Scalable Earthquake Simulation on Petascale Supercomputers

Y. Cui[1], K.B. Olsen[2], T. H. Jordan[3], K. Lee[1], J. Zhou[1], P. Small[3], D. Roten[2], G. Ely[3],
D.K. Panda[4], A. Chourasia[1], J. Levesque[5], S. M. Day[2], P. Maechling[3]
[1]San Diego Supercomputer Center, [2]San Diego State University,
[3]University of Southern California, [4]The Ohio State University, [5]Cray Inc.

***Abstract*— Petascale simulations are needed to understand the rupture and wave dynamics of the largest earthquakes at shaking frequencies required to engineer safe structures (> 1 Hz). Toward this goal, we have developed a highly scalable, parallel application (AWP-ODC) that has achieved "M8": a full dynamical simulation of a magnitude-8 earthquake on the southern San Andreas fault up to 2 Hz. M8 was calculated using a uniform mesh of 436 billion 40-m^3 cubes to represent the three-dimensional crustal structure of Southern California, in a 800 km by 400 km area, home to over 20 million people. This production run producing 360 sec of wave propagation sustained 220 Tflop/s for 24 hours on NCCS Jaguar using 223,074 cores. As the largest-ever earthquake simulation, M8 opens new territory for earthquake science and engineering—the physics-based modeling of the largest seismic hazards with the goal of reducing their potential for loss of life and property.**

***Keywords*—SCEC, computational seismology, earthquake ground motions, parallel scalability, extreme I/O, M8**

I. INTRODUCTION

Earthquake system science seeks to provide society with better predictions of earthquake causes and effects. Toward this goal, the Southern California Earthquake Center (SCEC) conducts a collaborative, inter-disciplinary research program within its Community Modeling Environment (CME) [25] that makes extensive use of large-scale, physics-based, numerical modeling of earthquake phenomena on TeraGrid and INCITE resources. The capability computing activities of the CME collaboratory—that is, our largest and computationally most demanding efforts—have been facilitated by the development of a highly scalable application called AWP-ODC. This paper describes for the first time AWP-ODC design and capabilities, focusing on the optimization techniques that have allowed the code to run efficiently on petascale supercomputers. The development of this scalable application as a community platform exemplifies the collaborative research now possible using petascale architectures. In particular, AWP-ODC now provides the capability to simulate the largest earthquakes anticipated on California's San Andreas Fault (SAF) system at the high shaking frequencies (> 1 Hz) required to understand seismic risk. SCEC's scientific goal is to apply this new tool in system-level efforts to mitigate life and property losses that could be caused by future earthquakes in California and elsewhere.

We demonstrate the scalability of AWP-ODC by simulating a magnitude-8 earthquake that ruptures the entire southern SAF from Cholame, in central California, to the Salton Sea, near the Mexican border—a total fault length of 545 km. This 'wall-to-wall' scenario, hereafter referred to as M8, required the computation of high-frequency ground motions throughout a very large simulation volume (810 km × 405 km × 85 km). The complex three-dimensional geologic structure of this vast region was derived from the SCEC Community Velocity Model V.4 (CVM4) and represented on a uniform mesh with 40-meter resolution, comprising a total of 436 billion cubic elements (Fig. 1). The source was calculated using a fully dynamic, spontaneous-rupture model that has been shown to produce ground motion levels in close agreement with ground motion prediction equations [39][41].

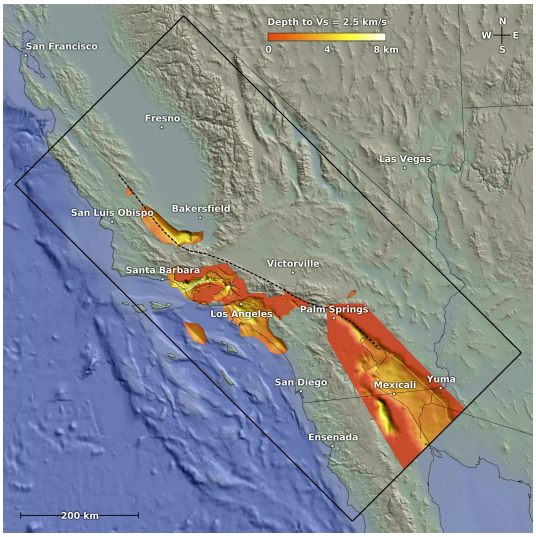

Fig. 1. Topographic location map for M8. The rectangle depicts the model area, the dashed line is the 545-km long stretch of the SAF that ruptured during M8, and the red/yellow/white shading shows the depth to the isosurface of a shear-wave velocity of 2.5 km/s.

SC10 November 2010, New Orleans, Louisiana, USA 978-1-4244-7558-2/10/$26.00

To our knowledge, M8 is the largest earthquake simulation ever conducted. It presented tremendous computational and I/O challenges that required collaboration of more than 30 seismologists and computational scientists. M8 was executed as a 24-hr production run on 223,074 cores of the Jaguar Cray XT5 at the National Center for Computational Sciences, which currently ranks first among the 'Top 500' of supercomputers [34].

Section 2 of this paper outlines the numerical method behind AWP-ODC. Section 3 details the implementation and components of AWP-ODC. Section 4 introduces some key tuning techniques we used for performance optimization. Section 5 demonstrates the parallel efficiency and summarizes the sustained performance achieved on petascale supercomputers. Section 6 reviews the SCEC milestone capability simulations based on AWP-ODC. Finally, Section 7 presents the ground-breaking scientific results of the M8 simulation on the Jaguar system at ORNL.

II. AWP-ODC Algorithm Description

A variety of numerical methods are available for modeling 3D modeling earthquake motion, including finite difference (FD), finite element, spectral element, and finite volume methods. While all of these computational techniques are capable of modeling spontaneous-rupture and wave propagation, the FD method provides the best trade-off in terms of accuracy, computational efficiency, and ease of implementation on massively parallel supercomputers.

AWP-ODC, which is an abbreviation of Anelastic Wave Propagation by Olsen, Day & Cui, is based on the FD code originally developed by Kim Bak Olsen at University of Utah [36]. The AWP-ODC code solves the 3D velocity-stress wave equations with the explicit staggered-grid FD scheme. This scheme is fourth-order accurate in space and second-order accurate in time. The code has undergone many developments and enhancements, which have transformed it from a 'personal' research code into a SCEC community modeling platform for capability simulations. The key components of the code include capabilities for spontaneous-rupture dynamic modeling on a vertical, planar fault (led by S. Day) as well as simulation of wave propagation (led by K. Olsen), with overall integration and enhancements for capability simulations led by Y. Cui.

A. Governing Equations

AWP-ODC solves a coupled system of partial differential equations. Let

$$\nu = \left(\upsilon_x, \upsilon_y, \upsilon_z\right)$$

denote the particle velocity vector, and

$$\sigma = \begin{pmatrix} \sigma_{xx} & \sigma_{xy} & \sigma_{xz} \\ \sigma_{yx} & \sigma_{yy} & \sigma_{yz} \\ \sigma_{zx} & \sigma_{zy} & \sigma_{zz} \end{pmatrix}$$

be the symmetric stress tensor. The governing elastodynamic equations can be written as

$$\partial_t \nu = \frac{1}{\rho} \nabla \cdot \sigma \tag{1a}$$

$$\partial_t \sigma = \lambda (\nabla \cdot \nu) \mathrm{I} + \mu (\nabla \nu + \nabla \nu^{\mathrm{T}}) \tag{1b}$$

where λ and μ are the Lamé coefficients and ρ is the density. Decomposing (1a) component-wise leads to three scalar-valued equations for the velocity vector components and six scalar-valued equations for the stress tensor components.

Seismic waves suffer anelastic losses in the Earth, and such attenuation must be included in realistic simulations of wave propagation. Anelastic attenuation can be quantified by quality factors for S waves (Q_s) and P waves (Q_p). Early implementations of attenuation models include Maxwell solids (e.g., [23]) and standard linear solid models (e.g., [6]). Day [17] and Day & Bradley [18] significantly improved the accuracy of the stress relaxation schemes by using a coarse-grained implementation of the memory variables, and this efficient technique has been implemented in AWP-ODC. This method closely approximates frequency-independent Q by incorporating a large number of relaxation times (eight in our calculations) into the relaxation function without sacrificing computational or memory efficiency [37].

B. Staggered-Grid Finite Difference Equations

The nine governing scalar equations are approximated by finite differences on a staggered grid in both time and space. Time derivatives are approximated by the following central difference equations

$$\partial_t \upsilon(t) \approx \frac{\upsilon(t + \frac{\Delta t}{2}) - \upsilon(t - \frac{\Delta t}{2})}{\Delta t} \tag{2a}$$

$$\partial_t \sigma(t + \frac{\Delta t}{2}) \approx \frac{\sigma(t + \Delta t) - \sigma(t)}{\Delta t} \tag{2b}$$

For the spatial derivatives, let Φ denote a generic velocity or stress component, and h be the equidistant mesh size. The FD approximation to $\partial_x \Phi$ at grid point (i,j,k) is

$$\partial_x \Phi_{i,j,k} \approx D_x^4(\Phi)_{i,j,k} = \frac{c_1(\Phi_{i+\frac{1}{2},j,k} - \Phi_{i-\frac{1}{2},j,k}) + c_2(\Phi_{i+\frac{3}{2},j,k} - \Phi_{i-\frac{3}{2},j,k})}{h} \tag{3}$$

with c_1=9/8 and c_2=-1/24. This equation is used to approximate each spatial derivative for each velocity and stress component.

C. Staggered-Grid Split-Node Algorithm for Dynamic Fault Rupture Modeling

A highly scalable dynamic fault rupture boundary condition has been integrated into AWP-ODC by Dalguer and Day. It is based on the accurate, and verified staggered grid, split-node (SGSN) scheme [14]. In this method, the fault divides the computational domain into two subregions (see Fig. 2), that are denoted as (+) and (-). Each side of the fault plane has its own set of 3 governing differential equations. Split nodes are introduced on the fault plane in order to account for velocity and stress disconti-

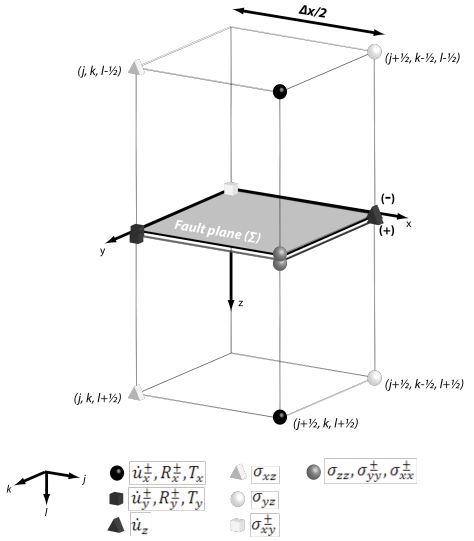

Fig. 2. Staggered-grid split-node geometry, illustrated for grid cells adjacent to the fault in the velocity-stress staggered-grid scheme. The fault plane grid points for u_x, u_y, σ_{xx}, σ_{yy}, and σ_{xy} are split into plus and minus sides. The finite difference equations of motion are partitioned to form separate elastic restoring forces (R) acting on the two halves. The two halves of a split velocity node interact only through shear tractions (T) at that node point [14].

nuities. The velocity or stress measured at a split node is labeled according to fault side on which it resides. In essence, the SGSN method calculates the traction that interacts between the two surfaces of the fault, and solves the equation of motion governing each side of this surface. The SGSN scheme can accommodate quite general friction laws and fault geometry, though AWP-ODC is currently limited to simulation of rupture on planar faults using a slip-weakening friction law.

Whereas time derivatives for the SGSN method remain centered 2nd-order accurate, the approximation of spatial derivatives depends on the proximity to the fault plane. The SGSN FD equations match the (4th-order accurate) staggered grid FD equations for points that are at least two spatial increments from the fault plane. For points less than two grid points from the fault plane, the accuracy of the FD equations is reduced to 2nd-order. Split nodes are introduced for certain stress and velocity components. Assuming that the fault plane is located at $y=j_0$, the following FD approximations to ∂_y are used for an arbitrary velocity or stress component Φ:

$$\partial_y \Phi_{i,j_0+\frac{3}{2},k} \approx \frac{c_1(\Phi_{i,j_0+2,k} - \Phi_{i,j_0+1,k}) + c_2(\Phi_{i,j_0+3,k} - \Phi^{+}_{i,j_0,k})}{h} \quad (4a)$$

$$\partial_y \Phi_{i,j_0+1,k} \approx \frac{\Phi_{i,j_0+\frac{3}{2},k} - \Phi_{i,j_0+\frac{1}{2},k}}{h} \quad (4b)$$

$$\partial_y \Phi_{i,j_0+\frac{1}{2},k} \approx \frac{\Phi_{i,j_0+1,k} - \Phi^{+}_{i,j_0,k}}{h} \quad (4c)$$

On the fault plane, spatial derivatives of velocity and shear stress components are computed by one-sided FD operators conforming to the traction continuity conditions.

D. External Boundary Conditions

Truncation of the 3D modeling domain on a computational mesh inevitably generates undesirable reflections. Absorbing boundary conditions (ABCs) are designed to reduce these reflections to the level of the numerical noise. AWP-ODC includes two different types of ABCs. The most efficient is the Perfectly Matched Layers (PML) scheme. PMLs were originally introduced for modeling of electromagnetic waves [2][3] and later adapted to elastodynamic waves on a staggered grid [30]. The PML ABCs can be formulated through a simple time-domain, equation-splitting procedure, where each wavefield equation is split into perpendicular and parallel components, with a damping term added to the perpendicular component. An example is wave propagation in the x-direction. Starting from (1a) and (1b), we split ∇ into normal and tangential operators $\nabla^{\perp x}$ and $\nabla^{\parallel x}$. If the velocity vector and stress tensor are similarly decomposed into normal and tangential components $v^{\perp x}$, $v^{\parallel x}$, $\sigma^{\perp x}$ and $\sigma^{\parallel x}$, the following decoupled partial differential equations are obtained:

$$\partial_t v^{\perp x} \cdot \sigma = \frac{1}{\rho} \nabla^{\perp x} \cdot \sigma \quad (5a)$$

$$\partial_t v^{\parallel x} \cdot \sigma = \frac{1}{\rho} \nabla^{\parallel x} \cdot \sigma \quad (5b)$$

$$\partial_t \sigma^{\perp x} = \lambda(\nabla^{\perp x} \cdot v)\mathrm{I} + \mu(\nabla^{\perp x} v + \nabla^{\perp x} v^{\mathrm{T}}) \quad (5c)$$

$$\partial_t \sigma^{\parallel x} = \lambda(\nabla^{\parallel x} \cdot v)\mathrm{I} + \mu(\nabla^{\parallel x} v + \nabla^{\parallel x} v^{\mathrm{T}}) \quad (5d)$$

Damping is applied to the wavefield components perpendicular to the boundary via a damping function $d(x)$, resulting in the updated equations:

$$\partial_t v^{\perp x} + d(x) v^{\perp x} = \frac{1}{\rho} \nabla^{\perp x} \cdot \sigma \quad (6a)$$

$$\partial_t \sigma^{\perp x} + d(x) \sigma^{\perp x} = \lambda(\nabla^{\perp x} \cdot v)\mathrm{I} + \mu(\nabla^{\perp x} \cdot v + \nabla^{\perp x} v^{\mathrm{T}}) \quad (6b)$$

Since the PML ABCs were introduced, Meza-Fajardo and Papageorgiou [32] developed the multi-axial PMLs (M-PMLs), which add efficiency by applying the damping separately for wavefields propagating parallel and perpendicular to the boundary. The M-PMLs were implemented on the sides and bottom of the grid in AWP-ODC, approximated with central differencing in both space and time. PML ABCs are efficient in both memory usage and computation time. Moreover, PMLs preserve the characteristics of nearest-neighbor interaction associated with the FD method, making it suitable for implementation on a massively parallel SIMD (single instruction, multiple data) computer.

Although efficient, the split-equation PMLs (and M-PMLs) are known to be numerically unstable in the presence of strong gradients of the media parameters inside the boundaries (e.g., [30]). In such cases, AWP-ODC implements a second kind of ABCs based on simple 'sponge layers' [9]. These ABCs apply a damping term to the full (un-split) wavefield inside the sponge layer and are unconditionally stable. However, the ability of the sponge layers to absorb reflections is poorer than PMLs. For our

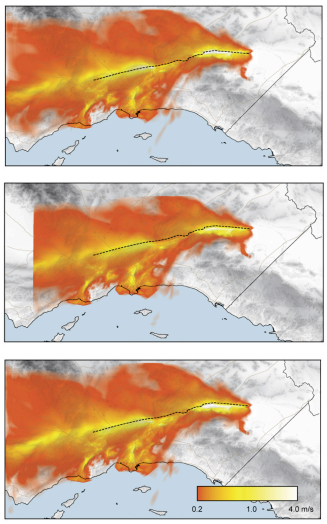

Fig. 3. PGVs for the ShakeOut simulation of a M7.8 scenario on the SAF using 3 independent codes. Top: finite-element (CMU); middle: FD (URS); bottom: AWP-ODC. The URS results were computed in a model smaller than that used by the other two studies.

M8 simulation we successfully used M-PMLs with a width of 10 grid points.

E. Free Surface Boundary Condition

At the top of our model we apply a zero-stress boundary condition, to simulate the (flat) free surface of the Earth. AWP-ODC uses the free surface boundary condition FS2 [22] that is defined at the vertical level of the σ_{xz} and σ_{yz} stresses.

F. Verification of AWP-ODC

AWP-ODC has been successfully verified by comparing dynamic rupture and wave propagation results to those from other numerical methods. For example, Fig. 3 shows the nearly identical peak ground velocities (PGVs) from three different 3D codes (one of which is AWP-ODC) for a simulation of a M7.8 earthquake ('ShakeOut') on the southern SAF [5]. Such verification is crucial during optimization and other code updates.

III. Implementation and Integration of AWP-ODC Components

The AWP-ODC software package has been designed in a modular fashion, aiming at providing efficient solvers for dynamic rupture and wave propagation at petascale computational levels. The implementation was developed with scalable mesh generation and partitioning in mind, with workflows applicable to diverse architectures. Although the development of an efficient FD code was a key challenge for AWP-ODC, input and output processing tools turned out to be equally important components for large-scale application. The final package was composed of pre-processing tools (Mesh Generator CVM2MESH, Petascale Mesh Partitioner PetaMeshP, Dynamic Source Generator dSrcG, Dynamic Source Partitioner PetaSrcP), solvers (Dynamic Fault Rupture Solver DFR, Wave Propagation Model AWM), and post-processing scripts (Validation Toolkit aVal, and derived Products dPDA, iRODS ingestion tool PIPUT). The I/O components are particularly critical for generating a high-resolution mesh and source, and delivering sub-mesh/source to hundreds of thousands of nodes. We have developed an end-to-end workflow package (E2EaW) to effectively consolidate distinct AWP-ODC modules. E2EaW contains a variety of tools and techniques that streamline the dynamic utilization of multiple components (Fig. 4).

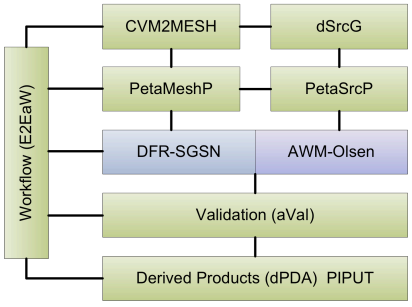

Fig. 4. Components of AWP-ODC

A. AWM and DFR Modules

The AWP-ODC FD scheme is designed for a structured grid in a Cartesian coordinate system. Communication in the code is performed through the Message Passing Interface (MPI) incorporating 3D domain decomposition. Data parallelism is the most efficient mechanism for parallel finite difference codes. Each processor is responsible for performing stress and velocity calculations within its own subgrid of the simulation volume. The processors allocated at the external edges of the volume must also process absorbing boundary conditions. Ghost cells, which occupy a two-cell padding layer, manage the most recently updated wavefield parameters exchanged from the edge of the neighboring subgrids (see Fig. 5 illustrated by M8).

The integration of AWM and DFR is illustrated in Fig. 6. The AWM and DFR algorithms begin with updates to the velocity vector at interior and boundary locations. Each processor then shares its locally updated velocity with its physically adjacent processors in all directions. The stress tensor is then updated in a similar way. Since wavefield information along the boundaries of the non-major axes is arranged into a large number of small non-contiguous chunks, intermediate staging buffers are introduced to accumulate and disseminate this information at the source

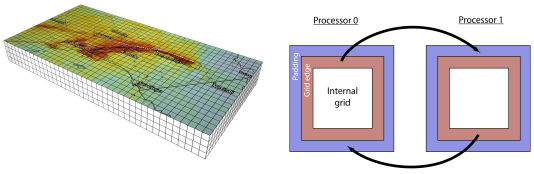

Fig. 5: (left) Decomposition of the M8 simulation region with 810 km long, 405 km wide and 85 km deep; (right) communication between neighboring subgrids

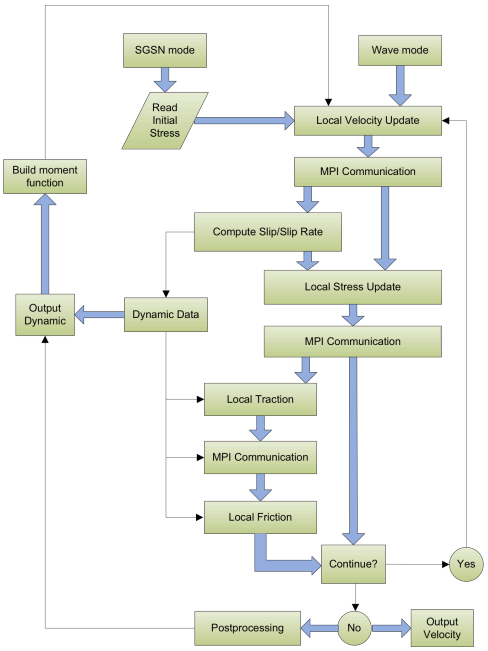

Fig. 6. Flow diagram showing the separation of AWP-ODC into a dynamic rupture modeling path ('SGSN mode') as well as a wave propagation path ('wave mode').

and destination, respectively.

B. Parallel Mesh Generator (CVM2MESH)

CVM2MESH is a software package containing a variety of mesh generation tools developed by Patrick Small and others at SCEC. The software supports both the SCEC CVM4 (rule-based interpolation approach) and Harvard CVM (CVM-H, static database at three resolutions) through a scalable parallel algorithm to extract material properties for the mesh at the required grid spacing. The program partitions the mesh region into a set of slices along the z-axis as illustrated in Fig. 7. Each slice is assigned to an individual core for extraction from the underlying CVM. This scheme achieves effective parallelization of the partitioning and extraction process. The cores interact only indirectly with the file system when the slices are merged into the final mesh file. Each core contributes its slice to the final mesh by computing the offset location of the slice within the mesh file, and uses efficient MPI-IO file operations to seek that location and write the slices. The parallel version of *CVM2MESH* has reduced the extraction time from hundreds of hours to minutes.

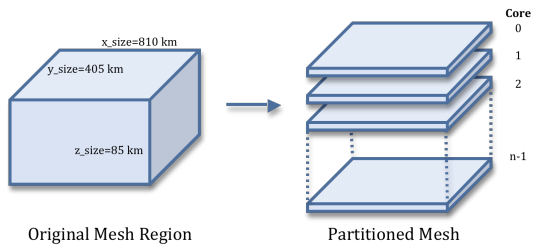

Fig. 7. The 3-D mesh region is partitioned into slices along the z-axis. Each slice is assigned to a core in the MPI job, and each core queries the underlying CVM for the points in its slice only.

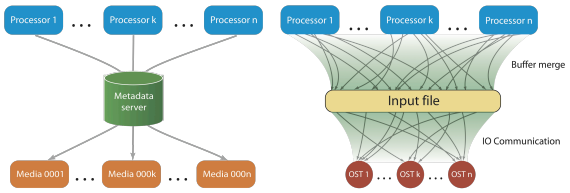

Fig. 8. (left) Serial approach to mesh partitioning; (right) parallel approach to mesh partitioning.

C. Petascale Mesh Partitioner (PetaMeshP)

CVM2MESH creates a single, global mesh file for the entire computational domain. A scalable and efficient mesh partitioning technique has been one of the critical challenges to prepare petascale seismic simulations on hundreds of thousands of processors, in particular when dealing with highly fragmented I/O data. Mesh partitioning aggregates regular but highly fragmented chunks of sub-mesh data and writes them back to per-process local contiguous files (pre-partitioning) or sends them to the dedicated processors (on-demand partitioning). Consequently, we have implemented two I/O models to deal with these issues: (1) serial I/O with input pre-partitioning, and (2) on-demand partitioning through MPI-IO [27] (Fig. 8).

The serial I/O model requires input partitioning prior to the simulation. Although many per-core partitioned small files are generated, this model provides efficient data locality and guarantees extensive use of the system I/O bandwidth. Unfortunately, such a per-processor file approach may encounter system-level issues by incurring excessive metadata operations and file system contention.

The MPI-IO model, on the other hand, allows all processes to issue parallel I/O accesses through the well-defined MPI-IO library. In general, the MPI-IO library provides good bandwidth and scalability when the input requests are well organized according to the system topology. Our *PetaMeshP* tools should theoretically work flawlessly on all systems. However, when dealing with hundreds of thousands of files, highly fragmented and scattered accesses can cause multiple unexpected system-level I/O issues on some file systems. Consequently, we try to minimize the fragmentation and scattering of the requests by restructuring the I/O scheme. First, a portion of processors ('readers') read highly contiguous big chunks of the mesh data with MPI-IO to maximize the I/O throughput. Second, the retrieved data are redistributed with point-to-point communication to the destination cores ('receivers') (see Fig. 9).

It is essential to minimize the average physical communication distance in the MPI-IO model. In the current implementation, each XY plane is read in parallel (based on MPI topology) and distributed to the associated receivers that require sub-XY layers to build a full local cube. We found that the communication overhead is highly tolerable using this approach, and the I/O performance and scalability are significantly enhanced. To further resolve potential

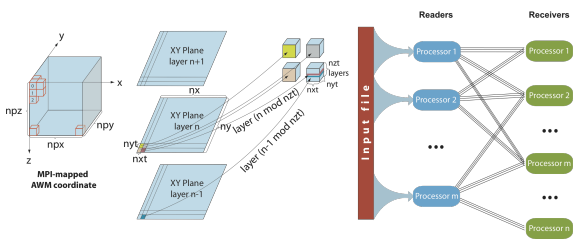

Fig. 9. (left) Cubes and (center) planes for contiguous burst reading and efficient data distributing; (right) high performance I/O with data redistribution

scalability problems caused by the excessive memory requirement on each reader (which can easily exceed the amount of usable memory of each core), a single XY plane can be subdivided along the Y-axis by a factor of *n*. This approach allows *n* times more readers to participate in reading sub-partitioned contiguous chunks of data (Fig. 9). Many experiments have been performed to test the performance of *PetaMeshP* on both GPFS and LUSTRE file systems. While the direct I/O model is good for the systems that have strong MDS tolerance, the advanced MPI-IO model works best for systems that provide highly scalable collective MPI file accesses.

D. Kinematic Source Generator (dSrcG) and Partitioner (PetaSrcP)

The AWM requires a kinematic source description formulated as moment rate time histories at a finite number of points (sub-faults). The moment rate time histories are stored in a file generated by the *dSrcG* (Kinematic Source Generator) tool. Once the moment-rate file is created, the Source Partitioner (*PetaSrcP*) distributes the source description to the associated processors. *PetaSrcP* incorporates an approach similar to parallel mesh partitioning. Special consideration must be taken for the irregular distribution of the source information. In general, the sources are highly clustered, and tens of thousands of sources can be concentrated in a given grid area, resulting in hundreds of gigabytes of source data assigned to a single core. To fit the large data into the processor memory, we further decompose the spatially partitioned source files by time. The scheme with both temporal and spatial locality significantly reduces the system memory requirements.

E. Parallel Output

Storing simulation results can be a significant bottleneck for large-scale FD simulations. AWP-ODC uses MPI-IO, allowing the velocity output to be concurrently written to a single file. To obtain efficient MPI-IO performance, we define new indexed data types at the initialization stage that represent segmented output blocks, and set logical file views for individual processors participating in file I/O. Instead of using individual file handles and associated offsets, we use explicit displacements to perform data accesses at the specific locations for all the participating processors [12]. By avoiding the use of file pointers, we save most of the I/O interactions.

As part of the MPI-2 standard, MPI-IO specifies the syntax and semantics of parallel I/O routines. However, an MPI-IO implementation is architecture dependent. Until a few years ago, AWP-ODC was one of the few data-intensive applications incorporating MPI-IO. Our experience from AWP-ODC has provided the opportunity to help compiler developers and system vendors, such as IBM, resolve multiple MPI-IO and GPFS related problems. To reduce I/O overhead, we set up a run-time environment that controls the frequency of I/O transactions at their lowest level. Consequently, the required velocity results are aggregated in memory buffers as much as possible before being flushed. This scheme works effectively in practice; in most cases, we have reduced the I/O overhead from 49% to less than 2%. The I/O aggregation is one of the critical features for scalability of the code and is particularly beneficial when many processors are involved.

We separate the generation of volume and surface velocity outputs for the convenience of data analysis and visualization. To track and verify the integrity of the simulation data collections, we generate MD5 checksums in parallel at each processor for each mesh sub-array. The parallelized MD5 approach substantially decreases the time needed to generate the checksums for several terabytes of data.

F. Checkpointing

A large-scale seismic simulation can take tens of hours to complete. For such cases, application level tolerance against various kinds of system failures is critical and should be dealt with by providing checkpoint/restart capabilities. All simulation states consisting of all the internal state variables on each processor are periodically saved into reliable storage where each processor is responsible for writing and updating its own checkpoint data. We use the system-level checkpointing library on BG/L, and incorporated IBM application assisted checkpoint/restart library calls into the application to synchronize all tasks and write the states of all processors to disk when needed. This approach helps restart in the case of unexpected termination.

Our current checkpoint approach, while an essential component of the program, may introduce a significant amount of overhead into some file systems. A more advanced application-based fault-tolerance checkpoint/restart feature is thus being developed [11]. Our fault tolerance framework is different in the sense that the surviving application processes will not be automatically aborted if only a small number of application processes fail. Instead, all non-failing processes will continue to run and the program environment adapts to the previous failures.

G. Adaptation of Algorithms Supporting Different Architectures

Careful coordination is required to handle each system's distinct characteristics including memory architectures, file systems, interconnects, and compilers. Our experience

indicates that application performance can be significantly affected by determination of fundamental system attributes [13]. Consequently, special algorithms have been implemented to adapt AWP-ODC to different architectures. A unique feature facilitates a run-time simulation configuration that is able to determine architecture-dependent handling to maximize our solver and/or I/O performance. The configuration specifies a memory buffer allocation for buffer aggregation. The optimizations are adjusted at run-time and depend on memory usage, I/O bandwidth, interconnect and other architectural characteristics. Alternative options also include selection of cache blocking size, communication models (asynchronous, computing/communication overlap), the selection of spatial and temporal decimation of outputs, serial pre-partitioned or parallel on-demand I/O, the inclusion of parallel checksums, and collection of performance characteristics.

H. Automated Verification (aVal)

The AWP-ODC code has been continuously updated as new capabilities improved efficiency and new platform adjustments are required. Acceptance testing of the updated software is therefore mandatory. We have developed a multi-step process of configuring a reference problem, running a simulation, and comparing results against a reference solution. This test uses a simple least-squares (L2 norm) fit of the waveforms from the new simulation and the 'correct' result in the reference solution [29].

I. End-to-End Workflow (E2EaW)

Typical large-scale seismic simulations use up to hundreds of thousands of cores that require terabytes of input and produce tens of terabytes of output. A large fraction of the output must be transferred to an archival storage and utilized for intensive analysis. These tasks challenge most computing and storage systems and they require coordination and an automated workflow. We have developed an end-to-end workflow that executes the simulation and automates archival to the SCEC digital library. The workflow uses GridFTP for high performance data transfer between sites and does not require human intervention [45]. Execution of the workflow is distributed across several TeraGrid machines and is controlled remotely. Fig. 10 depicts the automated end-to-end workflow procedures.

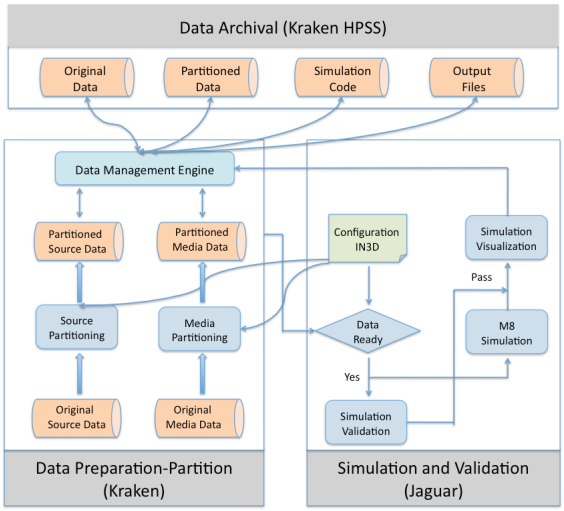

Fig. 10. Workflow E2EaW illustration: data partitioning on Kraken, solver simulation on Jaguar and data archival on Kraken HPSS.

The Globus toolkit is used to remotely spawn multiple parallel processes for MD5 checksum generation of all data files and places them in the destination machine. Once these tasks are completed, the verification of MD5 checksums is pipelined and performed in parallel. The parallel implementation significantly reduces the transfer time. The average transfer rate is above 200 MB/sec. In the event of file transfer failures, the transaction records are maintained to allow automatic recovery and retransfer.

Our 200 TB digital collection of science-based simulations [20] is managed through iRODS (integrated Rule Oriented Data System) [24]. We use our own toolkit (PIPUT) to ingest data into the SCEC digital library at an aggregated transfer rate of up to 177 MB/sec [45], more than ten times faster than direct use of single iRODS iPUT. These capabilities allow the modeler to access the files registered into the data grid, with associated integrity (MD5 checksum) and replica information. The workflow has been enhanced through the incorporation of derived data analysis products (dPDA) and our advanced vector visualization techniques [31].

IV. Performance Tuning

AWP-ODC has undergone extensive fine-tuning over the past 6 years. While some of the results have been addressed in the previous section (e.g., I/O), we especially highlight the recent performance optimizations in this section.

TABLE 1
Computers Used by Model for Production Runs

Computer	Location	Processor	Interconnect	Peak Gflops	Cores used
DataStar Power4	SDSC	1.5/1.7Ghz IBM Power4	Custom Fat Tree	6.0/6.8	2K
Ranger Sun Const.	TACC	2.3-Ghz AMD Barcelona	InfiniBand Fat Tree	9.2	60K
BGW BG/L	IBM Watson	700-Mhz PowerPC	3D Torus	2.8	40K
Intrepid BG/P	ANL	850-Mhz PowerPC	3D Torus	3.4	128K
Kraken Cray XT5	NICS	2.6-Ghz Istanbul	SeaStar2+ 3D Torus	10.4	96K
Jaguar Cray XT5	ORNL	2.6-Ghz Istanbul	SeaStar2+ 3D Torus	10.4	223K

Table 1 summarizes the key characteristics of the machines used in our study. The Jaguar system is the most powerful of those listed above. Jaguar's compute node contains two hex-core AMD Opteron processors, 16GB of memory, and a SeaStar 2+ router with a peak bandwidth of 57.6GB/s. The routers are connected in a 3D torus topol-

ogy that provides an interconnect with very high bandwidth, low latency, and good scalability in its interconnect [34].

A. Communication

AWP-ODC partitions the simulation volume into smaller sub-domains where the total number of subdomains matches the number of processors used in the simulation. In the partitioned space, wave propagation information must be shared among neighbors. The fourth-order FD scheme requires a two-cell padding for the outside of each sub-grid to correctly propagate waves. The number of variables to be propagated and the number of surrounding neighbors determine the amount of communication. To propagate waves from one end of the computational domain to the other, long linear communication links can be formed causing severe communication latency.

Generally, communication latency between cores is highly dependent on their physical interconnect distance and the node topology in NUMA (Non-Uniform Memory Architecture) systems. The link among cores in a node may also be asymmetrical, causing further latency variations. TACC Ranger, NCCS Jaguar, and ANL Intrepid systems are some of the largest systems that incorporate NUMA architecture.

AWP-ODC's main loop is composed of many computational, communication and I/O procedures. To ensure consistency, barriers are inserted. However, the local nature of computation and I/O within a subdomain makes many of the barriers unnecessary. Global synchronization created by barriers is redundant.

Originally, AWP-ODC adopted a synchronous communication model. In this model, the communication path is cascaded through multiple mpi_send/mpi_recv call pairs and latency is accumulated along the path. This makes the accrued latency on each core highly dependent on the number of cores in the communication path. The synchronous communication model is not a particular issue on single-socket based 3D torus architectures found on BG/L and Cray XT4. However, in NUMA systems, the number of sockets accessing the 3D torus network tends to increase the communication latency. For example, we observed a drop of parallel efficiency from 96% on BG/L to 40% on BG/P on 40K cores.

To deal with this NUMA architecture scalability issue, we redesigned the communication model with asynchronous communication. This change effectively reduces communication overhead caused by unnecessary interdependence among communication nodes. To use MPI asynchronous communication calls, a mechanism is needed to synchronize the overall communication flow. The former synchronous code was directly converted to an asynchronous code with unique tagging to avoid source/destination ambiguity. This new model allows out-of-order arrival and the unique tags maintain data integrity (Fig. 11). This scheme significantly reduces the latency caused by fine-grained arrival/departure order control and the redundant synchronization mechanism.

The extreme communication overhead inherent in the synchronous model on NUMA architectures causes critical performance and scalability degradation of the code. The asynchronous communication model effectively removes the interdependency among nodes that do not show any temporal dependence, resulting in highly balanced and low latency communication. The optimized communication code run on Ranger with 60K cores reduced the total time to 1/3 of that consumed by the synchronous version of code [26]. The parallel efficiency increased from 28% to 75%.

In addition to the asynchronous communication improvement, we removed redundant communication. During the communication, six stress tensor components (*xx, xy, xz, yy, yz, zz*) and three velocity components (*u, v, w*) need to be updated at each time step. Consider the following Fortran code.

```
u(i,j,k)=u(i,j,k)+(dth/d)*(
    +c1*(xx(i,j,k)-xx(i-1,j,k))+
    +c2*(xx(i+1,j,k)-xx(i-2,j,k))+
    +c1*(xy(i,j,k)-xy(i,j-1,k))+
    +c2*(xy(i,j+1,k)-xy(i,j-2,k))+
    +c1*(xz(i,j,k)-xz(i,j,k-1))+
    +c2*(xz(i,j,k+1)-xz(i,j,k-2)))
```

For the stress tensor component *xx*, only *xx(i,j,k)*, *xx(i+1,j,k), xx(i-1,j,k) and xx(i-2,j,k)* are needed to calculate the updated *u(i,j,k)*. This implies that we only need to update *xx* in the *x* direction rather than in all three directions. By sending two plane faces of *xx* information to the left neighbor and one plane to the right neighbor only in the *x* direction, we can reduce the *xx* message communication by 75%, achieving an additional 15% in wall clock time for the full-scale Jaguar execution.

B. Single-CPU Optimization

We used performance tools IPM and CrayPAT to analyze single CPU performance and find "hot spots" where most of the execution time is spent. The profiling results show that seven computing-intensive subroutines account for about 40% of the total application execution time.

Initially, the focus was to reduce expensive division operations in frequently used formulas. The Lamé parameter arrays *mu* and *lam* are computed once and remain un-

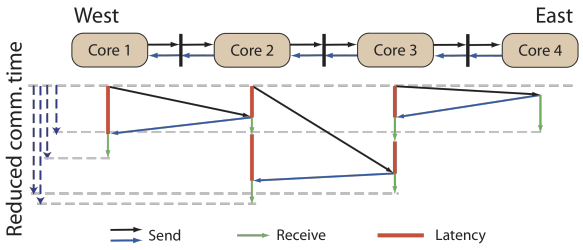

Fig. 11. Round-trip communication latency in the asynchronous model.

changed during the entire simulation. Elements in the arrays *mu* and *lam* are used in the forms *mu(i,j,k)* and *1./mu(i,j,k)*, but only the reciprocal form is used in frequently invoked subroutines [35]. For this reason, we store the reciprocals of *mu* and *lam* rather than the arrays themselves throughout the solver.

The critical subroutines share the same three nested-loop structure. Each node executes the loop with its processor's local mesh, which achieves good memory access behavior. However, the cache utilization rate is very low, primarily due to the requirement of assessing values in multiple 3D arrays with varying second or third indices. When one of these values is accessed, the whole cache line containing the value is fetched into the L1 cache. Since the number of variables in the inner loops is large, a cache line is usually evicted from the L1 cache right after being referenced. To improve cache utilization, we need to access as many values per cache lines loaded as possible.

The cache blocking technique provides better cache utilization. A 3D difference algorithm can be extremely limited in memory bandwidth, and the number of variables that must be fetched from memory is relatively high given the computation performed. If the required operands are in cache, the effectiveness of cache reuse in the difference code is low since the amount of data during the computation of an entire plane will exceed the size of the L1 and L2 caches. Consider the following code for calculating the harmonic mean of the Lamé parameter *lam*:

```
do k = nzb,nze
    do j = nyb,nye
    do i = nxb,nxe
         o          o          o
    xl=8./(lam(i,j,k)+lam(i+1,j,k) +
    +     lam(i,j-1,k)+lam(i+1,j-1,k) +
    +     lam(i,j,k-1)+lam(i+1,j,k-1) +
    +     lam(i,j-1,k-1)+lam(i+1,j-1,k-1))
         o          o          o
```

For any reasonably sized grid, the lines containing the variables from the *j-1* and *k-1* planes will not be located in cache when the difference equation is performed on the next plane. If the grid is subdivided into smaller sub-grids, the operands from the *j-1* and *k-1* planes may still be in cache as the computation progresses. This can be accomplished by forming a memory block for the *k* and *j* loops as described in the following Fortran code:

```
do kk = nzb,nze,kblock
    do jj = nyb,nye,jblock
    do 20 k= kk,min(kk+kblock-1,nze)
    do 20 j= jj,min(jj+jblock-1,nye)
    do i=nxb,nxe
         o          o          o
    xl=8./(lam(i,j,k)+lam(i+1,j,k) +
    +     lam(i,j-1,k)+lam(i+1,j-1,k) +
    +     lam(i,j,k-1)+lam(i+1,j,k-1) +
    +     lam(i,j-1,k-1)+lam(i+1,j-1,k-1))
         o          o          o
```

The values of *kblock* and *jblock* are chosen to guarantee that the operands on subsequent planes are still in cache while those planes are computed, so that each grid point will be accessed eight times. If the cache blocking is perfect, then a variable will only be fetched from memory once and the other seven fetches will be from the L1 or L2 cache. The values of *kblock* and *jblock* are dependent upon the number of operands accessed within the DO loop. For a typical loop length of 125, the optimal solution was found to be 16/8. The variation between different combinations is around 3%.

We also use explicit loop unrolling techniques to improve cache utilization, even though some compilers automatically incorporate this form of optimization. Since the number of registers on a processor is limited, over-unrolling could deteriorate loop performance. Empirical studies prove that unrolling by 2 iterations gives the best performance for the computing-intensive subroutines *xyq* and *xzq*.

We implemented additional manual transformations of the major computational subroutines that cannot be vectorized automatically. For example, we eliminated a *mod* function from the innermost computing loop by replacing it with *itx = 3 - itx,* where the value of *itx* alternates between 1 and 2. Removal of the *mod* enabled the compiler to vectorize the arithmetic in the compute-intensive loop.

In summary, the benefit of the proper restructuring of the computational intensive routines is significant. On Jaguar, for instance, we obtained a performance gain of 40% at full system scale, with contributions of 31% from arithmetic optimization, 2% from loop unrolling and 7% from cache blocking.

C. Communication and Computation Overlap

Most of the communication in AWP-ODC is point-to-point between nearest neighbors, making the performance of AWP-ODC heavily dependent on system interconnect bandwidth. DK Panda and his team at The Ohio State University (OSU) have developed non-blocking two-sided computation and communication overlap to further improve the parallel efficiency of AWP-ODC [42].

The velocity and stress exchange routines account for most of the MPI time in our application. The velocity and stress variables are composed of large amounts of small non-contiguous chunks in memory. To enhance the overall communication throughput, each process accumulates these chunks into contiguous staging buffers and the corresponding neighbor disseminates the received data. The computations of different velocity vector and stress tensor components are independent of one-another. Hence, we divide computation and communication per component and interleave them with each other. For example, velocity computation and exchange are split into three parts based on the *u, v* and *w* components. While the value of *v* is computed, the exchange of *u* can be performed simultaneously to provide efficient computation/communication

overlap. Similarly the exchange of *v* is overlapped with computation of *w*. For each component, non-blocking two-sided calls are used to initiate exchange of each velocity component and then wait on the completion of all these transfers using MPI_Waitall calls in the end after the exchange of *w* is initiated. A similar process is employed for the stress tensor components.

With this overlap implementation, we obtained an elapsed time performance gain of 11% and 21% on 65,610 cores on XT5 with PGI and Cray compilers, respectively. However, the performance was limited by the skew created by the load imbalance between the boundary and interior areas at full machine scale. For this reason, we combined the cache blocking technique with overlap to get consistent performance gains independent of processor counts. This was largely due to more efficient computation resulting in reduction of the skew. The OSU group has further used MPI-2 one-sided semantics with finer grained overlap to achieve additional performance gains for AWP-ODC on MVAPICH2-based TACC Ranger [42]. Unfortunately, the current MPI library of XT5 does not support MPI-2 one-sided operations and thus cannot take advantage of this optimization.

D. Load Balancing by Exploiting Hybrid Multithreads

By analyzing AWP-ODC with performance tools, we were able to reduce the load imbalance by more than 35% at full machine scale. This was achieved by incorporating an MPI/OpenMP hybrid approach. Such an approach can effectively resolve the load balancing issue and reduce memory traffic, independent of the architecture. In the hybrid method, multiple OpenMP threads, spawned from a single MPI process, directly access shared memory space within a node. On a NUMA architecture, hybrid methods can synchronize at memory requests instead of barriers, and reduce memory latency and data movement within a node by thread and data collocation. Hybrid methods also eliminate fine-grained intra-node domain decomposition.

We produced an MPI/OpenMP hybrid implementation of AWP-ODC to exploit the performance gain over the pure MPI implementation. While the hybrid approach reduces the load imbalance, it introduced significant idle thread overhead. When the processor count approaches the arithmetic limits of the subdomain decomposition, this overhead may offset the entire performance gain. Especially for the large-scale runs where communication and synchronization overhead dominate the simulation time, the pure MPI code still performs better than the MPI/OpenMP hybrid code. We will continue to investigate this issue as MPI thread support evolves.

E. Lustre I/O Optimization

At a large core count, per-process serial I/O performance can be hindered by the collection of metadata operations or file system contention. On BG/P, for example, the simultaneous reading of the pre-partitioned mesh at more than 100K cores failed due to the aforementioned problems. To minimize the impact of this issue, we implemented a simple I/O approach by constraining the number of synchronously opened files to control the number of concurrent requests hitting the metadata servers. This approach has also been applied to the checkpointing scheme to improve the read/write performance of checkpoint files or pre-partitioned velocity data at large core counts. Not only does the method avoid system contention, but it also improves the I/O rate. We observed a significant enhancement in aggregated I/O throughput to more than 20 GB/s on Jaguar.

File striping within the file system can also improve I/O performance. On Lustre file systems, we use the *lfs setstripe* command to distribute the given file contents across the maximally available object storage targets (OSTs). This enables concurrent reading/writing and provides an overall superior I/O rate. We place different classes of files into different directories for easy setting of distinct striping configurations. The stripe count is set to a manageable size for the single large source and for the mesh files, which are located in the *input* directory accessed by simultaneous reads from multiple processors through MPI-IO. The stripe size is set to unity for serial access of pre-partitioned input files and checkpoints, while simulation outputs folders use a large striping count that depends on the actual output operations.

V. Scalability

AWP-ODC exploits parallelism in various ways and at various levels of granularity as the capabilities of the code have evolved. Table 2 summarizes this evolution.

TABLE 2
Evolution of AWP-ODC

Year	Code version	Simulations	Optimization	SCEC alloc. SUs	Sustain. Tflop/s
2004	1.0	TeraShake-K	MPI tuning	0.5M	0.04
2005	2.0	TeraShake-D	I/O tuning	1.4M	0.68
2006	3.0	PN MQuake	partition. mesh	1.0M	1.44
2007	4.0	ShakeOut-K	incorp. SGSN	15M	7.29
2008	5.0	ShakeOut-D	asynchronous	27M	49.9
2009	6.0	W2W	single CPU opt	32M	86.7
2010	7.0		overlap		
	7.1	M8	cache blocking	61M	220
	7.2		reduced comm		

A. Parallel Efficiency

The execution time for each time step can be decomposed into five categories: computation, communication, synchronization, re-initialization of the source, and output generation:

$$T_{tot} = T_{comp} + T_{comm} + T_{sync} + \gamma T_{output} + \varphi T_{reini} \qquad (7)$$

In this time-dependent expression, T_{tot} denotes total execution time, T_{comp} is the pure computational time, T_{sync} and T_{comm} denote synchronization and communication times,

respectively, T_{output} is output time, and T_{reini} refers to the re-initialization time due to temporal partitioning of large-sized dynamic sources. The parameters γ and φ indicate the I/O operation rate, divided by the fixed number of iterations chosen before the simulation. Since reinitialization is performed infrequently, T_{reini} is significantly smaller than the other terms, due to the extremely fast local reading of the pre-partitioned sources, allowing it to be safely omitted (for M8 $\varphi = 1/3000$). For the M8 run, we aggregate outputs in buffers and write output every 20K time steps ($\gamma = 1/20000$), resulting in minimal I/O time per iteration. Reduction in total execution time through aggressive computation, synchronization, and communication optimization can positively impact the overall performance.

Minkoff [33] provided a useful estimate of communication cost that can be applied to AWP-ODC. To obtain a speedup formula, we define the following two factors: average latency denoted by α and average bandwidth denoted by $1/\beta$. With this notation, the cost to send a single message with k units of data is represented by $\alpha + k\beta$. Since our 3D grid contains $N = NX \times NY \times NZ$ grid points with a topology consisting of $P = PX \times PY \times PZ$ processors, our speedup can be re-defined as:

$$\frac{T(N,1)}{T(N,p)} = \frac{C_\tau \cdot N}{C_\tau \cdot \frac{N}{p} + 4 \cdot (3\alpha + 8\beta \frac{NX \cdot NY}{PX \cdot PY} + 8\beta \frac{NX \cdot NZ}{PX \cdot PZ} + 8\beta \frac{NY \cdot NZ}{PY \cdot PZ})} \quad (8)$$

where the factor C accounts for the number of floating point operations, the FD stencil, and the fact that there are nine output quantities to update (six stress tensor and three velocity vector components), see section 2. τ stands for machine computation time per flop. For example, on Jaguar, the following values were estimated: $\alpha = 5.5\times10^{-6}$ s, $\beta = 2.5\times10^{-10}$ s, and $\tau = 9.62\times10^{-11}$ s. This calculation, combining machine characteristics and our application code performance, demonstrates a 2.20×10^{5} speedup or 98.6% parallel efficiency on 223K Jaguar cores. Our production simulations at this scale display ideal speedup which is consistent with the theoretical parallel efficiency calculations discussed here.

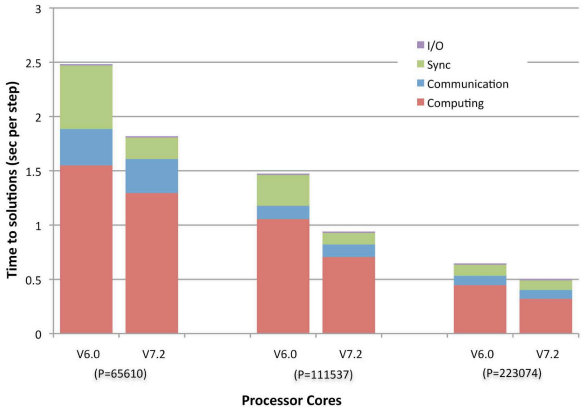

Fig. 12. Breakdown of execution time into computing, communication, synchronization, and I/O time for the M8 settings on Jaguar at ORNL. Execution time per step using (left) v. 6.0, and (right) v. 7.2 with cache blocking and reduced algorithm-level communication.

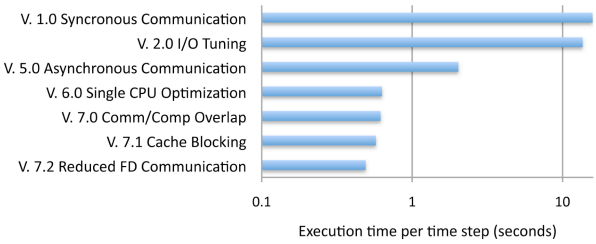

Fig. 13. Reduction of time-to-solution per time step achieved for each new version of AWP-ODC on NCCS Jaguar.

To further differentiate the parallel performance between the current version (v. 7.2) and the previous version (v. 6.0) that lacks cache blocking and the reduced algorithm-level communication, we collect the timing information for the communication and synchronization phases. Fig. 12 details T_{tot} spent for each fragment between 65,610 and 223,074 cores. I/O time is between 0.6% and 2% of the total time, and T_{comp} is heavily dependent on local floating-point operations. This demonstrates a super-linear speedup due to efficient cache utilization. As the problem size per processor reduces, the core data set sufficiently fitsinto L1/L2 cache, and the memory access time subsequently decreases. Our CrayPAT report indicated that the cache blocking optimization directly contributed to the reduction of T_{comp}, while the reduced communication optimization caused a simultaneous decrease of T_{comm} and T_{sync}. Here, T_{comm} includes the time spent at MPI_Waitall calls, since pure point-to-point communication time is only 0.2% of the total execution time. T_{sync} is mostly composed of a single MPI_Barrier call per iteration.

With regard to weak scaling, benchmarks on BG/L using a synchronous communication scheme demonstrated ideal scaling up to 32,768 cores. On Jaguar, we measured 90% parallel efficiency for weak scaling between 200 and 204K processor cores. The primary factor responsible for the degradation in performance was the load imbalance caused by the variability between boundary and interior computational loads and the increase of the communication-computation ratio.

The diverse optimizations of AWP-ODC significantly improved parallel efficiency. On Jaguar at full system scale, loop-level optimization improved the code performance by 40% (reduced division operations 31%, unrolling 2%, cache blocking 7%), computation/communication overlap by 11% (not included in v. 7.2), and reduced algorithm-level communication by 15% (see Fig. 13). Two other outstanding improvements include the incorporation of an asynchronous communication model (achieved more than ~7x reduction in wall clock time on 223K Jaguar cores) and reduction of I/O time from the original 49% to less than 2% of the wall clock time. Finally, the simultaneous reading of the multi-terabytes mesh/source inputs by thousands of processor cores has significantly reduced initialization time to a few minutes. Fig. 14 details the scalability improvements.

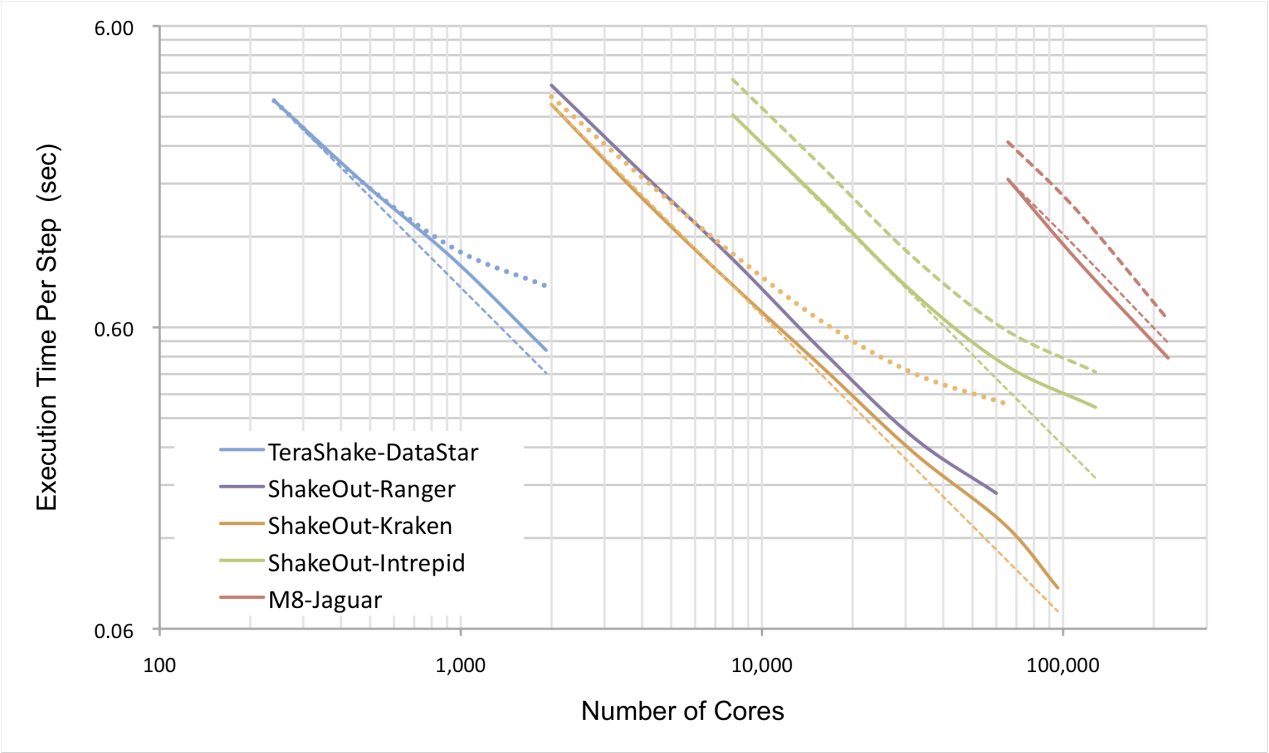

Fig. 14. Strong scaling of AWP-ODC on TeraGrid and DOE INCITE systems. Progressive improvements are illustracted for the following SCEC milestone simulations: 1.8 billion grid-point *TeraShake* (version 1.0/2.0 on DatarStar w/o I/O tuning), 14.4 billion grid point *ShakeOut* (v. 5.0/6.0 on Intrepid w/o single-CPU optimization, v. 6.0 on Ranger, v. 4.0/5.0 on Kraken w/o asyncronous communication), and 436 billion grid point *M8* (v. 6.0/7.2 on Jaguar w/o cache blocking and reduced algorithm-level communications). Solid lines are scaling after optimizations, square dotted lines denote scaling before optimization, and dashed lines are ideal cases. Super-linear speedup occurs for M8 on NCCS Jaguar.

B. Sustained Performance

We present two sustained performance estimates based on 223,074 Jaguar cores. The first estimate is a 2,000 time-step benchmark of a wall-to-wall scenario of dimensions 750 km × 375 km × 79 km with a spatial resolution of 25 meters and a maximum frequency of 2 Hz, discretized into 1.4 trillion mesh points. This benchmark was made in preparation for our 2011 PRAC project on NCSA Blue Waters, where we plan a slightly larger simulation. The second estimate is the M8 simulation.

Both the benchmark and M8 runs produced remarkable results. By documenting PAPI calls, we recorded the benchmark and M8 simulations to run at sustained rates of 260 Tflop/s and 220 Tflop/s, respectively. The average floating point operations per second is based on the report by PAPI_FP_OPS divided by measured wall-clock time. We emphasize that the sustained performance is based on the 24-hour M8 production simulation with 6.9 TB input and 4.5 TB output, *not a benchmark run*. Such performance results are particularly remarkable considering that stencil computations typically achieve a low fraction of theoretical peak performance.

VI. SCEC Milestone Capability Simulations Using AWP-ODC

In this section, we review significant SCEC milestone simulation efforts carried out in recent years based on the AWP-ODC application (see Table 3). Table 5.1 shows how the sustained performance of AWP-ODC increased as the NSF computer allocations grew in the past 5 years. The first important computational milestone obtained in 2004 was TeraShake-K (hereafter referred to as TS-K) [38], which modeled the effects of a M_w7.7 earthquake on a 200-km stretch of the southern SAF up to 0.5 Hz in a 600 km × 300 km area of southern California using a 1.8-billion grid point model. TS-K generated 53 TB of 3-D volume data for scientific analysis and was a breakthrough in computational seismology at the time. TS-K identified the critical role of a sedimentary waveguide along the southern border of the San Bernardino and San Gabriel Mountains in channeling seismic energy into the heavily populated San Gabriel and Los Angeles basin areas for rupture on the southern SAF from SE to NW. In contrast, NW-SE rupture on the same stretch of the SAF generated orders-of-magnitude smaller peak motions in Los Angeles (Fig. 15).

TABLE 3
SCEC SIMULATIONS BASED ON AWP-ODC BY NAME, MAXIMUM FREQUENCY, SOURCE DESCRIPTION, AND YEAR CONDUCTED

Sim w/ AWP-ODC	Description
TeraShake 240 DataStar cores 1024 IA-64 cores	TS-K: Mw7.7, 0.5Hz, kinematic source descriptions based on the Denali (2002) event. TS-D dynamic source based on 1992 Landers initial stress conditions. (2005-2006)
Pacific Northwest MegaThrust 6K SDSC BG/L cores	Long period (0-0.5Hz) ground motion for Mw8.5 and Mw9.0 earthquakes in a new 3D Community Velocity model of the Cascadia subduction zone. (2007)
ShakeOut 16K Ranger cores	SO-K, Mw7.8, 1.0Hz, kinematic source based on geological observations (2007) SO-D, SGSN-based dynamic source (2008)
FD3T ANL BG/P	Full 3D tomography of TS using full physics of 3D wave propagation (2008)
W2W 96K Kraken cores	Mw8.0, 1.0Hz, SGSN-based source, combined Mw7.8 sources (2009)
M8 223K Jaguar cores	Mw8.0, 2.0-Hz, 40m spacing and 436 billion mesh points, SGSN-based source, a new record (2010)

The TS-K simulations used a kinematic source description based on observations of the 2002 M7.9 Denali earthquake rupture scaled to a magnitude 7.7. The TS-K source was relatively smooth in its slip distribution and rupture characteristics, owing both to resolution limits of the Denali source inversion and simplifications imposed by the kinematic parameterization (see Fig. 16). Kinematic source descriptions are often strong simplifications of the earthquake rupture process, usually not constrained by physical properties of faults and earthquake ruptures such as friction laws and stress conditions.

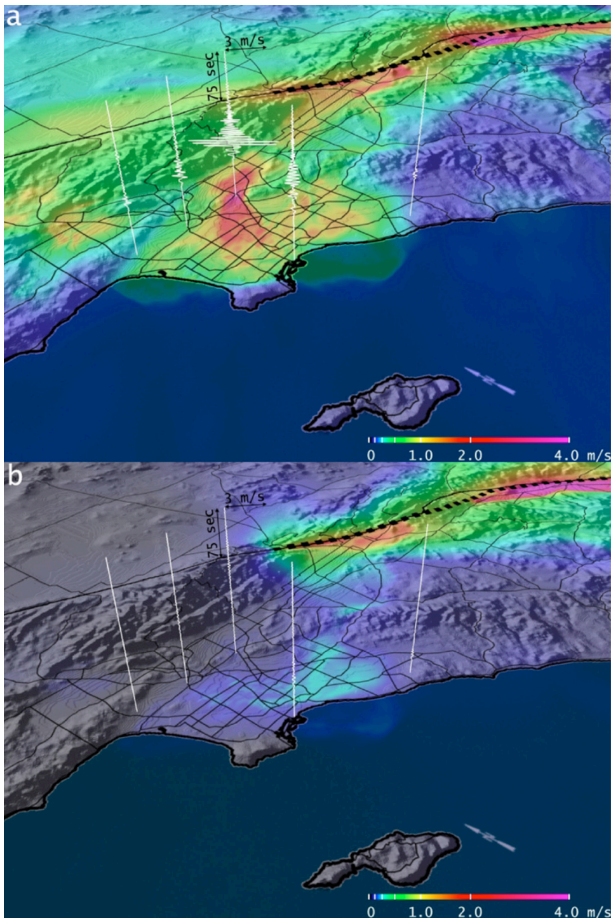

Fig. 15. Maximum RMS PGVs for the TS-K ruptures. N50W seismograms are superimposed at locations (from left to right) Westwood, downtown Los Angeles, Montebello, Long Beach, and Irvine. (a) SE-NW and (b) NW-SE scenarios. Modified after Olsen et al. [38].

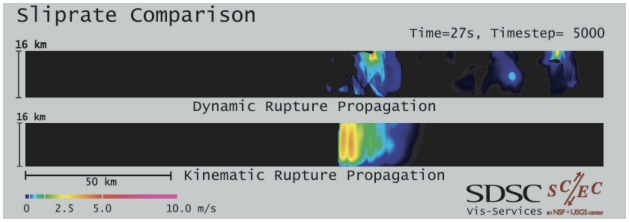

Fig. 16. Snapshots of slip rate for dynamic (top, TS-D) and kinematic (bottom, TS-K) rupture 27.5 sec after initiation. Modified after Olsen et al. [39].

In order to examine the effects of using a more complex, physically-constrained source, the TeraShake-D (TS-D) simulations were executed using dynamic rupture simulations [41]. The dynamic rupture simulations generated by TS-D were derived from a spontaneous rupture model with small-scale stress-drop heterogeneity based on inferences from the 1992 Landers earthquake (Fig. 16). The TS-D source models show average slip, rupture velocity and slip duration that are nearly the same as the corresponding values for the TS-K sources, but the ground motion predictions for the two source types are significantly different. In particular, the increased complexity of the TS-D sources decreases the largest peak ground motions associated with the wave guides and deep basin amplification by factors of 2-3. This general reduction in overall ground motion is attributed to the less coherent wavefield radiated by the TS-D sources.

Another notable characteristic feature in the TS-D ground motion distributions is the ‘star burst’ pattern of increased PGVs radiating out from the fault. These rays of elevated ground motions are generated in areas of the fault where the dynamic rupture pulse changes abruptly in speed, direction, or shape (Fig. 17). For this reason, the bursts of elevated ground motion are also correlated with pockets of large, near-surface slip rates on the fault. This pattern is absent from the PGV distributions for the TS-K simulations owing to the limited variation in rupture speed and constant shape of the source time functions. While dynamic rupture-based source descriptions are more computationally expensive to produce, they add important physical constraints to the source that are usually missing from kinematic descriptions.

TeraShake was followed by the ShakeOut simulations. The Great Southern California ShakeOut was an earthquake preparedness exercise mounted by the USGS, SCEC, and many other organizations to improve public readiness for a catastrophic earthquake along the southern SAF. The success of this exercise, which involved over 5 million people (the largest ever held in the United States), was attributable in part to the physical realism of the simu-

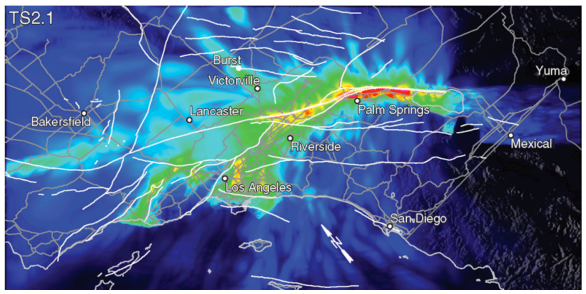

Fig. 17. PGVs for the TeraShake-D (2.1) simulation. White lines depict fault traces and county lines. The dotted line depicts the part of the San Andreas fault that ruptured in the TeraShake-D simulations. Modified after Olsen et al. [39].

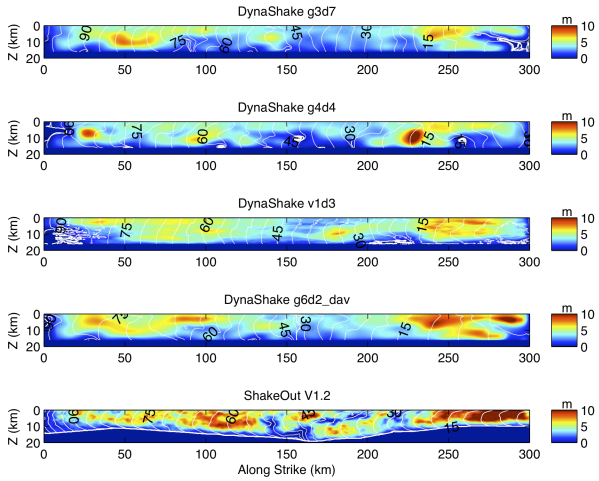

Fig. 18. Slip distributions for 4 of the 7 ShakeOut-D sources and ShakeOut-K (V1.2). The white contours and contour labels depict the rupture times. Modified after Olsen et al. [41].

lation used to drive the exercise. The M7.8 ShakeOut scenario was based on geological data and cumulative strain estimates from GPS records; the 300-km-long scenario rupture initiated at the southern terminus of the SAF near the Salton Sea and propagated unilaterally toward the northwest into the western Mojave desert. Seven dynamic source descriptions were used (Fig. 18) to assess the uncertainty in the site-specific peak motions. The ShakeOut project increased the upper frequency limit of the SCEC CME simulations to 1 Hz.

In addition to these large simulations on the southern SAF, AWP-ODC has been used for several other important modeling projects conducted by the CME collaboration. One of these projects produced 0-0.5 Hz simulations of large, M8.5-9.0 megathrust earthquake scenarios in the Pacific Northwest. This study demonstrated strong basin amplification and ground motion durations up to 5 minutes in metropolitan areas such as Seattle [40]. Another prominent project is F3DT, an I/O intensive 3D waveform tomography to iteratively improve the CVM4 in southern California. Here, AWP-ODC is used to calculate sensitivity kernels accounting for the full physics of 3D wave propagation, generating updated velocity models with substantial better fit to data as compared to the starting models [10]. Finally, preliminary wall-to-wall scenarios were carried out with a grid spacing of 100 m, leading us to M8.

VII. M8 Simulation

M8 was motivated by the need to understand the seismic hazard of the southern SAF, which has accumulated a significant slip deficit since the most recent large ruptures of this dangerous fault. The southernmost segments of the southern SAF, between Cajon Creek and Bombay Beach, have not participated a major earthquake since circa 1680, implying a slip deficit of 5–6 meters [44]. Further north, from the southern Mojave to Cholame, the most recent large event was in 1857. Estimated recurrence intervals between major events on the southern SAF from paleoseismic studies have recently been reduced to less than the time elaspsed since this earthquake, indicating that the entire southern San Andreas is “locked and loaded” [16].

There is no consensus from paleoseismic studies that a magnitude-8 rupture extending from the Salton Sea to Cholame (the ‘wall-to-wall’ event) has ever happened. In the current Uniform California Earthquake Rupture Forecast, in the 30-year probability of such a rupture is a modest 3% [21]. However, the paleoseismolgical record is consistent with such an event in the late 1400s, so ground motions from this type of earthquake need to be investigated.

We used the SGSN mode of AWP-ODC (Fig. 6) to generate a dynamic rupture model for M8. Since the SGSN mode can only be applied for a planar fault, a two-step method was employed. In a first step, we simulated a spontaneous rupture on a planar, vertical fault that is 545 km long (from Cholame to Bombay Beach) and 16 km deep. The source time histories obtained from the dynamic simulation were then transferred onto a segmented approximation of the southern SAF, and the wave-propagation for this source was solved with AWP-ODC. Surface topography was not included in the rupture or wave propagation models.

A. Source Description

We simulated the spontaneous rupture on a fault that is embedded in a seismic geologic model representing the average compressional-velocity, shear-velocity and density along the SAF. Friction in our model followed a slip-weakening law, with static (μ_s) and dynamic (μ_d) friction coefficients of 0.75 and 0.5, respectively, and a slip-weakening distance d_c of 0.3 m. In the top 2 km of the fault, we emulated velocity strengthening by forcing $\mu_d > \mu_s$, with a linear transition between 2 km and 3 km, causing the stress drop in this region to be negative. Additionally d_c was increased to 1 m at the free surface using a cosine taper in the top 3 km. The initial shear stress on the fault was derived from the assumption of depth-dependent normal stress [15]. Because the initial compressive normal stress $-\tau_n$ increased with depth as a consequence of increasing overburden, the frictional strength τ_f (and generally the

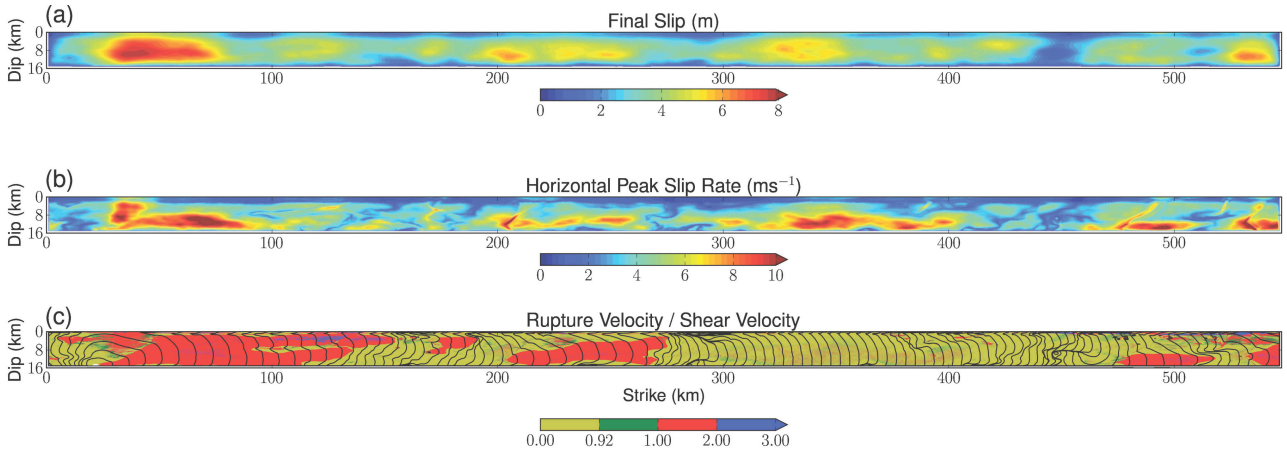

Fig. 19. M_w8's source model obtained from spontaneous rupture simulation (left at NW, right at SE). (a) Final slip (b) horizontal peak slip rate, and (c) rupture velocity normalized by the local shear-wave velocity. In (c), the yellow areas are dominated by sub-Rayleigh rupture velocities, while red and blue patches indicate areas where the rupture propagates at super-shear speed. The black contours show the rupture time in intervals of 1 second.

stress drop) increased as well. We also included cohesion of 1 MPa on the fault.

To define the initial shear stress τ_0 on the fault, we first generated a random stress field using a Van Karman autocorrelation function with lateral and vertical correlation lengths of 50 km and 10 km, respectively. The random stress field was then accommodated into the depth-dependent frictional strength profile in such a way that the minimum shear stress represented reloading from the residual shear stress after the last earthquake, and such that the maximum shear stress reached the failure stress [15]. The initial shear stress τ_0 generally increases with depth, despite the random component. The shear stress was tapered linearly to zero at the surface from a depth of 2 km. Rupture was initiated by adding a small stress increment to a circular area near the nucleation patch, located ~20 km from the northern end of the fault. We used a spatial discretization of 100 m and a temporal discretization of 6.25 ms. The extent of the rupture model included 40-km-wide zones between the fault and the PML absorbing boundaries on the sides and 24 km on the bottom. This discretization of the rupture model is adequate for good numerical resolution, as demonstrated by previous work [14][39][41].

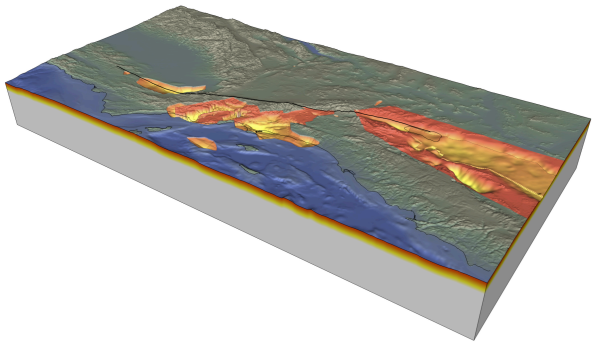

Fig. 20. Perspective view of the M8's 810 x 405 x 85 km model domain for central and southern California, and northern Baja California. Sedimentary basins are revealed by cutaway of material with S-wave velocity less than 2.5 km/s (as defined by the SCEC CVM 4). Depth below the surface is indicated by the red/yellow color scale.

The size of the computational domain was 629 × 80 × 40 km^3 (~2 billion nodes). The dynamic rupture was generated by *dSrcG* and *PetaSrcP* on NICS Kraken using 2160 cores during 7.5 hours, simulating 250 seconds of rupture. The moment rate time histories were defined on 881,475 subfaults with 108,000 time steps (2.1 TB). The source was further partitioned into 526 spatially separate grids. In addition to the spatial locality, we enabled temporal locality by splitting the source into 36 loops (each responsible for 3000 time steps) to reduce memory requirements.

The final slip (Fig. 19a) reached 7.8 m on the fault and 5.7 m on the surface, with an average slip of 4.5 m and a total seismic moment of 1.0×10^{21} Nm ($M_w = 8.0$). These values are in general agreement with worldwide observations from magnitude ~8 events (e.g., [43]). Peak slip rates were generally larger at depth, where they exceed 10 m/s (0-2 Hz) in a few patches (Fig. 19b). The rupture propagated both at sub-Rayleigh and super-shear speed until it reached the opposite end of the fault after 135 seconds (Fig. 19c). A large ~100 km patch of super-shear rupture velocity was located between 30 and 130 km along-strike, and smaller patches near 250 km, 500 km, and 540 km.

B. Wave Propagation

The spontaneous-rupture source was then inserted onto a 47-segment approximation of the southern SAF after applying temporal interpolation and a 4th-order low-pass filter with a cut-off frequency of 2 Hz. The source was imbedded in a 810 km × 405 km × 85 km volume extracted from the SCEC CVM 4 (see Fig. 20) using the Universal Transversal Mercator (UTM) projection. The volume was discretized into 436 billion 40-m^3 cubes using a minimum S-wave velocity (V_s) of 400 m/s.

P-wave and S-wave velocities and density values were stored on this mesh, while quality factors (specifying anelastic attenuation) were calculated on-the-fly (from an approximate empirical relationship - $Q_s = 50\ V_s$ where V_s is

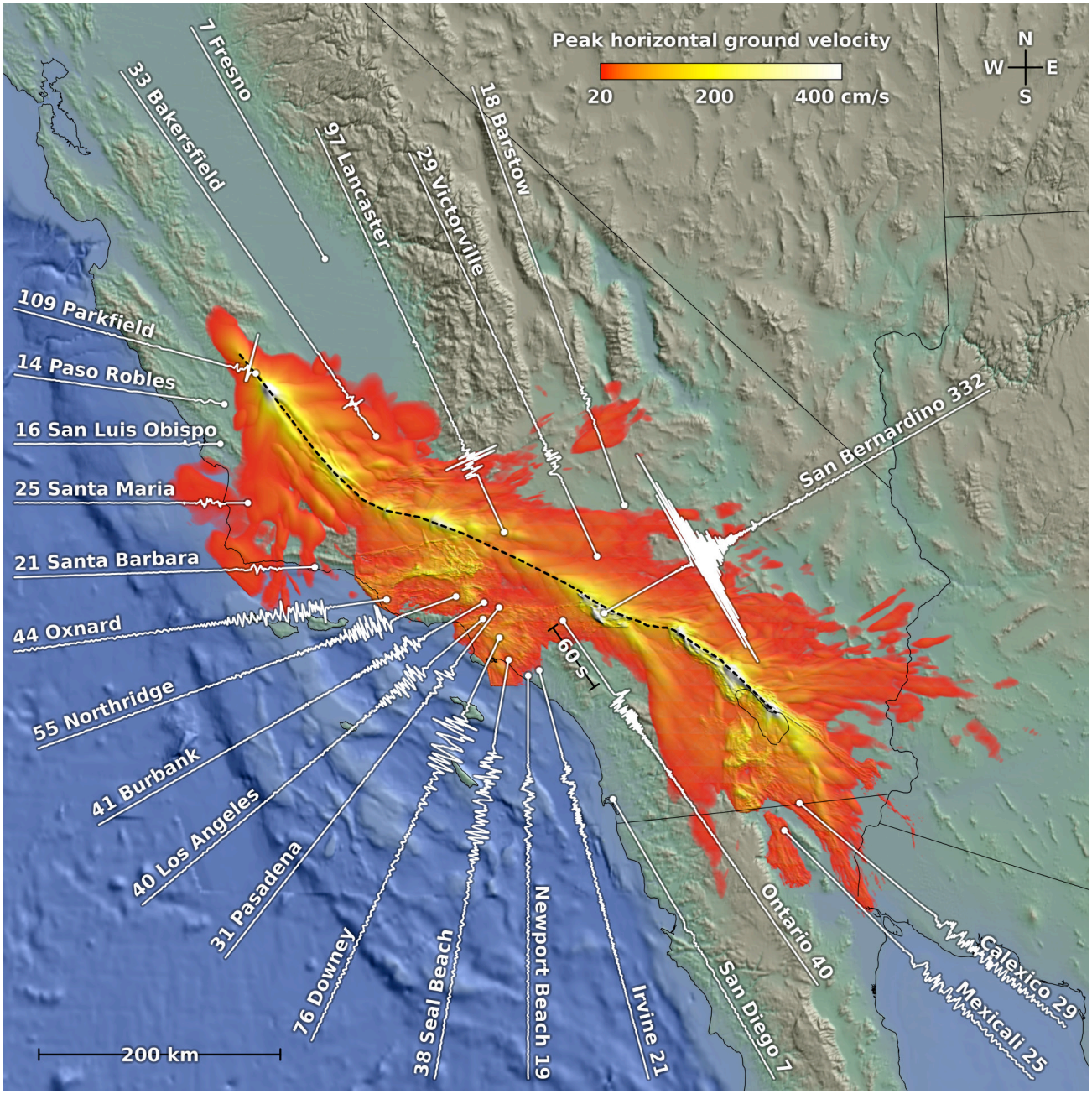

Fig. 21. PGVHs derived from M8 superimposed on the regional topography. N46E component seismograms are added at selected locations, with their peak velocities (cm/s) listed along the traces.

in units of km/s, and $Q_p = 2Q_s$). M8 mesh was pre-partitioned into 223,074 files on Jaguar using *PetaMeshP*. An alternative procedure (in case of hardware file system failure) used direct contiguous MPI-IO imbedded into the solver to directly read the single 4.8 TB mesh file by 21,600 readers, redistributing the partitioning to each process at solver time. For the final production run, we used the first approach to read in the pre-partitioned mesh files in 4 minutes. To avoid file contention, we limited the number of synchronous file open requests to 650 (maximum 670 OSTs on Jaguar) and as a result, achieved an aggregate read performance of 20 GB/s. The simulation of 360 seconds of wave propagation took 24 hours on Jaguar using 223,074 cores, sustained 220 Tflops, and produced 4.5 TB of surface synthetic seismograms. M8 consumed thirty times the computational resources that were required by each of the ShakeOut-D simulations (see section 6).

Checkpointing was not activated during the M8 production simulation to avoid additional potential stress to the file system writing the 49 TB checkpoint files at each time step. M8 saved the ground velocity vector at every 20th time step on an 80 m by 80 m grid (4.5 TB). Outputs were aggregated at run-time and written every 20K time steps to minimize I/O overhead. In total, M8 consumed 581 MB of memory per core, with 285 MB by the solver, 46 MB by buffer aggregation of outputs, 22 MB by the Earth model,

and 228 MB by the source after lowering the memory high water mark into 36 segments through temporal partitioning.

C. Scientific Advances From M8

Several important scientific advances have been gained from M8, primarily related to the increase of the largest frequencies included (2 Hz), the vast computational domain (810 km by 405 km), and complex spontaneous-rupture source description. Animations of the simulation can be downloaded from [28]. Fig. 21 shows horizontal peak ground velocities (PGVHs) calculated from M8 (as the root sum of squares of the horizontal components), along with synthetic seismograms at selected sites. The PGVHs show patterns in agreement with results from the previous TeraShake and ShakeOut-D simulations, including large near-fault values and strong directivity effects, and 'sun-bursts' radiating from the fault due to the complexity of the spontaneous rupture propagation. Although experiencing significant PGVHs on the order of 0.4 m/s, downtown Los Angeles is not excited by the waveguide amplification to the same extent as earthquake simulations on the southern SAF with SE-NW rupture directions. The NW-SE rupture direction for M8 is largely transverse to the waveguides, avoiding the intense focusing effect observed for NW-propagating TeraShake/ Shakeout ruptures.

Large near-fault ground motions were expected by M8 due to the strong directivity effects generally obtained from long strike-slip earthquakes. The largest near-fault peak velocities from M8 occurred immediately on top of the fault trace, in isolated locations exceeding 10 m/s (these most extreme velocities will be reduced when corrected for nonlinear soil response, not yet incorporated into the results). Some of largest near-fault PGVHs occur in connection with patches of super-shear rupture for M8 (e.g., at distances of about 30 km, 480 km, and 530 km from the northern end of the fault, Fig. 19). Previous analyses of super-shear rupture propagation and ground motions have focused on constant rupture velocities in simple homogeneous or layered media (e.g., [1][4][19]). These studies have shown that the Mach waves generated by supershear rupture (as obtained for M8 in Fig. 22) carry intense near-fault ground motions to much larger distances from the fault than is the case for sub-shear ruptures. Furthermore, the fault-parallel component of ground motion tends to display similar or larger amplitude, as compared to the fault-perpendicular component, which usually contains the largest peak velocities for subshear rupture propagation due to directivity. M8 shows similar wavefield characteristics and extends the analysis to complex, heterogeneous rupture models in 3D media. In particular, some of the largest M8 PGVs along the fault tend to occur where the rupture speed increases rapidly from sub-Rayleigh to super-shear rupture speeds. The rapid increase in rupture speed in these areas likely contribute to the exceptionally large ground motions.

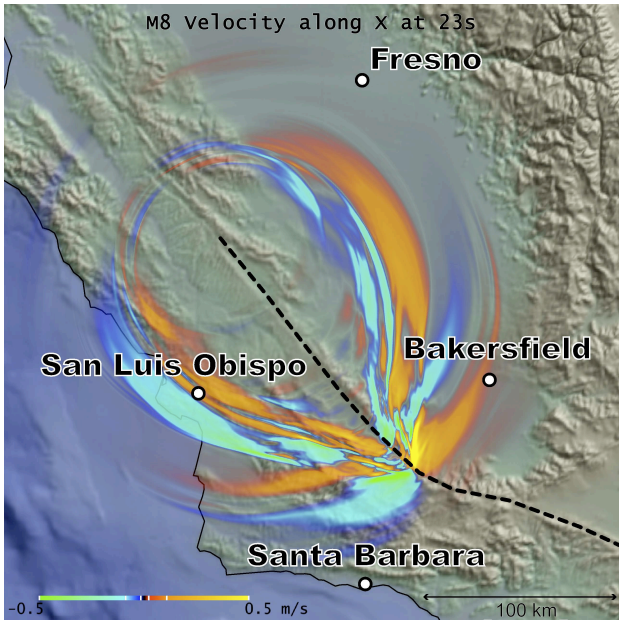

Fig. 22. N46E component velocity snapshot at 23 s from M8 illustrating super-shear wave propagation, with the Mach cone entering the 'Big Bend' section of the SAF.

The velocity time histories associated with the large PGVHs on the fault are generally characterized by a single, simple pulse, with a significant amount of energy between 1 and 2 Hz. In other cases, large near-fault shaking may occur associated with longer periods. An example of this is San Bernardino (see Fig. 21) where PGVHs reach 6 m/s. A spectral analysis shows that these peaks correspond to periods of 2-4 s. San Bernardino, like Los Angeles and Ventura, is built on top of a relatively deep sedimentary basin (the San Bernardino Basin, SBB, is up to 2 km deep). The combination of a location within kilometers of the SAF, the SBB, and the strong directivity from the NW-SE M8 rupture appears to be causing the large ground motions in San Bernardino. The Coachella Valley is another sedimentary basin located along the fault experiencing intense shaking from M8. Whether or not these large peak motions can prevail during real earthquakes requires studies on nonlinear soil response and alternative friction models for the rupture. Such studies are currently underway in SCEC.

In order to rank the ground motion levels from M8 relative to their expected frequency of occurrence for a generic site and event (of the same magnitude), we have made comparisons of the simulated PGVHs to those predicted by recent Next Generation Attenuation (NGA) relations. Such attenuation relations (ARs) are empirical regression estimates attempting to quantify the statistical distribution of ground motion amplitudes over all scenarios. We include the ARs proposed by Campbell and Bozorgnia [8] and Boore and Atkinson [7]. Note that for these comparisons we use the geometric mean of the PGVHs, since this measure is used by [8] and [7]. The geometric mean generates PGVHs typically 1.5-2 times smaller than those

values calculated from the root sum of squares, the measure used in Fig. 21. Since M8 calculates ground motions up to 2 Hz, the PGVHs are not expected to be significantly biased by the simulation bandwidth.

Fig. 23 shows a comparison of horizontal geometric-mean PGVHs for all rock sites in M8 at distances up to 200 km from the fault to the values predicted by the [8] and [7]. The rock sites were defined by a surface V_s > 1000 m/s for M8 and a depth of 400 m to the V_s = 2500 m/s isosurface for [8] (and V_{s30} = 760 m/sec). For most distances from the fault, the median M8 and AR PGVs agree very well, and the M8 median ± 1 standard deviation are very close to the AR 16% and 84% probability of exceedance (POE) levels, respectively. The good agreement provides independent evidence that the fault area (and therefore the average stress drop) used for M8 are consistent with a moment magnitude 8 event.

The comparison in Figure 23 shows that, at sites where region-specific propagation effects are relatively unimportant, M8 matches remarkably closely the ground motion statistics encoded in the NGA empirical relationships. Of course the empirical relationships, by their very nature (i.e., since they are based on empirical fits to worldwide data), cannot predict path-specific effects associated with the southern California geological model that are captured by M8, mainly in the sedimentary basins. Examples of ground motions predicted by M8 that are high amplitude (i.e., low POE relative to the corresponding generic NGA predictions) can be seen in Fig. 22: Oxnard in the Ventura basin with a PGVH of 33 cm/s (72 km from the fault, at about 2% POE for [7]) and Downey in the Los Angeles basin with a PGVH of 65 cm/s (65 km from the fault, at about 0.13% POE for [7]). In particular, the large peak ground motions simulated in the SBB fall well below 0.1% POE levels for [7], illustrated by a PGVH of 430 cm/s at a distance of 10 km from the fault. However, these extreme ground motions, caused by complex source propagation and 3D basin amplification effects, are not expected to be captured by the ARs, which cannot replicate such geographically specific effects.

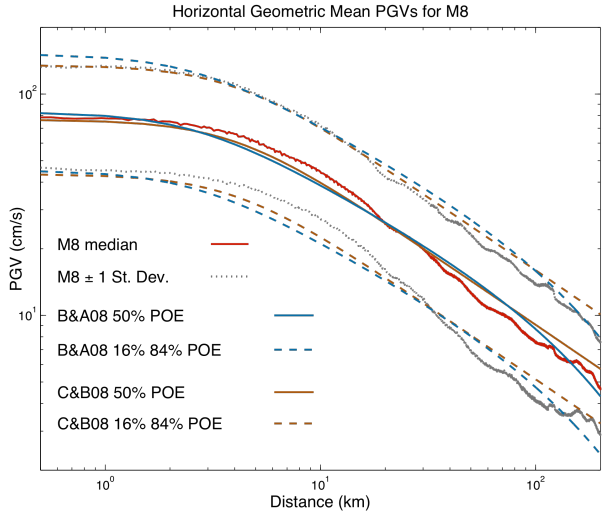

Fig. 23. Comparison of PGVs from M8 to those calculated for M_w 8 events from C&B08 [8] and B&A08 [7] at rock sites. The rock sites were defined by a surface Vs > 1000 m/sec for M8 and a depth of 400 m to the V_s =2500 m/sec isosurface for [8] (and Vs_{30} = 760 m/sec). POE stands for probability of exceedance.

VIII. Summary and Outlook

We have developed a highly scalable, parallel finite-difference application (AWP-ODC) targeting petascale earthquake hazard calculations. It combines the state-of-the-art SGSN-based dynamic rupture simulations with seismic wave propagation simulations in three dimensions. AWP-ODC is an integral part of the Community Modeling Environment at SCEC, and has been widely used in the community for applied scientific research. Recent AWP-ODC performance gains made it possible to simulate M8, a great earthquake scenario on the SAF. M8 produced 368 s of 0-2 Hz ground motion in a 810 km × 405 km × 85 km region within southern California using 3D structure properties from the SCEC Community Velocity Model V4.0 (CVM-4). M8 sustained 220 Tflop/s, at approximately 10% of peak performance, with nearly ideal scaling to the entire Jaguar system of 223K cores. These values demonstrate an unprecedented performance for an explicit, stencil-based solver, and M8 is a breakthrough in seismology both in terms of computational size and scalability. M8 is the largest and most detailed physics-based ground motion simulation of a large scenario earthquake thus far performed.

Important new scientific insight was gained from M8 about the ground motion levels to be expected for a Great earthquake on the SAF, in particular at the large population centers within the model area. For example, M8 generates large amplification with peak ground velocities exceeding 300 cm/s at some locations in the Ventura basin. Such large amplification was expected for SAF scenarios with SE-NW rupture directions due to wave-guide channeling of the seismic energy from the fault [41]. However, M8 shows that directivity effects for a great SAF earthquake initiating near Cholame and propagating SE may generate similar levels of amplification in the Ventura basin, despite the wave field enters almost perpendicularly to the wave guide. On the other hand, the largest peak motions in the deeper LA basin reach about 120 cm/s, with 'only' 40 cm/s in downtown Los Angeles. San Bernardino appears to be the area hardest hit by M8, due to directivity effects coupled with basin amplification and proximity to the fault.

M8 also provides new insight about the effects of super-shear rupture propagation. In particular, M8 suggests that exceptionally large ground motions can be generated along the fault trace above locations where the earthquake rupture transitions from sub-Rayleigh to super-shear speeds. In addition, the reduced attenuation of the Mach waves generated during the super-shear rupture NW of the 'Big

Bend' of the SAF may be a contributing factor to the large peak motions obtained in the Ventura area. Additional near-future M8 rupture scenarios planned within SCEC will include source descriptions with generally sub-shear rupture speeds, to test whether the ground motions will be characterized by less extreme directivity effects (and associated coherent wave fields and extreme ground motions). In addition, scenarios with different hypocentral locations (i.e., SE-NW rupture propagation, bi-lateral rupture) in several different CVMs will be carried out, to obtain an estimate of the ground motion uncertainty related to a Great SAF event.

Increased complexity with multicore NUMA architectures has pushed the burden of obtaining good performance to the application level. In this paper we have addressed some critical technical issues regarding large-scale earthquake simulations: careful design of efficient algorithms and adaptive software packages, optimization of message passing, effective communications, and in particular understanding the underlying characteristics of the parallel file system and building highly customizable parallel I/O libraries at extreme scale. Further enhancements of AWP-ODC are planned for the near future, including development of a highly scalable algorithm-dependent fault tolerance technique based on work by Dongarra et al. [11], adding HDF5 and ADIOS features to allow changes in the I/O external configuration files, and facilitating an extremely large volume data analysis using our state-of-the-art 4D vector visualization technique. Finally, we will continue to port and re-engineer AWP-ODC to increasingly larger computing platforms, such as the upcoming NCSA Blue Waters through a Petascale Computing Resource Allocation (PRAC) award, sponsored by the NSF PetaApps program.

The path to successful completion of SCEC milestone simulations in recent years (with M8 as the most prominent example) has demonstrated that optimization and enhancement of major application codes are essential for using large resources (i.e., number of processors, number of CPU-hours, terabytes of data produced). M8 also showed that multiple types of resources are needed for large problems, namely initialization, run-time execution, analysis resources, and long-term data collection management. The development and improvements to AWP-ODC leading to M8 have created a community code that has been used by the wider SCEC community to perform petascale earthquake simulations. In the future, well-verified and validated simulations of ground motions using realistic 3D structural models will provide better estimates of strong ground motions at frequencies of interest to engineering, emergency management, and seismological communities.

ACKNOWLEDGMENT

The authors acknowledge the Office of Science of the U.S. Department of Energy (DOE) for providing HPC resources that have contributed to the research results reported within this paper through an Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program allocation award. Computations were performed on Jaguar, which is part of the Oak Ridge Leadership Facility at the Oak Ridge National Laboratory which is supported by under DOE Contract No. DE-AC05-00OR22725. This research used resources of the Argonne Leadership Computing Facility at Argonne National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under contract DE-AC02-06CH11357. This research was supported by an allocation of advanced computing resources provided by the National Science Foundation. Computations were performed on Kraken (a Cray XT5) at the National Institute for Computational Sciences (www.nics.tennessee.edu). Computations and data management were performed at San Diego Supercomputer Center (www.sdsc.edu), where the iRODS Data System was used (www.diceresearch.org). The Texas Advanced Computing Center (TACC) at The University of Texas at Austin (www.tacc.utexas.edu) provided HPC resources that have contributed to the research results reported within this paper. Computations for the work described in this paper were supported by the University of Southern California Center for High-Performance Computing and Communications (www.usc.edu/hpcc). The Ohio State University One-sided MPI Communication research was supported through NSF HECURA-1 (CCF-0833169/139/155). This research received technical and user support through the Advanced Support for TeraGrid Applications (ASTA) program (www.teragrid.org). This research was supported by the Southern California Earthquake Center (www.scec.org). SCEC is funded by NSF Cooperative Agreements EAR-0106924 and USGS Cooperative Agreement 02HQAG0008, and NSF awards EAR-074493, EAR-0949443, OCI-0832698, and OCI-0832698. The SCEC contribution number for this paper is 1443.

REFERENCES

[1] B.T. Aagaard and T.H. Heaton, "Near-Source Ground Motions from Simulations of Sustained Intersonic and Supersonic Fault Ruptures," *Bull. Seis. Soc. Am.,* vol. 94, no.6, 2004, pp. 2064-2078.

[2] J. Berenger, "A Perfectly Match Layer for the Absorption of Electromagnetic Waves," *J. Comput. Phys.,* vol. 114, no. 2, 1994, pp. 185-200.

[3] J. Berenger, "Three-Dimensional Perfectly Matched Layer for the Absorption of Electromagnetic Waves," *J. Comput. Phys.,* vol. 127, no. 2, 1996, pp. 363-379.

[4] P. Bernard and D. Baumont, "Shear Mach Wave Characterization for Kinematic Fault Rupture Models with Constant Supershear Rupture Velocity," *Geophys. J. Int'l,* vol. 162, no. 2, 2005, pp. 431-447, doi:10.1111/j.1365-246X.2005.02611.x.

[5] J. Bielak, R. Graves, K.B. Olsen, R. Taborda, L. Ramirez-Guzman, S.M. Day, G. Ely, D. Roten, T.H. Jordan, P. Maechling, J. Urbanic, Y. Cui, and G. Juve, "The ShakeOut Earthquake Scenario: Verification of Three Simulation Sets," *Geophys. J. Int'l*, vol. 180, no. 1, 2010, pp. 375-404.

[6] J.O. Blanch, J.O.A. Robertsson, and W.W. Symes, "Modeling of a Constant Q: Methodology and Algorithm for an Efficient and Op-

timally Inexpensive Viscoelastic Technique," *Geophysics*, vol. 60, no. 1, 1995, pp. 176-184, doi:10.1190/1.1443744.

[7] D.M. Boore and G.M. Atkinson, "Ground-motion prediction equations for the average horizontal component of PGA, PGV, and 5%-damped PSA at spectral periods between 0.01 s and 10.0 s", *Earthquake Spectra,* vol. 24, 2008, pp. 99-138.

[8] K.W. Campbell and Y. Bozorgnia, "NGA ground motion model for the geometric mean horizontal component of PGA, PGV, and 5%-damped PSA at spectral periods between 0.01 s and 10.0 s", *Earthquake Spectra,* vol. 24, 2008, pp. 139-171.

[9] C. Cerjan, D. Kosloff, R. Kosloff, and M. Reshef, "A Nonreflecting Boundary Condition for Direct Acoustic and Elastic Wave Equations," *Geophysics,* vol. 50, no. 4, 1985, pp. 705-708, doi:10.1190/1.1441945.

[10] P. Chen, L. Zhao, and T.H. Jordan, "Full 3D Tomography for the Crustal Structure of the Los Angeles Region," *Bull. Seis. Soc. Am.*, vol. 97, no. 4, 2007, pp. 1094-1120, doi: 10.1785/0120060222.

[11] Z. Chen, J.J. Dongarra, "Algorithm-Based Fault Tolerance for Fail-Stop Failures", *IEEE Trans. Parallel and Distributed Systems,* vol. 19, no. 12, 2008, pp. 1628-1641, doi:10.1109/TPDS.2008.58.

[12] Y. Cui, R. Moore, K.B. Olsen, A. Chourasia, P. Maechling, B. Minster, S. Day, Y. Hu, J. Zhu, A. Majumdar, and T.H. Jordan, "Enabling Very-Large Scale Earthquake Simulations on Parallel Machines," *Proc. Int'l Conf. Comput. Science*, *Lecture Notes in Computer Science series,* vol. 4487, Springer-Verlag, 2007, pp. 46-53.

[13] Y. Cui, K.B. Olsen, A. Chourasia, R. Moore, P. Maechling, and T. H. Jordan, "The TeraShake Computational Platform for Large-Scale Earthquake Simulations," *Advances in Geocomputing*: *Lecture Notes in Earth Sciences,* vol. 119, Springer-Verlag, 2009, pp. 229-278.

[14] L.A. Dalguer and S.M. Day, "Staggered-Grid Split-Node Method for Spontaneous Rupture Simulation," *J. Geophys. Res.,* vol. 112, B02302, 2007, doi:10.1029/2006JB004467.

[15] L.A. Dalguer and P.M. Mai, "Implications of Style-of-Faulting and Loading Characteristics on the Dynamic Rupture Process," *EOS Trans., Am. Geophys. Union 89(53), Fall Meet. Suppl.*, 2008.

[16] T.E. Dawson, T.K. Rockwell, R.J. Weldon II, and C.J. Wills, "Summary of Geologic Data and Developments of A Priori Rupture Models for the Elsinore, San Jacinto, and Garlock Faults," *Appendix F to USGS Open File Report 2007-1437F*, 2008.

[17] S.M. Day, "Efficient Simulation of Constant Q Using Coarse-Grained Memory Variables," *Bull. Seis. Soc. Am.* vol. 88, no. 4, 1998, pp. 1051–1062.

[18] S.M. Day and C.R. Bradley, "Memory-Efficient Simulation of Anelastic Wave Propagation," *Bull. Seis. Soc. Am,* vol. 91, no. 3, 2001, pp. 520–531, doi:10.1785/0120000103.

[19] E.M. Dunham and H.S. Bhat, "Attenuation of Radiated Ground Motion and Stresses from Three-Dimensional Supershear Ruptures," *J. Geophys. Res.,* vol. 113, 2008.

[20] M. Faerman, R. Moore, Y. Cui, Y. Hu, J. Zhu, B. Minister, and P. Maechling, "Managing Large Scale Data for Earthquake Simulations," *J. Grid Comp.,* vol. 5, no. 3, 2007, pp. 295-302, doi: 10.1007/s10723-007-9072-x.

[21] E.H. Field, T.E. Dawson, K.R. Felzer, A.D. Frankel, V. Gupta, T.H. Jordan, T. Parsons, M.D. Petersen, R.S. Stein, R.J. Weldon II, and C.J. Wills, "Uniform California Earthquake Rupture Forecast, Version 2 (UCERF 2)," *Bull. Seis. Soc. Am.* vol. 99, no. 4, 2009, pp. 2053-2107, doi:10.1785/0120080049.

[22] E. Gottschammer and K.B. Olsen, "Accuracy of the Explicit Planar Free-Surface Boundary Condition Implemented in a Fourth-Order Staggered-Grid Velocity-Stress Finite-Difference Scheme," *Bull. Seis. Soc. Am.,* vol. 91, no. 3, 2001, pp. 617-623, doi: 10.1785/0120000244.

[23] R.W. Graves, "Simulating Seismic Wave Propagation in 3D Elastic Media Using Staggered-Grid Finite Differences," *Bull. Seis. Soc. Am.,* vol. 86, no. 4, 1996, pp. 1091-1106.

[24] iRODS: Data Grid, Digital Libraries, Persistent Archives, and Real Time Data Systems, http://www.irods.org/.

[25] T.H. Jordan and P. Maechling, "The SCEC community Modeling Environment: An Information Infrastructure for System-Level Earthquake Science," *Seis. Res. Lett*er, vol. 74, no. 1, 2003, pp. 324-32.

[26] K. Lee, Y. Cui, P. Maechling, K.B. Olsen, and T.H. Jordan "Commnication Optimizations of SCEC CME AWP-Olsen Application for Petascale Computing," *Supercomputing conf. (SC'09),* 2009, Poster.

[27] K. Lee, Y. Cui, T. Kaiser, P. Maechling, K.B. Olsen, and T.H. Jordan, "I/O Optimizations of SCEC AWP-Olsen Application for Petascale Earthquake Computing," *Supercomputing conf. (SC'09),* 2009, Poster.

[28] M8 Simulation Visualization, http://visservices.sdsc.edu/projects/scec/m8/1.0.

[29] P. Maechling, E. Deelman, and Y. Cui, "Implementing Software Acceptance Tests as Scientific Workflows," *PDPTA CSREA Press*, 2009, pp. 317-323.

[30] C. Marcinkovich and K.B. Olsen, "On the Implementation of Perfectly Matched Layers in a 3D Fourth-Order Velocity-Stress Finite-Difference Scheme," *J. Geophys. Res.*, vol. 108 (B5), 2276, 2003, doi: 10.1029/2002JB002235.

[31] E. Mcquinn, A. Chourasia, B. Minster, and J. Schulze, "Visualizing Time-Dependent Seismic Vector Fields With Glyphs," Feb 2010, http://visservices.sdsc.edu/projects/scec/vectorviz/.

[32] K.C. Meza-Fajardo and A.S. Papageorgiou, "A Nonconventional, Split-field, Perfectly Matched Layer for Wave Propagation in Isotropic and Anisotropic Elastic Media: Stability Analysis," *Bull. Seis. Soc. Am.,* vol. 98, no. 4, 2008, pp. 1811-1836, doi: 10.1785/0120070223.

[33] S.E. Minkoff, "Spatial Parallelism of a 3D Finite Difference Velocity-Stress Elastic Wave Propagation Code," *SIAM J. Sci. Comput.*, vol. 24, no. 1, 2002, pp. 1-19.

[34] NCCS, "Jaguar", 2010, http://www.nccs.gov/jaguar.

[35] H. Nguyen, Y. Cui, K.B. Olsen, K. Lee, "Single CPU optimization of SCEC AWP-Olsen," *SCEC Annual Meeting*, 2009, Poster.

[36] K.B. Olsen, "Simulation of Three-Dimensional Wave Propagation in the Salt Lake Basin," doctoral dissertation, Univ. of Utah, 1994, p. 157.

[37] K.B. Olsen, S.M. Day, C.R. Bradley, "Estimation of Q for Long-Period (>2 s) Waves in the Los Angeles Basin," *Bull. Seis. Soc. Am.* vol. 93, no. 2, 2003, pp. 627–638, doi: 10.1785/0120020135.

[38] K.B. Olsen, S.M. Day, J.B. Minster, Y. Cui, A. Chourasia, M. Faerman, R. Moore, P. Maechling, T.H. Jordan, "Strong Shaking in Los Angeles Expected from Southern San Andreas Earthquake," *Geophys. Res. Letter,* vol. *33*, 2006, L07305, doi:10.1029/2005GL025472.

[39] K.B. Olsen, S.M. Day, J.B. Minster, Y. Cui, A. Chourasia, D. Okaya, P. Maechling, and T.H. Jordan, "TeraShake2: Spontaneous Rupture Simulations of Mw 7.7 Earthquakes on the Southern San Andreas Fault," *Bull. Seism. Soc. Am.,* vol. 98, no. 3, 2008, pp. 1162-1185, doi: 10.1785/0120070148.

[40] K.B. Olsen, W. J. Stephenson, and A. Geisselmeyer, "3D crustal structure and long-period ground motions from a M9.0 Megathrust Earthquake in the Pacific Northwest region," *J. Seism.*, vol. 12, no. 2, April 2008, pp. 145-159.

[41] K.B. Olsen, L.A. Dalguer, S.M. Day, Y. Cui, J. Zhu, V.M. Cruz, D. Roten, J. Mayhew, P. Maechling, T.H. Jordan, A. Chourasia, and D. Okaya, "ShakeOut-D: Ground Motion Estimates Using an Ensemble of Large Earthquakes on the Southern San Andreas Fault with Spontaneous Rupture Propagation," *Geophys. Res. Letter*, vol. 36, Feb. 2009, L04303, doi: 10.1029/2008GL036832.

[42] S. Potluri, P. Lai, K. Tomko, S. Sur, Y. Cui, M. Tatineni, K. Schulz, W. Barth, A. Majumdar, and D.K. Panda, "Quantifying Performance Benefits of Overlap using MPI-2 in a Seismic Modeling Application", *Proceedings of the 24th ACM Int'l Conference on Supercomputing*, 2010, pp. 17-25, doi: 10.1145/1810085.1810092.

[43] D.P. Schwartz, and J. Coppersmith (1984), "Fault behaviour and characteristic earthquakes: examples from Wasatch and San Andreas faults", *J. Geophys. Res*., vol. 89**,** 1984, pp. 5681–5698.

[44] R. Weldon, K. Scharer, T. Fumal, and G. Biasi, "Wrightwood and the Earthquake Cycle: What a Long Recurrence Record Tells Us about How Faults Work," *Geol. Soc. Am.,* vol. 14, no. 9*,* 2004, pp. 4-10.

[45] J. Zhou, Y. Cui, S. Davis, C.C. Guest, "Workflow-Based High Performance Data Transfer and Ingestion to Petascale Simulations on TeraGrid", *IEEE Comput. Sciences and Optimization (CSO'10),* vol. 1, 2010, pp. 343-347.