Contents

11

SCEC CyberShake Workflows - Automating Probabilistic Seismic Hazard Analysis Calculations

Philip Maechling[1], Ewa Deelman[2], Li Zhao[1], Robert Graves[3], Gaurang Mehta[2], Nitin Gupta[1], John Mehringer[1], Carl Kesselman[2], Scott Callaghan[1], David Okaya[1], Hunter Francoeur[1], Vipin Gupta[1], Karan Vahi[2], Thomas Jordan[1], and Edward Field[4]

[1] Southern California Earthquake Center, University of Southern California, Los Angeles CA, 90089
maechlin,zhaol,niting,jmehring,scottcal,okaya,francoeu,vgupta,tjordan@usc.edu
[2] Information Sciences Institute, University of Southern California, Marina Del Rey, CA 90292
deelman,gmehta,carl,vahi@isi.edu
[3] URS Corporation, Pasadena, CA 91101
robert_graves@urscorp.com
[4] US Geological Survey, Pasadena, CA 91106
field@caltech.edu

Keywords: probabilistic seismic hazard analysis, seismology, ground motion, earthquake, grid computing, high throughput computing, provisioning, Condor, Virtual Data System, Pegasus.

11.1 Introduction to SCEC CyberShake Workflows

The Southern California Earthquake Center (SCEC) is a community of more than 400 scientists from over 54 research organizations that conducts geophysical research in order to develop a physics-based understanding of earthquake processes and to reduce the hazard from earthquakes in the southern California region [1].

SCEC researchers are developing physics-based models of earthquake processes and integrating these models into a scientific framework for seismic hazard analysis and risk management. This research requires both structural geological models such as fault models and three dimensional earth density models, as well as a variety of earthquake simulations programs, such as earthquake wave propagation simulation codes and dynamic fault ruptures simulation codes. The goal of this model-oriented approach to earthquake science is

to transform seismology into a predictive science with forecasting capabilities similar to climate modeling and weather forecasting.

SCEC research has several common characteristics. The science is collaborative; a wide variety of organizations and disciplines work together. The science is integrative; techniques and approaches from different disciplines are combined in new ways. The science is physics-based; the scientists are continuously trying to incorporate more physics into their models and to ensure that their simulations are consistent with physical laws. The science is model-driven; theoretical results are incorporated into predictive computational models. The science is validated; predictive model results are compared to observation and to each other for validation.

The output data for many SCEC earthquake simulations are predicted ground motions for a specific earthquake. For example, a researcher can model a "scenario" earthquake on the San Andreas Fault and predict the ground motions that will be produced in Los Angeles if that earthquake actually occurs. While ground motion predictions for a particular earthquake are of significant interest, they are not a solid basis for understanding the earthquake hazards in an area.

To characterize the earthquake hazards in a region, seismologists and engineers utilize a technique called Probabilistic Seismic Hazard Analysis (PSHA). PSHA attempts to quantify the peak ground motions from **all possible earthquakes** that might affect a particular site and to establish the probabilities that the site will experience a given ground motion level over a particular time frame. An example of a PSHA hazard curve at a specific site in Los Angeles is shown in Figure 11.1. Because Los Angeles has widely varying geological regions (mountains, deserts, and sedimentary basins), hazards curves for sites fairly close together can differ significantly. PSHA information is used by city planners and building engineers to estimate seismic hazards prior to construction of significant buildings and PSHA results are often the basis for building codes in a region.

A probabilistic seismic hazard curve describes the seismic hazards at a particular site. Probabilistic seismic hazard curves can be combined into probabilistic seismic hazard maps [**?**]. To construct a hazard map, one of the two variables used in the curve (either (1) the ground motion level, or (2) the probability of exceedence) is fixed, and then color variations indicate how the other parameter varies by location on the map. A set of hazard curves, typically from a set of regularly spaced sites, can be combined into a hazard map by interpolating the site specific data values and plotting the resulting contours. In the United States, the United States Geological Survey (USGS), as well as several state agencies, publish hazard maps. An example PSHA map, produced by the U.S.G.S, and the California Geological Survey (CGS) is shown in Figure 11.2. This map fixes the probability of exceedence at 10% in 50 years and the color variations indicate predicted levels of peak accelerations with the darker colored regions predicted to experience stronger ground motions than the lighter colored regions.

Because of the significant role PSHA information has in public safety, improvements in PSHA techniques are of great interest to seismologists, public safety officials, building engineers, and emergency management groups. PSHA researchers recognize that current PSHA techniques have not fully integrated recent advances in earthquake simulations capabilities. As a result, researchers working on the SCEC Community Modeling Environment Project (SCEC/CME) [**?**] [**?**] recently initiated the CyberShake Project to develop new techniques for calculating PSHA seismic hazard curves. The goal of the CyberShake Project is to utilize earthquake wave propagation simulations to produce the ground motions estimates used in PSHA hazard curves. The geoscientists and computer scientists working on CyberShake have successfully calculated probabilistic seismic hazard curves for several sites in the Los Angeles area using peak ground motions values produced by earthquake wave propagation simulations. This new class of PSHA hazard curves has the potential to transform probabilistic seismic hazard analysis because the earthquake wave propagation simulations used to produce these new curves produce more physically realistic peak ground motion values than the techniques used to calculate peak ground motions used in earlier hazard curve calculations.

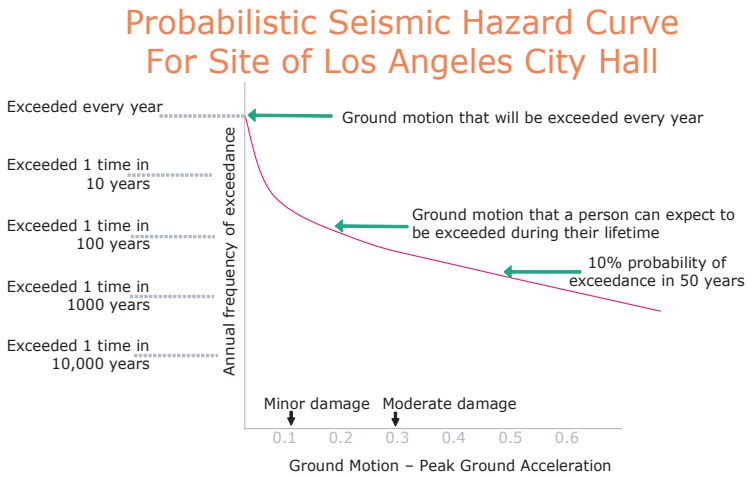

Fig. 11.1. Probabilistic Seismic Hazard Curve for the site of Los Angeles City Hall. This curve predicts that this site will experience Peak Ground Accelaration of 0.5G about every 500 years.

We refer to all the steps in the CyberShake hazard curve calculation process (including preparation, simulation, post-processing, and analysis) as the CyberShake computational pathway. The CyberShake computational pathway can be divided into two main computational phases; 1) a high performance, MPI-based, finite difference earthquake wave propagation simulation phase, and 2) a post-processing phase in which thousands of serial data analysis jobs must be executed.

We model the CyberShake computational pathway as a scientific workflow to be executed within the SCEC grid-based computing environment. The SCEC scientific workflows utilize a software stack based on Pegasus and the Virtual Data Toolkit (VDT). VDT is a modular suite of software that enables execution of workflows in a grid-based environment.

We anticipated several benefits by implementing our computations as scientific workflows. The VDT grid-based workflow-based computing model offers the possibility of developing one workflow that will execute in the distributed, highly heterogeneous SCEC computing environment. Also, the VDT tools provide automated job submission and job tracking. In addition, the VDT tools offer data management tools that include support for data replication and metadata management.

As we implemented the CyberShake computational pathway as a scientific workflow, we recognized that our workflows tools provided some additional, somewhat un-anticipated, benefits. For example, we found that the Virtual Data Language (VDL) (Chapter **??**) allowed us to express our workflows at a high level of abstraction. Our workflows can be expressed without referring to particular computer resources or particular file copy. This workflow virtualization is performed by the Pegasus meta-schedule which automatically converts our abstract workflows into concrete executable workflows by selecting particular computer resources to be used, and by adding implied, but unspecified, actions such as directory creation and file transfers.

We also uncovered some of the limitiations of these workflow tools. For example, we found that only portions of our CyberShake computational pathway benefited from modeling as scientific workflows. Our original goal of configuring the entire CyberShake processing sequence to run as a workflow proved to be impractical.

In the following sections, we describe the CyberShake computational pathway and our efforts to convert this conceptual sequential processing into an executable scientific workflow. We outline issues related to the modeling of computations as workflows and describe where we gained significant benefits from workflow technology.

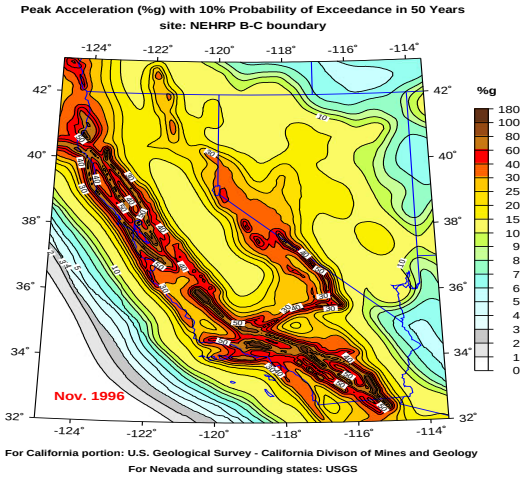

Fig. 11.2. This USGS and CGS PSHA map for California and Nevada is based on a large number of PSHA hazard curves. This map fixes the Probability of Exceedance at 10% in 50 years and uses color variations to indicate expected peak ground motion levels throughout the mapped region.

11.2 SCEC Probabilistic Seismic Hazard Analysis Research

Prior to the start of the CyberShake Project, SCEC researchers outlined a conceptual framework for improving probabilistic seismic hazard analysis. This conceptual framework describes a series of what are termed computational pathways. They represent a series of four increasingly more accurate, but also more complex approaches to calculating the anticipated intensity of ground motions at a particular site. The computational pathways are shown in Figure 11.3.

The SCEC Computational Pathways can be described as follows: Pathway 1 is standard probabilistic seismic hazard analysis that uses empirical attenuation relationships. Pathway 2 replaces attenuation relationships with anelastic waveform modeling (AWM) and site response models. Pathway 3 introduces physics-based dynamic rupture models to describe the earthquakes that produce the ground motions. Pathway 4 represents a series of inverse problems in which simulated data is used to constrain and improve the geophysical models used in the simulations.

The Pathway 1 group has developed a component-based software suite call OpenSHA [**?**] that implements standard PSHA models, such as Earthquake Rupture Forecasts (ERFs) and Intensity Measure Relationships (IMRs),

within a common framework. OpenSHA is a stable and robust suite of software that allows researchers to combine PSHA components in ways never before possible.

The Pathway 2 group has developed and validated techniques for running large, regional-scale, earthquake wave propagation simulations. These simulations, validated for low frequencies against observed earthquakes, demonstrate important effects such as earthquake directivity and amplification of ground motions over sedimentary basins.

One reason for the effectiveness of the SCEC computational pathway conceptual framework is that each subsequent pathway builds on elements of previous pathways. The higher level pathways leverage the work done by other group on lower numbered pathways. The SCEC CyberShake simulations are an integration of our Pathway 2 wave propagation simulations with our Pathway 1 Probabilistic Seismic Hazard Analysis techniques. This integration has not been without challenges. This integration of pathways requires levels of computational, data management, and data analysis that exceed any previous SCEC computational models.

11.3 Computational Requirements of SCEC Workflows

The computational challenges of CyberShake begin with the number of earthquakes that must be simulated. We would like to represent **all possible** earthquakes that might occur, but for practical reasons, we must settle for representing **all somewhat probable** earthquakes that might occur within 200KM of the site being considered. Geophysical models, implemented as computer programs, that can provide a list of somewhat probable future earthquakes are called Earthquake Rupture Forecasts (ERFs). For sites near Los Angeles, current ERFs produce a list of over 20,000 earthquakes within 200km. Applying an attenuation relationship to 20,000 earthquakes is a fairly modest computational challenge within the capabilities of a desktop computer. However, running the state-of-the-art wave propagation simulations for 20,000 earthquakes is prohibitively expensive in CPU-Hours and wall-clock time. The exact computational time required to run an earthquake wave propagation simulation varies by the size of the volume, the length of time the wave propagation is simulated, and the frequencies supported by the simulation. Earthquake simulations of approximately the required size and resolution, such as SCEC's Pathway 2 TeraShake simulation require approximately 15,000 CPU-Hours and approximately 3 days of wall-clock time. So, for the 20,000 ERF ruptures, it would require 300 Million CPU-Hours and well over 100 years to complete all the simulations needed to calculate a PSHA hazard curve.

While these processing requirements are well beyond the scale of the computer resources available to SCEC, we have not yet represented the full scale

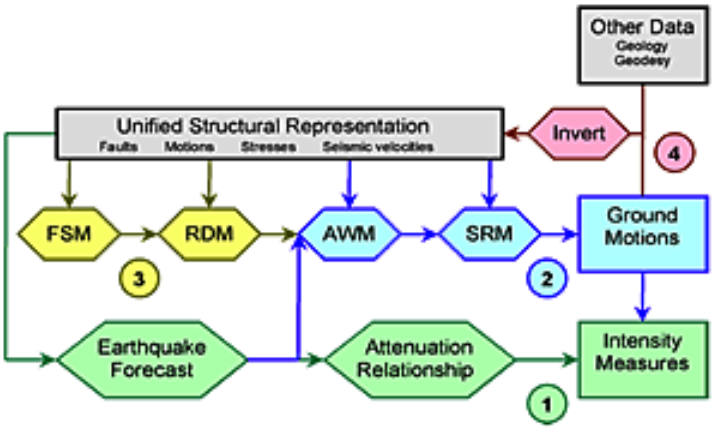

Fig. 11.3. SCEC/CME computational pathways represent a series of increasingly more accurate ways of performing probabilistic seismic hazard analysis.

of the problem. These numbers underestimate the required calculation because the ERF list of 20,000 earthquakes does not represent the full list of earthquakes that must be simulated. The ERF indicates only the fault surface and the magnitude of the earthquakes that are likely to occur. This is sufficient when using an attenuation relationships. However, when using waveform modeling, one must consider how the earthquake rupture occurs. For example, if the earthquake rupture starts at the bottom of the fault and propagates upward towards the surface, the ground motions at the surface will be larger than if the earthquake starts near the surface and propagates downward into the ground. For a given fault, there are many ways that earthquakes can occur. Each possible, or somewhat likely, earthquake variation must be simulated in order to properly perform the PSHA analysis.

To capture the possible differences between earthquakes in the PSHA analysis, one or more variations of each earthquake mentioned in the ERF must be simulated. For small earthquakes (e.g. Magnitude 5.0 or smaller), typically only one variation will be considered. But for large faults, there are many ways the fault may rupture and a reasonable number of these ruptures scenarios must be simulated in the calculation. There is no widely accepted approach for identifying all reasonable rupture variations; however, some basic heuristics have been developed for creating a reasonable number of rupture variations. When these heuristic are applied to the ERF list for Los Angeles area sites, the total number of earthquakes that must be simulated to create a probabilistic seismic hazard curve is over 100,000 earthquake simulations. At 15,000 CPU-Hours per simulation, a fully probabilistic hazard curve calculation would require approximately 1,500,000,000 CPU-Hours. The computation time is not the only challenge. There are also significant data management issues. Each rupture variation will produce two seismograms (horizontal com-

ponents only), which, depending on the data storage format, may results in a large number of seismogram files. These seismogram files and their associated metadata must be managed to support the analysis of the results.

In order to implement the CyberShake 3D waveform-based IMR, all of these scientific questions, computational scale, and the data management challenges had to be addressed by the CyberShake development group.

11.4 The SCEC Hardware and Software Computational Environment

CyberShake was implemented within the distributed computational environment that was designed and implemented as a part of SCEC/CME Project [**?**]. The SCEC/CME computational system uses a grid-based architecture that allows us to share heterogeneous computing resources with other collaborating computing organizations in a consistent and secure manner. The SCEC/CME execution environment is composed of the NSF TeraGrid [**?**], the University of Southern California (USC) High Performance Computing and Communications center (USC HPCC)[**?**], a large academic Linux cluster with approximately 1,800 processors, and the local SCEC computing resources that include a variety of Linux and Solaris servers. Significant disk storage, in excess of 10 TB, is available at all sites, including SCEC's local cluster.

The SCEC, USC HPCC, and TeraGrid sites are linked into an extensible grid-based computing environment through the NSF National Middleware Initiative software stack [**?**]. Grid security is managed using Grid Security Infrastructure (GSI). Certificate policy was negotiated between the three organizations, SCEC, USC, and TeraGrid allowing acceptance of each others host and user grid-security certificates.

In addition to the grid layer, the SCEC/CME computational system has implemented a workflow software layer based on the Virtual Data Toolkit (VDT) [16] . The Virtual Data Toolkit, in turn, includes the Virtual Data System (VDS) which includes Chimera [47] and Pegasus (Chapter **??**). VDT also includes data management tools such as the Replica Location Service (RLS) [34]. An overview of the grid-based hardware and software used in the CyberShake calculations are shown in Figure 11.4.

11.5 Introduction of a Reciprocity-based Site Calculation

The key to reducing the computational demands for CyberShake PSHA hazard curves was the introduction of a non-intuitive scientific technique for calculating synthetic seismograms called reciprocity.

Typically, synthetic seismograms are created through what are termed "forward calculations". Motions are introduced in a volume at the point of the earthquake and the resulting waves are propagated throughout the volume. An

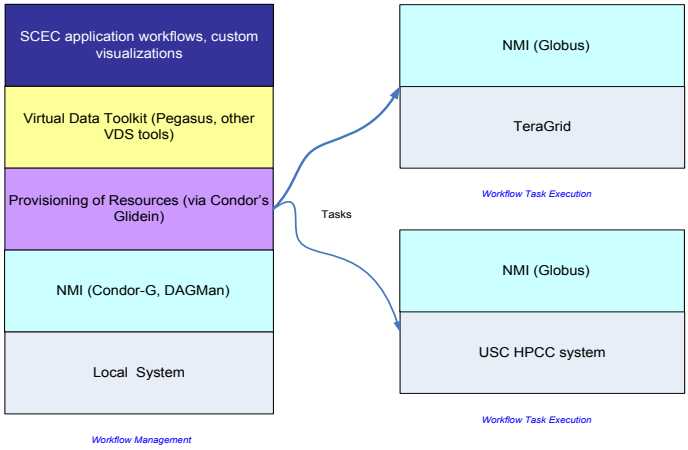

Fig. 11.4. SCEC/CME workflow system software stack, based on the Virtual Data Toolkit, provides SCEC workflows with secure access to a distributed, heterogenous, grid-based computing environment.

alternative method for calculating synthetic seismograms, called reciprocity, can be used. A reciprocity-based approach places a unit-force at the site of interest. Then the waves from this force are propagated throughout the volume to "illuminate the volume". The response of the volume to the unit force is saved as Strain Green Tensors (SGT). Given SGT data for a volume, it is very computationally inexpensive to calculate a synthetic seismogram for an earthquake located anywhere within the volume using a technique called seismogram synthesis.

Using a reciprocity-based approach, the computational estimates for calculating a probabilistic seismic hazard curve for a single site is approximately 25,000 CPU-Hours. This includes the two unit-force simulations, and the reciprocity-based seismogram synthesis for 100,000 earthquakes. This reciprocity-based technique brings the computational cost of within reach of SCEC computing resources.

There is, as might be expected, a tradeoff involved in using this reciprocity-based approach. These calculations only produce seismograms for the one site and, consequently, only one hazard curve. Since each hazard curve requires approximately 25,000 CPU-Hours, producing a small 50km x 50km hazard map that requires 625 hazard curves requires approximately 15,625,000 CPU-Hours using this approach.

These estimates indicate that even using a reciprocity-based approach, it is still prohibitively computationally expensive to produce a PSHA hazard map. However, it is now possible to calculate a small number of full probabilistic hazard curves with this technique. The CyberShake group felt that the calculation of waveform-based PSHA hazard curves for a number of sites

provides an excellent basis for an initial analysis of waveform-based PSHA hazard curves. If the CyberShake research demonstrates that waveform-based PSHA hazard curves are significantly more accurate that traditional PSHA hazard curves, then there will be a strong justification to obtain the necessary computational resources to calculate full PSHA maps using this new approach.

11.6 Outlining the CyberShake Computational Elements

A CyberShake hazard curve calculation can be described algorithmically in seven steps. Each processing step has specific computational and workflow implications.

Processing Step Number	**CyberShake Simulation Algorithm Description**
1	Select a site for which a hazard curve is of interest.
2	Use an earthquake rupture forecast (ERF) to identify all probable ruptures within 200KM of the site of interest.
3	For each rupture, convert the rupture description from the ERF into a suite of rupture variations with slip-time history.
4	Calculate Strain Green Tensors (SGT) for the two horizontal components for a volume containing all the ruptures and save the volume data.
5	Using a reciprocity-based approach, calculate synthetic seismograms for each rupture variation.
6	Calculate the peak intensity measure of interest such as peak spectral acceleration for each synthetic seismogram.
7	Using the peak intensity measures for each rupture, and the probabilities of the rupture, calculate a probabilistic hazard curve.

CyberShake Step 1: Select a Site

Probabilistic seismic hazard curves are site specific, so a natural and meaningful unit of work on the CyberShake project is a **site**. We perform a series of calculations and at the end we can calculate one or more hazard curves for one particular site. We typically measure the computational and storage information by the amounts required for a single site. These numbers can then be multiplied by the number of sites required.

Sites selected for our initial CyberShake hazard curve calculations must be in a region for which both a 3D velocity model and an earthquake rupture forecast have been defined. These items are available for most parts of southern

California. Also, to facilitate comparison to others types of IMRs, we selected sites for which hazard curves currently exist. The selection of sites is currently manual. No software is used at this step in the processing.

CyberShake Step 2: Identify Probable Ruptures

Given a particular site, an Earthquake Rupture Forecast is used to identify all probable ruptures within 200km of the site. The magnitude and probability of each identified rupture must also be specified. The table below shows six of the initial CyberShake sites and the number of ruptures that the ERF identifies within 200km of each site.

Site Name	**Number of Ruptures in ERF within 200KM of Site**
USC	24,421
Pasadena	24,870
Downtown Los Angeles	24,620
Port of Long Beach	24,484
Santa Ana Business District	25,363
Whittier Narrows Golf Course	25,056

In this stage in the CyberShake processing, one program, the Earthquake Rupture Forecast generator is used. The Earthquake Rupture Forecast Generator is the first computational step in our scientific workflow.

CyberShake Step 3: Calculate Rupture Variations

Rupture descriptions produced by current ERF's are static descriptions of earthquakes indicating the fault surface and the magnitude of each earthquake. However, earthquake wave propagation simulations require more detailed information about the ruptures, and several variations of each earthquake rupture must be considered. As a general rule, the larger the earthquake in the ERF, the larger the number of rupture variations that will be used in the CyberShake calculation. For each earthquake in the ERF, the CyberShake system will calculate a series of rupture variations using a heuristic-based method developed by SCEC scientists.

The table below shows an example of how the ERF ruptures fan out into a series of rupture variations for site USC. The ERF contains a large number of small earthquakes. These earthquakes do not produce many variations. The ERF produces only a small number of very large earthquakes. However, each of these very large earthquakes produces a large number of variations. The result is that the CyberShake processing must produce seismograms for over 100,000 ruptures. Other sites have similar distribution of ruptures by magnitude.

CyberShake Step 4: Calculate Strain Green Tensors

Site U.S.C	**Ruptures By Magnitude**	**Rupture Variations By Magnitude**
Magnitude < 5.0	0	0
Magnitude ≥ 5 and < 6.0	20,450	64320
Magnitude ≥ 6 and < 7.0	2524	14600
Magnitude ≥ 7.0 and < 8.0	1435	47066
Magnitude ≥ 8	12	12864
Totals	**24421**	**109806**

The next step in the CyberShake computational pathway is to calculate strain Green tensors for the site of interest. A strain tensor quantifies the strain of an object, in this case the earth, undergoing a 3D deformation which is, in this case, caused by an earthquake. For small deformations, the strain tensor can be described by a Strain Green Tensor (SGT). SGT calculations are the high performance (parallel), computing aspect of the CyberShake simulation. Our current SGT code is a fourth order, finite difference code. One SGT calculation is run for each horizontal component of motion. So for CyberShake, two SGT simulations are run per site. Before the MPI code can be run, an input velocity mesh must be generated. This is done in two steps. First a regular mesh with the appropriate dimensions and grid spacing is created. Then a 3D velocity model program is run to assign properties such as P-wave velocity, S-wave velocity, density, and attenuation to each mesh point. The velocity mesh properties vary by location in the mesh. For example, the P-wave velocity, S-wave velocity, and density values all typically increase with depth.

CyberShake Step 5: Synthesize Synthetic Seismograms

The CyberShake reciprocity-based seismogram synthesis processing stage generates thousands, or hundreds of thousands of seismograms for a site. To do this, we must run the seismogram synthesis code twice for each rupture, which amounts to tens of thousands of times. The seismogram synthesis program will generate output files containing the synthetic seismograms. Metadata must be maintained for each file so that we can associate the seismogram with the ruptures that it represents.

CyberShake Step 6: Calculate Peak Intensity Measure

Once one or more seismograms have been calculated, the next step is to extract a peak ground motion from the data. SCEC scientists have decide that spectral acceleration at 3.0 seconds (SA3.0) is a ground motion intensity measure type that is consistent with the frequency content of the seismograms we have generated. To calculate peak SA3.0 values from our seismogram we use codes that can filter the seismograms, differentiate to acceleration, and then calculate peak SA3.0.

CyberShake Step 7: Calculate Hazard Curve

When all the peak SA3.0 values have been calculated, the final step is to calculate a hazard curve. To do this, the peak SA3.0 values for each rupture are read and a geometric average of the horizontal components is calculated. Then, the peak SA3.0 values are associated with the probability of the given rupture. These calculations are done for each rupture, the results are combined, and a hazard curve is calculated.

11.7 SCEC Workflows Solutions to Key Workflow Requirements

Scientific workflow tools may be modeled, in general terms, as a set of tasks with data dependencies between them. Scientific workflow tools must then meet three essential requirements; 1) user definition of the tasks and their data dependencies, 2) an execution engine for running the tasks in an appropriate order, and 3) tools for maintaining data and metadata input and output by the tasks in the workflow.

The SCEC workflow system uses the Virtual Data Toolkit (VDT) [16]. VDT, in turn, relies on the Virtual Data System (VDS) for constructing and executing workflows. VDS in turn relies on the Pegasus Planner and the Condor System (Chapter **??**) for executing the workflows. VDS-based workflows try to avoid specification of particular executables and particular physical files by referring to elements in the workflow in "abstract", "logical", or "virtual" terms. This delayed specification relieves the operator of some specific decisions and provides opportunities for optimization by the workflow system.

The SCEC workflow system satisfies the first essential requirement (user definition of workflow tasks and data dependencies) by expressing the tasks in the workflow and the data dependencies between the tasks using an abstract Directed Acyclic Graph (DAG). The abstract workflow DAG captures the programmatic and data dependencies in the workflow. The abstract workflow DAG describes both the program names and file names in logical, not physical terms. For example, when the workflow refers to a file, it uses a file ID, rather than a physical path to the file. Later programs in the workflow system will convert the file ID to a physical file name.

Our large CyberShake workflows require large abstract DAGS so creation of the abstract DAG is non-trivial. We typically use a Java language program that can generate the abstract workflow DAG directly. In the future, we plan to use technologies such as Wings and CAT (Chapter **??**).

The VDT tools that convert an abstract DAG to a concrete DAG require a collection of appropriate configuration files such as configuration file describing available computer resources (resource pool config) and a list of executable programs (transformation catalog). Once these configuration files and the abstract DAG are available, Pegasus can be invoked to convert the

abstract DAG to a concrete DAG. Once the concrete DAG is available, it can be submitted to the Condor DAGMan (Chapter **??**).

Condor DAGMan functions as a job manager, and it will run the jobs in the DAG in the appropriate order. Pegasus makes a significant number of runtime decisions. For example, by the time the workflow is expressed as a concrete DAG, the specific run-time hosts, the specific file replicas, and implied but unspecified actions such as file creation and data transfers have been incorporated into the concrete DAG by the Pegasus planning tool.

The SCEC workflow system satisfies the second essential workflow requirement (an execution engine for running the tasks) with a series of Globus and Condor tools. Condor-G [49] and Condor DAGman are used as job management tools for the SCEC workflows. Globus GRAM [37] is used as the resource management tool. Condor DAGman manages the job submissions by interacting with other job scheduling software. Condor DAGman works with the Condor-G job management software to ensure that the jobs expressed in the DAG are executed in the correct order.

The SCEC workflow system satisfies the third essential workflow requirement (data and metadata management) by using the Replica Location Service (RLS) [34] software to maintain a mapping between logical and physical file names. Logical File Names (LFN) are basically ID numbers assigned to files used in SCEC workflows. Physical File Names (PFN) used in SCEC workflows are typically GridFTP accessible URL's [19]. Metadata is managed through the use of the Metadata Catalog Service (MCS) [95]. The RLS and MCS system are modular and grid-enabled.

We utilize a second file preservation system, the Storage Resource Broker (SRB) [25], for long term storage of valuable data sets. Files created by SCEC workflows that are deemed of long term value are registered into the SRB for archival purposes.

11.8 Benefits of Modeling CyberShake as Workflows

Implementing the workflow on top of a grid-based architecture provides distributed computing capabilities and the ability to add or remove computing resources from the environment without significant changes to software. The grid layer provides secure management of job submission and data transfers. The grid architecture also provides standardized service interfaces to security, job management, resource monitoring and communication for a heterogeneous environment. This, in theory, allows our workflows to utilize these standardized interfaces which then enables them execute in a heterogeneous computing environment.

The use of Condor DAGman as a job management tools provides significant benefits. The DAGman job manager will analyze the dependencies in a workflow and will run jobs in parallel if there are no dependencies between them. This capability is particularly valuable in a distributed grid-based

environment where there are multiple computing resources available for job execution.

The Condor-G and DAGman job management tools provide significant other capabilities such as job management and failure recovery. Since workflow jobs are run as Condor DAGs, error recovery files called rescue DAGs are created and maintained by the system. Workflow jobs that fail can be automatically retried, or manually re-submitted, and they will continue from the point of the error without re-running the entire workflow.

The SCEC workflow system utilizes the common data management practice of separating the logical file name from the physical file name. This technique helps in two main ways. First, references to the file are not tied to the physical location of the file. When the file is moved, workflow references to the file do not need to be changed. Second, this technique supports copies of files, or replicas. For each file, multiple versions can be maintained and the workflow system has the opportunity to select the most appropriate copy.

Another benefit in this workflow system, contributed by the Pegasus planner, is the ability to express the workflow at a high level of abstraction. When the user expresses the workflow and its dependencies, either using VDL, or in an XML DAG format (DAX), they specify the workflow by referring to logical programs (transformations) and logical files. A significant amount of information can be omitted at the workflow specification stage. For example, the computers that will be used and the location of the files to be used are not needed at the workflow specification stage. These details are provided by the Pegasus Planner as the abstract workflow is converted to a concrete workflow. In addition, Pegasus is smart enough to recognize that the workflow must be elaborated in order execute within an actual physical computing environment. Processing steps such as directory creation, registration of created file into the RLS, and file transfers to and from the program execution hosts are automatically added into the workflow by the Pegasus planner.

11.9 Cost of Using the SCEC Workflow System

While the SCEC workflow offers a number of clear benefits, it also imposes a series of requirements, or costs, on system developers and users. These costs are distinct from the costs of personnel or hardware.

First, establishing and maintaining a widely distributed, grid-based, computing environment requires a significant amount of work involving issues such as security agreements, certificate exchange, software version coordination, installation, operations and maintenance. The grid-based environment provides an outstanding foundation on which to build a workflow system but it also requires significant investment in system and software maintenance.

Next, two different relational databases must be installed, configured, and managed in this system. The Replica Location Service uses a MySql database to maintain the LFN to PFN mapping. The Metadata Catalog System uses a

MySQL database to associate LFN to metadata attribute to metadata value pairs and to provide search capabilities across metadata values. Our experience is that introduction of one or more relational databases into a system require a specialized staff.

The SCEC workflow system also requires a significant amount of configuration before a workflow can be executed. The Pegasus ability to work at a high level of abstraction is implemented by utilizing data stores that map between abstractions and actual computing resources. This means that before a workflow can be executed, a series of data stores must be developed and populated. For example, computing resources available to be used are defined in a pool.config file that defines the computing resource and describes their capabilities. Also, each executable program or script used in a workflow must be defined in a transformation catalog. The transformation catalog contains a definition of the input parameter, output parameters, and the run-time environment required by the program.

Also, all files to be used in workflows must be registered into the RLS and MCS and staged at a URL that is accessible by a GridFTP server. This creates a fairly sharp distinction between files "in the system" and files "not in the system". This puts a burden on users that want to create new files by hand or that want to import files into the system. While the data management tools such as RLS provides interfaces for registering files, it has been necessary for us to write user-oriented tools to help users with this task.

While it may seem obvious, it is worth noting that the SCEC workflow system is designed to execute programs with file-oriented inputs and outputs. Programs that support the standard "Unix" computing model work well within the SCEC workflow system. These programs have common characteristics such as file or piped inputs, quiet execution unless there are problems, zero return on success, and non-zero return on problems. The SCEC workflow system is not designed to execute programs with GUI-interfaces or with interactive user inputs.

The SCEC workflow system imposes specific requirements on the programs that will be used in the workflow. To facilitate the data management tools, programs used in workflows should not use hard-coded input or output file names. The workflow system will dynamically assign LFN to files as they are created. Many of the SCEC programs used hard-coded file names. In some cases, we modified these programs so that both input and output file names could be specified as input parameters at runtime. If this modification was difficult, we developed wrapper scripts that would accept arbitrary input and output file names. The wrapper script would then rename these files to the hard coded file names, call the SCEC programs, then rename the output file to the file name assigned by the workflow system.

One additional requirement for using the SCEC workflow system is the need to create a DAX before the workflow can be run. In order to create a DAX, the user is faced with a couple of options; a) use VDL to describe the workflow then use Chimera to convert the VDL to a DAX, or b) or write code

that can construct a DAX directly. Because the SCEC CyberShake workflows were fairly static, we chose to develop a DAX generator program and output our DAX's directly. The other option, using VDL, may be the more general solution. Both of these approaches require training and investment of time by users. Often users are not willing to invest significant training time until the benefits to their science is obvious.

11.10 From Computational Pathway to Abstract Workflow

We began our modeling of CyberShake as a workflow by assembling our programs end to end and identifying the data dependencies between them. Figure 11.5 shows the programs involved in the CyberShake hazard curve calculation and their data dependencies.

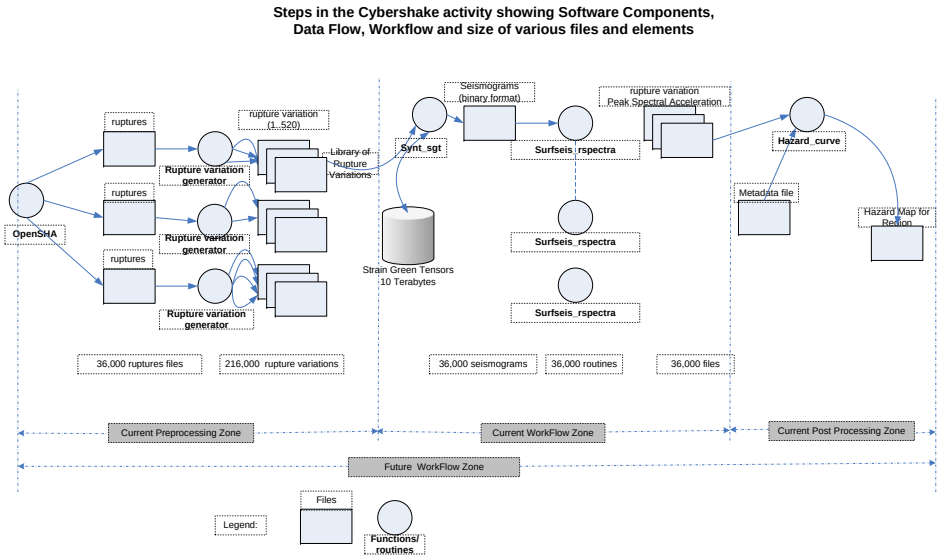

Fig. 11.5. CyberShake computational pathway is an end to end computation of a CyberShake Hazard Curve

Our intention was to model our CyberShake computational pathway as an abstract workflow, then model the abstract workflow as a DAX, and then

use our workflow tools to convert our DAX into an executable workflow and run it until a hazard curve was completed. However, our workflow eventually was reduced to a small portion of this chain of processing. The process of converting our CyberShake computational pathway into an abstract workflow led to a significant reduction in the number of programs in the workflow. As we will describe, programs were excluded from the actual workflow for a variety of reasons.

At the beginning of the CyberShake computational pathway, we found that the Rupture_Forecast_Generator, a GUI-based Java program, required user interactions during execution. The operator uses a series of dropdown menu's and text boxes to enter information about the site being specified such as location site, cutoff distance, and other configurable parameters. Then the program is run once.

We did not want to integrate a GUI-based program into the workflow. We considered re-writing the program into a version without a GUI but the Rupture_Forecast_Generator is a part of an externally maintained software suite called OpenSHA. So we excluded these programs from the CyberShake workflow.

The next three steps we considered were the Mesh_Maker, Velocity_Mesh_Maker, and the Strain_Green_Tensor calculations. These three programs are run to create the large Strain Green Tensor data set. The Strain_Green_Tensor program is an MPI-based finite difference code that requires high performance computing cluster. The SGT calculations used in the CyberShake simulations require approximately 140GB of RAM at run-time. On our target clusters, we can utilize approximately 500MB of RAM per processor. In order run the SGT successfully, we must divide the 140GB across approximately 280 processors, or about 140 nodes on dual processor systems the TeraGrid IA-64 or USC HPCC clusters.

As we developed our CyberShake abstract workflow, we noted that scheduling large MPI-based programs onto a cluster often have interactive aspects that are not easily managed by a workflow system. For example, the CPU-Hours allocation available to the workflow should be verified prior to running. Sufficient disk space must be available in the output storage location. In some cases, a specialized queue, or a reservation for a set of compute nodes, is used, in which case the job should be run in a specific queue or at a specific time. Although it is possible to include these constraints into the workflow system, we decided to leave the MPI-based calculations out of the workflow for now since they are run only once per target site. However, we plan to make them a part of the abstract workflow in the future.

We wanted to take advantage of the SCEC workflow system's ability to automatically restart jobs that fail. However, we recognized that special care must be taken when restarting large, multi-day, 280 processor jobs. We looked for ways to utilize the restart capabilities of the workflow with our SGT calculation. One way to address the restart capability is to model the SGT calculation as a series of smaller steps with checkpoint files. Then a failure would get

restarted from the last checkpoint rather than from the beginning. However, to accomplish this we needed to elaborate our definition of the workflow to identify a series of restart-able calculations. This added complexity into our workflow which, in our judgment, did not add sufficient value.

The first two programs in the sequence Mesh_Maker and Velocity_Mesh_Maker are run to create a velocity mesh that represent the region of interest at the appropriate resolution with the properties of the geological structure. The sequence of these two programs could easily be incorporated into a workflow. However, we noted that these two programs are run only once during the workflow. Running a program and submitting a workflow for execution require essentially the same amount of operator interaction. If we are running the program only once, adding in the overhead of workflow construction simply adds complexity. More importantly, these programs are executed before the execution of the Strain_Green_Tensor (SGT) calculation, but the SGT was not to be run as part of the workflow due to the interactive aspects of its execution. Running these two programs as an isolated workflow did not seem worthwhile.

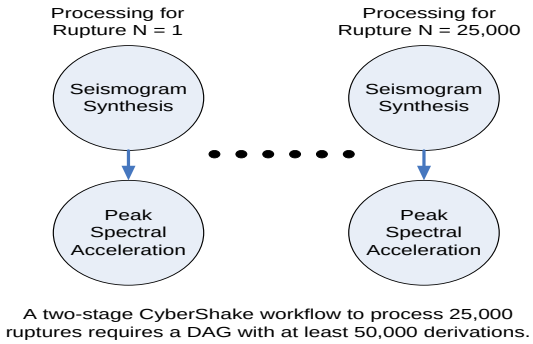

Fig. 11.6. The CyberShake abstract workflow has two processing steps. These two processing steps must be repeated for approximately 25000 times each to produce a single PSHA hazard curve.

By the end of the abstract workflow generation process, the abstract workflow that emerged consisted of only two steps, Seismogram_Synthesis and Peak_Spectral Acceleration. These two steps are shown in Figure 11.6.

As this final abstract workflow design emerged, we recognized two additional criteria worth considering when identifying appropriate elements of a workflow. One, because DAG's do not contain loops, the number of times a particular program is called can greatly impact the construction and execution of the workflow. Two, we found it is worth considering the impact of long delayed program execution.

First, since DAG's may not contain loops, each call to a program (referred to as a derivation in the VDS literature) must be explicitly mentioned in the workflow. In our workflows, we had approximately 25,000 derivations for each step in the dataflow, leading to over 50,000 derivations per workflow. This led to very large and cumbersome DAX's. While we subdivided our workflows to make the derivation count per DAG more manageable, we did end up with tens of thousands of derivations in each DAG. The need to keep the number of derivations at a workable level had some impact on the workflow design. The VDS workflow tools can support large workflows but there are practical upper limits on the size of DAX files and the number of derivations specified in any one particular DAX. Our experience suggests that DAX's with no more than 50,000 transformations work best.

One more issue we identified in converting our CyberShake computational pathway to an abstract workflow is related to delayed execution of programs. At the very end of our computational pathway, once all the calculations are performed, we ran a small program that calculates the actual hazard curve. However, based on the time required to execute all the jobs that lead up to this last summary stage, the execution time for this final job could be days, even weeks after the workflow is submitted. When a job in a workflow has an expected execution time that is days, or weeks, in the future, there is a reasonable possibility that the computing environment will change between now and then. Currently many of the grid systems we target in our work do not provide an easy way of programmatically accessing up-to-date system information and thus make it impossible for workflow management systems to make good scheduling decisions over time. This leads to lower reliability that the workflow will execute to successful completion. We recognized that long periods of time between submission and execution present practical problems for workflows.

11.11 Resource Provisioning in the CyberShake Workflows

Once the CyberShake abstract workflows were developed, a new issue emerged related to resource provisioning. In the context of CyberShake workflows, provisioning means reserving computer resources for our use during the workflow execution. Our workflow system provides special capabilities that allow us provision the computing resources required by our workflows even though the structure of our workflows does not match the scheduling policies of our underlying computing resources.

To understand this issue, we must examine the requirements of the workflow and characteristics of the execution environment. Once our abstract workflow is converted to an executable workflow, and all the data movement jobs, directory creation jobs, and data registration jobs are added, the total number of jobs in the workflow exceeded 100,000. While some of these are long

running, I/O intensive programs, all of them are single processor, serial programs. The only available computing resources that will run this number of jobs within a reasonable amount of time are the high performance clusters at the TeraGrid and USC HPCC.

However, none of these computational clusters are configured to run a large number of serial jobs; rather, they are configured to run a few, large, parallel jobs. The supercomputer sites implement this policy in a couple of ways. First, the job submission managers at these sites will typically allow a single user to submit only 100 jobs at a time. None of the supercomputer sites would allow us to dump 100,000 jobs in a queue all at once. This issue could be addressed to some extent by the job submission throttling capabilities of our Pegasus meta-scheduler but the number of jobs we need to schedule represents a real issue for the CyberShake workflows.

Second, the supercomputer facilities give preference to large parallel jobs through the job priorities used by the underlying job scheduling systems. The supercomputer facilities used to give priority to jobs that required a small number of nodes, and that ran for a short amount of time. However, the sites recognized that many of these small jobs could be run elsewhere. The sites decided that they preferred to support the very large jobs that could only run on the large supercomputer clusters. Job scheduling algorithms were changed so that large, highly-parallel, long running jobs, that is, supercomputer class jobs, received scheduling priority.

The SCEC researchers recognized that the CyberShake computations were supercomputer class computations even though they were not written as MPI-based jobs. Rather than re-write all the CyberShake programs into parallel codes, our workflow system was able to address these problems through the provisioning techniques offered by the Condor glide-in system [3]. The Condor tools allow us to run small scheduler program, called startd, on one, or many, cluster compute nodes. Once startd programs are running on cluster nodes, we can send CyberShake jobs from our Condor-G job submission host directly to the startd program which then execute the our CyberShake job on its node. Once a CyberShake job completes on a compute note, Condor-G sends another job to startd for execution.

We utilize this Condor glide-in provisioning approach in the following way. We submit a Condor glide-in provisioning job to the high performance clusters that we want to use in which we request a significant number of nodes, such as 50 or 100 nodes. Once this large multi-node jobs starts to run, we submit our CyberShake workflow to the Condor-G and Condor DAGman job managers on our submit host. Condor-G and Condor DAGman send our small serial jobs out to the startd programs running on the compute nodes. The Condor glide-in jobs help to reserve the cluster nodes as the CyberShake jobs are run one after another. Using this provisioning technique, we were able to schedule and run large numbers of CyberShake jobs on the high performance clusters available to us.

11.12 CyberShake Workflow Results

The analysis, software development, configuration, testing and validation work that led up to the first full-scale CyberShake workflows was performed over approximately 6 months. The first two, full-scale, CyberShake workflows were run over a period of approximately one month. At this point the computational rate increased dramatically. Eight additional CyberShake curves have now been calculated at a rate of approximately one a week.

During our first two full-scale CyberShake workflow runs, we executed over one 261,000 separate jobs, at four different computing centers (SCEC, SDSC, NCSA, and USC), and we used over 1.8 CPU-Years of processing time. Over 80,000 separate files were created, and registered into our data management system. We are still collecting statistics on the subsequent eight site calculations.

The CyberShake workflows made good use of our grid-based environment. SCEC computers were used as job submission hosts and as storage location for the resulting seismograms, spectral acceleration, and hazard curve data files. The CyberShake workflow executed at multiple sites include SCEC, USC HPCC, SDSC, and NCSA. The SCEC workflow system allowed us to create file replicas at two TeraGrid sites, and then to divide our workflows across two different supercomputer facilities with the results ending up back at SCEC. This flexible use of available computing resources underscores the value of specifying workflows in a computer resource independent manner. It also underscores the capabilities that can be built on top of a grid-based infrastructure.

11.13 Conclusions

The first ten CyberShake probabilistic seismic hazard curves are currently under analysis. The CyberShake results are so new that conclusions regarding the scientific benefits of using 3D waveform-based intensity measure relationship in probabilistic seismic hazard analysis are still pending. Regardless of the final judgment on this new class of PSHA hazard curves, CyberShake represents an important research effort that has provided SCEC scientists with results needed to evaluate this promising new approach to PSHA.

Our scientific workflow tools provided scalability of calculation through automation. These tools allow us to work at a computational scale that would be very difficult to achieve without them. But we recognize that the computational demands of SCEC science is increasing just as quickly as our computational capabilities.

In order to meet the computational requirements of SCEC science in the near future, we need to improve our workflow automation. We plan to begin by increasing the number of programs executed as a part of the workflow. In order to do this, we must resolve the issues that caused us to remove computational

jobs from the workflow. At this point, it appears that the portions of our computational pathway that benefit from modeling as a workflow share two characteristics, **high repetitiveness** and **low interactivity**. We believe that these characteristics may be used to identify which parts of a series of scientific calculations can be most readily expressed as a scientific workflow regardless of the underlying workflow technology.

These principles suggest that a high priority for us should be to integrate any highly repetitive steps of the computational pathway that is not yet in the workflow. They also suggest we should make an effort to reduce the interactivity of the programs used in our computational pathways so these programs can be more easily used in the workflow. In addition to removing operator interactions with the program, we must also look for ways to remove operator interactions in the development, configuration, and submission of the workflows themselves.

We believe that scientific workflow tools provide the current best technology for working at the computational scale needed to perform SCEC's transformative seismic hazard analysis research. If SCEC research goals required only one or two hazard curves, it may have been faster to calculate them without the use of a workflow system. However, since SCEC researchers want to calculate hundreds, or thousands, of these hazard curves, we needed a system that will allow us to scale-up the large CyberShake computational pathway calculation by one or two orders of magnitude. We believe that as SCEC workflow tools evolve and improve, they will make this level of scientific processing and data management possible.

11.14 Acknowledgements

This work was performed by a large group of people at SCEC, ISI, USC High Performance Computing and Communications Center (USC HPCC), the San Diego Supercomputer Center (SDSC), the National Center for Supercomputing Applications (NCSA), the USGS, and URS Corporation. SCEC contributors include Thomas H. Jordan, Li Zhao, David Okaya, Scott Callaghan, John Mehringer, Nitin Gupta, Vipin Gupta, and Hunter Francoeur. ISI contributors include Gaurang Mehta, Karan Vahi, Gurmeet Singh, Carl Kesselman and Sridhar Gullapalli. USC HPCC contributors include Maureen Dougherty, Garrick Staples, and Brian Mendenhall. SDSC contributors include Yifeng Cui, Amit Majumdar, Don Frederick, and Reagan Moore. NCSA contributors include Randy Butler, Tim Cockerill, John Towns, and Dan Lapine. USGS contributors include Ned Field. URS Corporation contributors include Robert Graves. This work was support by the National Science Foundation (NSF) under contract EAR-0122464 - The SCEC Community Modeling Environment (SCEC/CME): An Information Infrastructure for System-Level Earthquake Research. This research was also supported by the Southern California Earthquake Center.SCEC is funded by NSF Cooperative Agreement EAR-0106924

and USGS Cooperative Agreement 02HQAG0008. The SCEC contribution number for this article is 972.

References

1.
2. The 2mass project. http://www.ipac.caltech.edu/2mass.
3. Condor glidein. http://www.cs.wisc.edu/condor/glidein.
4. Einstein@Home Project. See web site at: `http://www.physics2005.org/events/einsteinathome/`.
5. Flexible image transport system.
6. The galactic legacy infrared mid-plane survey extraordinaire (glimpse). http://www.astro.wisc.edu/sirtf/.
7. Glimpse validation images. http://www.astro.wisc.edu/sirtf/2massimages/2massimages.html.
8. Iphas image gallery. http://astro.ic.ac.uk/Research/Halpha/North/gallery.shtml.
9. Iphas: the int h-alpha emission survey. http://iapetus.phy.umist.ac.uk/IPHAS/iphas.html.
10. The montage project page. http://montage.ipac.caltech.edu/docs/download.html.
11. The montage project web page.
12. Montage version 1.7.x documentation and download. http://montage.ipac.caltech.edu/docs/.
13. Montage version 1.7.x. photometric and calibration accuracy. http://montage.ipac.caltech.edu/docs/accuracy.html.
14. Mopex, the spitzer science center mosaic engine. http://ssc.spitzer.caltech.edu/postbcd/doc/mosaicer.pdf.
15. The teragrid project. http://www.teragrid.org/.
16. The virtual data toolkit. http://www.vdt.org.
17. WSRF::Lite - Perl Grid Services. `http://www.sve.man.ac.uk/Research/AtoZ/ILCT`.
18. Instant-Grid – A Grid Demonstration Toolkit. http://instant-grid.de/, 2006.
19. W. Allcock, J. Bester, J. Bresnahan, A. Chervenak, I. Foster, C. Kesselman, S. Meder, V. Nefedova, D. Quesnel, and S. Tuecke. Data management and transfer in high-performance computational grid environments. *Parallel Computing*, 28(5):749–771, May 2002.
20. M. Alt, H. Bischof, and S. Gorlatch. Algorithm design and performance prediction in a Java-based Grid system with skeletons. In B. Monien and R. Feldmann, editors, *Euro-Par 2002*, volume 2400 of *Lecture Notes in Computer Science*, pages 899–906. Springer-Verlag, Aug. 2002.

21. M. Alt, A. Hoheisel, H.-W. Pohl, and S. Gorlatch. A Grid Workflow Language Using High-Level Petri Nets. In R. W. et al., editor, *Proceedings of the 6-th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2005*, Poznan, Poland, 2005.
22. I. Altintas, C. Berkley, E. Jaeger, M. Jones, B. Ludäscher, and S. Mock. Kepler: An extensible system for design and execution of scientific workflows. In *SSDBM*, pages 423–424. IEEE Computer Society, 2004.
23. T. Andrews, F. Curbera, H. Dholakia, Y. Goland, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana. Business Process Execution Language for Web Services Version 1.1.
24. J. E. Barnes and P. Hut. A hierarchical $O(N \log N)$ force-calculation algorithm. *Nature*, 324(4):446–449, 1986.
25. C. Baru, R. Moore, A. Rajasekar, and M. Wan. The SDSC storage resource broker. In *Proceedings of CASCON*, 1998.
26. R. A. Benjamin et al. First glimpse results on the stellar structure of the galaxy. *Astrophysical Journal*, 600:L149–L152, 2005.
27. L. Bernardinello and F. de Cindio. A survey of basic net models and modular net classes. In *Advances in Petri Nets 1992, The DEMON Project*, volume 609 of *LNCS*, pages 304–351, London, UK, 1992. Springer-Verlag.
28. L. Bocchi, C. Laneve, and G. Zavattaro. A calculus for long running transactions. In E. Najm, U. Nestmann, and P. Stevens, editors, *FMOODS 2003*, number 2884 in Lecture Notes in Computer Science, pages 124–138, 2003.
29. S. Bowers, D. Thau, R. Williams, and B. Ludäscher. Data procurement for enabling scientific workflows: On exploring inter-ant parasitism. *Lecture Notes in Computer Science*, 3372:57–63, 2005.
30. M. Brune, G. Fagg, and M. Resch. Message passing environments for metacomputing. *Future Generation Computer Systems*, 15(5-6):699–712, 1999.
31. F. Camilo, D. Lorimer, P. Freire, A. Lyne, and R. Manchester. Observations of 20 millisecond pulsars in 47 tucanae at 20 cm. *The Astrophysical Journal*, (535):975, 2000.
32. Convention on Biodiversity Article 2 (Rio Earth Summit), 1992. http://www.biodiv.org/convention/default.shtml.
33. CCA Forum. The Common Component Architecture Technical Specification - version 0.5. Technical report, Common Component Architecture Forum, 2001.
34. A. Chervenak, E. Deelman, I. Foster, L. Guy, W. Hoschek, A. Iamnitchi, C. Kesselman, P. Kunszt, M. Ripeanu, B. Schwartzkopf, H. Stockinger, K. Stockinger, and B. Tierney. Giggle: A Framework for Constructing Scalable Replica Location Services. In *Supercomputing '02: Proceedings of the 2002 ACM/IEEE conference on Supercomputing*, pages 1–17. IEEE Computer Society Press, November 2002.
35. E. Churchwell et al. Rcw 49 at mid-infrared wavelengths: A glimpse from the spitzer space telescope. *The Astrophysical Journal Supplement Series*, 154:322–327, 2004.
36. Condor Team. DAGMan: A Directed Acyclic Graph Manager, July 2005. `http://www.cs.wisc.edu/condor/dagman/`.
37. K. Czajkowski, I. Foster, N. Karonis, C. Kesselman, S. Martin, W. Smith, and S. Tuecke. A Resource Management Architecture for Metacomputing Systems. In *Proc. IPPS/SPDP '98 Workshop on Job Scheduling Strategies for Parallel Processing*, pages 62–82. IEEE Computer Society, 1998.

38. E. Deelman, J. Blythe, Y. Gil, and C. Kesselman. Workflow management in griphyn. In J. Nabrzyski, J. Schopf, and J. Weglarz, editors, *Grid Resource Management*. Kluwer, 2003.
39. E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, S. Patil, M.-H. Su, K. Vahi, and M. Livny. Pegasus : Mapping scientific workflows onto the grid. In *2nd EUROPEAN ACROSS GRIDS CONFERENCE*, Nicosia, Cyprus, 2004.
40. E. Deelman, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, K. Blackburn, A. Lazzarini, A. Arbree, R. Cavanaugh, and S. Koranda. Mapping abstract complex workflows onto grid environments. *Journal of Grid Computing*, 1(1):25–39, 2003.
41. E. Deelman, G. Singh, M.-H. Su, J. Blythe, Y. Gil, C. Kesselman, G. Mehta, K. Vahi, G. B. Berriman, J. Good, A. Laity, J. C. Jacob, and D. Katz. Pegasus: a framework for mapping complex scientific workflows onto distributed systems. *Scientific Programming Journal*, 13(2), November 2005.
42. M. J. Duftler, N. K. Mukhi, A. Slominski, and S. Weerawarana. Web Services Invocation Framework (WSIF). In *OOPSLA 2001 Workshop on Object-Oriented Web Services*, 2001.
43. L. Dutka, B. Kryza, K. Krawczyk, M. Majewska, R. Slota, L. Hluchy, and J. Kitowski. Component-expert Architecture for Supporting Grid Workflow Construction Based on Knowledge. In P. C. P and M. Cunningham, editors, *Innovation and the Knowledge Economy. Issues, Applications, Case Studies*, volume 2, pages 239–246. IOS Press, 2005.
44. A. Faulkener, I. Stairs, M. Kramer, A. Lyne, G. Hobbs, A. Possenti, D. Lorimer, R. Manchester, M. McLaughlin, N. DÁmico, F. Camilo, and M. Burgay. The parkes multibeam pulsar survey: V. finding binary and millisecond pulsars. *Mon. Not. R. Astron. Soc.*, (355):147–159, 2004.
45. A. Faulkner. *Search methods for binary and millisecond pulsars*. PhD thesis, Faculty of Physics and Astronomy, University of Manchester, UK, December 2004.
46. I. Foster, C. Kesselman, J. Nick, and S. Tuecke. The Physiology of the Grid: An Open Grid Services Architecture for Distributed Systems Integration. Technical report, Open Grid Service Infrastructure WG, Global Grid Forum, 2002.
47. I. Foster, J. Voeckler, M. Wilde, and Y. Zhao. Chimera: A virtual data system for representing, querying, and automating data derivation. In *14th International Conference on Scientific and Statistical Database Management (SSDBM'02)*, Edinburgh, Scotland, 2002.
48. Fraunhofer-Gesellschaft. The Fraunhofer Resource Grid. http://www.fhrg.fraunhofer.de/, 2006.
49. J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke. Condor-G: A Computation Management Agent for Multi-Institutional Grids. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPCD-'01)*, 2001.
50. E. Greisen and M. Calabretta. Representation of celestial coordinates in fits.
51. T. Gubala, M. Bubak, M. Malawski, and K. Rycerz. Semantic-based Grid Workflow Composition. In R. W. et al., editor, *Proceedings of the 6-th Intl. Conf. on Parallel Processing and Applied Mathematics PPAM'2005*, Poznan, Poland, 2005.
52. S. Hinz, K. Schmidt, and C. Stahl. Transforming BPEL to Petri Nets. In W. van der Aalst, B. Benatallah, F. Casati, and F. Curbera, editors, *Proceed-*

ings of the Third International Conference on Business Process Management (BPM 2005), volume 3649 of *Lecture Notes in Computer Science*, pages 220–235, Nancy, France, September 2005. Springer-Verlag.
53. A. Hoheisel and U. Der. An XML-Based Framework for Loosely Coupled Applications on Grid Environments. In P. M. A. Sloot, D. Abramson, A. V. Bogdanov, J. J. Dongarra, A. Y. Zomaya, and Y. E. Gorbachev, editors, *Computational Science – ICCS 2003*, volume 2657 of *LNCS*, pages 245–254. Springer-Verlag, 2003.
54. A. Hoheisel and U. Der. Dynamic workflows for grid applications. In *Proceedings of the Cracow Grid Workshop '03*, Cracow, Poland, 2003.
55. IBM and BEA. BPELJ: BPEL for Java . http://www.ibm.com/developerworks/webservices/library/ws-bpelj/, 2004.
56. ISO/IEC 10026-1. Information technology – Open Systems Interconnection – Distributed Transaction Processing – Part 1: OSI TP Model, 1998.
57. ISO/IEC 15909-1. High-level Petri nets – Part 1: Concepts, definitions and graphical notation, 2004.
58. ISO/IEC 15909-2. High-level Petri nets – Part 2: Transfer Format, 2005. Working Draft.
59. K. Jensen. An introduction to the theoretical aspects of Coloured Petri Nets. In J. W. de Bakker, W.-P. de Roever, and G. Rozenberg, editors, *A Decade of Concurrency, Lecture Notes in Computer Science*, volume 803, pages 230–272. Springer-Verlag, 1994.
60. A. C. Jones, R. J. White, W. A. Gray, F. A. Bisby, N. Caithness, N. Pittas, X. Xu, T. Sutton, N. J. Fiddian, A. Culham, M. Scoble, P. Williams, O. Bromley, P. Brewer, C. Yesson, and S. Bhagwat. Building a biodiversity grid. In A. Konagaya and K. Satou, editors, *Grid Computing in Life Science: First International Workshop on Life Science Grid, Revised selected and invited papers (LNCS/LNBI 3370)*, pages 140–151. Springer Verlag, 2005.
61. A. C. Jones, X. Xu, N. Pittas, W. A. Gray, N. J. Fiddian, R. J. White, J. S. Robinson, F. A. Bisby, and S. M. Brandt. SPICE: a flexible architecture for integrating autonomous databases to comprise a distributed catalogue of life. pages 981–992.
62. M. Jüngel, E. Kindler, and M. Weber. The Petri Net Markup Language. In S. Philippi, editor, *7. Workshop Algorithmen und Werkzeuge für Petrinetze*, pages 47–52, http://www.informatik.hu-berlin.de/top/pnml/, 2000. Universität Koblenz-Landau.
63. Project JXTA, July 2005. `http://www.jxta.org`.
64. G. Kandaswamy, L. Fang, Y. Huang, S. Shirasuna, S. Marru, and D. Gannon. Building web services for scientific grid applications. *IBM Journal of Research and Development*, 50, 2006.
65. D. S. Katz, J. C. Jacob, G. B. Berriman, J. Good, A. C. Laity, E. Deelman, C. Kesselman, G. Singh, M.-H. Su, and T. A. Prince. A comparision of two methods for building astronomical image mosaics on a grid. In *34th International Conference on Parallel Processing Workshops ICPP 2005 Workshops, 14-17 June 2005, Oslo, Norway*, pages 85–94. IEEE Computer Society, 2005.
66. Linked Environments for Atmospheric Discovery. http://lead.ou.edu/.
67. LEAD Year-2 Annual Report. http://lead.ou.edu/pdfs/LEAD_Year-2_Report.pdf, 2005.

68. F. Leymann and D. Roller. *Production Workflow: Concepts and Techniques*. Prentice Hall, 1999.
69. M. Litzkow, M. Livny, and M. Mutka. Condor - a hunter of idle workstations. In *Proceedings of the 8th International Conference on Distributed Computing Systems*, pages 104–111, June 1988.
70. C. J. Lonsdale et al. Swire: The sirtf wide-area infrared extragalactic survey. *Pub Ast Soc Pac*, 115:897, 2003.
71. M. Lorch and D. Kafura. Symphony — A Java-based Composition and Manipulation Framework for Computational Grids. In *Proceedings of the CCGrid2002*, Berlin, Germany, 2002.
72. H. D. Lord. Improving the Application Environment with Modular Visualization Environments. *Computer Graphics*, 29(2):10–12, 1995.
73. D. Lorimer and M. Kramer. *A Handbook of Pulsar Astronomy*. Cambridge University Press, Cambridge, 2005.
74. A. G. Lyne and G. Smith. *Pulsar Astronomy, 3rd Edition*. Cambridge University Press, Cambridge, 2005.
75. M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley Series in Parallel Computing. John Wiley and Sons, 1995. http://www.di.unito.it/~greatspn/GSPN-Wiley/.
76. S. Microsystems. Java Remote Method Invocation (RMI). http://java.sun.com/products/jdk/rmi/.
77. N. Mulyar and W. van der Aalst. Patterns in Colored Petri Nets. In *BETA Working Paper Series*, WP 139. Eindhoven University of Technology, Eindhoven, 2005.
78. myLEAD. http://www.cs.indiana.edu/dde/projects/lead.html.
79. OASIS. OASIS Web Services Business Process Execution Language (WSBPEL) TC. http://www.oasis-open.org/committees/wsbpel.
80. OASIS. WS-Resource Properties (WSRF-RP), June 2004. http://docs.oasis-open.org/wsrf/2004/06/wsrf-WS-ResourceProperties-1.2-draft-04.pdf.
81. Object Management Group (OMG). The Common Object Request Broker Architecture (CORBA). http://www.corba.org.
82. T. Oinn, M. Addis, J. Ferris, D. Marvin, M. Senger, M. Greenwood, T. Carver, K. Glover, M. R. Pocock, A. Wipat, and P. Li. Taverna: A tool for the composition and enactment of bioinformatics workflows. *Bioinformatics*, 20(17):3045–3054, Nov 2004.
83. S. Parastatidis and J. Webber. The SOAP Service Description Language. http://www.ssdl.org/docs/v1.3/html/SSDL%20v1.3.html, April 2005.
84. S. Parastatidis, J. Webber, S. Woodman, D. Kuo, and P. Greenfield. An Introduction to the SOAP Service Description Language v1.3. http://www.ssdl.org/docs/v1.3/html/SSDL%20whitepaper%20v1.3.html, April 2005.
85. S. Parastatidis, J. Webber, S. Woodman, D. Kuo, and P. Greenfield. Using SSDL to Describe and Reason about Asynchronous and Multi-Message Interactions between Web Services. *IEEE Internet Computing*, Jan/Feb 2006.
86. C. A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1962.

87. S. Pickles, J. Brooke, F. Costen, E. Gabriel, M. Müller, M. Resch, and S. Ord. Metacomputing across intercontinental networks. *Future Generation Computer Systems*, 17(5-6):911–918, 2001.
88. W. Recommendation. Architecture of the World Wide Web, Volume One. http://www.w3.org/TR/webarch.
89. W. Reisig. *Elements Of Distributed Algorithms: Modeling and Analysis with Petri Nets*. Springer-Verlag, sep 1998.
90. W. Reisig. An Informal Introduction to Petri Nets - Running Example: A Producer/Consumer System. In *Petri Nets 2000, 21st International Conference on Application and Theory of Petri Nets*, 2000.
91. M. Resch, R. D., and S. R. Metacomputing experience in a transatlantic wide area application testbed. *Future Generation Computer Systems*, 15(5-6):807–816, 1999.
92. M. P. Robertson, N. Caithness, and M. H. Villet. A PCA-based modelling technique for predicting environmental suitability for organisms from presence records. *Diversity & Distributions*, 7:15–27, 2001.
93. D. J. Schlegel, D. Finkbeiner, and M. Davis. Maps of dust infrared emission for use in estimation of reddening and cosmic microwave background radiation foregrounds. *Astrophysical Journal*, 500:525, 1998.
94. S. Shirasuna. X Baya Workflow Composer. http://www.extreme.indiana.edu/xgws/xbaya.
95. G. Singh, S. Bharathi, A. L. Chervenak, E. Deelman, C. Kesselman, M. Manohar, S. Patil, and L. Pearlman. A metadata catalog service for data intensive applications. In *SC*, page 33, 2003.
96. M. Snir, S. W. Otto, S. Huss-Lederman, D. W. Walker, and J. Dongarra. *MPI: The Complete Reference*. The MIT Press, Cambridge, MA, 1996.
97. Simple Object Access Protocol (SOAP) 1.2. Technical report, W3C, 2003.
98. R. D. Stevens, A. J. Robinson, and C. A. Goble. mygrid: personalised bioinformatics on the information grid. In *ISMB (Supplement of Bioinformatics)*, pages 302–304, 2003.
99. D. Stockwell and D. Peters. The GARP modelling system: problems and solutions to automated spatial prediction. *International Journal of Geographical Information Science*, 13(2):143–158, 1999.
100. A. S. Szalay, editor. *An Architecture for Access to a Compute Intensive Image Mosaic Service in the NVO.*, volume 4686. Proceedings of SPIE, 2002.
101. The K-Wf Grid Project. Knowledge-based Workflow System for Grid Applications. http://www.kwfgrid.net/, 2006.
102. The K-Wf Grid Project. The Grid Workflow Description Language Toolbox. http://www.gridworkflow.org/kwfgrid/gworkflowdl/docs/, 2006.
103. W. Tim Berners-Lee. Web Architecture from 50,000 feet. http://www.w3.org/DesignIssues/Architecture.html.
104. W. van der Aalst. Pi calculus versus Petri nets: Let us eat humble pie rather than further inflate the pi hype. www.bptrends.com, 2005.
105. W. van der Aalst and A. ter Hofstede. Workflow Patterns: On the Expressive Power of (Petri-net-based) Workflow Languages. Technical report, Department of Technology Management, Eindhoven University of Technology P.O. Box 513, NL-5600 MB, Eindhoven, The Netherlands, 2002.

106. G. von Laszewski, B. Alunkal, K. Amin, S. Hampton, and S. Nijsure. GridAnt – client-side workflow management with Ant, 2006. http://www-unix.globus.org/cog/projects/gridant/.
107. W3C. Semantic Web Activity Statement. http://www.w3.org/2001/sw/Activity.
108. W3C. Web Services Description Language (WSDL) Version 2.0 Part 2: Adjuncts. http://www.w3.org/TR/2005/WD-wsdl20-adjuncts-20050803.
109. Wikipedia. Choreography — wikipedia, the free encyclopedia, 2006. http://en.wikipedia.org/w/index.php?title=Choreography&oldid=33366853 [Online; accessed 18-January-2006].
110. Wikipedia. Orchestration — wikipedia, the free encyclopedia, 2006. http://en.wikipedia.org/w/index.php?title=Orchestration&oldid=34882858 [Online; accessed 18-January-2006].
111. Wikipedia. Petri net — wikipedia, the free encyclopedia, 2006. http://en.wikipedia.org/w/index.php?title=Petri_net&oldid=35563598 [Online; accessed 18-January-2006].
112. Workflow Management Coalition. Terminology & Glossary. Technical report, WfMC, 1999. http://www.wfmc.org/.