Physics-based Seismic Hazard Analysis on Petascale Heterogeneous Supercomputers

Y. Cui[1], E. Poyraz[1], K.B. Olsen[2], J. Zhou[1], K. Withers[2], S. Callaghan[3], J. Larkin[4], C. Guest[1], D. Choi[1], A. Chourasia[1], Z. Shi[2], S. M. Day[2], J. P. Maechling[3], T. H. Jordan[3]

[1]University of California, San Diego, [2]San Diego State University, [3]University of Southern California, [4]NVIDIA Inc.

[1](yfcui,epoyraz,j4zhou,cguest,djchoi,amit@ucsd.edu),
[2](kbolsen,withers,zshi,sday@mail.sdsu.edu),
[3](scottcal,maechlin,tjordan@usc.edu) , [4](jlarkin@nvidia.com)

ABSTRACT

We have developed a highly scalable and efficient GPU-based finite-difference code (AWP) for earthquake simulation through high throughput, memory locality, communication reduction and communication / computation overlap, achieving perfect linear speedup on Cray XK7 Titan at ORNL and NCSA's Blue Waters system. AWP's excellent performance is demonstrated by simulating realistic 0-10 Hz earthquake ground motions, as required by building engineering design, through small-scale complexity in the fault surface and surrounding crustal structure. Moreover, we show that AWP provides a speedup in key strain tensor calculations critical to probabilistic seismic hazard analysis by a factor of 110. This achievement, coupled with improved co-scheduling capabilities of our workflow-managed systems, makes a statewide hazard model a goal reachable with existing supercomputers. The performance of the GPU-based AWP is expected to take physics-based seismic hazard analysis to a new level using Petascale heterogeneous computing resources, saving millions of core-hours over the next few years.

Keywords

SCEC, seismic hazard analysis, earthquake ground motions, parallel scalability, GPU, CyberShake, hybrid heterogeneous.

1. INTRODUCTION

Economic exposure to earthquake devastation is skyrocketing, primarily because urban environments are growing rapidly in seismically active regions. Probabilistic seismic hazard analysis (PSHA) in its standard form—i.e. derived empirically assuming the hazard is time-independent (Poissonian)—has been effective in helping decision-makers reduce seismic risk and increase community resilience. However, the earthquake threat is highly time-dependent and involves terribly violent, but known, physics. To understand risk and improve resilience, we need to quantify earthquake hazards in physics-based models that can be coupled to engineering models of the built environment [1]. For example, the performance of critical distributed systems (water, medical, energy, etc.) depends on how complex earthquake wavefields interact nonlinearly with the mechanical heterogeneities of an entire cityscape, below as well as above the ground. Earthquake system science seeks to represent these complexities and interactions in system-level models.

The enabling technology of earthquake system science and physics-based PSHA is the numerical simulation of fault rupture dynamics and seismic wave propagation in realistic 3D models of the crust's heterogeneous structure. Research to achieve high performance has ranged worldwide [2] [3] [4], but some of the most advanced computational efforts have been organized under the Southern California Earthquake Center (SCEC). In 2001, SCEC established its "Community Modeling Environment" as a collaboratory for earthquake simulation. In 2005, we demonstrated the jump to terascale with the TeraShake simulations at the San Diego Supercomputer Center [5] [6]. We discovered how the rupture directivity of the southern San Andreas fault, a source effect, could couple to the excitation of sedimentary basins, a site effect, to substantially increase the seismic hazard in Los Angeles [5]. We also used dynamic rupture simulations to investigate how increases in source complexity can act the other way, reducing the directivity effect [6].

In 2008, the USGS adopted a SCEC simulation of a magnitude-7.8 San Andreas earthquake [7] as the basis for the first Great Southern California ShakeOut, the largest earthquake preparedness exercise of its time. This simulation was verified by comparisons of simulations from three different codes (Graves, Hercules, AWP-ODC) running at three national supercomputer centers [8]. The ShakeOut simulation provided system engineers, emergency responders, and disaster planners with a realistic, high-resolution scenario, prompting many detailed studies of a Katrina-scale catastrophe that have led to reductions in risk and improved resilience [9]. ShakeOut exercises are now performed in most of the United States and a growing number of other countries [10].

Improvements in earthquake simulation have closely tracked the development of leadership-class computational facilities. A major goal of the SCEC program—to simulate the largest expected ("wall-to-wall") earthquake on the San Andreas fault up to seismic frequencies exceeding 1 Hz—was achieved in 2010 on *Jaguar*, the first petascale machine at ORNL's Oak Ridge Leadership Computing Facility (OLCF). This Magnitude-8 (M8) scenario involved running the AWP-ODC finite-difference code on a uniform mesh comprising 436-billion elements for 24 hours at a sustained speed of 220 Tflops [11]. The computational size of the M8 simulation (mesh points × time steps) was almost 10^{17}, more than three orders of magnitude greater than the TeraShake

Conference'13, Nov, 2013, Denver, Colorado, USA

simulations. M8 highlighted the importance of two aspects of earthquake physics in hazard analysis: the propagation of dynamic fault ruptures at speeds exceeding the shear-wave speed, “super-shear rupturing”, which substantially modifies the seismic wavefield, and the existence of stresses high enough in the near-source and near-surface regions to require a nonlinear treatment of the deformations.

The arrival of petascale computing has opened the door to full-scale, physics-based PSHA. For example, earthquake simulations have recently been validated against ground motions recorded up to 4 Hz, with promising results [12], and we are pushing these comparisons to even higher frequencies. However, in order to calculate seismic hazards in California and other tectonically active regions, simulating just a few earthquakes won’t do; we must adequately sample earthquake distributions from probabilistic models, such as the Uniform California Earthquake Rupture Forecast (UCERF) [13]. Using standard “forward” simulation methods, computing three-component seismograms from M sources at N sites requires M simulations. For the UCERF2 model in Southern California, $M > 10^5$; i.e., hundreds of thousands to millions of possible earthquake sources must be modeled, which cannot be done directly, even at petascale.

To overcome this scale limitation, SCEC has built a special simulation platform, CyberShake, which uses the time-reversal physics of seismic reciprocity to turn the problem around [14]. A complete tensor-valued wavefield (the strain Green tensor or SGT) is calculated for a system of point forces at surface sites; seismic reciprocity then allows us to compute seismograms at those sites by fast (embarrassingly parallel) quadratures of the SGT over the fault surfaces. This “reciprocal” simulation method can generate 3-component seismograms for M sources at N sites with only $3N$ simulations. For the Los Angeles region, the near-surface geologic structure can be interpolated to produce high-resolution seismic hazard maps with N as small as 200-250, reducing the computations by a factor of 2,000. Scientific workflow software is used to manage the hundreds of millions of jobs needed to populate a CyberShake model [15].

Using the CyberShake platform, we have created the first physics-based PSHA models of the Los Angeles region from suites of simulations comprising ~10^8 seismograms. These models are “layered”, allowing earthquake engineers and other users to access ensembles of hazard curves (representing epistemic uncertainties), to disaggregate the calculations and identify the ruptures that dominate the hazard at a particular site, and to retrieve the actual seismograms, which can then be used to drive full-physics engineering models (Figure 1).

CyberShake brings the computational challenges of physics-based PSHA into sharp focus. The current models are limited to low seismic frequencies (≤ 0.5 Hz). Goals are to increase this limit to above 1 Hz and produce a California-wide CyberShake model using the new UCERF3 rupture forecast, which is scheduled to be released this year. The computational size of the statewide model will be more than 100 times larger than the current Los Angeles models. Our progress towards exascale is also being driven by the application of full-3D waveform tomography to the development of the seismic velocity models [16] [17], which are required as input to CyberShake. Full-3D tomography using 3-component seismograms from M sources observed at N stations requires at least $3N + M$ wavefield simulations per iteration [18].

This paper demonstrates that the computational power needed to accelerate CyberShake, and other applications of earthquake simulations will likely come from accelerator-based, many-core architectures. We present initial results on the performance of a finite-difference code, Anelastic Wave Propagation by Olsen, Day, and Cui (AWP-ODC) [11], that has been GPU-accelerated on the new OLCF system, *Titan*. Perfectly linear speedup has been achieved on up to 8192 Cray XK7 nodes.

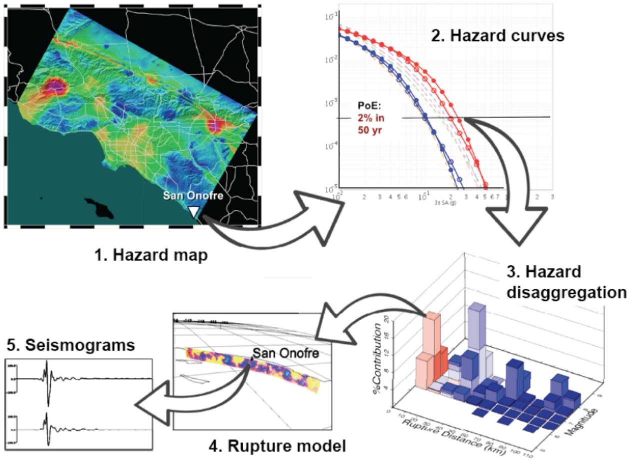

Figure 1: The CyberShake hazard model, showing the layering of information. (1) Hazard map for the LA region (hot colors are high hazard). (2) Hazard curves for a site near the San Onofre Nuclear Generating Station. (3) Disaggregation of hazard in terms of magnitude and distance. (4) Rupture with the highest hazard at the site (a nearby offshore fault). (5) Seismograms simulated for this rupture. Arrows show how users can query the model starting at high levels (e.g. hazard map) to access information of progressively lower levels (e.g. seismograms).

Section 2 of this paper states the computational problem in a general context, and Section 3 outlines our solution for AWP-ODC, which is based on an effective memory reduction scheme. Section 4 introduces single-GPU and multi-GPU implementation of the MPI-CUDA-based code. Section 5 demonstrates the parallel efficiency and summarizes the sustained performance achieved on petascale supercomputers. Section 6 applies these new capabilities to obtain two scientific results: (1) the first 10-Hz simulation on the *Titan* system and (2) the first CyberShake hazard curve generated using AWP-ODC-GPU on the NCSA *Blue Waters* system.

2. STATEMENT OF THE PROBLEM

To advance and apply our earthquake system science research within the practical limits of currently available open science HPC resources, significant computational performance improvements must be developed. Using GPUs, scientific applications in molecular dynamics, signal processing, magnetic materials, electromagnetics, and a host of other research areas have achieved orders-of-magnitude speedup [19] [20] [21] [22] [23] [24]. These applications are typically compute-bound, in contrast to a wide range of memory-bound scientific and engineering applications, such as earthquake and weather forecasting. The performance of memory-bounded codes is dominated by the memory system and arithmetic throughput. In GPU computing, the memory-bounded stencil calculations for large-scale production runs are limited in compute performance primarily due to their low computational intensity and poor data locality, which require significant amounts of costly data movement between CPUs and GPUs through the slow PCI Express (PCIe) bus because of MPI communication.

One successful stencil application on GPUs is the Phase-Field model, with which Shimokawabe et al. [25] achieved 1 Pflops (single precision) on the TSUBAME 2.0 Supercomputer in 2011. This application is less memory-bound, because its 2nd-order stencil equations require only single-layer ghost cells and two compute variables.

In contrast, popular finite difference (FD) seismic wave propagation codes use 4th-order, 13-point stencils with two ghost-cell layers [4] [5] [7] [26] [27] [28]. These stencils increase memory usage significantly because they require global memory sweeps through data structures that are much larger than the data caches available on GPUs. Memory-bound stencil calculations are known to achieve only a low fraction of peak performance [29], which explains why only a few FD seismic wave propagation codes have been ported to GPUs [29] [30] [31] [32] [33] [34]. FD GPU codes can be fine-tuned with CUDA asynchronous memory copy operations to overlap CPU/PCIe data transfer with GPU computation [30]. However, the performance is still burdened by CPU-GPU data communication. One of the few seismic applications tuned to a petascale heterogeneous system is the spectral-element package SPECFEM3D [35] [36]. The compute kernel of this application runs on GPUs while other computations remain on CPUs; hence, full advantage cannot be taken of the GPU accelerators. To our knowledge, no seismic application has thus far demonstrated sustained petascale performance for a science run.

This paper presents the capabilities and initial performance of AWP-ODC-GPU, a restructured CUDA-MPI code that solves 3D velocity-stress wave equations with an explicit, staggered-grid FD scheme. A hardware-oriented design has been developed to achieve high performance. The main problem is to make full use of GPU power by overlapping slow data communication with an extended computing region. The algorithms used are highly scalable because they carefully rearrange the order of computing and communication to hide latency, resulting in exceptional speedup and parallel efficiency. We implement a communication model that reduces the intra-node frequency of data movement between CPU and GPU and enables complete overlap of communication and computation. This model can be extended to general stencil computing on a structured grid. We have validated the simulation outputs in comparison to both reference models and earthquake observations.

3. NUMERICAL METHODS

AWP-ODC-GPU (hereafter abbreviated AWP) is based on the FD code originally developed by Kim Bak Olsen at University of Utah [37]. The AWP 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. This application has two computing modes: dynamic rupture and wave propagation mode. In this paper, we will focus on the wave propagation mode.

3.1 Wave Propagation Equations

The seismic forward problem calculates the propagation of seismic waves through spatially heterogeneous soil materials. AWP solves a coupled system of partial differential equations. The governing elastodynamic equations can be written as

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

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

where λ and μ are the Lamé coefficients and ρ is the density, ν and σ are particle velocity vector and symmetric stress tensor respectively. 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.

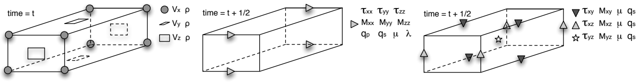

Figure 2: Staggering of the wavefield parameters, where V_i are particle velocities, τ_{ij} are stress tensor components, M_{ij} are memory variables, ρ, λ, μ are elastic parameters, q_p and q_s are quality factors for *P* and *S* waves, respectively.

3.2 Staggered-Grid Finite Difference Equations

The nine governing scalar equations are approximated by finite differences on a staggered grid in both time and space (see Figure 2). Time derivatives are approximated by

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

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

For the spatial derivatives, let *Φ* denote a generic velocity or stress component, and *h* be the equidistant mesh size. The FD approximation to the partial derivative with respect to x at grid point (*i,j,k*) is

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

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.

Truncation of the 3D modeling domain on a computational mesh inevitably generates undesirable reflections. Absorbing boundary conditions (ABCs) are designed and optimized to reduce these reflections to the level of numerical noise. AWP implements ABCs based on simple 'sponge layers' [38]. The ABCs apply a damping term to the full wavefield inside the sponge layer and are unconditionally stable.

3.3 Anelastic attenuation in AWP-ODC

Seismic waves are subjected to 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., [39]) and standard linear solid models (e.g., [40]). Here, we implemented an efficient coarse-grained methodology in AWP [41] [42], which significantly improves the accuracy of the stress relaxation schemes. 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 performance or memory. The quality factor *Q* (separate for S and P waves) is in this formulation expressed as

$$Q^{-1}(\omega) \approx \frac{\delta M}{M_u}\sum_{i=1}^{N}\frac{\lambda_i \omega \tau_i}{\omega^2\tau_i^2+1} \tag{4}$$

where δM is the relaxation of the modulus, M_u is the unrelaxed modulus, λ_i are weights used in the associated quadrature calculations, τ_i are the relaxation times, and ω is angular frequency. Each stress component has associated with it N memory variables $\varsigma_i(t)$ (one variable co-located with each stress component in the staggered grid, see Figure 2).

$$\sigma(t) = M_u\left[\varepsilon(t) - \sum_{i=1}^{N} \varsigma_i(t)\right] \qquad (5)$$

where $\sigma(t)$ is stress and $\varepsilon(t)$ is strain. We use N=8 to obtain sufficient accuracy in the implementation.

3.4 Strain Green Tensor Calculations

Alternatively, the strain Green tensor (SGT) can be simulated and utilized in reciprocal methods to produce waveforms. The strain Green tensor can be calculated as

$$\boldsymbol{H}(\boldsymbol{r}, t; \boldsymbol{r}_s) = \frac{1}{2}\left[\partial_i^S G_{jn}(\boldsymbol{r}, t; \boldsymbol{r}_s) + \partial_j^S G_{in}(\boldsymbol{r}, t; \boldsymbol{r}_s)\right] \qquad (6)$$

where G_{in} is the ith component of the displacement response to the nth component of a point force at $\boldsymbol{r}_S$, and the spatial gradient operator acts on the field coordinate $\boldsymbol{r}$ [43]. The SGT can be computed from the stress-field by applying the stress-strain constitutive relation. The displacement field is linearly related to the seismic moment tensor $\boldsymbol{M}$:

$$u_n(\boldsymbol{r}, t; \boldsymbol{r}_s) = \boldsymbol{H}(\boldsymbol{r}, t; \boldsymbol{r}_S) : \boldsymbol{M} \qquad (7)$$

Therefore, the elements of the SGT can be used in earthquake source parameter inversions to obtain the partial derivatives of the seismograms with respect to the moment tensor elements. By directly using the strain Green tensor, we can improve the computational efficiency in waveform modeling while eliminating the possible errors from numerical differentiation [43]. Seismic reciprocity can then applied to compute synthetic seismograms from SGTs, from which peak spectral acceleration values are computed and combined into hazard curves [43] [44].

4. AWP IMPLEMENTATION DETAILS

In this section we present implementation details of our GPU application AWP. We introduce a C/CUDA/MPI implementation whose initial development was part of co-author Zhou's graduate research [45] [46]. We will emphasize the key points that led to the extraordinary scaling performance we obtained for the GPU application.

4.1 Computation Kernel

In AWP, two computation kernels for velocity and stress are carried out in sequence for wave propagation simulations based on the numerical approximation of the partial differential equations [1-3]. At each time step in the main loop, for each mesh point in the domain, first the velocity computation kernel updates three velocity components (in *X, Y*, and *Z* directions) by using the six stress components (on *XX, YY, ZZ, XY, XZ,* and *YZ* faces), and then the stress computation kernel employs these updated velocity components to update the six stress components. We have twenty-one 3D arrays to be maintained in the memory to process the wave propagation, including velocity, stress and coefficients. The size of each 3D array is the same as the 3D simulation domain. Figure 3 shows three examples for the memory access pattern for velocity v_x, stress *xx* and stress *xy* computation kernels. Approximately 136 reads, 15 writes and 307 FLOPs calculations are involved for each point of the 3D domain in one iteration. The Flops to bytes ratio is around 0.5, with low computational intensity. Improving the data locality has been the key to achieve high performance.

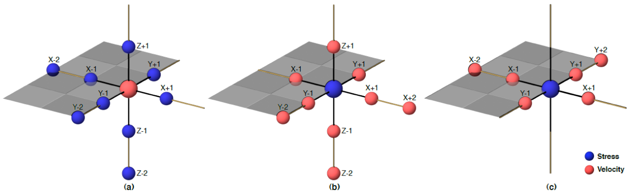

Figure 3: (a) 13-point asymmetric stencil computation for velocity v_x: a velocity center point computation requires 13 points stress input including 4 from the same center location and 9 others from neighborhood. Computation for velocity v_y and v_z has similar format with different neighborhood input. (b) 13-point asymmetric stencil computation for stress (*xx, yy, zz*) with very similar stencil format as velocity v_x. (c) 9-point asymmetric stencil computation for stress *xy*: the input velocity only involves *x* and *y* directions. Computation for stress *yz* and *xz* has similar format with different directions.

4.2 AWP-ODC Fortran/MPI Code

The CPU-based AWP-ODC software is highly scalable, composed of solvers (dynamic rupture and wave propagation), pre-processing tools (PetaSrcP, PetaMeshP) and other post-processing workflow tools [11]. The code achieves excellent strong scaling up to 223K cores on XT5. Scalable IO in the code uses MPI-IO to handle petabytes of simulation data [11]. Newly added features include checkpointing using ADIOS, and outputting in HDF5 format which enable time and storage space optimizations.

4.3 Single-GPU AWP Implementation

The AWP code was re-structured from scratch to enable GPU computation. The initial programming effort was to convert the Fortran/MPI code to a serial CPU program in C. Then we added CUDA calls and kernels to the application for GPU computation [45]. Each GPU is controlled by an associated CPU. The design is implemented with maximum throughput for heterogeneous computing environments in mind.

Various optimization approaches are implemented to improve the data locality: 1) memory is coalesced for continuous CUDA thread data access, 2) register usage is optimized to reduce global memory access, 3) L1 cache or shared memory usage is optimized for data reuse and register savings, and 4) read-only memory is employed to store constant coefficient variables because of read-only cache benefits [45].

4.4 Multi-GPU AWP Implementation

The novel MPI-GPU implementation includes algorithm-level communication reduction, effective overlap of communication and computation and scalable IO.

4.4.1 Communication and Computation

Unlike the CPU code with 3D decomposition, our multi-GPU code uses a two layer decomposition where each is 2D (see Figure 4). The 3D domain (*NX, NY, NZ*) is partitioned into (*PX, PY*, 1) sub-domains. Each GPU is responsible for the computation of its own sub-domain with dimension of (nx, ny, NZ). The sub-domain is further partitioned along *Y* and *Z* axes inside the GPU for different streaming multiprocessors (SM).

One of the benefits of using a 2D decomposition is that, for the 3D arrays in the GPU memory, two consecutive locations correspond to data related for two neighboring mesh points in the *Z* direction, i.e. GPU memory is managed as fast-*Z*. In this fashion, memory locality is increased and the memory access latency is reduced. Another benefit of this decomposition is that there are no neighboring sub-domains along the *Z* direction, and hence the number of neighbors is reduced from 6 to 4 for inner sub-domains. With this approach, the total amount of MPI communication is reduced by approximately 33%.

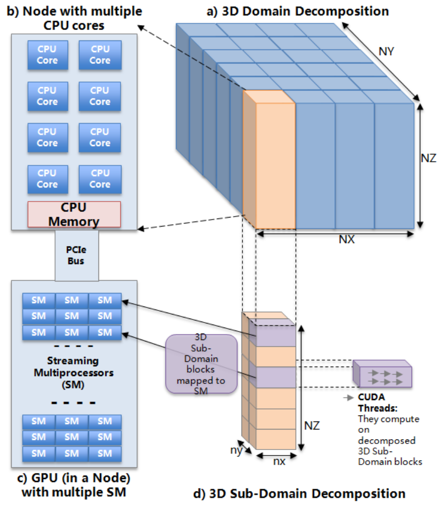
b) Node with multiple CPU cores
a) 3D Domain Decomposition
CPU Core
CPU Memory
NY
NZ
NX
PCIe Bus
SM
Streaming Multiprocessors (SM)
3D Sub-Domain blocks mapped to SM
CUDA Threads: They compute on decomposed 3D Sub-Domain blocks
NZ
ny
nx
c) GPU (in a Node) with multiple SM
d) 3D Sub-Domain Decomposition

Figure 4: Two-layer 3D domain decomposition: *X&Y* decomposition for GPUs and *Y&Z* decomposition for GPU SMs.

We take an innovative approach to reduce the amount of communication and latency. The primary concept is to extend the ghost cell region by adding two additional layers, and hence manage a ghost cell region with thickness of 4 mesh points in total, in both *X* and *Y* directions. We exchange 4 layers of velocity data of ghost cells, resulting in up-to-date velocity data for (*nx*+8, *ny*+8, *NZ*). After this, each GPU computes stress for a domain of size (*nx*+4, *ny*+4, *NZ*) including 2 layers of ghost cells. The computed stress is then used to compute velocity for the sub-domain of size (*nx, ny, NZ*). That means at each iteration we exchange twice as much velocity data but no stress data. Note that we now exchange 33% less data with halved communication frequency, as velocity has three variables and stress has six. This is a significant saving in communication with single exchange per iteration, compared to two exchanges per iteration in the CPU code. Moreover, we gain more time to overlap communication with computation without synchronizing stress for ghost cell data. The slight increase in memory and computation requirements is upper bounded by $NZ \times (4 \times (nx + ny) + 16)$, which can easily be obtained by setting sub-domain size as (*nx*+4,*ny*+4,*NZ*) rather than (*nx,ny,NZ*). For our benchmark block size of 160 × 160 × 2048, this upper bound corresponds to a 5% increase in the memory requirement.

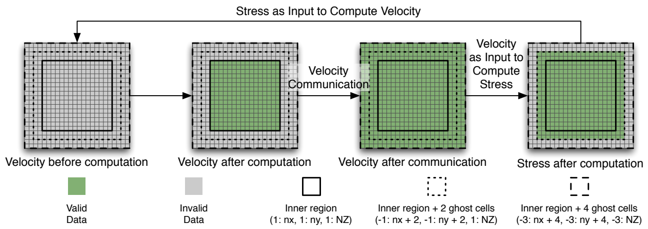
Stress as Input to Compute Velocity
Velocity Communication
Velocity as Input to Compute Stress
Velocity before computation
Velocity after computation
Velocity after communication
Stress after computation
Valid Data
Invalid Data
Inner region (1: nx, 1: ny, 1: NZ)
Inner region + 2 ghost cells (-1: nx + 2, -1: ny + 2, 1: NZ)
Inner region + 4 ghost cells (-3: nx + 4, -3: ny + 4, -3: NZ)

Figure 5: Communication reduction - extend ghost cell region with extra 2-layers and utilize computation instead of communication to update the ghost cell region before stress computation. The 2D *XY* plane represents 3D sub-domain, no communication is required in *Z* direction due to 2D decomposition for GPUs.

The communication approach introduced in Figure 5 requires two extra layers of ghost cells, for we need data for all four corners. We introduced an in-order communication method - first west/east, then north/south. As a result we are able to exchange diagonal cell information without adding additional MPI messages [46].

We employ an ordered scheduling to manage asynchronous communication and computation efficiently as illustrated in Figure 6. We first compute the velocity of the boundary region, which corresponds to ghost cells of a neighboring sub-volume (V1-V4). While this data is asynchronously copied to CPU and being sent to neighbors through MPI, GPU computes the velocity (V5) and stress (S5) for the inner region. When the data exchange is done and velocity data for the ghost cells is received, it is copied back to GPU asynchronously. After the velocity data for the ghost cells is copied, GPU computes stress for the boundary region (S1-S4) [46].

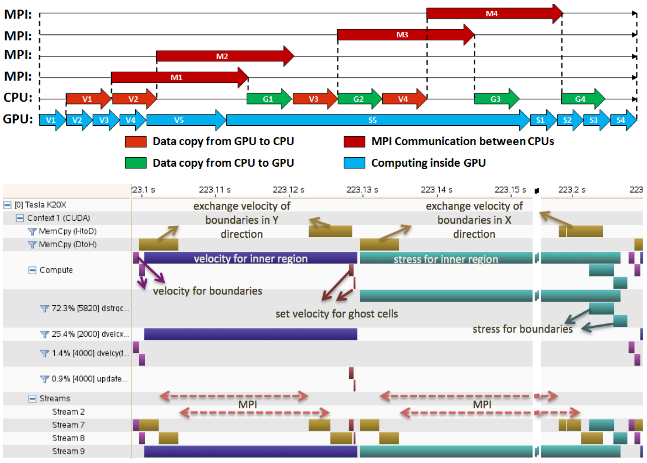

Figure 6: Overlap of computation and communication overlapping. Top: concept scheme. Bottom: *nvvp* profiler output matches well with the design, achieving complete overlap.

4.4.2 I/O

The AWP code is capable of handling large number of dynamic sources and petabytes of heterogeneous mesh inputs. The dynamic sources consist of the positions of earthquake source stations, and stress data associated with each source station. In our 10-Hz simulation case (Section 6), the mesh input is 4.9 TB, and the source is as large as 1.9 TB. These dynamic sources are computed based on the accurate and verified staggered grid, split-node scheme [47]. Multi-million sources are highly clustered in a concentrated grid area, resulting in hundreds of gigabytes of source data assigned to a single core. Copying this data to GPUs through PCIe is an additional challenge at runtime.

We support the sources and mesh in 3 different modes: serial reading of a single file, concurrent reading of pre-partitioned files, and concurrent reading through MPI-IO. Source partitioning involves both spatial and temporal locality required to fit in the GPU memory. Parameters are introduced to control how often the partitioned source is copied from CPUs to GPUs. This feature allows CPUs to read in large chunks of source data to avoid frequent access to file system, while GPU only copies over the amount it can afford. Our implementation has demonstrated excellent scalability in handling the initial dataset.

AWP uses MPI-IO to write the simulation outputs to a single file concurrently. This works particularly well as more memory is available on CPUs to allow effective aggregation of outputs in CPU memory buffers before being flushed. We support run-time parameters to select a subset of the data by skipping mesh points as needed.

4.4.3 AWP API Implementation

We also implemented a generic API that employs the pthreads to take advantage of the idle CPU cores which can work on other independent tasks in parallel. For each computing node with multiple CPU cores, only 1 core/thread is requested to run the regular GPU solver since each node has only 1 GPU on XT7. Hence the other cores/threads are available during the running period, and a pthread-based API has been introduced to run some other workloads simultaneously.

Our first pthread task is the output. We separated the output related operations from the computation code and implemented them in the output thread. After the main thread finishes the initialization, the output thread starts and passively waits until the main thread signals that it is time to save velocity data. Then the output thread wakes up, launches a kernel on GPU to save velocity data into a buffer on the GPU device, and copies that buffer back to the CPU host. After data copy is done, the output thread gives signal back to the main thread to compute the next timestep. At the same time, the output thread prepares the aggregated data to write into the disk while waiting for signal for the new generated velocity data. Therefore, the output task reduces some non-computation activities in the main thread and makes full use of the computing resource. Other potential tasks will be related to post-processing tools, visualization, analysis tools to gather statistics from the run, or interactive control tools.

4.5 Implementation of SGT Calculations

The SGT generation step is by far the most time-consuming processing step in the CyberShake workflow, accounting for approximately 90% of the CPU-hours. Therefore, we have adapted AWP for CyberShake, using the GPU solver to accelerate the process of calculating SGTs (hereafter abbreviated AWP-SGT). We implemented two effective IO communication schemes for calculating SGTs. The first uses serial IO with the velocity mesh partitioned in advance. The second utilizes run-time partitioning inside the solver, using MPI-IO. The code supports 2D decomposition on CPUs, where each processor is responsible for performing stress and velocity calculations within its own subgrid of the simulation volume, while allowing GPUs to handle SGT calculations. This code has been extensively verified by comparing stress and strain outputs of earthquake sources to those from a reference model. Such verification is crucial during optimization and code updates.

4.5.1 Co-scheduling

When the SGT calculations are performed on GPUs, the CPUs on the same nodes are mostly idle except for handling IO and communications, which could be a potential waste of the resources. We present a runtime environment for co-scheduling across CPUs and GPUs. We motivate this work because the CyberShake workflow consists of two parts: a parallel AWP-SGT calculation, and high-throughput reciprocity calculations with each rupture variation to produce seismograms and intensity measures of interest. This is described in more detail in section 6.2. Co-scheduling enables us to perform both calculations simultaneously on XK7 nodes, reducing our time-to-solution and making efficient use of all available computational resources.

To enable co-scheduling, we launch multiple MPI jobs on XK7 nodes via multiple calls to *aprun*, the ALPS utility to launch jobs on compute nodes from a mom node. We use core specialization when launching the child *aprun* calls to keep a core available for GPU data transfer and communication calls, as both the GPU and CPU codes use MPI. Testing has shown that this approach results in little to no impact on the GPU performance. To prevent overloading the mom node with too many simultaneous *aprun* calls, we limit the number of child *aprun* calls to 5-10.

Since calculating a pair of SGTs requires approximately 60 GPU hours, and the CyberShake post-processing requires about 1000 CPU hours, the post-processing is able to complete on the 15 available CPUs per XK7 node while SGTs are calculated on the GPUs. We have successfully tested co-scheduling with the first half of CyberShake post-processing, calculating SGTs on 50 GPUs while performing post-processing with 10 child jobs of 5 nodes each. We anticipate full CyberShake co-scheduling capabilities in the near future.

4.5.2 Hazard Curve Calculation

PSHA results are typically delivered by hazard curves, which relate ground motion on the *X*-axis to probability of exceeding that level of ground motion on the *Y*-axis, for a site of interest. To verify AWP-SGT, we calculated a CyberShake hazard curve using the GPU version of AWP-SGT, and compared it to a hazard curve using the CPU version; the two are numerically almost identical. Calculation of a hazard curve involves SGT timeseries data from over half a million locations in the volume, providing rigorous verification.

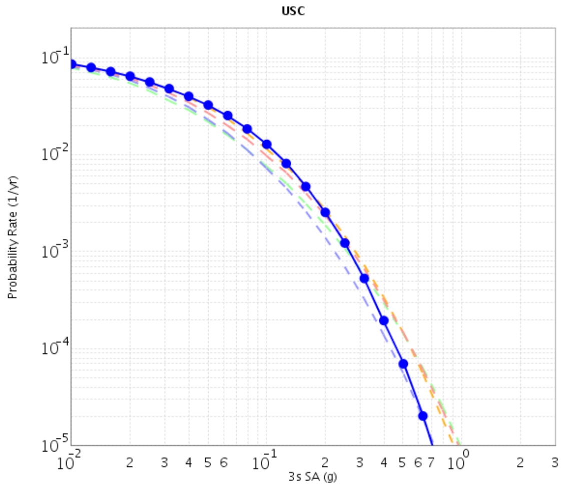

Figure 7: PSHA hazard curve calculated for the University of Southern California (USC) site. The horizontal axis represents ground motion at 3 seconds spectral acceleration, in terms of *g* (acceleration due to gravity). The vertical axis gives the probability of exceeding that level of ground motion. The blue line is the curve calculated using CyberShake with AWP-SGT. The dashed lines are hazard curves calculated using four common attenuation relationships which provide validation of the CyberShake methodology.

4.6 Verification

We performed a variety of tests to ensure that AWP produces results comparable in accuracy to those for widely used and validated SCEC community codes running on HPC systems. We started with a wave propagation simulation of the magnitude-5.4 Chino Hills earthquake at frequencies up to 2.5 Hz using 128

Keeneland GPUs, with extended sources active for 2.5 seconds [46]. The results are verified with those from our CPU code, showing almost identical simulation results.

Comparing the velocities with negligible error is necessary, but not sufficient for the execution of accurate simulation of ground motions. Even small errors can accumulate over time if they are correlated or biased. We then examined further the correction of the seismograms using the SGT calculations. We demonstrated that the results from the GPU code and reference model are nearly identical, in a 1.2 billion mesh point volume for 20K timesteps.

5. PERFORMANCE ANALYSIS

We present the strong and weak scaling results obtained on OLCF Titan, NCSA Blue Waters and Georgia Tech Keeneland.

AWP has undergone extensive fine-tuning on NVIDIA Fermi GPUs, but the team has had only limited time to analyze performance and optimize for NVIDIA Kepler GPUs, like those in Titan and Blue Waters. We have observed that for small sub-domain sizes, accessing input arrays through the GPU's texture cache sped up the two primary compute kernels by a combined 1.9X. This speed-up is due to a reduction in global memory transactions. At larger sub-domain sizes, like those used for the scaling results below, the local data becomes too large for the texture cache, which negates the benefit of this change. We did see more modest gains by loading some, but not all, input arrays through the texture cache. In the future we intend to explore the use of per-SM shared memory to more selectively stage data arrays to achieve the same reduction in global memory transactions. This may have the additional benefit of reducing the register usage per thread and increasing occupancy.

5.1 Benchmark Machine Specifications

The OLCF Titan is a Cray XK7 supercomputer located at the Oak Ridge Leadership Computing Facility (OLCF), with a theoretical peak double-precision, floating point performance of more than 20 petaflops. Titan consists of 18,688 physical compute nodes, where each compute node is comprised of one 16-core 2.2GHz AMD Opteron™ 6274 (Interlagos) CPU, one NVIDIA Kepler (K20X) GPU, and 32 GB of RAM. Two nodes share a Gemini™ high-speed interconnect router, which are connected in a 3D torus [48]. The Blue Waters system is a Cray XE6/XK7 hybrid machine composed of AMD 6276 "Interlagos" processors (nominal clock speed of at least 2.3 GHz), NVIDIA K20X accelerators, and Cray Gemini interconnect [49]. The Keeneland Full Scale (KFS) system consists of a 264-node cluster based on HP SL250 servers. Each node has 32 GB of host memory, two Intel Sandy Bridge CPU's, three NVIDIA M2090 (Fermi) GPUs, and a Mellanox FDR InfiniBand interconnect. The total peak double precision performance is around 615 TFlops [50].

5.2 Strong Scaling and Weak Scaling

The strong scaling benchmarks were performed on NCSA Blue Waters and OLCF Titan. The small fixed size benchmark was run on Blue Waters whereas others were on Titan (Figure 8). The degradation in performance with the increase of the number of GPUs is expected, as the application becomes bounded by communication overhead that arises from less compute work. As the number of GPUs is increased, so does the outer halo region to total sub-volume size ratio in proportion, making our application less effective in overlapping communication and computation.

With regard to weak scaling, the perfect linear speedup was observed on 90 Keeneland Initial Delivery System (KIDS) nodes equipped with 3 NVIDIA M2090 GPUs per node, where 10% of the peak performance was achieved. Figure 9 and Table 1 show the AWP code's extraordinary scaling performance with 100% parallel efficiency for weak scaling from 16 up to 8192 Titan nodes. In this benchmark, each GPU carries out stencil calculations for a sub-domain with size 160 × 160 × 2048. The total number of points in the domain becomes 160 × 160 × 2048 × N, where N represents the number of GPUs used. To the best of our knowledge, this is a record speedup from a highly memory-bounded scientific application achieved on Cray XK7. Perfect linear weak scaling indicates that our careful design of communication model is able to hide communication latency by computation efficiently.

Notable slowdown was observed in the case of 16,384 nodes, although we still achieve 93.5% parallel efficiency. Since the application performs only nearest-neighbor communications, we would expect continued linear scaling. The source of this performance degradation is not yet fully understood, but we believe that the topology of the network may have played a significant role. We intend to explore the effect of node topology and evaluate the benefit of topology-aware node placement in the future.

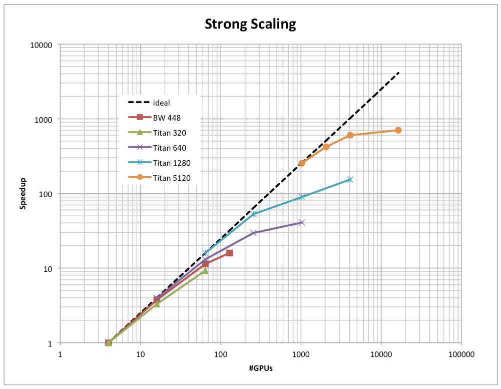

Figure 8: Speedup of strong scaling on Cray XT7 at ORNL, with 2D square configuration (Z direction fixed as 2048) for problem size of 320, 640, 1280 and 5120.

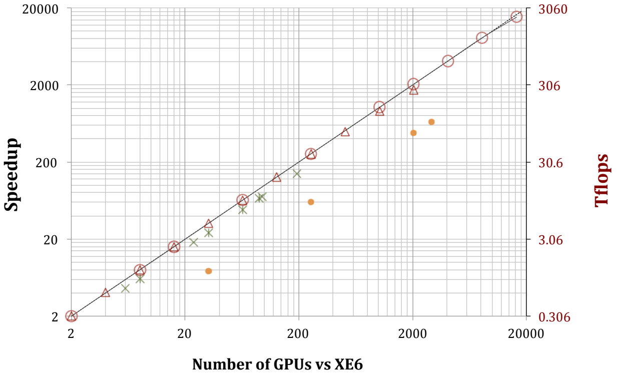

Figure 9: Weak scaling and sustained performance using AWP-ODC-GPU in single precision. XK7 exceeds XE6 performance by a factor of 4.2. Solid (dashed) black line is (ideal) speedup on Titan, Rounds/triangle/cross points are FLOPS performance on Titan/Blue Waters/Keeneland. Solid round points are FLOPS on Blue Waters XE6. A perfect linear speedup is observed between 16 and 8,192 nodes. A sustained 2.3 Pflop/s performance was recorded on 16,384 Titan nodes.

5.3 Sustained Performance

We calculate the performance by measuring the average time spent on one time step after running a benchmark test for 2,000 time steps. The number of floating point operations is counted in the code based on 307 FLOP per mesh point per time step. Initialization and output writing parts are excluded from this calculation. The IO time is negligible when time iterations of tens to hundreds of thousands of time steps are involved. We obtained a sustained performance estimate of 2.33 PetaFlops on 16,384 Titan GPUs. This was a 2,000 time-step benchmark run of a problem size of 20,480 × 20,480 × 2,048 or 859 billion mesh points.

Our main scientific findings using the code were obtained from a rough-fault simulation with a domain size 416 km × 208 km × 41 km with a spatial resolution of 20 meters at a maximum frequency resolution of 10-Hz, discretized into 443 billion mesh points. The size of this run is slightly larger than the record M8 San Andreas fault simulation [11]. The run took only 5 hours and 30 minutes to complete 170 seconds of simulation time whereas M8 ran on approximately 220K CPU cores for 24 hours. We emphasize that the 10-Hz rough-fault simulation included 6.8 TB input and 170 GB output. To our best knowledge, this is the first sustained petaflop seismic production simulation to date, and a new record for earthquake simulation in terms of scale. These results are particularly remarkable considering that memory-bounded stencil computations typically achieve a low fraction of theoretical peak performance.

Table 1: Time-to-solutions and Parallel efficiency

XK7 Nodes used	Elements (Thousands)	Wall Clock Time	Parallel Efficiency
16 (4 × 4)	838,860	0.1085	100%
32 (4 × 8)	1,677,721	0.1084	100%
64 (8 × 8)	3,355,443	0.1085	100%
128 (8 × 16)	6,710,886	0.1085	100%
256 (16 × 16)	13,421,772	0.1085	100%
512 (16 × 32)	26,843,545	0.1085	100%
1024 (32 × 32)	53,687,091	0.1085	100%
2048 (32 × 64)	107,374,182	0.1084	100%
4096 (64 × 64)	214,748,364	0.1085	100%
8192 (64 × 128)	429,496,729	0.1085	100%
16384 (128 × 128)	858,993,459	0.1159	93.2%

Both the benchmark and rough fault runs produced remarkable scaling results for the GPU-enable AWP code. We also compared the performance against CPU systems. The benchmark results indicate that using GPU accelerators on Cray XK7 improves the performance by a factor of 5.2 compared to CPU-only usage of XK7 nodes. Furthermore the performance of XK7 exceeds Cray XE6 by a factor of 2.5 when 512 nodes are in use. We expect our code's performance on XK7 nodes to improve further compared to XE6 as the number of nodes increases. The reason is that the CPU code suffers more from the increasing communication costs because of the lack of effective overlap.

5.4 Time-to-solution and Performance-to-cost Analysis for CyberShake Calculations

One of the primary motivations of implementing AWP is to accelerate CyberShake calculations. We are planning to use CyberShake to calculate a California state-wide seismic hazard map with a maximum frequency of 1 Hz. When using the heavily optimized CPU code AWP-ODC, it is expected to require 662 million allocation hours to complete. Our AWP-SGT GPU code running on XK7 demonstrates a performance improvement of a factor of 3.7 compared to the CPU code running on XE6. Table 2 provides some detailed comparisons of calculating SGTs on XK7 versus XE6, and demonstrates the saving of 579 millions of allocation hours when using the accelerated (CPU+GPU) AWP.

Table 2: CyberShake Strain Green Tensor Calculations

CyberShake	CPU[1]only	GPU[2] only	CPU+GPU[2]
XE6[1]/XK7[2] nodes	400	400	400
WCT[3] per site	10.36 hr	2.80 hr	2.80 hr
Total SUs charged[4]	662 M	168 M	168 M
Saved in Million SU[5]		495 M	**579 M**

1) XE6 node (dual Interlagos); 2) XK7 (Operaton+Kepler K20X); 3) Wall clock time based on measurements on Cray XE6/XK7 at NCSA for two Strain Green Tensor calculations per site; 4) Based on total 5000 sites required for the generation of California state-wide seismic hazard map at a maximum frequency resolution of 1-Hz; 5) CPU+GPU saving counts the use of XK7 CPUs for post-processing of seismogram extraction as co-scheduling, involving 6.2 million rupture variations calculations per site.

6. SCIENTIFIC RESULTS

We have applied these new AWP-ODC-GPU capabilities to obtain the first 10-Hz deterministic simulation on the *Titan* system and the first CyberShake hazard curve on the NCSA *Blue Waters* system.

6.1 Ground Motion Up To 10-Hz

High-frequency (>1 Hz) deterministic ground motion predictions are critical input to performance-based building design. The accuracy of the simulations is limited by the small-scale complexity of the source and by high-frequency wave scattering in the crust. To investigate this problem, we have simulated high-frequency ground motions on a mesh comprising 443-billion (20,800 × 10,400 × 2,048) elements in a calculation that includes both small-scale fault geometry and media complexity. Specifically, we have computed the ground motion synthetics using dynamic rupture propagation along a rough fault imbedded in a velocity structure with heterogeneities described by a statistical model. We first carried out simulations of dynamic ruptures using a support operator method [51], in which the assumed fault roughness followed a self-similar fractal distribution with wavelength scales spanning three orders of magnitude, from $\sim 10^2$ m to $\sim 10^5$ m. We then used AWP to propagate the ground motions out to large distances from the fault in a characteristic 1D rock model with and without small-scale heterogeneities. The latter employed the moment-rate time histories from the dynamic rupture simulations as kinematic sources. Figure 10 shows snapshots of the rupture surface wave propagation for crustal models with and without the media heterogeneities. The fractal roughness is controlled by a Hurst number, which we set at 0.2, and the size of the heterogeneity by a standard deviation, which we set at 5%, as constrained by near-surface and borehole velocity data. Note how the wavefield in the bottom snapshot is scattered the small-scale heterogeneities, which generates realistic high-frequency synthetics. A few seismograms are shown to compare models with and without the small-scale structure.

The simulation results show realistic features. The acceleration spectra from the simulation are nearly flat up to almost 10 Hz, in agreement with theoretical predictions. Moreover, the simulated response spectra compare favorably with spectra obtained from the empirical ground motion prediction equations (GMPEs) currently used by building engineers, which are calibrated to high-frequency recordings of earthquake ground motions.

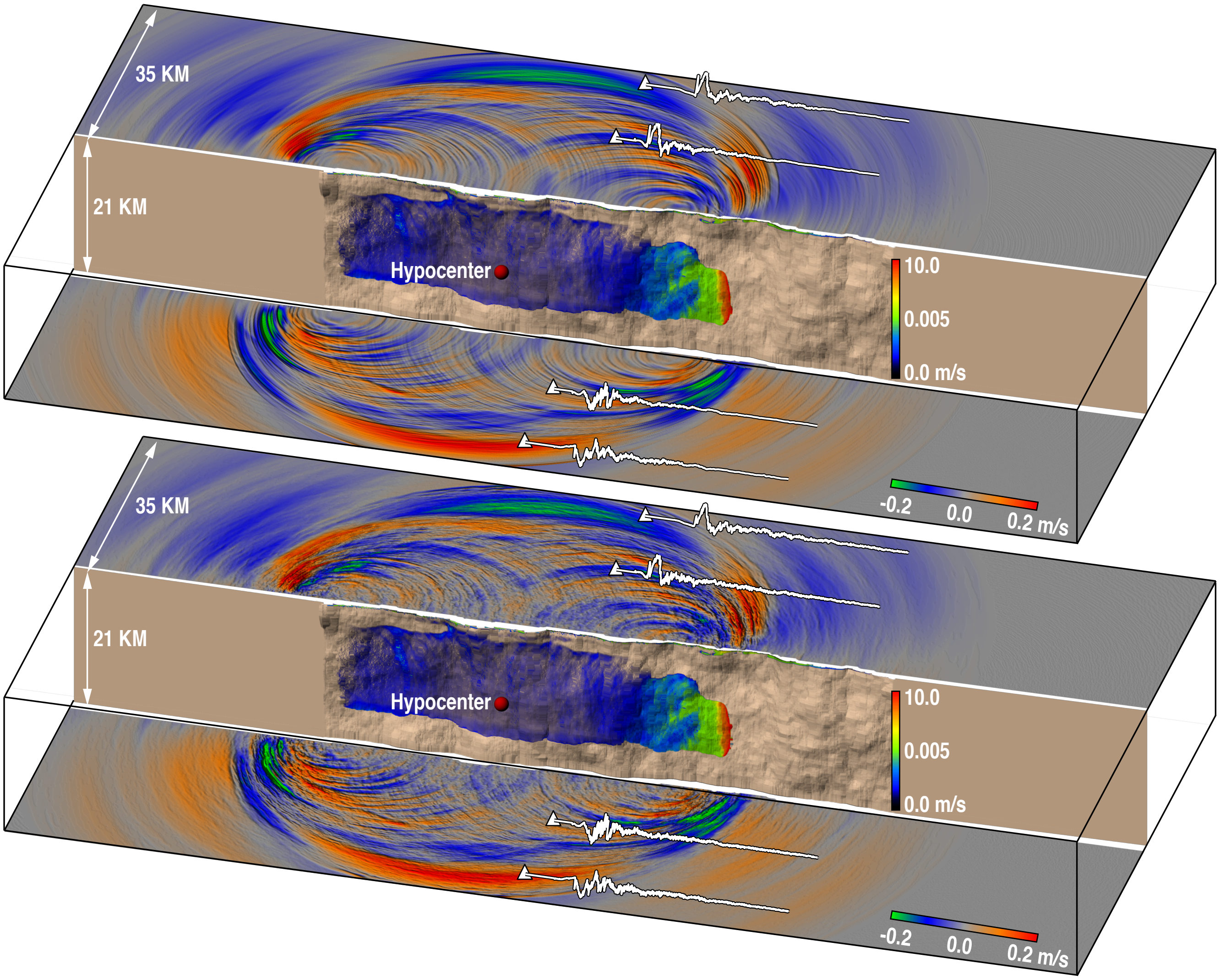

Figure 10: Snapshots of 10-Hz rupture propagation (slip rate) and surface wavefield (strike-parallel component) for a crustal model (top) without and (bottom) with a statistical model of small-scale heterogeneities. The displayed geometrical complexities on the fault were included in the rupture simulation. The associated synthetic strike-parallel component seismograms are superimposed as black traces on the surface at selected sites. The part of the crustal model located in front of the fault has been lowered for a better view. Note the strongly scattered wavefield in the bottom snapshot due to the small-scale heterogeneities.

6.2 Cybershake Hazard Model

PSHA estimates the probability that earthquake ground motions at a location of interest will exceed some intensity measure, such as peak ground velocity or peak ground acceleration, over a given time period. Results are delivered in the form of hazard curves for a site of interest and hazard maps for a region (see Figure 1). These kinds of estimates are highly useful for civic planners, building engineers, and insurance agencies, and, through building codes, they influence billions of dollars of construction yearly.

As described in the introduction, physics-based PSHA requires very large ensembles of deterministic forward simulations. SCEC has developed the CyberShake methodology to incorporate 3D ground motion simulations into seismic hazard calculations [14]. To calculate a waveform-based seismic hazard estimate for a site of interest, we begin with UCERF2 [13] and generate multiple rupture realizations with differing hypocenter locations and slip distributions (sampled from an appropriate stochastic rupture model). A geo-referenced mesh of approximately 1.2 billion points is then constructed and populated with seismic velocity information from a SCEC Community Velocity Model. A body-force impulse is placed at the site of interest and the resulting 20K timestep simulation illuminates the volume, calculating SGTs. Seismic reciprocity is used to post-process the SGTs and obtain synthetic seismograms and peak intensity measures for each rupture variation [43]. These are combined with the UCERF2 rupture probabilities to produce probabilistic seismic hazard curves for the site using the OpenSHA hazard analysis code [52]. Figure 11 illustrates the CyberShake workflow.

A major computational challenge how to increase the overall computational efficiency of the CyberShake workflow, which must combine the execution of the massively parallel SGT calculations with many embarrassingly parallel post-processing jobs. We have successfully utilized workflow tools to manage the data and job dependencies [15]. Looking ahead, we plan to increase the frequency of the model from 0.5 Hz to 1.0 Hz, which will require simulation volumes with eight times the mesh points and simulations with twice the timesteps. In addition, the new UCERF3 earthquake rupture forecast, which will be released this year, will include more numerous and more complex ruptures, increasing the number of seismograms per site by a factor of about 15. These computational requirements drive the need for scalable, heterogeneous approaches to workflow execution.

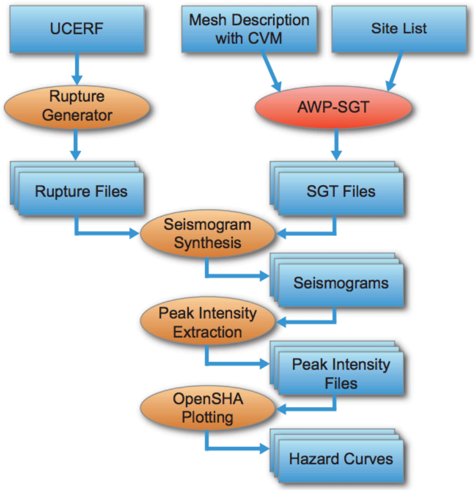

Figure 11: CyberShake workflow. Circles indicate computational modules and rectangles indicate files and databases.

Through an innovative co-scheduling approach, we have shown how CyberShake can make efficient use of GPUs and CPUs in heterogeneous systems. By running AWP-SGT on GPUs and doing high-throughput computations on the CPUs, we are able to run CyberShake workflows at a scale which now brings a 1-Hz California-wide CyberShake hazard model within reach.

7. CONCLUSIONS AND FUTURE WORK

We have re-designed the AWP-ODC code to accelerate wave propagation simulations on GPU-powered heterogeneous systems An aggressive architecture-oriented optimization has maximized throughput and memory locality, providing much better performance than our highly optimized CPU-based code. Algorithm-level communication reduction, effective overlap of communication and computation, and scalable IO have produced a GPU-based AWP code that achieves perfect speedup and a sustained petaflops capability.

AWP provides scientists, for the first time, with the ability to simulate ground motions from large fault ruptures to frequencies as high as 10 Hz in a physically realistic way. We have demonstrated this capability with simulations that incorporate both the fractal roughness of faults, which is thought to enhance the generation of high-frequency seismic waves, and the fractal heterogeneity of the crust, through which the waves are strongly scattered. The resulting ground motions compare favorably with leading GMPEs and provide guidance to further refine high-frequency simulations.

These results will change how synthetic seismograms are produced for use in earthquake engineering. Currently, the only way to compute synthetic seismograms across the full bandwidth of engineering interest (0.1-10 Hz) is to combine low-frequency deterministic simulations with high-frequency stochastic simulations [53] [54] [55]. The latter are obtained from *ad hoc* models that match the observed spectral content of the observations but do not satisfy the anelastic wave equations. The lack of a physics-based model makes it difficult to transport what is learned about the high-frequency behavior of one earthquake into forecasting the effects of future earthquakes. Our research shows how better physics can be incorporated into solutions of this problem.

We have also transformed the GPU-powered AWP to calculate SGTs. Our results show that it can serve as the main computational engine for CyberShake. The use of the AWP-SGT code is expected to save up to 500 million hours of computation required for the proposed statewide CyberShake 3.0 model, in addition to reducing dramatically the time-to-solution.

In the near future, we will refine our co-scheduling strategy for the CyberShake calculations to allow full utilization of both CPUs and GPUs on heterogeneous computational systems such as *Blue Waters* and *Titan*. Factor-of-three reductions in time-to-solution are anticipated, which will enable on-demand hazard curve calculations. We also plan to facilitate co-scheduling of in-situ volume data analysis. We will continue optimization of the GPU code on Kepler, develop resilience features, and implement ADIOS for solver check-pointing. Finally, we are in the process of adding more physics to AWP-SGT simulations, incorporating more realistic media, different realizations of fault roughness, plasticity, and other features, which will greatly advance our objectives to improve the accuracy of seismic hazard analysis.

The GPU-based AWP-SGT code will provide highly scalable solutions for other problems of interest to SCEC as well as the wider scientific community, including full-3D waveform inversions to obtain better velocity models for use in structural studies of the Earth across a range of geographic scales.

8. ACKNOWLEDGMENTS

We thank Jeffrey Vetter, Matthew R. Norman and Bob Fiedler of ORNL, Carl Ponder and Roy Kim of NVIDIA, Mitchel D. Horton of Georgia Tech, Sreeram Potluri and DK Panda of OSU, Didem Unat and Scott Baden of UCSD, for their major contributions of technical support; Jack Wells and Judith C. Hills of ORNL, Jay Alameda and Gregory H. Bauer of NCSA, Bruce Loftis of NICS for their resource support. 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 Titan, 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. NSF XSEDE Compute resources used for this research are supported by XSEDE under NSF grant number OCI-1053575. Blue Waters computing resources was provided by NCSA. Research funding was provided through XSEDE's Extended Collaborative Support Service (ECSS) program, UCSD Graduate Program, Petascale Research in Earthquake System Science on Blue Waters PRAC (Petascale Computing Resource Allocation) under NSF award number OCI-0832698, SCEC's core program of 2012, NSF Geoinformatics award: Community Computational Platforms for Developing Three-Dimensional Models of Earth Structure (EAR-1226343), and NSF Software Environment for Integrated Seismic Modeling (OCI-1148493). This research was supported by the SCEC funded by NSF Cooperative Agreement EAR-0529922 and USGS Cooperative Agreement 07HQAG0008. The SCEC contribution number for this paper is 1753.

9. REFERENCES

[1] National Research Council, "National earthquake resilience: research, implementation, and outreach," *National Academies Press*, p. 198, 2011.

[2] D. Komatitsch, S. Tsuboi, C. Ji, and J. Tromp, "A 14.6 billion degrees of freedom, 5 teraflops, 2.5 terabyte earthquake simulation on the Earth simulator," in *Proceedings of the 2003 ACM/IEEE conference on Supercomputing (SC)*, 2003, p. 4.

[3] T. Bui-Thanh, C. Bursteddey, O. Ghattas, J. Martin, G. Stadler, and L. C. Wilcox, "Extreme-scale UQ for bayesian inverse problems governed by PDEs," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, Salt Lake City, Utah, 2012.

[4] T. Furumura. (2013, April) Japan Agency for Marine-Earth Science and Technology. [Online]. http://www.jamstec.go.jp/hpci-sp/strategy/pamphlet_en.pdf

[5] K. B. Olsen, S. M. Day, J. B. Minster, Y. Cui, A. Chourasia, M. Faerman, R. Moore, P. Maechling, and T. H. Jordan, "Strong shaking in Los Angeles expected from southern San Andreas earthquake," *Geophysical Research Letters*, vol. 33, no. 7, April 2006.

[6] 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," *Bulletin of the Seismological Society of America*, vol. 98, no. 3, pp. 1162-1185, June 2008.

[7] R. Graves, B. Aagaard, K. Hudnut, L. Star, J. Stewart, and T. H. Jordan, "Broadband simulations for Mw 7.8 southern San Andreas earthquakes: ground motion sensitivity to rupture speed," *Geophysical Research Letters*, vol. 35, no. 22, November 2008.

[8] J. Bielak, R. Graves, K. B. Olsen, R. Taborda, L. Ramirez-Guzman, S. Day, G. Ely, D. Roten, T. Jordan, P. Maechling, J. Urbanic, Y. Cui, and G. Juve, "The ShakeOut earthquake scenario: Verification of three simulation sets," *Geophysical Journal International*, vol. 180, no. 1, pp. 375-404, January 2010.

[9] K. Porter, K. Hudnut, S. Perry, M. Reichle, C. Scawthorn, and A. Wein, "Forward to the special issue on ShakeOut," *Earthquake Spectra*, vol. 27, no. 2, pp. 235-237, 2011.

[10] Southern California Earthquake Center. (2013, April) ShakeOut. [Online]. shakeout.org

[11] Y. Cui, K. B. Olsen, T. H. Jordan, K. Lee, J. Zhou, P. Small, D. Roten, G. Ely, D. K. Panda, A. Chourasia, J. Levesque, S. M. Day, and P. Maechling, "Scalable earthquake simulation on petascale supercomputers," in *Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, New Orleans, 2010, pp. 1-20.

[12] R. Taborda and J. Bielak, "Ground-motion simulation and validation of the 2008 Chino Hills," *Bulletin of the Seismological Society of America*, vol. 103, pp. 131-156, 2013.

[13] 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)," *Bulletin of the Seismological Society of America*, vol. 99, no. 4, pp. 2053-2107, August 2009.

[14] R. Graves, T. H. Jordan, S. Callaghan, E. Deelman, E. Field, G. Juve, C. Kesselman, P. Maechling, G. Mehta, K. Milner, D. Okaya, P. Small, and K. Vahi, "CyberShake: A physics-based seismic hazard model for southern california," *Pure and Applied Geophysics*, vol. 168, no. 3, pp. 367-381, March 2011.

[15] S. Callaghan, E. Deelman, D. Gunter, G. Juve, P. Maechling, C. Brooks, K. Vahi, K. Milner, R. Graves, E. Field, D. Okaya, and T. Jordan, "Scaling up workflow-based applications," *Journal of Computer and System Sciences*, vol. 76, no. 6, pp. 428–446, September 2010.

[16] P. Chen, L. Zhao, and T. H. Jordan, "Full 3D tomography for the crustal structure of the Los Angeles region," *Bulletin of the Seismological Society of America*, vol. 97, no. 4, pp. 1094-1120, 2007.

[17] C. Tape, Q. Liu, A. Maggi, and J. Tromp, "Seismic tomography of the southern California crust based on spectral-element and adjoint methods," *Geophysical Journal International*, vol. 180, no. 1, pp. 433-462, 2010.

[18] P. Chen, T. H. Jordan, and L. Zhao, "Full three-dimensional tomography: a comparison between the scattering- integral and adjoint-wavefield methods," *Geophysical Journal International*, vol. 170, no. 1, pp. 175-181, 2007.

[19] J. C. Phillips, J. E. Stone, and K. Schulten, "Adapting a message-driven parallel application to GPU-accelerated clusters," in *International Conference for High Performance Computing, Networking, Storage and Analysis, (SC)*, 2008, pp. 1-9.

[20] E. H. Phillips, Y. Zhang, R. L. Davis, and J. D. Owens, "Rapid aerodynamic performance prediction on a cluster of graphics processing units," in *Proceedings of the 47th AIAA Aerospace Sciences Meeting*, vol. 565, 2009.

[21] W. M. Brown, T. D. Nguyen, M. A. Fuentes-Cabrera, J. D. Fowlkes, P. D. Rack, and M. Berger, "An evaluation of molecular dynamics performance on the hybrid Cray XK6 supercomputer," in *International Conference on Computational Science, ICCS 2012*, Omaha, NE, 2012.

[22] M. A. Clark, P. C. La Plante, and L. J. Greenhill, "Accelerating radio astronomy cross-correlation with graphics processing units," *Submitted to the International Journal of High Performance Computing Applications (IJHPCA), preprint arXiv:1107.4264*, 2011.

[23] S. Li, R. Chang, A. Boag, and V. Lomakin, "Fast electromagnetic integral-equation solvers on graphics processing units," *Antennas and Propagation Magazine, IEEE*, vol. 54, no. 5, pp. 71-87, 2012.

[24] M. Eisenbach, "Future Proofing WL-LSMS: Preparing for First Principles Thermodynamics Calculations on Accelerator and Multicore Architectures," 2011.

[25] T. Shimokawabe, T. Aoki, T. Takaki, A. Yamanaka, A. Nukada, T. Endo, N. Maruyama, and S. Matsuoka, "Peta-scale phase-field simulation for dendritic solidification on the TSUBAME 2.0 supercomputer," in *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, 2011, pp. 1-11.

[26] T. Furumura and L. Chen, "Large scale parallel simulation and visualization of 3D seismic wavefield using the Earth simulator," *Computer Modeling in Engineering and Sciences*, vol. 6, pp. 153-168, 2004.

[27] P. Moczo, J. Kristek, M. Galis, P. Pazak, and M. Balazovjech, "The finite-difference and finite-element modeling of seismic wave propagation and earthquake motion," *Acta Physica Slovaca. Reviews and Tutorials*, vol. 57, no. 2, pp. 177-406, 2007.

[28] H. Aochi, T. Ulrich, A. Ducellier, F. Dupros, and D. Michea, "Finite difference simulations of seismic wave propagation for understanding earthquake physics and predicting ground motions: Advances and challenges," in *Proceedings of Computational Physics*, 2012.

[29] K. Datta, M. Murphy, V. Volkov, S. Williams, J. Carter, L. Oliker, D. Patterson, J. Shalf, and K. Yelick, "Stencil computation optimization and auto-tuning on state-of-the-art multicore architectures," in *Proceedings of the 2008 ACM/IEEE conference on Supercomputing*, 2008, p. 4.

[30] P. Micikevicius, "3D finite difference computation on GPUs using CUDA," in *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units*, 2009, pp. 79-84.

[31] D. Michea and D. Komatitsch, "Accelerating a three-dimensional finite-difference wave propagation code using GPU graphics cards," *Geophysical Journal International*, vol. 182, no. 1, pp. 389-402, 2010.

[32] R. Abdelkhalek, H. Calandra, O. Coulaud, J. Roman, and G. Latu, "Fast seismic modeling and reverse time migration on a GPU cluster," in *IEEE International Conference on High Performance Computing & Simulation, HPCS'09*, 2009, pp. 36-43.

[33] T. Okamoto, H. Takenaka, T. Nakamura, and T. Aoki, "Accelerating large-scale simulation of seismic wave propagation by multi-GPUs and three-dimensional domain decomposition," *Earth, planets and space*, vol. 62, no. 12, pp. 939-942, 2010.

[34] S. Song, T. Dong, Y. Zhou, D. A. Yuen, and Z. Lu, "Sesmic wave propagation simulation using support operator method on multi-GPU system," University of Minnesota, Technical Report 2010.

[35] D. Komatitsch, D. Goddeke, G. Erlebacher, and D. Michea, "Modeling the propagation of elastic waves using spectral elements on a cluster of 192 GPUs," *Computer Science-Research and Development*, vol. 25, no. 1-2, pp. 75-82, 2010.

[36] M. Rietmann, P. Messmer, T. Nissen-Meyer, D. Peter, P. Basini, D. Komatitsch, O. Schenk, J. Tromp, L. Boschi, and D. Giardini, "Forward and adjoint simulations of seismic wave propagation on emerging large-scale GPU architectures," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis (SC)*, Salt Lake City, Utah, 2012, p. 38.

[37] K. B. Olsen, "Simulation of three-dimensional wave propagation in the Salt Lake basin," University of Utah, Doctoral dissertation 1994.

[38] C. Cerjan, D. Kosloff, R. Kosloff, and M. Reshef, "A nonreflecting boundary condition for discrete acoustic and elastic wave equations," *Geophysics*, vol. 50, no. 4, pp. 705-708, 1985.

[39] R. W. Graves, "Simulating seismic wave propagation in 3D elastic media using staggered-grid finite differences," *Bulletin of the Seismological Society of America*, vol. 86, no. 4, pp. 1091-1106, 1996.

[40] J. O. Blanch, J. O. Robertsson, and W. W. Symes, "Modeling of a constant Q: methodology and algorithm for an efficient and optimally inexpensive viscoelastic technique," *Geophysics*, vol. 60, no. 1, pp. 176-184, 1995.

[41] S. M. Day, "Efficient simulation of constant Q using coarse-grained memory variables," *Bulletin of the Seismological Society of America*, vol. 88, no. 4, pp. 1051-1062, 1998.

[42] S. M. Day and C. R. Bradley, "Memory-efficient simulation of anelastic wave propagation," *Bulletin of the Seismological Society of America*, vol. 91, no. 3, pp. 520-531, 2001.

[43] L. Zhao, P. Chen, and T. H. Jordan, "Strain Green's tensors, reciprocity, and their applications to seismic source and structure studies," *Bulletin of the Seismological Society of America*, vol. 96, no. 5, pp. 1753-1763, 2006.

[44] D.J. Wald and R.W. Graves, "Resolution analysis of finite fault source inversion using one-and three-dimensional Green's functions: 2. Combining seismic and geodetic data," *Journal of Geophysical Research*, vol. 106, 2001.

[45] J. Zhou, D. Unat, D. Choi, C. Guest, and Y. Cui, "Hands-on performance tuning of 3D finite difference earthquake simulation on GPU fermi chipset," in *Proceedings of International Conference on Computational Science (ICCS)*, vol. 9, Omaha, Nebraska, 2012, pp. 976-985.

[46] J. Zhou, Y. Cui, E. Poyraz, D. Choi, and C. , ICCS 2013, Barcelona, June 5-7, 2013 (in press) Guest, "Multi-GPU implementation of a 3D finite difference time domain earthquake code on heterogeneous supercomputers," in *Accepted to International Conference on Computational Science (ICCS)*, 2013.

[47] L. A. Dalguer and S. M. Day, "Staggered-grid split-node method for spontaneous rupture simulation," *Journal of Geophysical Research: Solid Earth (1978–2012)*, vol. 112, no. B2, 2007.

[48] Oak Ridge Leadership Computing Facility. (2013, April) Titan User Guide. [Online]. https://www.olcf.ornl.gov/support/system-user-guides/titan-user-guide/,.

[49] University of Illinois NCSA. (2013, April) Blue Waters System Overview. [Online].

https://bluewaters.ncsa.illinois.edu/user-guide

[50] XSEDE. Georgia Tech Keeneland User Guide. [Online]. https://www.xsede.org/gatech-keeneland

[51] Z. Shi and S. M. Day, "Rupture dynamics and ground motion from 3-D rough-fault simulations," *Journal of Geophysical research*, vol. 118, pp. 1-20, 2013.

[52] E. H. Field, T. H. Jordan, and C. A. Cornell, "OpenSHA: A developing community-modeling environment for seismic hazard analysis," *Seismological Research Letters*, vol. 74, no. 4, pp. 406-419, 2003.

[53] R. W. Graves and A. Pitarka, "Broadband ground-motion simulation using a hybrid approach," *Bulletin of the Seismological Society of America*, vol. 100, no. 5A, pp. 2095-2123, 2010.

[54] P. M. Mai, W. Imperatori, and K. B. Olsen, "Hybrid broadband ground motion simulations: combining long-period deterministic synthetics with high frequency multiple S-to-S back-scattering," *Bulletin of the Seismological Society of America*, vol. 100, no. 5A, pp. 2124-2142, 2010.

[55] J. Schmedes, R. J. Archuleta, and D. Lavallée, "Correlation of earthquake source parameters inferred from dynamic rupture simulations," *Journal of Geophysical Research: Solid Earth (1978–2012)*, vol. 115, no. B3, 2010.

[56] J. Zhou, U. Didem, D. Choi, C. Guest, and Y. Cui, "Hands-on performance tuning of 3D finite difference earthquake simulation on GPU fermi chipset," in *Proceedings of International Conference on Computational Science (ICCS)*, vol. 9, Omaha, Nebraska, 2012, pp. 976-985.