SeisSol Optimization, Scaling and Synchronization for Local Time Stepping

JOSEP DE LA PUENTE* - MARTIN KÄSER - JOSÉ MARÍA CELA*****

* Institut de Ciències del Mar, CMIMA-CSIC Barcelona, Spain.
** Department of Earth and Environmental Sciences, Ludwig-Maximilians-University Munich, Germany
*** Computer Applications on Science and Engineering Department, Barcelona Supercomputing Center, Spain

Abstract. – *The SeisSol code is a solver for the elastic wave equations developed in the past three years. It uses a novel technique, essentially a DG (Discontinuous Galerkin) method, called ADER-DG which is a high-order scheme using tetrahedra as computational cells. The code has been extended to many special cases interesting for seismologists e.g. anisotropy, viscoelasticity, poroelasticity and finite sources. It is written in FORTRAN 95 and uses meshes created externally with a variety of commercial software, mostly Gambit and ICEM CFD. Mesh partitioning is performed using the Metis software. Parallelization has been implemented using MPI (Message Passing Interface) libraries and the code is successfully working in different local clusters as well as the Munich Supercomputer HLRB2. A novel asynchronous explicit time stepping scheme can strongly reduce the overall number of iterations but makes the number of iterations performed by each processor time-dependent. We study the time evolution of the workload for each processor for a large-scale simulation setup and speculate on possible solutions.*

The SeisSol code is an application for the simulation of seismic waves in complex 3D media [1, 2]. It is based upon a discontinuous Galerkin method and uses tetrahedral meshes to discretize the models. SeisSol has been working on the cluster of the Geophysics department of the LMU (160 AMD Opteron 250 processors, Ethernet connection) and at the HLRB2 supercomputer at the LRZ Munich (9728 Itanium2 processors, NUMAlink connection). During the month spent at the BSC, the code has been successfully ported to the IBM architecture of the MareNostrum supercomputer. Some monitoring using PARAVER has been added with the aim of controlling the time spent in communications during local-time stepping (LTS) asynchronous solvers [3]. Such solvers use a different time step, locally optimal in terms of stability of explicit schemes, for each element in the domain. As a consequence, in highly heterogeneous models, a large reduction on the amount of iterations required can be achieved. The code performs time cycles in which a different amount of elements are updated according to the current time level and the local time step of each element. Therefore, at each cycle a different amount of elements require an update.

For parallel computations, the mesh is partitioned in subdomains using the Metis [4] algorithm and each subdomain is assigned to a different core. After each cycle information is passed among cores using MPI routines, if needed. When LTS schemes are used, for each cycle a different amount of iterations must be performed at different subdomains. All cores then wait for the slowest (the one which performs more iterations) to finish in order to compute the next cycle. Thus there is a loss of speed-up for LTS in parallel because of the blocking structure of the algorithm. Nevertheless, the often strong reduction in the total amount of iterations required by LTS schemes makes it preferable in terms of run-time compared to the more commonly applied global timestepping (GTS) schemes. The cases studied have essentially been simulations of a hypothetic earthquake in the area of Grenoble, a region of largely heterogeneous material properties and complex geometry. For the simulations a total of 256 cores have been used and simulations lasted for 31 hours in order to retrieve 30 seconds of seismograms. For comparison, the same simulation in the HLRB2 system requires about 25 hours of run-time in the same number of cores. In Figure 1 we can see the evolution of the minimum and maximum number of iterations performed at each processor. The global speed of the code will always be limited by the maximum number of iterations per cycle, which has a roughly stable value throughout the time marching after a few cycles. However, the maximum value is not achieved always by the same processor. The thin lines in the figure depict the trend of 5 sample processors. It can be seen that the maximum value among them is achieved by different processors at different cycles. Moreover, if we consider an optimal distribution of the iterations among processors we find that the code could reach an improvement of more than a 30% in terms of run time. In the next figure we can observe how each core perfoms a different amount of iterations at each cycle.

Efforts have been undertaken towards optimizing the load balance in LTS schemes by redistributing the elements of the mesh among the processor. Unfortunately, improvements in the load balance by element redistribution lead to an under-

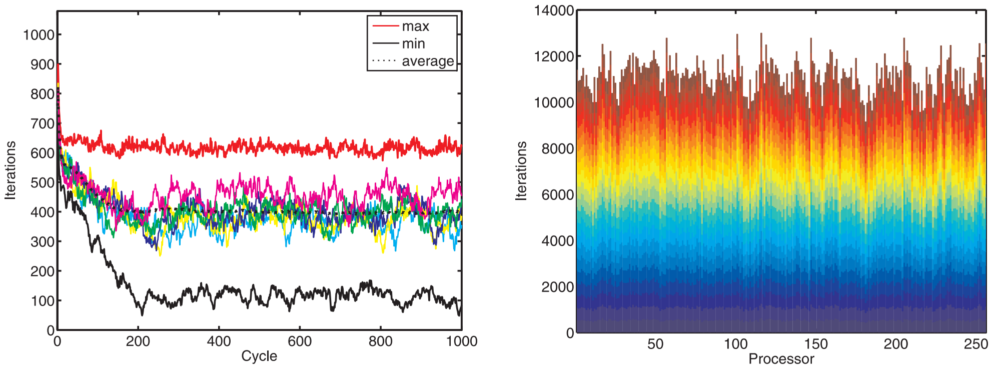

FIGURE 1. – *Left: Evolution of the maximum and minimum iterations performed among all processors at each cycle. Thin lines depict the trends of the sample processors* 10 *to* 14. *Right: Workload of each processor between cycles* 40 *and* 60. *Each color depicts a single cycle.*

performing topological distribution of the elements and thus strongly increase the amount of communication required. Thus an optimal load balance reverts into an increase of communication overhead.

Future work should lead to one possible direction. A first approach could be to develop a new iterative inversion process that obtains such distribution of elements that optimizes load balance and communication overhead. This would have little impact on the present structure of the *SeisSol* code. A more radical approach would be to dynamically accomplish a good load balance on-the-fly, possibly by redistributing elements at run-time. This *dynamic load balancing* would imply a remarkable amount of coding and the overall benefit is yet to be assessed.

To conclude we wish to remark that some of the runs performed at the BSC have been used to obtain synthetic seismograms of translational and rotational ground motion for a recent study [5].

References

[1] M. Käser and M. Dumbser, *An Arbitrary High Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes I: The Two-Dimensional Isotropic Case with External Source Terms*, Geophys. J. Int., **166**(2), 855-877 (2006).

[2] M. Dumbser and M. Käser, *An Arbitrary High Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes II: The Three-Dimensional Case*, Geophys. J. Int., **167**(1), 319-336 (2006).

[3] M. Dumbser, M. Käser and E.F. Toro, *An Arbitrary High Order Discontinuous Galerkin Method for Elastic Waves on Unstructured Meshes V*, Geophys. J. Int., **171**(2), 695-717 (2007).

[4] G. Karypis and V. Kumar, *Multilevel k-way partitioning scheme for irregular graphs*, J. *PARALLEL DISTRIB. COMPUT*, **48**, 96-129 (1998).

[5] M. Stupazzini, J. de la Puente, C. Smerzini, M. Käser, H. Igel and A. Castellani, *Study of rotational ground motion in the near field region*, Bull Seism. Soc. Am., in press (2009).