Reducing Time-to-Solution Using Distributed High-Throughput Mega-Workflows – Experiences from SCEC CyberShake

Scott Callaghan[1], Philip Maechling[1], Ewa Deelman[2], Karan Vahi[2], Gaurang Mehta[2], Gideon Juve[1], Kevin Milner[1], Robert Graves[3], Edward Field[4], David Okaya[1], Dan Gunter[5], Keith Beattie[5], Thomas Jordan[1]

(1) University of Southern California, Los Angeles, CA 90089
{scottcal, maechlin, juve, kmilner, okaya, tjordan}@usc.edu
(2) USC Information Sciences Institute, Marina Del Rey, CA 90292; {deelman, vahi, gmehta}@isi.edu
(3) URS Corporation, Pasadena, CA 91101; Robert_Graves@URSCorp.com
(4) U.S. Geological Survey, Pasadena, CA 91106; field@caltech.edu
(5) Lawrence Berkeley National Laboratory, Livermore, CA 94551; dkgunter,ksbeattie}@lbl.gov

Abstract

Researchers at the Southern California Earthquake Center (SCEC) use large-scale grid-based scientific workflows to perform seismic hazard research as a part of SCEC's program of earthquake system science research. The scientific goal of the SCEC CyberShake project is to calculate probabilistic seismic hazard curves for sites in Southern California. For each site of interest, the CyberShake platform includes two large-scale MPI calculations and approximately 840,000 embarrassingly parallel post-processing jobs. In this paper, we describe the computational requirements of CyberShake and detail how we meet these requirements using grid-based, high-throughput, scientific workflow tools. We describe the specific challenges we encountered and we discuss workflow throughput optimizations we developed that reduced our time to solution by a factor of three and we present runtime statistics and propose further optimizations.

1. Introduction

Researchers from the Southern California Earthquake Center (SCEC) [1] are using high performance computing to advance their program of earthquake system science research. As a part of this research program, SCEC scientists have developed a new technique in which full 3D waveform modeling is used in probabilistic seismic hazard analysis calculations (PSHA). However, the complexity and scale of the required calculations prevented actual implementation of this new approach to PSHA. Since that time, SCEC computer scientists have gathered the seismological processing codes and the cyberinfrastructure tools required for this research. The codes and tools have been assembled into what we call the CyberShake computational platform [2]. In SCEC terminology, a computational platform is a vertically integrated and well-validated collection of hardware, software, and people that can produce a useful research result. The CyberShake computational platform is designed to perform physics-based PSHA calculations for sites in the Southern California region. In this paper, we describe CyberShake and our experiences using it to produce this important new type of PSHA hazard curves.

2. CyberShake Science Description

Seismologists quantify the earthquake hazard for a location or region using the PSHA technique by estimating the probability that the ground motions at a site will exceed some intensity measure (IM) over a given time period. These intensity measures include peak ground acceleration, peak ground velocity, and spectral acceleration. PSHA seismic hazard estimates are expressed in statements such as: a specific site (e.g. the USC University Park Campus) has a 10% chance of experiencing 0.5g acceleration in the next 50 years. This type of ground motion estimate is useful for civic planners and building engineers because it provides a probable "upper limit" on the ground motions.

PSHA estimates are delivered as PSHA hazard curves (Fig. 1) that present ground motion values (e.g. accelerations in g) on one axis, and the probability of exceeding these ground motion level within one year on the other axis.

A set of hazard curves, for many sites in a geographical region, may be integrated into a regional

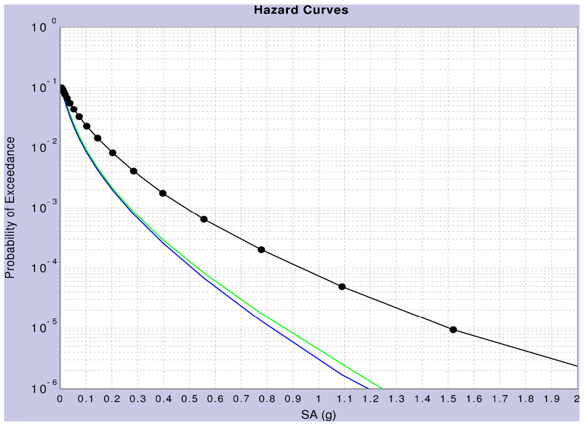

Figure 1. Three hazard curves for the Whittier Narrows site near Los Angeles. The black line is the physics-based CyberShake simulation results. The blue and green lines are two common attenuation relationships.

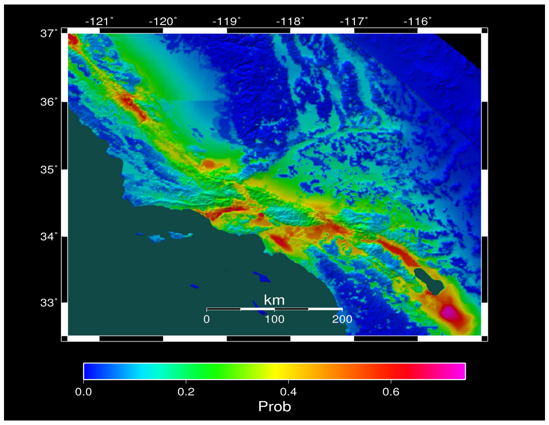

Figure 2. An example hazard map for the Southern California area generated using OpenSHA with attenuation relationships. Colors show the probability that ground motions will exceed 0.1g within next 50 years.

hazard map (Fig. 2) by keeping either the IM or probability constant and plotting the other as a function of geographic location. PSHA calculations require two essential inputs. First, a list of all possible future earthquakes that might affect a site is needed (this information is available from an Earthquake Rupture Forecast (ERF)). Secondly, a way of calculating the ground motions that will be produced by each earthquake in the list is needed. Traditionally, the ground motions produced by each earthquake are calculated using empirical attenuation-based relationships. However, this approach has limitations. The data used to develop the attenuation relationships do not cover the full range of possible earthquake magnitudes. Also, attenuation relationships do not include basin or rupture directivity effects.

To address these issues, CyberShake uses physics-based 3D ground motion simulations with anelastic wave propagation models to calculate the ground motions that would be produced by each of the earthquakes in the ERF. The scientists involved in CyberShake believe that this new approach to PSHA has clear scientific advantages and can improve the accuracy of the seismic hazard information currently available. For a typical site in Los Angeles, the latest ERF available from the USGS (UCERF 2.0) identifies more than 7,000 earthquake ruptures with magnitude > 5.5 that might affect the site. For each rupture, we must capture the possible variability in the earthquake rupture process. So we create a variety of hypocenter, slip distribution, and rise times for each rupture to produce over 415,000 rupture variations, each representing a potential earthquake. In CyberShake processing, there is a fairly technical, but important, distinction between ruptures (~7,000) and rupture variations (~415,000). These distinctions impact our workflows. The SGTs calculations generate data for each rupture, but post-processing must be done for each rupture variation.

Once we define the ruptures and their variations, CyberShake uses an anelastic wave propagation simulation to calculate strain Green tensors (SGTs) around the site of interest. Seismic reciprocity is used to post-process the SGTs and obtain synthetic seismograms [3]. These seismograms are then processed to obtain peak spectral acceleration values, which are combined into a hazard curve (Fig. 1). Fig. 3 contains a workflow illustrating these steps. Ultimately, CyberShake hazard curves from hundreds of sites will be interpolated to construct a physics-based seismic hazard map for Southern California.

In 2005, the CyberShake curves were calculated using a smaller number of ruptures, and a smaller number of rupture variations, only about 100,000 rupture variations per site [4]. Since then, a new ERF (UCERF 2.0) was released by the Working Group on California Earthquake Probabilities [5]. This new ERF identifies more possible future ruptures than were used in 2005. In addition, based on initial CyberShake results, SCEC scientists decided to specify more variability in the rupture processes. As a result, the number of rupture variations we must manage increased by more than a factor of 4.

3. Computing Requirements

The CyberShake platform must run many jobs, and manage many data files. Thus, it requires extensive computational resources. An outline of computational and data requirements is given in Table

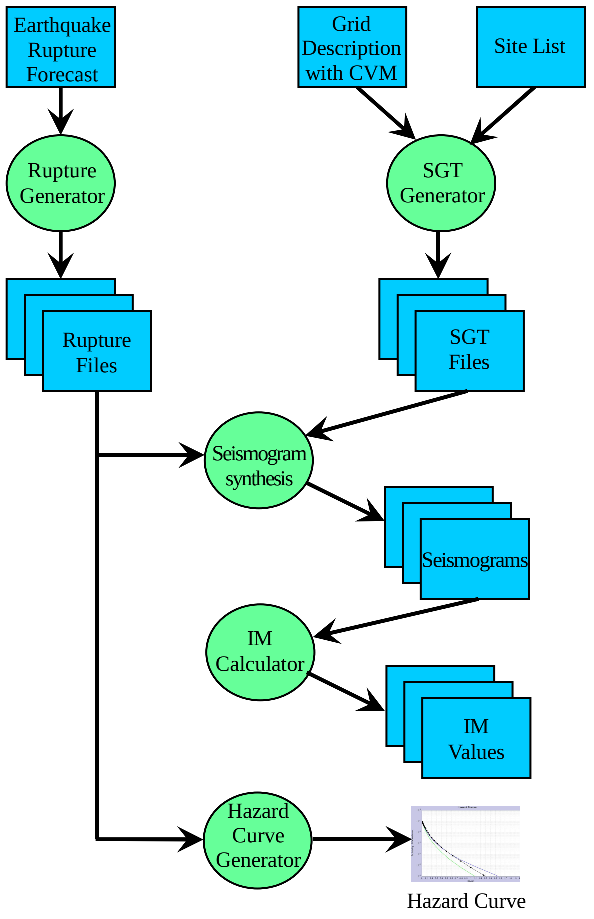

Figure 3: A high-level Cybershake workflow.

1. To compute the SGTs, a mesh of about 1.5 billion points must be constructed and populated with seismic wave velocity information. The velocity mesh is then used in a wave propagation simulation for 20,000 time steps.

Once the SGTs are calculated, the post-processing is performed. Processing is done for each of the rupture variations and there are approximately 415,000 rupture variations for each hazard curve. Post-processing begins by selecting a rupture variation. Then, SGTs corresponding to the location of the rupture are extracted from the volume and used to generate synthetic seismograms which represent the ground motions that the rupture variation would produce at the site we are studying. Next the seismograms are processed to obtain the IM of interest, which, in our current study, is peak spectral acceleration at 3.0 seconds. Each execution of these post processing steps takes no more than a few minutes, but SGT extraction must be performed for each rupture, and seismogram synthesis and peak spectral acceleration processing must be performed for each rupture variation. On average, 7,000 ruptures and 415,000 rupture variations must be considered for each site. Therefore, each site requires approximately 840,000 executions, 17,000 CPU-hours and generates about 66 GB of data.

Considering only the computational time, performing these calculations on a single processor would take almost 2 years. In addition, the large number of independent post-processing jobs necessitates a high degree of automation, to help submit jobs, manage data, and provide error recovery capabilities should jobs fail. The velocity mesh creation and SGT simulation are large MPI jobs which run on a cluster using spatial decomposition. The post-processing jobs have a very different character, as they are "embarrassingly parallel" – no communication is required between jobs.

These processing requirements indicated that the CyberShake computational platform requires both high-performance computing (for the SGT calculations) and high-throughput computing (for the post-processing). To make a Southern California hazard map practical, time-to-solution per site needs to be short, on the order of 24-48 hours. This emphasis on reducing time-to-solution, rather than categorizing the system as a capability or capacity platform, pushes the CyberShake computational platform into the high productivity computing category which is emerging as the key capability needed by science groups. In contrast to capability computing (a single, large job) and capacity computing (smaller, multiple jobs, often in preparation for a capability run), high productivity computing focuses on high throughput jobs with extremely short runtimes. The challenge is to minimize overhead and increase throughput to reduce end-to-end wallclock time.

Table 1: Data and CPU requirements for the CyberShake components, per site of interest.

Component	**Data**	**CPU hours**
Mesh generation	15 GB	150
SGT simulation	40 GB	10,000
SGT extraction	1 GB	250
Seismogram synthesis	10 GB	6,000
PSA calculation	90 MB	100
Total	66 GB	17,000

4. Technical Approach

We developed a technical approach to CyberShake which meets the computational requirements defined by the domain scientists and also minimizes the time-to-solution. First, we recognized that we need to distribute the calculations because SCEC does not own the necessary computational resources. However, we do have HPC allocations at the USC Center for High Performance Computing and Communications and on the NSF TeraGrid [6]. Grid computing enables us to acquire and use resources on large, remote clusters. Second, the large number of small jobs suggests a distributed, high-throughput solution, such as Condor [7]. Third, the automation required including job submission, data management, and error recovery led us to the use of scientific workflow tools. Using workflows increases the degree of automation and provides a concise way to enforce dependencies between jobs. We constructed the CyberShake computational platform using grid computing tools, high throughput capabilities of Condor, and scientific workflows using Pegasus-WMS [8, 9] to plan and run the very large workflows.

To calculate a PSHA hazard curve, we first create an abstract representation of a workflow called a DAX (directed acyclic graph in XML format), which contains jobs related by dependencies. The DAX is supported by Pegasus and uses logical filenames for executables and input and output files, making it execution platform independent. Next, Pegasus is used to plan the DAX to run on a specific platform. The planning process converts the abstract DAX into a concrete DAG. Pegasus uses its Transformation Catalog [10] to resolve the logical names into physical paths, specific to the remote execution system. Additionally, Pegasus automatically augments the DAG to include the transfer in (stage-in) of required input files, found using the Globus Replica Location Service (RLS) [11], and the transfer out (stage-out) of final data files. Pegasus also wraps the jobs with kickstart [12], which allows us to check the return codes for successful execution and is easy to mine for usage statistics using Netlogger-based tools [http://acs.lbl.gov/NetLoggerWiki].

The DAG is then submitted to the workflow execution component of Pegasus-WMS, DAGMan [http://www.cs.wisc.edu/condor/dagman]. DAGMan manages the job execution by determining when jobs are ready to run, submitting jobs via Globus to remote resources, and throttling the number of jobs so as to not overwhelm either the submission host (where the jobs originate) or the remote platform (where the jobs execute). If a job fails, DAGMan will automatically retry the failed job and create a rescue DAG if the job cannot be completed successfully. This restart capability is critical for CyberShake. Jobs will fail for a variety of reasons, and being able to resume execution easily is a major benefit. These tools enable CyberShake to be run on a variety of platforms and be managed from a single local job submission host.

Since the SGT simulation requires large MPI calculations while the post-processing involves high throughput embarrassingly parallel jobs, we decided to separate these stages into two independent workflows. This gives us flexibility to run the two pieces at different times on different platforms, since some environments may be optimized for MPI jobs while others are more efficient for short serial jobs.

Many remote execution environments limit the number of jobs a single user can put in their queue. Additionally, remote cluster schedulers often have a scheduling cycle of several minutes, meaning that when jobs finish it can take some time for another job to be scheduled. To address these concerns, we use Condor glideins, a type of multi-level scheduling. Fig. 4 shows the steps involved in using glideins to create a temporary Condor pool:

(1) The user requests a group of nodes for a specified duration via Condor. Condor sends the request to the Globus gatekeeper on the remote host, which submits the request on to the batch scheduler. (2) After waiting in the remote queue, the glidein job begins on the requested nodes. (3) The nodes start up the Condor startd process and advertise themselves back to the Condor collector on the local job submission host as available nodes. This creates a temporary Condor pool on the remote platform. (4) The local submission host can then schedule directly on the remote nodes by matching queued jobs to available resources and sending the job to the node's startd process, which then begins the job.

5. Implementation

We targeted the NSF-funded TeraGrid as an execution environment due to their support for grid computing as well as past SCEC success.

To run CyberShake on the TeraGrid, we have established interoperability between the local SCEC grid and the TeraGrid through a TeraGrid science gateway. We initiate the CyberShake workflows on our local SCEC job submission machine, which communicates to TeraGrid resources using Globus. Our CyberShake jobs run in a TeraGrid gateway account that exists on multiple TeraGrid resources. We map the SCEC researcher submitting CyberShake workflows on SCEC machines to an authorized account on the TeraGrid.

Initially, we targeted our CyberShake workflows at the NCSA's Abe cluster for both SGT and post-processing workflows. However, we ran into complications. We attempted to verify the SGTs by using them to generate a seismogram for a certain source and comparing it against a previously generated seismogram for the same source. However, we were unable to successfully verify the SGTs and despite investigation were unable to pinpoint the source of the

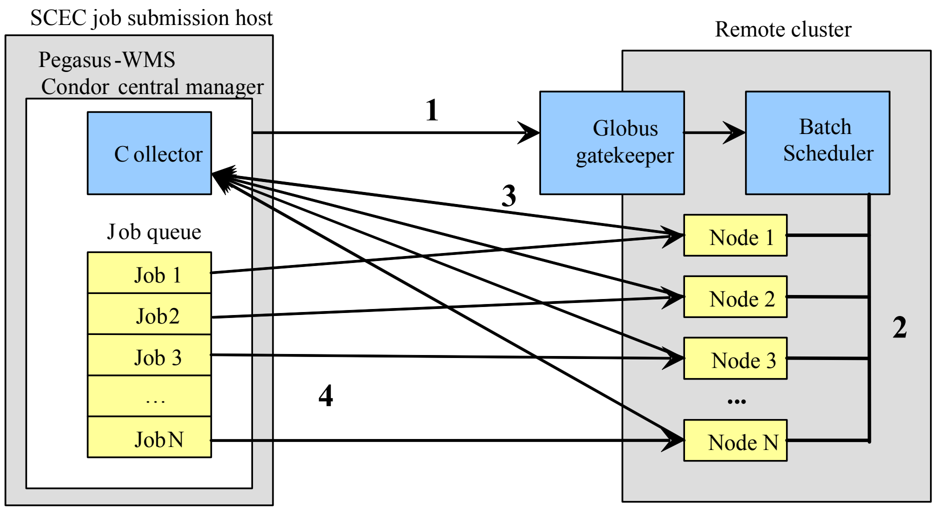

Figure 4: Steps involved in acquiring glideins

error. The post-processing workflow also presented unexpected challenges. Sometimes seismogram synthesis jobs would experience a segmentation fault which would not reoccur if the job was rerun. After extensive memory profiling of the seismogram synthesis code (written in C) and examination of the runtime environment, we concluded that the version of the Linux kernel used by Abe, 2.6.9, may contain a memory management bug. On multi-core systems memory claimed by the cache is not made properly available to running code and therefore segmentation faults can occur even though there is sufficient memory available. This problem was exacerbated by the short runtimes (<1 min) and high memory requirements (1.6 GB) of the synthesis jobs. As a result of these issues, we postponed running CyberShake on Abe until the kernel is upgraded.

We switched our target platforms to NCSA's Mercury cluster for the post-processing workflow and USC's HPCC for the SGT generation. Separating the two stages onto different platforms was trivial; we only had to transfer the two SGT files that are the result of the first workflow to Mercury and register them in the Globus RLS. This also permitted us to utilize computational resources at two remote sites, more than at either site alone. This demonstrates the value of the Pegasus workflow approach which defines abstract workflows. Modifying the workflow to run on a difference physical system is simply a matter of changing configuration information, and replanning the workflow.

6. Workflow challenges and solutions

An immediate problem was how to execute over 800,000 jobs without overwhelming the scheduler on either the submit host or on the execution platform. The average runtime for SGT extraction jobs ("extraction jobs") is 139 seconds; for seismogram synthesis jobs ("synthesis jobs"), 48 seconds; and for peak spectral acceleration jobs ("PSA jobs"), 1 second. Running these short runtime jobs individually through the scheduler would mean that computing resources would sit idle much of the time, waiting for the scheduler to schedule jobs, and the scheduler would be under intense load to schedule new jobs to waiting idle processors.

To reduce the load on the remote platform, we use Condor glide-ins, discussed in Section 4. Since the remote scheduler (on Mercury, PBS) sees only a single job, minimal load is added.

On our local job submission host, we use a Pegasus technique called "clustering" in which jobs of the same type are grouped together. Pegasus alters the DAX so that a number of individual jobs are replaced by a single call to seqexec (a Pegasus remote-site executable), which iterates over a Pegasus-created file containing a list of jobs and sequentially executes them on a remote resource. Each DAX, and therefore each DAG, contains fewer jobs, reducing the load on the local Condor scheduler. We selected clustering values of 20 for the extraction jobs, 40 for seismogram synthesis jobs, and 200 for PSA jobs.

Another challenge was to create and plan the DAX. Previously CyberShake was run with a smaller number of jobs, only about 200,000, using a single DAX, with label-based workflow partitioning to divide up the DAX into 40 partitions. Workflow partitioning separates a workflow into multiple pieces ("partitions"), using user-assigned labels. It is used by Pegasus-WMS to improve the scalability and the reliability of the workflow mapping and execution. Dependencies between the partitions follow the dependencies of the original workflow. Partitions are then mapped and executed according to the dependencies. Since each partition corresponds to a execution directory on the remote host, running multiple partitions reduces the number of files per directory, improving filesystem response. Initially we attempted to extend this technique to all 840,000 jobs and continue to use label-based partitioning. However, the size of the single DAX file – about 1 GB -- made it too memory-intensive to partition or plan on any of SCEC's servers. We altered the workflow generator to create 40 independent DAXes rather than a single DAX with 40 partitions. This gave us similar benefits to partitioning but reduced the memory requirements of planning, only requiring about two minutes of wall-time and 1 GB of memory to plan each DAX.

7. Optimizations

We were able to introduce optimizations to decrease the runtime of a single workflow and reduce the manual effort involved.

Monitoring our execution, we discovered we had poor utilization; many available processors spent long periods of time idle. This was due to high clustering values. Each DAX has only about 20 extraction jobs, but each extraction job has between 2 and 1568 synthesis child jobs. In order to keep processors busy, the extraction jobs must be completed quickly so the more numerous synthesis jobs can begin execution and claim idle processors. When clustering is used, dependencies are moved to the cluster level. So instead of a synthesis job having an extraction job parent, a synthesis cluster has an extraction cluster parent. Using a cluster factor of 20, it took at least 45 minutes before any synthesis clusters would start execution. We found that for most of the duration of the workflow not all the processors were used as a result.

The clustering parameters were adjusted to 2 for extraction jobs, 10 for synthesis jobs, and 60 for PSA jobs. With these new parameters, extraction clusters completed in a few minutes, allowing synthesis clusters to start more quickly and run on previously idle processors. Although this increased the total number of clusters fourfold, it reduced average workflow completion times by a factor of three since the processors spent much less time idle. We also doubled the number of DAXes, from 40 to 80, so that the number of files per directory on the remote host was kept to a reasonable level (around 30,000).

Increasing to 80 DAXes made manual submission and monitoring unwieldy and labor-intensive. To increase automation, we use a Pegasus construct called a PDAX. A PDAX is a recursive workflow description, in the form of a DAX. The PDAX is constructed so that it contains Condor prescript jobs, which plan the DAXes into DAGs, and main jobs which submit the DAGs to Condor. The PDAX is planned into a PDAG and submitted to Condor. Since having all 80 DAXes running simultaneously could put excessive load on the submission host and the GridFTP servers [13] we initially set up "pipelines", chains of DAXes with parent-child relationships so the chain executes serially. We used 10 pipelines, each with 8 DAXes, so that 10 DAXes run simultaneously. When one of the 10 initial DAXes finished, its child DAX would be planned and run, and so on until all 10 pipelines completed.

If a DAX failed, the pipeline of the failed DAX would stop execution, since jobs further down the pipeline cannot run until their parent DAX finishes. The artificial dependencies we introduced to throttle w had an inadvertent impact on failure recovery. Over time the number of concurrent DAXes running would decrease as pipelines stalled. This resulted in poor use of the available processors. Eventually the entire workflow had to be manually halted and resumed, which would retry the failed jobs and begin again with 10 concurrent DAXes.

To eliminate the negative impact of the dependencies, we altered the workflow to contain a pool of 80 independent DAXes rather than pipelines. Using the Condor parameter maxjobs, we were able to limit the number of concurrent DAXes. If one failed, it did not hold up other DAXes; instead, a new DAX was pulled from the pool, planned, and run. Changing to a pool approach increased the amount of automation, and decreased runtime for the workflow.

We noticed that the most common cause of DAX failure was an error with the GridFTP stageout job at the end of the workflow. The transfer jobs were inserted automatically by Pegasus to transfer data products back to SCEC servers after completion of each DAX. If the transfer failed, the workflow was configured to retry it three times before failing and throwing a rescue DAG, which registered as a failure in the PDAX. The PDAX would respond by starting the DAX over from the planning stage, and since for reasons of space we did not use the Pegasus checkpoint feature, all of the successful computations were thrown out and rerun. To avoid redoing completed computations, the PDAX was altered to tap into the auto-restart feature of Condor 7.1, which checks to see if a rescue DAG exists and, if so, restart from it. This prevented us from duplicating computations which had already been successfully executed.

Further investigation of the transfer failures revealed that the main factor was the large number of

Table 2: Number of jobs by type and site

Site	Pegasus jobs	Extraction	Synthesis	PSA	Zip	Total jobs	Failed jobs	Walltime (min)
1	504	7000	417886	417886	80	843356	264	1042
2	950	7093	418256	418256	154	844709	483	1066
3	1074	7026	417954	417954	156	844164	624	1287
4	896	6932	415416	415416	158	838818	422	988
5	736	6912	415778	415778	156	839360	268	1542
6	703	7045	426740	426740	156	861384	235	964
7	946	6823	416090	416090	158	840107	472	792
8	1552	6919	418946	418946	156	846519	1084	1287
9	923	6947	417772	417772	156	843570	455	890
All	8284	62697	3764838	3764838	1330	7601987	4307	1095

files being transferred. Each DAX had two transfer jobs of approximately 5,000 files, one job for seismograms and one for peak spectral acceleration files. Additionally, the files are quite small; the seismogram files are 24 KB each, and the spectral acceleration files are just 216 bytes. Using GridFTP to transfer so many small files was not efficient. We added two additional jobs to each DAX which would zip all of each kind of file before transfer, reducing the number of files to transfer from over 800,000 to 160, two per DAX. Additionally, transferring larger files lets us utilize the speedup advantage of GridFTP. Even with the additional time required for zipping, these improvements sped up the stageout of data products by a factor of ten. This helped us improve our processor utilization, as DAXes could now quickly perform their transfers, finish, and clear the way for a new DAX to start execution and claim any available resources. Modifying the workflow also had the side effect of improving our local processing. After stageout, we insert the peak spectral acceleration values into a local database. We then use the database to generate a hazard curve, the final result. Performing the database insertions from a zip file is much faster than from a large number of small files, by a factor of four in our case.

8. Results

The computational results reported in this paper were the result of performing ten CyberShake runs, occurring from June 26, 2008 – July 18, 2008. All calculations were performed using a reservation of 400 nodes (800 processors) on NCSA's IA-64 Mercury TeraGrid cluster. The first site was calculated using a mix of varied methods, and so is not included in the total statistics. A detailed discussion of the science results can be found in [14]. The results below were obtained using the Netlogger toolkit [15]. To normalize and correlate the resulting flood of information, the NetLogger Toolkit uses a relational database back-end. In order to analyze large amounts of data, often consisting of a million records, it leverages existing log processing capabilities. The log parser converts the kickstart and Condor logs from their native format to NetLogger's standard format. The database loader loads NetLogger-formatted log files into a pre-defined schema. Troubleshooting and performance analysis proceeds from these database tables, e.g., through direct SQL queries, Python programs, or with the "R" statistical language.

The specific numbers of jobs for each of the other nine sites are listed in Table 2. The number of Pegasus jobs varies since it includes transfer job failures, which varied among sites. We used a single glidein job to claim all 800 processors. Since the maximum job length allowed in the main queue on Mercury is 25 hours, we resubmitted the glidein request daily to reclaim the processors.

Average wall-time for each workflow was 18.3 ± 3.9 hours. In total, approximately 90 GB of output was produced in 7.5 million individual data files. There were a total of 4,307 job failures, or approximately 0.06% of the total jobs. All but one occurred on stageout. This increases our confidence that the segmentation faults we saw on Abe were due to operating system memory management issues, rather than application code problems.

The execution time by job type for all nine sites is shown in Table 3.

Table 3: Runtime per job type

Job type	Hours	% of total time
Directory creation	0.16	0.0003
Registration	0.82	0.002
Transfer	150.42	0.3
Extraction	2423.72	4.5
Synthesis	50443.52	92.7
PSA	1212.23	2.2
Zip jobs	1028.05	0.3
Total	54420.77	100.0

The Pegasus jobs have been broken down further, into directory creation, registration (in the Globus RLS), and transfer jobs. Clearly, the majority of time is spent in synthesis jobs. This suggests examining both the workflow environment and the synthesis C code itself for optimization opportunities.

9. Conclusions

The CyberShake computational platform uses high-throughput grid computing to produce a new and important type of probabilistic seismic hazard curve. Full 3D waveform modeling is currently not in general use by PSHA researchers because, until now, it has been considered too computationally expensive and computationally challenging to perform. Through the work we are performing on CyberShake, we hope to bring the use of full 3D waveform modeling into regular use within the seismological community by showing that the computational challenges can be met and that CyberShake is a practical technology. CyberShake is an outstanding example of how advances in high performance computing provide opportunities for improved scientific information.

Through the process of designing and executing CyberShake, we made a series of optimizations enabling us to run end-to-end workflows of over 800,000 jobs in 18 hours on 800 processors. Over the course of three weeks, a single individual was able to execute over 7.5 million jobs using grid computing. There is still potential for further improvement. Approximately one third of the end-to-end time is overhead, suggesting further optimizations may be possible. We plan to continue the optimization process as we progress with new CyberShake sites, including further modifications of the clustering factor, staggered DAX start times, and adjustments to the Condor scheduler parameters.

The improvements suggested in this paper have application outside of CyberShake. Many of our developments are applicable to a wide range of embarrassingly parallel, high throughput computing applications. As more simulations and more resource providers support this kind of HPC, there will likely be continued investigation into optimization for the execution of large-scale embarrassingly parallel scientific workflows.

Acknowledgments

This work was partially supported by the National Center for Supercomputing Applications under TG-MCA03S012 (SCEC PetaScale Research: An Earthquake System Science Approach to Physics-based Seismic Hazard Research) and utilized NCSA's Mercury cluster. This work was supported in part by the Center for High Performance Computing and Communications at the University of Southern California. The Southern California Earthquake Center is funded by NSF Cooperative Agreement EAR-0106924 and USGS Cooperative Agreement 02HQAG0008. The SCEC contribution number for this paper is 1237. This work was also supported by NSF under OCI-0722019 and OCI-0749313.

References

[1] "Southern California Earthquake Center," http://www.scec.org.

[2] Jordan, T.H., P. Maechling, P. "The SCEC Community Modeling Environment -- An Information Infrastructure for System-Level Earthquake Science." *Seismol. Res. Lett., 74* (3), pp 324-328.

[3] Zhao, L., 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, 96* (5), pp 1753-1763.

[4] Deelman, E., et al. "Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance tracking: The CyberShake Example." *IEEE e-Science and Grid Computing 2006*, Amsterdam, Netherlands, December 4-6 2006.

[5] 2007 Working Group on California Earthquake Probabilities, 2008, The Uniform California Earthquake Rupture Forecast, Version 2 (UCERF 2): U.S. Geological Survey Open-File Report 2007-1437 and California Geological Survey Special Report 203 [http://pubs.usgs.gov/of/2007/1437/].

[6] "TeraGrid," http://www.teragrid.org.

[7] "Condor project homepage," http://www.cs.wisc.edu/condor.

[8] "Pegasus," http://pegasus.isi.edu

[9] Deelman, E., et al. "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems." *Scientific Programming Journal*, 2005. 13(3): p. 219-237.

[10] Deelman, E., C. Kesselman, and G. Mehta. "Transformation Catalog Design for GriPhyN." Technical Report, GriPhyN-2001-17, 2001.

[11] Chervenak, A., et al. "Giggle: A Framework for Constructing Sclable Replica Location Services." *Proceedings of Supercomputing 2002* (SC2002). 2002.

[12] Voeckler, J., Mehta, G., Zhao, Y., Deelman, E., Wilde., M. Kickstarting Remote Applications. Presented at *GCE06 Second International Workshop on Grid Computing Environments*, Tampa, Florida.

[13] Allcock, W., et al. "Secure, Efficient Data Transport and Replica Management for High-Performance Data-Intensive Computing." *Mass Storage Conference*. 2001.

[14] Graves, R. "Physics Based Probabilistic Seismic Hazard Calculations for Southern California." *14th World Conference on Earthquake Engineering*, Beijing, China, Oct 12-17 2008.

[15] Tierney, B. and D. Gunter. "NetLogger: A Toolkit for Distributed System Performance Tuning and Debugging." *Proceedings of the 8th IFIP/IEEE International Symposium on Integrated Network Management*, 2003.