*noz + (n_y-1)*noz*nox;
            node3 = n_z+1  + (n_x-1)*noz + (n_y-1)*noz*nox;
            node4 = n_z+1  + n_x*noz + (n_y-1)*noz*nox;
            node5 = n_z + (n_x-1)*noz + n_y*noz*nox;
            node6 = n_z + n_x*noz + n_y*noz*nox;
            node7 = n_z+1 + (n_x-1)*noz + n_y*noz*nox;
            node8 = n_z+1 + n_x*noz + n_y*noz*nox;
            local_element=node1-(n_x-1)-(n_y-1)*(nox+noz-1);

            E->Tracer.THETA_LOC_ELEM[local_element]=(E->sx[m][1][node1]+E->sx[m][1][node2])/2;
            E->Tracer.FI_LOC_ELEM[local_element]=(E->sx[m][2][node1]+E->sx[m][2][node5])/2;
            E->Tracer.R_LOC_ELEM[local_element]=(E->sx[m][3][node1]+E->sx[m][3][node3])/2;

        }
    }
}
```

Figure 1. An excerpt from the parallelized particle tracing code, where each processor holds its number of tracers and the global mesh is split in a local mesh according to the processor map.

In the second part of this code, the tracers are advected within each computing core. An additional MPI function was used, MPI_Request, which is intended to send and receive requests. Here the velocity array from CitcomS is imported into the tracer code, and is used to advect tracers using the mid-point method with two iterations (second order accuracy). Basically, for each processor, all the tracers inside were iterated and the finite element that actually contains the tracers was found. In order to advect the traces, it was needed to calculate the velocity of the tracer located inside a finite element. For this purpose, the velocities were interpolated from the eight element corners to the tracer position using volumetric weighting functions (figure 2).

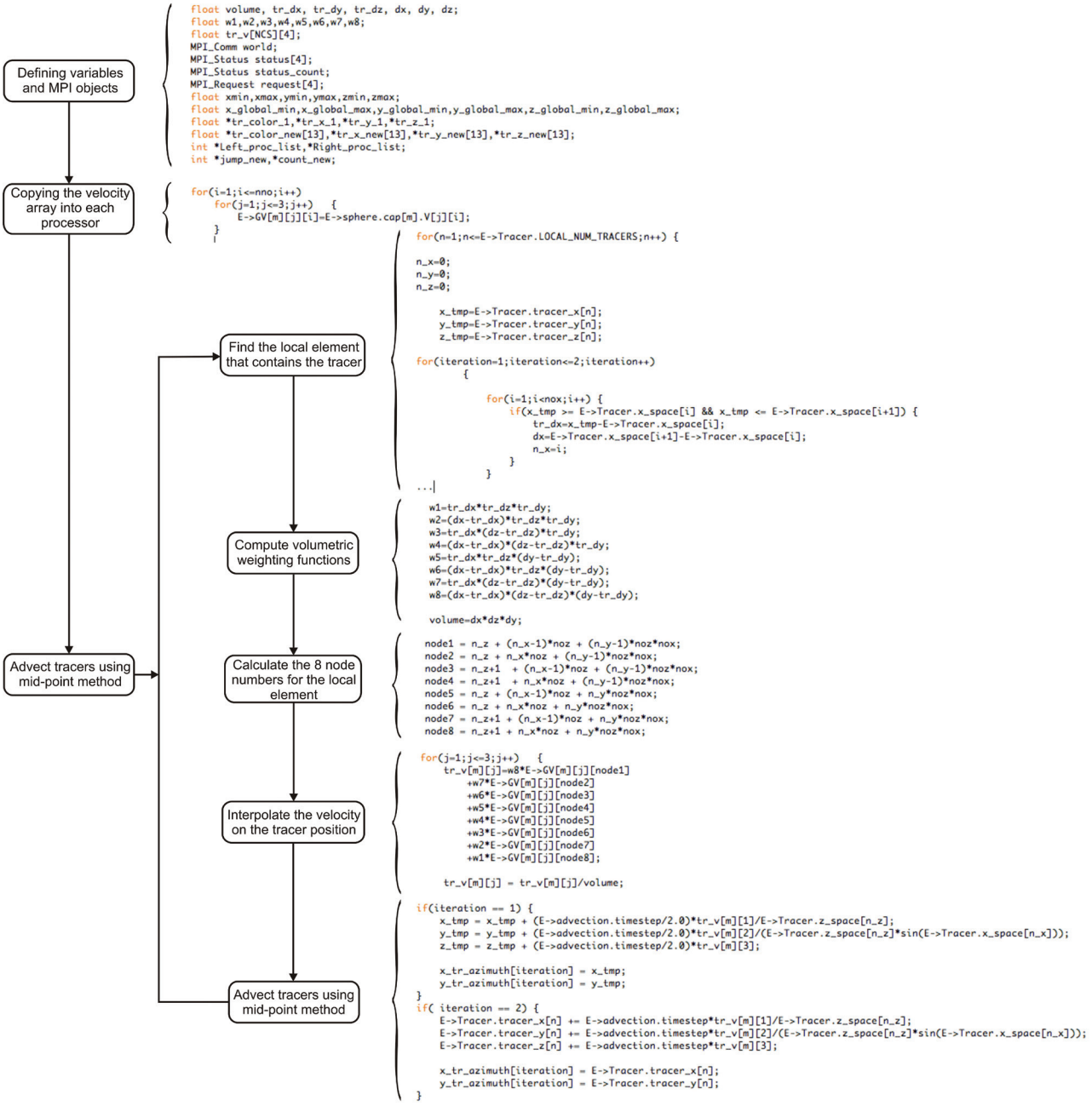


Figure 2. An excerpt from the parallelized particle tracing code, where the tracers are advected using mid-point method (2nd accuracy order).

The last part of the tracer code is dedicated entirely to code parallelization. It starts by defining the processor map. Essentially, the goal was to find out which are the processor neighbors for every processor used to split the computational domain. The general rule is to find which processor is to the left/right, front/back and up/down to the current processor (figure 3).

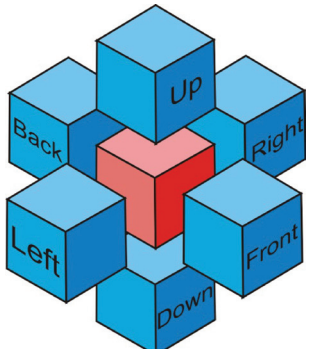


Figure 3. Processor map used for tracer code parallelization using MPI. Cubes represent computing cores. Red cube depicts the current processor and the position of the six neighbor processors.

They were used four MPI objects like MPI_Irecv, MPI_Send, MPI_Waitall and MPI_Get_count. MPI_Irecv starts a nonblocking receive, which means that the system may start writing data into the receive buffer (figure 4a and 4b). It actually receives a number of tracers that cross the boundary between two adjacent processors. The number of tracers that remained in the sender processor (MPI_Send) needs to be compacted, and the number of tracers in the receiving processor needs to be expanded. At each time step, every processor needs to update its number of tracers, according to the number of tracers received and sent among processors. The structure MPI_Waitall imposes restriction until all communications among processors for one time step are completed. With MPI_Get_count, it is possible to track accurately the number of tracers transmitted from one processor to another.

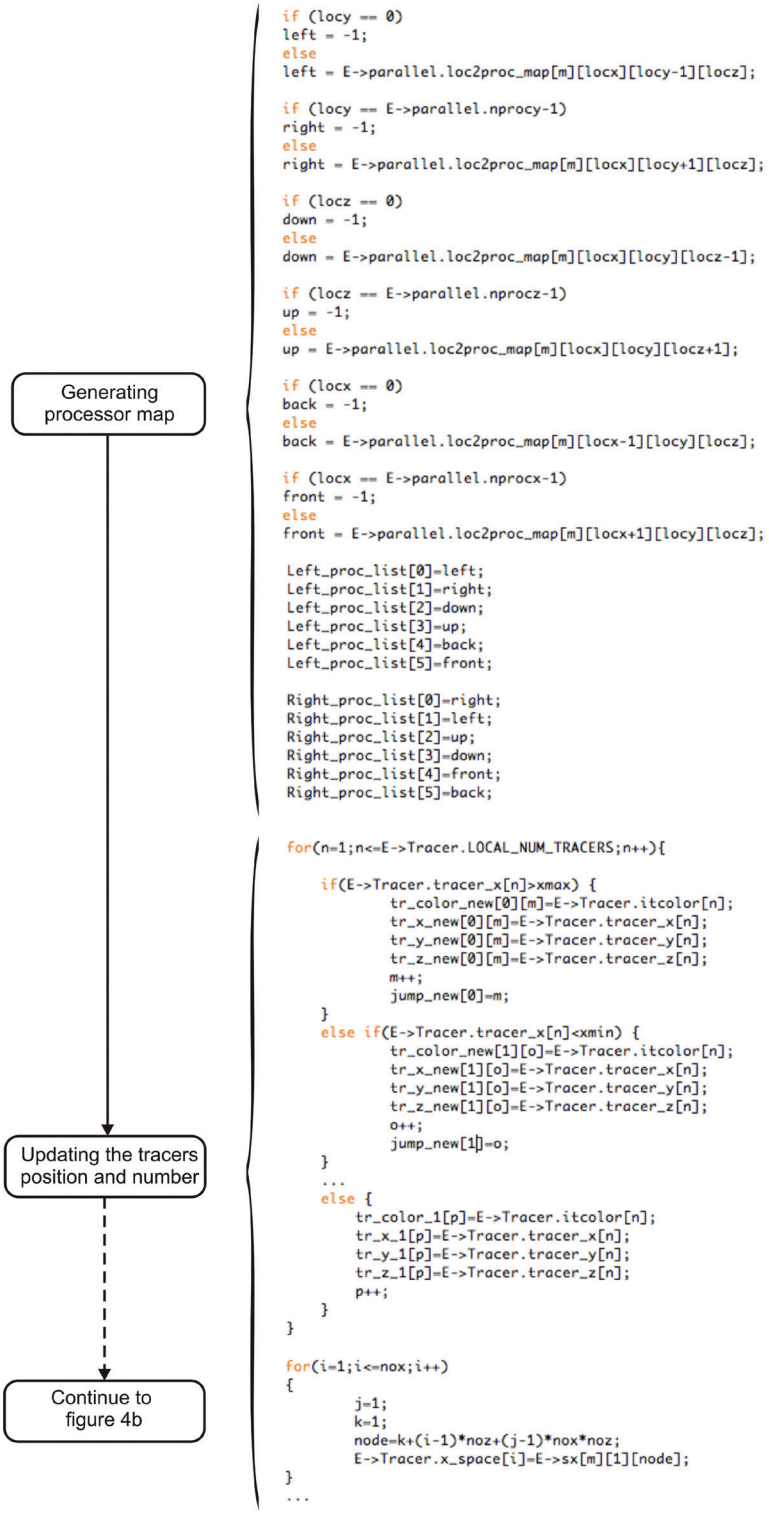
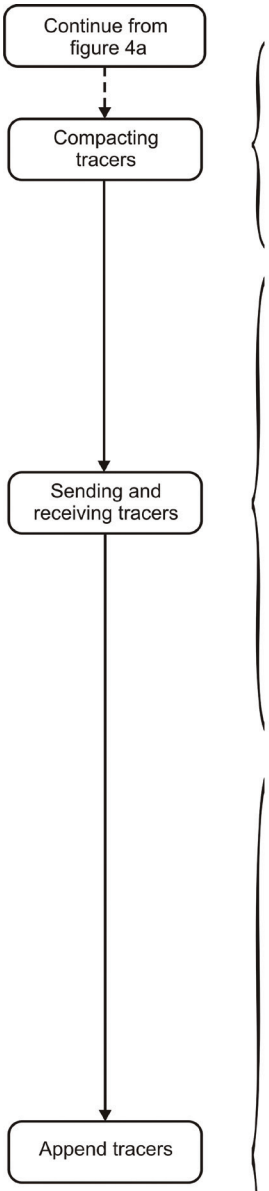


Figure 4a. Logical scheme and the corresponding sample of parallelized code, where the tracers are sent and received between processors using MPI functions.




```
Previous_num_tracers=E->Tracer.LOCAL_NUM_TRACERS;
Current_num_tracers=Previous_num_tracers-jump_new[0]-jump_new[1]-
-jump_new[2]-jump_new[3]-jump_new[4]-jump_new[5];

for(p=1;p<=Current_num_tracers;p++){
E->Tracer.itcolor[p]=tr_color_1[p-1];
E->Tracer.tracer_x[p]=tr_x_1[p-1];
E->Tracer.tracer_y[p]=tr_y_1[p-1];
E->Tracer.tracer_z[p]=tr_z_1[p-1];
}
```

```
for(i=0;i<=5;i++){

j=jj[i];

    if (Left_proc_list[i] >= 0) {
        proc=Left_proc_list[i];
        MPI_Irecv(tr_color_new[i+6], E->Tracer.NUM_TRACERS, MPI_FLOAT, proc, 11+i, world, &request[0]);
        MPI_Irecv(tr_x_new[i+6], E->Tracer.NUM_TRACERS, MPI_FLOAT, proc, 12+i, world, &request[1]);
        MPI_Irecv(tr_y_new[i+6], E->Tracer.NUM_TRACERS, MPI_FLOAT, proc, 13+i, world, &request[2]);
        MPI_Irecv(tr_z_new[i+6], E->Tracer.NUM_TRACERS, MPI_FLOAT, proc, 14+i, world, &request[3]);
    }

    if (Right_proc_list[i] >= 0) {
        proc=Right_proc_list[i];
        MPI_Send(tr_color_new[i], jump_new[i], MPI_FLOAT, proc, 11+i, world);
        MPI_Send(tr_x_new[i], jump_new[i], MPI_FLOAT, proc, 12+i, world);
        MPI_Send(tr_y_new[i], jump_new[i], MPI_FLOAT, proc, 13+i, world);
        MPI_Send(tr_z_new[i], jump_new[i], MPI_FLOAT, proc, 14+i, world);
    }

    if (Left_proc_list[i] >= 0) {
        MPI_Waitall(4, request, status);
        status_count = status[0];
        MPI_Get_count(&status_count, MPI_FLOAT, &count_new[i]);
    }
```

```
temp_tracers=temp_tracers+count_new[i]-jump_new[i];
E->Tracer.LOCAL_NUM_TRACERS=Previous_num_tracers+temp_tracers;

 if(i <= 1){
        for(n=Current_num_tracers+count_new[j];
            n<=Current_num_tracers+count_new[i]+count_new[j]-1;n++) {
            m=Current_num_tracers+count_new[j];
            E->Tracer.itcolor[n+1]=tr_color_new[i+6][n-m];
            E->Tracer.tracer_x[n+1]=tr_x_new[i+6][n-m];
            E->Tracer.tracer_y[n+1]=tr_y_new[i+6][n-m];
            E->Tracer.tracer_z[n+1]=tr_z_new[i+6][n-m];

        }
 }


 else if (i <= 3) {
        for(n=Current_num_tracers+count_new[0]+count_new[1]+count_new[j];
            n<=Current_num_tracers+count_new[0]+count_new[1]+count_new[i]+count_new[j]-1;n++) {
            m=Current_num_tracers+count_new[0]+count_new[1]+count_new[j];
            E->Tracer.itcolor[n+1]=tr_color_new[i+6][n-m];
            E->Tracer.tracer_x[n+1]=tr_x_new[i+6][n-m];
            E->Tracer.tracer_y[n+1]=tr_y_new[i+6][n-m];
            E->Tracer.tracer_z[n+1]=tr_z_new[i+6][n-m];

        }
 }

 else  {
        for(n=Current_num_tracers+count_new[0]+count_new[1]+count_new[2]+count_new[3]+count_new[j];
            n<=E->Tracer.LOCAL_NUM_TRACERS-1;n++) {
            m=Current_num_tracers+count_new[0]+count_new[1]+count_new[2]+count_new[3]+count_new[j];
            E->Tracer.itcolor[n+1]=tr_color_new[i+6][n-m];
            E->Tracer.tracer_x[n+1]=tr_x_new[i+6][n-m];
            E->Tracer.tracer_y[n+1]=tr_y_new[i+6][n-m];
            E->Tracer.tracer_z[n+1]=tr_z_new[i+6][n-m];

        }
 }
```

Figure 4b. Logical scheme and the corresponding sample of parallelized code, where the tracers are sent and received between processors using MPI functions.

RESULTS

Code benchmark

The benchmark tests are performed using a series of 3D geodynamic settings, where various layers of tracers at the bottom limit in the models were introduced. Two simulations with different temperature anomalies imposed at the bottom of the model were performed. The models were applied in a 3D regional domain within the Earth's mantle with different types of thermal anomalies specified as initial conditions. The tests were conducted on the CyberDyn HPC machine with 1 344 computing cores, Infiniband QDR interconnect and Rmax ~12 TFlops [29]. To submit jobs, it was used Sun Grid Engine (SGE) configured as "round-robin"; each compute node was filled sequentially. The focus was on how the number of tracers and computing cores influence the wall time. A number of 0 (no tracers), 104, 105, 106 and 107 tracers, and from 4 to 1 024 computing cores were used. The results show that the optimum number of computing cores for the simulation with no tracers is between 32 and 64. Using a larger number of cores (128 or even 1024) results in a significant increasing in computing time, and the time increases more when a larger number of cores is used. When an increasing number of tracers (from 104 to 107) is included, it is obtained the same optimum number (minimum wall time) of computing cores of 32-64.

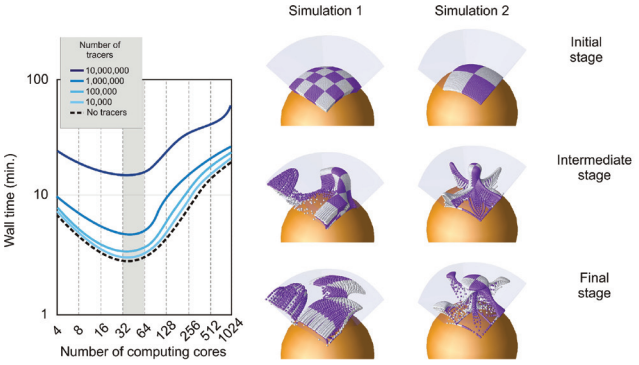


Figure 5. Benchmark result of the parallelized tracer code on the HPCC CyberDyn for thermal convection FEM simulations in a regional model, which shows the influence of number of tracers on wall time as a function of number of processors. To the right, it is shown the tracers evolution for two simulations at three different evolutionary stages of the simulation (visualization performed with open source software OpenDX). Orange sphere at the initial stage represents the Earth's iron core.

CONCLUSIONS

As fluid dynamics computations in Earth Sciences grow in size, parallel particle tracing algorithms are needed to analyze in details the resulting mantle flows. Moreover, a tracer particle capability is a valuable addition to all modern flow simulation codes. However, efficiently parallelizing algorithms on large HPCC systems, whose communication requirements control run time, represents an open challenge for scientists. In this paper, the technique to efficiently parallelize a tracer code using MPI is presented. The performance tests show that the tracers code scales well with the number of cores and, although a large number of particle was used (up to 107 tracers) -which involves a massive amount of communication among computing nodes-, no wall time penalty was observed (figure 5). This is an indication that the parallelized tracer code is integrated well in the main CitcomS parallelized code, and that the high-speed QDR network fabric (40 Gb/s) used by the HPCC CyberDyn does not represent a bottleneck for the MPI traffic involved in tracers advection.

ACKNOWLEDGMENTS

The authors gratefully acknowledge the use of the HPCC CyberDyn for all numeric simulations at the Institute of Geodynamic of the Romanian Academy. This research has been conducted through the CYBERDYN project (POS CCE O212_ID 593).

REFERENCES

[1] Gerya, T. (2009). *Introduction to Numerical Geodynamic Modelling.* Cambridge University Press. Cambridge, UK.

[2] Ismail-Zadeh, A. and Tackley, P. (2010). *Computational Methods for Geodynamics.* Cambridge University Press.

[3] Kaus, B. J. P., Schmid, D. W. and Schmalholz, S. M. (2011). Numerical modelling in structural geology. *Journal of Structural Geology,* submitted. Available online: http://www.gfd.geophys.ethz.ch/~kausb/kss10_Review_MS_JSG.pdf. Consulted: 2011.

[4] van Keken, P. (1997). Evolution of starting mantle plumes: a comparison between numerical and laboratory models. *Earth and Planetary Science Letters* 148(1-2): pp. 1-11.

[5] Burg, J.-P. and Podladchikov, Y. (1998). Lithospheric scale folding: numerical modelling and application to the Himalayan syntaxes. *International Journal of Earth Sciences* 88(2): pp. 190-200.

[6] Gerya, T. and Yuen, D. A. (2007). Robust characteristics method for modelling multiphase visco-elasto-plastic thermo-mechanical problems. *Physics of the Earth and Planetary Interiors* 163(1-4): pp. 83-105.

[7] Boutelier, D., Chemenda, A. and Burg, J.-P. (2003). Subduction versus accretion of intra-oceanic volcanic arcs: insight from thermo-mechanical analoque experiments. *Earth and Planetary Science Letters* 212(1-2): pp. 31-45.

[8] Becker, C., Buijssen, S. and Turek, S. (2008). *FEAST: Development of HPC Technologies for FEM Applications.* Part 5. High Performance Computing in Science and Engineering `07. Springer Berlin-Heidelberg: pp. 503-516.

[9] Zhang, Y., Sun, J., Yuan, G. and Zhang, L. (2010). Perspectives of China's HPC system development: a view from the 2009 China HPC TOP100list. *Frontiers of Computer Science in China* 4(4): pp. 437-444.

[10] Xie, X., Fang, X., Hu, S. and Wu, D. (2010). Evolution of supercomputers. *Frontiers of Computer Science in China* 4(4): pp. 428-436.

[11] Gropp, W. D., Kaushik, D. K., Keyes, D. E. and Smith, B. F. (2001). High Performance Parallel Implicit CFD. *Parallel Computing* 27: pp. 337–362.

[12] Moresi, L. N., Dufour, F. and Muhlhaus, H.-B. (2003). A lagrangian integration point finite element method for large deformation modeling of viscoelastic geomaterials. *Journal of Computational Physics* 184: pp. 476-497.

[13] Choi, E., Gurnis, M., Kientz, S. and Stark, C. (2010). *SNAC User Manual. Version 1.2.* Available online: www.geodynamics.org/cig/software/snac/snac.pdf. Consulted: 2011.

[14] Choi, E., Lavier, L. and Gurnis, M. (2008). Thermomechanics of mid-ocean ridge segmentation. *Physics of the Earth and Planetary Interiors* 171: pp. 374-386.

[15] Landry, W., Hodkinson, L. and Kientz, S. (2011). *GALE User Manual. Version 2.0.0. Availabl*e online: www.geodynamics.org/cig/software/gale/gale.pdf. Consulted: 2011.

[16] Tan, E., Gurnis, M., Armendariz, L., Strand, L. and Kientz, S. (2012). *CitcomS User Manual. Version 3.2.0.* Available online: www.geodynamics.org/cig/software/citcoms/citcoms.pdf. Consulted: 2011.

[17] King, S. (2008). *ConMan User Manual. Version 2.0*. Available online: www.geodynamics.org/cig/software/conman/conman.pdf. Consulted: 2011.

[18] King, S. D., Raefsky, A. and Hager, B. H. (1990). ConMan: Vectorizing a finite element code for incompressible two-dimensional convection in the Earth's mantle. *Physics of the Earth and Planetary Interiors* 59: pp. 195-208.

[19] Zhong, S., Zuber, M. T., Moresi, L. N. and Gurnis, M. (2000). The role of temperature-dependent viscosity and surface plates in spherical shell models of mantle convection. *Journal of Geophysical Research* 105: pp. 11 063-11 082.

[20] Tan, E., Choi, E., Thoutireddy, P., Gurnis, M. and Aivazis, M. (2006). GeoFramework: Coupling multiple models of mantle convection within a computational framework. *Geochemistry, Geophysics, Geosystems* 7(Q06001), doi:10.1029/2005GC001155.

[21] Gerya, T. and Yuen, D. (2003). Rayleigh–Taylor instabilities from hydration and melting propel "cold plumes" at subduction zones. *Earth and Planetary Science Letters* 212(1–2): pp. 47–62.

[22] Gerya, T. and Burg, J. (2007). Intrusion of ultramafic bodies into the continental crust: numerical simulation. *Physics of the Earth and Planetary Interiors* 160: pp. 124–142.

[23] Christensen, U. R. and Hofmann, A. W. (1997). Segregation of subducted oceanic crust in the convecting mantle. *Journal of Geophysical Research* 99: pp. 19867-19884.

[24] Tackley, P. J. and King S. (2003). Testing the tracer ratio method for modeling active compositional fields in mantle convection simulations. *Geochemistry Geophysics Geosystems* 4(8302), doi: 10.1029/2001GC000214.

[25] Xie, S. and Tackley, P. J. (2004). Evolution of U-Pb and Sm-Nd systems in numerical models of mantle convection and plate tectonics. Journal of Geophysical Research 109(B11204), doi: 10.1029/2004JB003176.

[26] McNamara, A. K. and Zhong, S. (2004). Thermochemical structures within a spherical mantle: Superplumes of piles? *Journal of Geophysical Research* 109(B07402), doi: 10.1029/2003JB002847.

[27] van Keken, P. E., King, S. D., Schmeling, H., Christensen, U. R., Neumeister, D. and Doin, M.-P. (1997). A comparison of methods for the modeling of thermochemical convection, *Journal of Geophysical Research* 102: pp. 22 477- 22 496.

[28] Gropp, W., Lusk, E. and Skjellum, A. (1999). *Using MPI.* MIT Press. Cambridge.

[29] Manea, V. C., Manea, M., Pomeran, M., Besutiu, L. and Zagnean, L. (2012). A parallelized tracer code for CFD simulations in Earth Sciences [abstract]. En *3rd International supercomputing conference in Mexico. Where supercomputing Science and Technologies Meet, S09.* Memoria del Congreso. Procesamiento en Paralelo: p. 103.