High-Frequency Simulations of Global Seismic Wave Propagation Using SPECFEM3D_GLOBE

Laura Carrington[a], Dimitri Komatitsch[b,c], Michael Laurenzano[a], Mustafa M Tikir[a], David Michéa[b], Nicolas Le Goff[b], Allan Snavely[a], Jeroen Tromp[d]

[a] Performance Modeling and Characterization Lab, San Diego Supercomputer Center, La Jolla, CA, USA

[b] Université de Pau, CNRS and INRIA Magique-3D, Laboratoire de Modélisation et d'Imagerie en Géosciences, Pau, France

[c] Institut Universitaire de France, Paris, France

[d] Seismological Laboratory, California Institute of Technology, Pasadena, CA, USA

Abstract

SPECFEM3D_GLOBE is a spectral-element application enabling the simulation of global seismic wave propagation in 3D anelastic, anisotropic, rotating and self-gravitating Earth models at unprecedented resolution. A fundamental challenge in global seismology is to model the propagation of waves with periods between 1 and 2 seconds, the highest frequency signals that can propagate clear across the Earth. These waves help reveal the 3D structure of the Earth's deep interior and can be compared to seismographic recordings. We broke the 2 second barrier using the 62K processor Ranger system at TACC. Indeed we broke the barrier using just half of Ranger, by reaching a period of 1.84 seconds with sustained 28.7 Tflops on 32K processors. We obtained similar results on the XT4 Franklin system at NERSC and the XT4 Kraken system at University of Tennessee Knoxville, while a similar run on the 28K processor Jaguar system at ORNL, which has more memory per processor, sustained 35.7 Tflops (a higher flops rate) with a 1.94 shortest period. For the final run we obtained access to the ORNL Petaflop System, a new very large XT5 just coming online, and achieved **1.72** shortest period and **161 Tflops** using **149,784** cores.

With this landmark calculation we have enabled a powerful new tool for seismic wave simulation, one that operates in the same frequency regimes as nature; in seismology there is no need to pursue periods much smaller because higher frequency signals do not propagate across the entire globe.

We employed performance modeling methods to identify performance bottlenecks and worked through issues of parallel I/O and scalability. Improved mesh design and numbering results in excellent load balancing and few cache misses. The primary achievements are not just the scalability and high teraflops number, but a historic step towards understanding the physics and chemistry of the Earth's interior at unprecedented resolution.

1 Introduction

The calculation of accurate synthetic seismograms for 3D global Earth models poses a significant computational challenge, both in terms of the demands on the numerical algorithm and with regards to computer hardware (i.e., memory and CPU requirements). Global seismologists routinely analyze recorded seismic signals with period between 1 and 2 seconds. Previous large-scale simulations in 3D Earth models have only been capable of reaching 3.5 seconds [11]. Therefore, our objective is to simulate global seismic wave propagation down to periods between 1 and 2 seconds, the highest frequency signals that can propagate clear across the Earth. Shorter periods get attenuated before reaching the other side of the Earth[1]. These waves at periods of 1 to 2 seconds, generated when large earthquakes (typically of magnitude 6.5 or above) occur in the Earth, help reveal the detailed 3D structure of the Earth's deep interior, in particular near the core-mantle boundary (CMB), the inner core boundary (ICB), and in the enigmatic inner core composed of solid iron. The CMB region is highly heterogeneous with evidence for ultra-low velocity zones, anisotropy, small-scale topography, and a recently discovered post-perovskite phase transition. The Earth's inner core appears to be anisotropic, with dramatic differences between its eastern and western hemispheres, and there are suggestions that it rotates at a slightly different rate than the Earth's mantle. Being able to simulate 3D global seismic wave propagation at these frequencies will thus help us understand and image these complex structures, an endeavor that will enhance our understanding of the physics and chemistry of the Earth's interior. The SPECFEM3D_GLOBE package has been designed to compute these simulations.

Since the record-breaking 3.5 second frequency run of 2003 which used the Earth Simulator[11], the team has expended a major R&D effort towards breaking the 2 second barrier. Achieving this goal required radical algorithmic changes to SPECFEM3D enabling peta-scalability (beyond 10Ks of processors) and in-

[1] A period of 1 second corresponds to a wavelength of compressional waves of 15 km or less, and a wavelength of shear waves of 8 km or less.

corporation of new algorithms that are both more scientifically accurate and more computationally scalable. Recent algorithm and tuning work is described in Section 4, previous such work involved optimizations to reduce cache misses, a new mesh design to improve spatial resolution for the seismic waves and to nearly eliminate load imbalance, and improvements to the inner Earth core resolution based upon an "inflated" central cube instead of a real cube with flat faces [7]; reduction of the "central cube" bottleneck by cutting the cube in two, reduction of MPI messages by 33% inside each chunk by handling crust mantle and inner core simultaneously, and finally non-iterative coupling between fluid and solid based on the displacement vector [4] instead of velocity as in previous versions of the application. In addition to these enhancements and optimizations, the model has been improved to include more complex Earth models and the capacity to compute sensitivity kernels for inverse problems in addition to forward problems [13]. Thus with the advanced domain science and computer science incorporated in SPECFEM3D it amounts to practically a new code and we were able to break the 2 second barrier using it.

The paper is laid out as follows: in Section 2, a description of the spectral-element method used to solve the seismic wave propagation problem is given. Section 3 briefly describes the current usage for the SPECFEM3D application and challenges in moving to shorter seismic periods. In Section 4, we describe the challenges associated with running at large scales (e.g. >10K+ cores), plus the performance analysis, code modifications, and tuning we carried out to address those challenges. Section 5 presents the results; ground breaking simulations of global seismic wave propagation down to wave periods of 1 to 2 seconds at more than 160 Tflops sustained, and section 6 illustrates the deep investigations that will be carried out now with this tool to explore the Earth's inner structures.

2 Description of the method

To simulate global seismic wave propagation in 3D anelastic, anisotropic, rotating and self-gravitating Earth models we have developed and implemented a spectral-element method (SEM). The SEM was introduced more than twenty years ago in computational fluid dynamics [14]. It has gained interest for problems related to 2-D [5, 15] and 3-D [8, 9, 12] seismic wave propagation (for instance following a large earthquake). The method accurately represents the propagation of both body waves and surface waves, and lends itself well to parallel computation with distributed memory [6, 11].

2.1 Equations of motion

We seek to determine the displacement field produced by an earthquake in a finite Earth model, as shown in Figure 1. The equations of motion that govern the propagation of seismic waves in the Earth may be solved based upon either a strong or a weak formulation of the problem. In the strong formulation one works directly with the equations of motion and associated boundary conditions written in differential form; this approach is used, for instance, in finite-difference or global-pseudo spectral modeling techniques. In the weak formulation one uses an integral form of the equations of motion, as in finite-element (FEM) and direct solution methods. The SEM is based upon a weak formulation of the equations of motion.

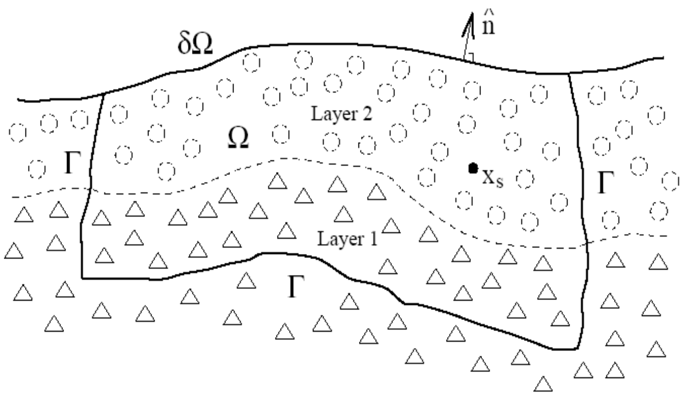

Figure 1. Finite Earth model with volume Ω and free surface $\partial\Omega$. An artificial absorbing boundary Γ is introduced if the physical model is not of finite size, and $\hat{n}$ denotes the unit outward normal to all boundaries. The model can be fully heterogeneous or composed of any number of layers.

2.1.1 Strong form

The displacement field s produced by an earthquake is governed by the momentum equation

$$\rho\partial_t^2 s = \nabla \cdot \mathrm{T} + f \tag{1}$$

The distribution of density is denoted by ρ. The stress tensor T is linearly related to the displacement gradient ∇ s by Hooke's law, which in an elastic, anisotropic solid may be written in the form

$$\mathrm{T} = c : \nabla s \tag{2}$$

The elastic properties of the Earth model are determined by the fourth-order elastic tensor c, which has

21 independent components in the case of general anisotropy.

The earthquake source is represented by the point force f, which may be written in terms of a moment tensor M

$$f = -\mathrm{M} \cdot \nabla \delta(x - x_0) S(t) \tag{3}$$

The location of the point source is denoted by x_s, $\delta(x - x_0)$ denotes the Dirac delta distribution located at x_s, and the source-time function is given by S(t).

The momentum equation (1) must be solved subject to a stress-free boundary condition at the Earth's surface $\partial\Omega$:

$$\mathrm{T} \cdot \hat{n} = 0 \tag{4}$$

2.1.2 Weak form

Rather than using the equations of motion and associated boundary conditions directly, one can use an integrated form. This is accomplished by dotting the momentum equation (1) with an arbitrary vector w, integrating by parts over the model volume Ω, and imposing the stress-free boundary condition (4). This gives

$$\int_\Omega \rho w \cdot \partial_t^2 s d^3x = -\int_\Omega \nabla w : \mathrm{T} d^3x + \mathrm{M} : \nabla w(x_0) S(t) \tag{5}$$

where the stress tensor T is determined in terms of the displacement gradient ∇s by Hooke's law (2). The source term has $\int_\Omega f \cdot w d^3x$ been explicitly integrated using the properties of the Dirac delta distribution.

2.2 Definition of the mesh

As in a classical FEM, the model volume Ω is subdivided into a number of non-overlapping elements Ω_e, e = 1,…,n_e, as shown in Figure 2. Each hexahedral volume element Ω_e is mapped to a reference cube. The mapping is defined by the so-called classical Jacobian matrix. Points within this reference cube are denoted by the vector $\xi = (\xi,\eta,\zeta)$, where $-1 \le \xi \le 1$, $-1 \le \eta \le 1$ and $-1 \le \zeta \le 1$.

2.3 Representation of functions and numerical integration on the elements

To solve the weak form of the equations of motion (5), integrations over the volume Ω are subdivided in terms of smaller integrals over the volume elements Ω_e. A high-degree Lagrange interpolant is used to express functions on the elements. The control points needed in the definition of the Lagrange polynomials of degree n_ℓ are chosen to be the classical $n_\ell + 1$ so-called Gauss-Lobatto-Legendre (GLL) quadrature points. Note that they always include +1 and −1; therefore in a SEM some points always lie exactly on the boundaries of the elements.

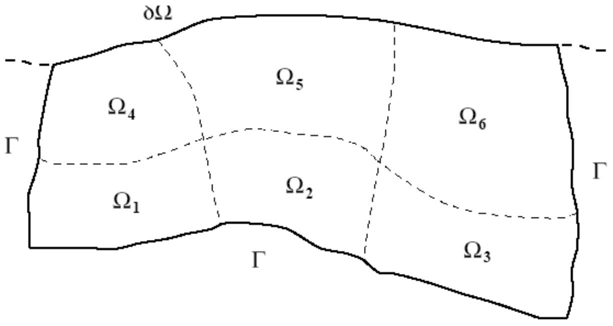

Figure 2. For the purpose of computations, the Earth model Ω shown in Figure 1 is subdivided into curved hexahedra whose shape is adapted to the edges of the model ∂Ω and Γ and to the main geological interfaces.

Any smooth function *f* can then be interpolated in a 3D hexahedral element by triple products of Lagrange polynomials of degree n_ℓ at these GLL points. In a SEM for seismic wave propagation problems one typically uses a polynomial degree n_ℓ between 4 and 10 to represent a function on the element [8]. The derivative of function *f* can then be computed by computing the derivative of the Lagrange polynomials. And numerical integration of this function over volume elements Ω_e may be approximated using the Gauss-Lobatto-Legendre integration rule, whose weights can easily be computed numerically and stored once and for all [3].

2.4 Assembling and marching the global system in time

In the SEM mesh, grid points that lie on the sides, edges, or corners of an element are shared amongst neighboring elements, as illustrated in Figure 3. Therefore, the need arises to distinguish between the grid points that define an element, the *local mesh*, and all the grid points in the model, many of which are shared amongst several spectral elements, the *global mesh*. One needs to determine a mapping between grid points in the local mesh and grid points in the global mesh; efficient routines are available for this purpose from finite-element modeling. Before the system can be marched forward in time, the contributions from all the elements that share a common global grid point need to be summed. In a traditional FEM this is referred to as the *assembly* of the system. Computationally, this assembly stage is a costly part of the calculation on parallel computers, because information from individual elements needs to be shared with neighboring elements, an operation that involves communication between distinct CPUs (based on message passing with MPI in our case, see for instance Komatitsch et al. [11]).

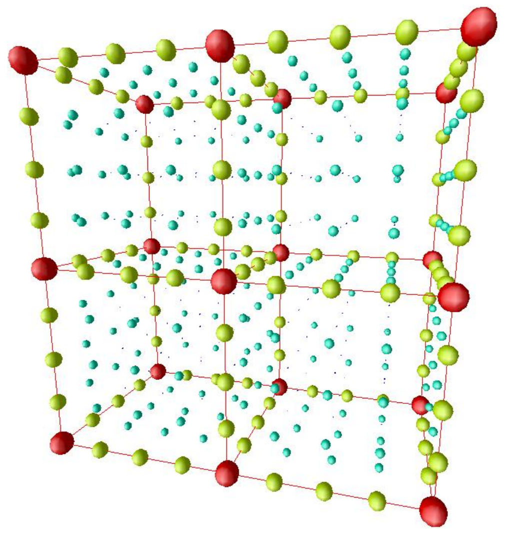

Figure 3. Illustration of the local and global meshes for a four-element spectral-element discretization with polynomial degree *N* = 4. Each spectral element contains $(N + 1)^3$ = 125 Gauss-Lobatto-Legendre points that constitute the local mesh for each element. These points are non-evenly spaced, but have been drawn evenly spaced here for simplicity. In the global mesh, points lying on faces, edges or corners are shared between elements. The contributions to the global system of degrees of freedom, computed separately on each element, have to be summed at these common points. Exactly two elements share points inside a face, while corners can be shared by any number of elements depending on the topology of the mesh, which can be non-structured.

Let U denote the displacement vector of the global system, i.e., U contains the displacement vector at all the grid points in the global mesh, classically referred to as the global degrees of freedom of the system. The ordinary differential equation that governs the time dependence of the global system may be written in the form

$$M\ddot{U} + KU = F, \qquad (6)$$

where M denotes the global mass matrix, K the global stiffness matrix, and F the source term. Explicit expressions for the local contributions to the mass and stiffness matrices and further details on the construction of the global mass and stiffness matrices from their elemental expression may be found for instance in [8],[9].

A highly desirable property of a SEM, which allows for a very significant reduction in the complexity and cost of the algorithm, is the fact that the mass matrix M is diagonal by construction. Therefore, no costly linear system resolution algorithm is needed to march the system in time.

Time discretization of the second-order ordinary differential equation (6) is achieved based upon a classical explicit second-order finite-difference scheme, which is conditionally stable (i.e., the time step has an upper limit above which the simulation becomes unstable).

3 The SPECFEM3D GLOBE package

The SPECFEM3D_GLOBE package was designed to simulate three-dimensional global and regional seismic wave propagation based upon the SEM to solve the equations described in Section 2. The package is maintained under GNU GPL license on a source code release server at Computational Infrastructure for Geodynamics (CIG) [1], and is being actively developed by a core group of approximately 15 scientists. The package has been extensively benchmarked against semi-analytical normal-mode synthetic seismograms (i.e., curves showing the evolution of displacement with time after the earthquake at a given mesh point) for spherically-symmetric Earth models [9][10]. These benchmarks are very challenging because they involve solid-fluid domain decomposition and coupling, attenuation, anisotropy, self-gravitation, and the effect of the ocean layer located at the surface of the Earth. Our simulations incorporate effects due to topography and bathymetry as well as fluid-solid boundaries, such as the ocean floor, the core-mantle boundary (CMB), and the inner-core boundary (ICB). Thus far, only SPECFEM3D_GLOBE has been capable of accurately incorporating all of these effects.

SPECFEM3D_GLOBE consists of two major subprograms: meshfem3D, the mesher, which generates the spectral-element mesh and specfem3D, the solver, which uses the generated mesh to run the simulation.

The mesher is designed to generate a spectral-element mesh for either regional or entire globe simulations. This work focuses on simulations of the entire globe, which are the most expensive and therefore by far the most challenging. These simulations use a spectral-element mesh which is based upon an analytical mapping from the cube to the sphere called the ‘gnomonic mapping’ or the ‘cubed sphere’ (see e.g. [17],[16]), which splits the globe into 6 chunks, each of which is further subdivided into n^2 mesh slices for a total of 6 x n^2 slices, as shown in Figure 4. The work for the mesher code is distributed to a parallel system by distributing the slices.

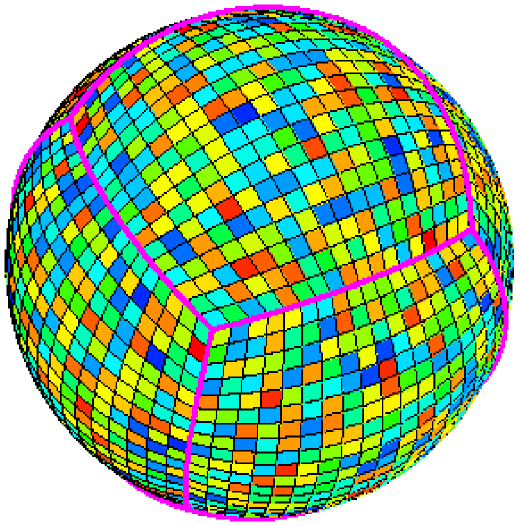

Figure 4. cubed-sphere mapping of the globe: here we represent a mesh of 6 x 18^2 = 1944 slices.

Given the shortest desired period, the grid spacing is determined by a requirement of at least 5 grid points (GLL points) per shortest seismic wavelength that we want to accurately model, and the Courant stability condition determines the upper bound of the associated time step. Current tomographic models reveal only large-scale features of the Earth's interior, features with dimensions much larger than the wavelengths of 1-second to 2-second waves.

4 Overcoming large-scale challenges

To meet our objective to simulate global seismic wave propagation down to seismic wave periods of 1 to 2 seconds (i.e., up to maximum seismic frequencies of 0.5 to 1 Hz) the mesher and solver would each require at least 37 TBs of data. This would require around 62K cores of an HPC system having around 1.85 GB of memory per core available to the running application. Running any application at this scale can create bottlenecks and challenges not seen at smaller scale.

There were four separate efforts working on the SPECFEM3D_GLOBE package to enable efficient runs at large scale. The first was to remove the I/O bottleneck created between the mesher and solver. The second was to make sure the mesh layout was optimal. The third effort was to do some single processor optimization on the computation routine that dominates the runtime. The fourth effort was to implement a faster way of assigning material properties to mesh elements and to design a simpler algorithm to locate seismic recording stations in order to improve performance and get higher overall FLOPS. These are each expanded upon below.

4.1 Removing the I/O bottleneck

The original mode of running the SPECFEM3D code was to first run the MESHFEM3D code which generated the mesh and wrote it to local disks. The SPECFEM3D code then ran immediately after this and read in those local disk files. For a system with good local disks this method of running can be quite efficient (although sensitive to hardware failures of these disks, or to the fact that one of them can be full or almost full etc.). But many newly installed larger systems use diskless nodes (to decrease power consumption and to increase node stability by getting rid of mechanical parts). This means that each of these mesher files would then have to be written to a globally mounted file system (for instance LUSTRE or GPFS), creating a large bottleneck for both the mesher and the solver due to I/O contention. The original (current stable) version of the code (version 4.0) writes and reads up to 51 files per core. At around 62K cores, this corresponds to over 3.2 million files that would have to be written and then subsequently read. Furthermore, the amount of data transferred between the two parts of the application will become a factor for a large-scale simulation. Figure 5 shows a simple regression model of the disk space used for a series of resolutions along with the actual disk usage. This model predicts that in order to obtain a simulation accurate down to a seismic period of 2 seconds, over 14 TB of data would have to be transferred between the mesher and the solver; and to obtain a simulation accurate down to a seismic period of 1 second, over 108 TB of data transfer is required.

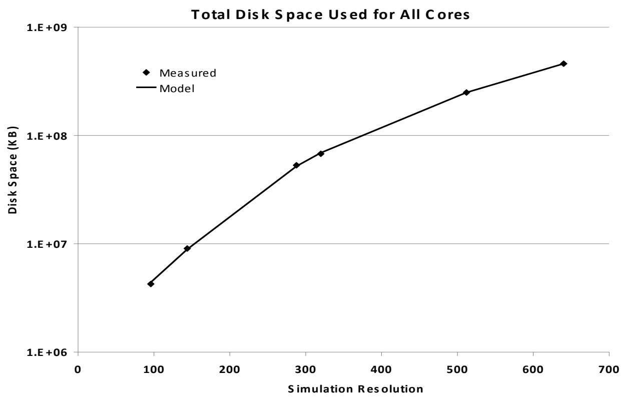

Figure 5. Total disk space used for communication between MESHFEM3D and SPECFEM3D in the initial stable version of the package. Resolution = 256*17 / Wave Period. (Higher resolution is higher frequency).

This amount of data transfer was deemed a major performance bottleneck; the bottleneck was removed by merging the mesher and solver into a single application and making them communicate via shared memory rather than with I/O. Merging the codes was technically difficult because it brought challenges in memory management and bookkeeping across the two originally separate applications.

We were able to completely remove the use of I/O to communicate between the two parts of the application, eliminating the need to use any disk space for intermediate files along with the associated I/O penalties of using these files. Initially, removing the I/O bottleneck created an additional challenge by using more memory because in the initial merged version some of the arrays from the mesher and from the solver had to be present in memory simultaneously. This was problematic because the more memory per core required, the more cores we will need to meet the goal of simulations accurate down to seismic periods of 1 to 2 seconds. To reduce this memory usage, optimizations were performed to lower the memory high water mark of the merged application. This was achieved by reusing the data structures allocated by the mesher in the solver via allocating these data structures on the data segment of the application as well as allocating some of the data structures on the call stack such that memory fragmentation would be prevented.

4.2 Point renumbering and multilevel Cuthill-McKee sorting

In the SEM algorithm, one spends a lot of time looping on all the elements (the so-called spectral elements) of the 3D mesh and computing local contributions (local forces and resulting acceleration vectors) at all the internal grid points of each element. Contributions computed at element faces, edges or corners shared between two or more elements are then summed. Therefore in principle (i.e. mathematically) one can loop on the elements in any order and get the same final result because of the associativity and commutativity of the sum operator. (Note that formally this ceases to be true on a computer because of different roundoff depending on the order in which the sub-sums are performed, but in practice only the last one or two decimals are affected and therefore one can still choose any order, and the result is “almost” invariant by permutation down to the last digits). We have checked this experimentally: the same mesh computed with different loop orders on the elements give two sets of synthetic seismograms that are indistinguishable when plotted superimposed.

However, processors have caches and therefore it is important to try to maximize cache reuse and also maximize the effect of prefetching by trying to loop on the neighbors of an element first once the calculations in that element are finished; this way we will increase the probability for common faces, edges or corners to already be in the cache.

To increase spatial and temporal locality for the global access of the points that are common to several elements, the order in which we access the elements can then be optimized. The goal is to find an order that minimizes the memory strides for the global arrays. We use the classical reverse Cuthill-McKee [17] algorithm, which consists of renumbering the vertices of a graph to reduce the bandwidth of its adjacency matrix. Sorting the elements with the Cuthill-McKee algorithm before renumbering the global index table also increases the spatial and temporal locality: spatial locality, because the common points of the connected elements will be stored statistically closer in memory; temporal locality, because these common points will be re-accessed sooner. We have designed an improved version of that algorithm in which we use multi-level sorting to define groups of typically 50 to 100 elements which all fit together in the L2 cache. Tests performed with SPECFEM3D_GLOBE on the same mesh with and without sorting show that unfortunately we do not gain much based on sorting: at most 5% in practice. But this is probably in fact good news: it means that previous work we performed to reduce cache misses based on point renumbering [7], which is crucial, has worked very well and there are already so few L2 cache misses that it is difficult to further reduce them. An additional explanation is the fact that in the SEM we perform a lot of local operations in each element therefore in percentage the time it takes to move new data in the L2 cache is not crucial compared to the total time it takes to perform the calculations in that element. This implies that using more modern element renumbering algorithms such as Peano/Hilbert curves instead of Cuthill-McKee sorting would probably not help much.

4.3 Manual use of SSE instructions

The initial performance model for the SPECFEM3D_GLOBE application indicates that a large fraction of time (greater than 70%) is spent in two computational routines in which we compute the internal forces and related acceleration vectors in each spectral element of the mesh in two regions of the Earth: the large solid mantle and crust, and the smaller fluid outer core. Inside these two routines, which have a very similar structure, we perform small matrix-matrix products (each matrix has a size of 5 x 5 typically) along cutplanes of 3D arrays (first cut along the i axis, then cut along the j axis, and then cut along the k axis).

It is therefore important to study how we can optimize this crucial section of the two routines. When talking about matrix-matrix products, one immediately thinks about calling a vendor-optimized implementation of the Basic Linear Algebra Subprograms (BLAS-3) subroutine SGEMM, but in our case this turns out to be a poor idea for two reasons. First. the matrices are very small (5 x 5) and therefore the overhead of the BLAS routine is higher than what we can hope to gain. Second, because we have to handle cutplanes along three different directions of a 3D memory block, several of these calls to BLAS would be for blocks not linearly aligned in memory and would therefore first require a memory copy to an aligned 2D block, before the call; this would be more expensive than any potential gain from the BLAS routine.

Tests that we have performed have confirmed that using BLAS calls actually significantly slows down the code compared to our existing regular Fortran loops. We therefore tried another option, which is to use vector instructions provided by a SSE unit (for instance on Intel or AMD processors) or an Altivec/VMX unit (for instance on IBM PowerPC processors). These units can handle four single-precision floating-point operations in a vector and are very well suited for our small matrix products since we can load a vector unit with 4 floats, perform several “multiply and add” (MADD) operations to compute the matrix-matrix product, and store the results in four consecutive elements of the result matrix (Note that MADD does not exist explicitly in SSE but is rather implemented as a combination of "multiply" and then "add").

These three types of operations (load, MADD and store) are standard in both SSE and Altivec. Note that, since our matrices are of size 5 x 5 and not 4 x 4, we use vector instructions for 4 out of each set of 5 values and compute the last one serially in regular Fortran. Also note that to improve performance we align our 3D blocks of 5 x 5 x 5 = 125 floats on 128 in memory using padding with three dummy values set to zero. This induces a negligible waste of memory of 128 / 125 = 2.4%.

The tests we performed show that we typically gain between 15% and 20% (with respect to the stable version 4.0 of our code) both with SSE on AMD processors and with Altivec on another machine equipped with IBM PowerPC970 processors. The relative gain is limited by two factors: first, the limited number of vector registers present in the hardware (16 for SSE and 32 for Altivec); and second the fact that modern compilers can automatically unroll loops and generate SSE or Altivec instructions to perform something similar to what we implement manually; therefore the reference time may already include some of the effects of using SSE instructions.

4.4 Optimizations to improve FLOPS

When using 10K+ cores, many things that have worked fine for years in the application on tens or hundreds of cores can start to either fail or become very slow and significantly reduce performance and may need to be partially or entirely redesigned. In SPECFEM3D_GLOBE we found and fixed two such problems:

1. Due to legacy code, the mesher was actually run twice internally: once to generate the mesh of elements (i.e., the geometry) and a second time to populate this geometry with material properties (i.e., the velocity of the seismic waves and the density of the rocks in each mesh element); this slowed down the mesher by a factor of two, which may be acceptable on a small in-house cluster but not on 10K+ cores on a machine shared with other users; we therefore merged these two steps (assigning properties to each mesh element right after its creation)
2. At low resolution, the mesher used to use a costly non linear algorithm to locate the seismic recording stations in the mesh (the location of these stations may not fall exactly on a grid point and at low resolution choosing the closest point leads to a large error, therefore one needs to use a more precise algorithm to locate them between grid points; as a result, a costly interpolation process also had to be used in the solver to compute the wave field at the right location between grid points. At very high resolution, this resulted in a significant slowdown of the whole application and significant load imbalance because some mesh slices carry more seismic stations than others and

therefore would spend more time performing the interpolation. We noticed that at high resolution the best option was to suppress the costly interpolation process and to locate these stations at the closest grid point because the mesh is so dense that the error made is then very small (and negligible from a geophysical point of view)

5 Performance measurements and models

To meet our objective to simulate global seismic wave propagation down to seismic wave periods under 2 seconds we needed to run on 30K cores or more. We used four different systems to investigate how to reach this goal.

The first is Texas Advanced Computing Center (TACC) Sun Constellation Linux cluster, named Ranger, which has 62,976 processing cores connected with a full-CLOS InfiniBand interconnect. Each compute node in Ranger consists of four 2.0 GHz quad-core AMD Opteron processors with a theoretical peak performance of 32 Gflops and 8 GBytes of memory. The theoretical peak performance of Ranger is thus about 504 Tflops (its Rmax is 326 Tflops).

The second is National Energy Research Scientific Computing Center (NERSC) Cray XT4 system, named Franklin. Each of its compute nodes consists of a 2.6 GHz dual-core AMD Opteron processor with a theoretical peak performance of 10.4 Gflops and 4 GBytes of memory. The theoretical peak performance of Franklin is thus about 101.5 Tflops, its measured Rmax is 85 Tflops. Each compute node is connected to a dedicated SeaStar2 router through Hypertransport with a 3D torus topology

The third one is National Institute for Computational Sciences' (NICS) Kraken is a Cray XT4 system. Kraken has a total of 4512 compute nodes where each compute node contains a 2.3 GHz quad-core AMD Opteron processor and 4 GB of memory resulting in a total of 18048 compute cores. The theoretical peak performance of Kraken is about 166 Tflops. (Rmax unknown at time of publication). Kraken runs Compute Node Linux (CNL) on each compute node. Each node is connected to a Cray SeaStar router through HyperTransport, and the SeaStars are all interconnected in a 3-D-torus topology.

The fourth one is Oak Ridge National Laboratory's (ORNL) Cray XT4 system, named Jaguar. Jaguar has a total of 7,832 XT4 compute nodes where each compute node contains a quad-core 2.1 GHz AMD Opteron processor and 8GB of memory. The overall theoretical peak performance of Jaguar is 263 Tflops. (Rmax is 205 Tflops). Each node is connected to a Cray SeaStar router through HyperTransport, and the SeaStars are all interconnected in a 3-D-torus topology.

The initial step was to model the communication behavior of SPECFEM3D. To accomplish this we ran several experiments varying the input resolution and the number of processors. In SPECFEM3D, resolution can be changed based on an input parameter called NEX_XI, which defines the number of elements at the surface along the two sides of each of the six chunks, whereas the number of processor cores can be changed based on an input parameter called NPROC_XI, which defines the number of MPI processor cores to be used along the two sides of each of the six chunks. For our initial investigation, we varied the processor count from 24 to 1536 and the mesh resolution from 96 to 640 (which corresponds to minimum seismic periods from 45.3 seconds to 6.8 seconds, respectively).

We measured the communication time for each run with IPM (Integrated Performance Monitoring) tool [2], which is a portable profiling tool that provides a performance summary of the computations and communications in a parallel program. IPM has extremely low overhead and is scalable to thousands of processors, which makes it ideal for this purpose.

We measured the total communication time spent in the main loop of the solver component for each run. We ran these experiments on Franklin. Even though we used only Franklin for our modeling runs, we expected similar behavior on other balanced systems for SPECFEM3D. The results showed that the communication time spent in the main loop of the solver component ranges from 1.9% to 4.2% (with an average of 3.2%) of the overall execution time for the runs. More importantly, the lower communication percentages indicate that SPECFEM3D_GLOBE is dominated by the computation time and is a good candidate to scale up to tens of thousands of processors before the communication time becomes a bottleneck.

The results of modeling runs also showed that the total communication time spent for all processors tends to increase both when the resolution increases and when the number of processors increases. However, it also shows that for a given resolution, the communication time per core decreases as the number of processor increases. Using these observations and measured overall communication time for all processors, we fitted a function to the actual measured communication times for a given resolution. Figure 6 presents the measured and modeled total communication times for all cores for two resolutions. Other resolutions were fitted with similar results. Based on the fitted models for all resolutions used in our modeling runs, we were also able to model the increase in overall communication time for all cores as the resolution increases.

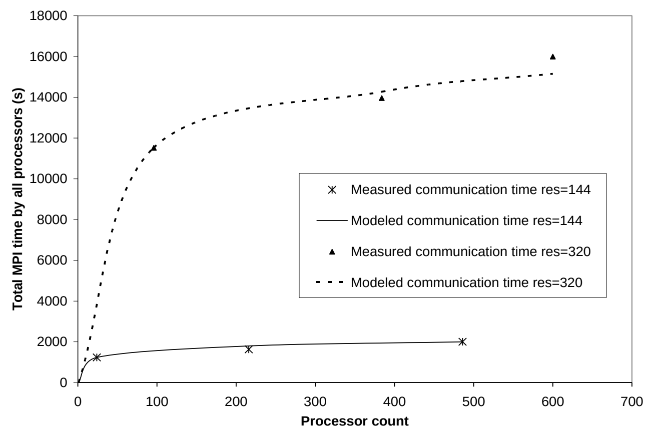

Figure 6. Fitted curves for total communication time (in seconds) for all cores for different resolutions.

Using the overall model, we were able to predict the total communication time for all cores of a hypothetical SPECFEM3D run with 12K processors and a resolution of NEX_XI = 1440 to be around 7.3E6 seconds, which corresponds to 599 seconds per core and 3.2% of overall execution time. Similarly, we predict the communication time per core for a SPECFEM3D run with 62K processors and a resolution of NEX_XI = 4848 to be around 28K seconds, which also corresponds to 4.7% of overall execution time. More importantly, the results of modeling runs as well as the models we devised using these results indicate that the overall execution time of a SPECFEM3D run is dominated by the computation time and communication is not expected to be the bottleneck for scaling the application to tens of thousands of processors.

Similar to the communication model we also modeled the total runtime for all cores in order to estimate the runtime of a run with a minimum seismic period under 2 seconds and also confirm that the larger 12K core run did not exhibit any unforeseen bottlenecks. The results of modeling experiments showed that the overall execution time totaled for all computation cores is defined by the resolution used and is independent of the number of cores used. That is, for a given resolution, the execution time per core decreases but the totaled execution time for all cores is almost always the same.

Figure 7 shows the actual (e.g. measured) and fitted total execution times for all cores (normalized with respect to the minimum) for different resolutions.

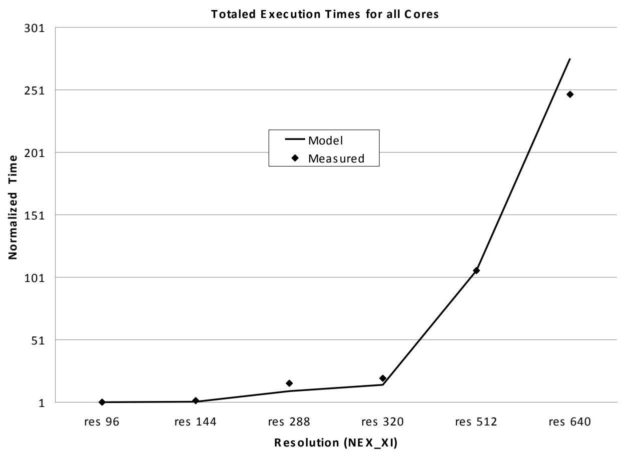

Figure 7. Predicted and actual total time spent for all cores for different resolutions.

Figure 7 shows that total execution time of SPECFEM3D for all cores increases significantly (quadratic) as the resolution increases. Using the fitted function, we were able to predict the totaled execution time of all cores of SPECFEM3D run with a 12K processors and a resolution of NEX_XI = 1440 within 12% error, indicating that no unforeseen bottlenecks emerged as the scaling was increased.

Similar to modeling communication we developed a model for the overall sustained FLOPS rate of the application using the modeling runs. The results show that the sustainable FLOPS rate for SPECFEM3D increases directly proportional to the number of processors it is run on and for the same number of processors slightly increases as the resolution increases.

6 Results of actual large simulations

The merged SPECFEM code run on the NERSC system Franklin was successfully completed on 12,150 cores running for nearly 6 hours achieving around 24 Tflops (44% of Rmax) to model a shortest seismic period of 3 seconds. We experimented with turning attenuation (i.e., loss of energy due to the fact that the rocks are viscoelastic) on and off. Attenuation was turned off initially to reduce the runtime in our initial modeling runs. Once the initial modeling runs confirmed the scaling, attenuation was turned on for the final science runs. This resulted in a 1.8 increase in execution time but only an almost imperceptible drop in Tflops.

Next, simulation of a few seconds of an earthquake in Argentina with attenuation turned on was run successively on 9,600 cores (12.1 Tflops sustained), 12,696 cores (16.0 Tflops sustained), and then 17,496 cores of NICS's Kraken system. The 17K core run sustained 22.4 Tflops and had a seismic period length of 2.52 seconds; temporarily a new resolution record. The Tflops number in these and subsequent reported runs was measured using PSiNSlight [18].

On the Jaguar system at ORNL we simulated the same event and achieved a seismic period length of 1.94 seconds and a sustained 35.7 Tflops (our current flops record) using 29K cores.

On the Ranger system at TACC the same event achieved a seismic period length 1.84 seconds (our current resolution record) with sustained 28.7 Tflops using 32K cores.

Finally, we obtained "friendly user" access to the new ORNL Petaflop System, the world's largest XT5-Kraken is being upgraded to XT5 the end of the year. There are 200 cabinets, each holding 768 cores. They are 2.3 GHZ with 2 GB/core. The nodes contain two Barcelona Sockets with Seastar 2+ interconnect. The interconnect is 4 GB/sec bi-directional on a 3-D torus, theoretical peak is 1.4 P, Rmax unknown at time of publication. Using this system we simulated the same event and achieved **1.72** shortest period and **161** Tflops using **149,784** cores. This is the shortest wave period ever obtained in seismic wave propagation, the highest level of parallelism, the first sustained performance of seismic wave propagation > 160 TFlops. But of more significance, it now enables simulations at the resolution of nature.

7 Future work and conclusion

The runs reported here are just precursors that modeled a few seconds of each earthquake event. It takes about 25 minute of real time and about 1 week we estimate of dedicated ORNL Petaflop (in other words a true petascale calculation) to model wave propagation clear through the Earth to predict structure.

The simulations we enabled, at under 2 seconds will help reveal the detailed 3D structure of the Earth's deep interior, in particular near the core-mantle boundary (CMB), the inner core boundary (ICB), and in the enigmatic inner core. Earth, help reveal the detailed 3D structure of the Earth's deep interior, in particular near the core-mantle boundary (CMB), the inner core boundary (ICB), and in the enigmatic inner core composed of solid iron. The CMB region is highly heterogeneous with evidence for ultra-low velocity zones, anisotropy, small-scale topography, and a recently discovered post-perovskite phase transition. The Earth's inner core appears to be anisotropic, with dramatic differences between its eastern and western hemispheres, and there are suggestions that it rotates at a slightly different rate than the Earth's mantle. Being able to simulate 3D global seismic wave propagation at these frequencies will thus help us understand and image these complex structures, an endeavor that will enhance our understanding of the physics and chemistry of the Earth's interior.

Acknowledgements

Some improvements in the package were implemented while D. Komatitsch and D. Michéa were visitors at the Barcelona Supercomputing Center (BSC, Catalonia, Spain) in the context of the HPC-Europa program. The help of Jesús Labarta, Sergi Girona, José Cela, David Vincente, and of their ParaVer analysis tool was invaluable. The authors would also like to thank Rogeli Grima from BSC and Sébastien Deldon from the Portland Group Inc. for fruitful discussion regarding Altivec and SSE instructions. This research used resources of the National Energy Research Scientific Computing Center, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231. This research used resources of the National Center for Computational Sciences at Oak Ridge National Laboratory (ORNL), which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725. We especially thank Don Maxwell and Steve Poole of ORNL for their help and time. This research used resources provided by the National Institute for Computational Sciences, operated by the University of Tennessee and funded by the National Science Foundation. We especially thank Phil Andrews, Victor Hazelwood, Bruce Loftis, and John Walsh of NICS and Arthur Funk of Cray for their help and time.

A very special thank-you to John Levesque of Cray stationed at ORNL; he ran the record setting calculations on the new ORNL Petaflop System. Chapeau!

This research used resources of Texas Advanced Computing Center (TACC) at The University of Texas at Austin. We especially thank John R. Boisseau, Tommy Minyard, and Karl W. Schulz of TACC for their help and time.

This work was supported in part by Performance Evaluation Research Institute (PERI), (DE-FC02-06ER25760), a DoE Office of Science SciDAC2 Institute, The Cyberinfrastructure Evaluation Center, (NSF-OCI-0516162), and Workshop on Petascale Computing and the Geosciences, (NSF-GEO-0621611). This material is based in part upon research supported by French ANR grant NUMASIS ANR-05-CIGC-002 and European FP6 Marie Curie International Reintegration Grant MIRG-CT-2005-017461.

References

1. Computational Infastructure for Geodynamics (CIG).

2. IPM: Integrated Performance Monitoring.

3. Canuto, C., Hussaini, M.Y., Quarteroni, A. and Zang, T.A. *Spectral methods in fluid dynamics*. Springer-Verlag, New York, 1998.

4. Chaljub, E. and Valette, B. Spectral element modelling of three-dimensional wave propagation in a self-gravitating Earth with an arbitrarily stratified outer core. *Geophys. J. Int.*, *158* (131-141).

5. Cohen, G., Joly, P. and Tordjman, N. Construction and analysis of higher-order finite elements with mass lumping for the wave equation. *Proceedings of the second international conference on mathematical and numerical aspects of wave propagation, SIAM*. 152-160.

6. Fischer, P.F. and Rønquist, E.M. Spectral-element methods for large scale parallel Navier-Stokes calculations. *Comput. Methods Appl. Mech. Engrg.*, *116*. 69-76.

7. Komatitsch, D., Labarta, J. and Michéa, D. A 21 billion degrees of freedom, 2.5 terabytes simulation of seismic wave propagation in the inner core of the Earth on MareNostrum. *Proceedings of the 8th World Congress on Computational Mechanics (WCCM8) and the 5th European Congress on Computational Methods in Applied Sciences and Engineering (ECCOMAS 2008)*.

8. Komatitsch, D. and Tromp, J. Introduction to the spectral-element method for 3-D seismic wave propagation. *Geophys. J. Int.*, *139* (3). 806-822.

9. Komatitsch, D. and Tromp, J. Spectral-element simulations of global seismic wave propagation-I. Validation. *Geophys. J. Int.*, *149* (2). 390-412.

10. Komatitsch, D. and Tromp, J. Spectral-element Simulations of Global Seismic Wave Propagation-II. 3-D Models, Oceans, Rotation, and Self-Gravitation. *Geophys. J. Int.*, *150*. 303-318.

11. Komatitsch, D., Tsuboi, S., Ji, C. and Tromp, J., A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the Earth Simulator. in *Proceedings of the ACM/IEEE Supercomputing SC'2003 conference*, (Phoenix, Arizona, USA, 2003).

12. Komatitsch, D. and Vilotte, J.P. The Spectral-element method: an efficient tool to simulate the seismic response of 2D and 3D geological structures. *Bull. Seismol. Soc. Am.*, *88* (2). 368-392.

13. Liu, Q. and Tromp, J. Finite-Frequency Kernel Based on Adjoint Methods. *Bulletin of the Seismological Society of America*, *96* (6). 2383-2397.

14. Patera, A.T. A Spectral element method for fluid dynamics: laminar flow in a channel expansion. *J. Comput. Phys.*, *54*. 468-488.

15. Priolo, E., Carcione, J.M. and Seriani, G. Numerical simulation of interface waves by high-order spectral modeling techniques. *J. Acoust. Soc. Am.*, *95* (2). 681-693.

16. Ronchi, C., Ianoco, R. and Paolucci, P.S. The "Cubed Sphere": a new method for the solution of partial differential equations in spherical geometry. *J. Comput. Phys.*, *124*. 94-114.

17. Sadourny, R. Conservative finite-difference approximations of the primitive equations on quasi-uniform spherical grids. *Mon. Wea. Rev*, *100*. 136-144.

18. PSiNS Tracer and Simulator, Performance Modeling and Characterization Lab, SDSC, San Diego, CA, http://www.sdsc.edu/pmac/projects/psins.html