In Proceedings of the e-Science 2006 Conference in Amsterdam, the Netherlands, Dec. 2006
Best Paper Award

Managing Large-Scale Workflow Execution from Resource Provisioning to Provenance tracking: The CyberShake Example

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

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

Abstract

This paper discusses the process of building an environment where large-scale, complex, scientific analysis can be scheduled onto a heterogeneous collection of computational and storage resources. The example application is the Southern California Earthquake Center (SCEC) CyberShake project, an analysis designed to compute probabilistic seismic hazard curves for sites in the Los Angeles area. We explain which software tools were used to build to the system, describe their functionality and interactions. We show the results of running the CyberShake analysis that included over 250,000 jobs using resources available through SCEC and the TeraGrid.

1. Introduction

Increasingly, Grid infrastructure is becoming available on a production basis [1-3]. Deployed as part of the general purpose computing infrastructure, the services and capabilities being provided are generally shared across many users from a variety of communities. One of the compelling motivations for the creation of the Grid infrastructure is that it provides a foundation on which distributed collaborations, or Virtual Organizations (VO)[4] may be formed.

One notable example of a VO is the Southern California Earthquake Center (SCEC)[5] an organization that consists of 50 earth science research institutions in the United States and several other countries. As a VO, SCEC has created its own production Grid infrastructure that enables sharing of resources owned by participants across the collaboration [6]. However, there are problems of great interest to the SCEC community that requires capabilities far beyond those that can be brought to bear by the participating institutions. One example is the calculation of Probabilistic Seismic Hazard Analysis (PSHA) maps, which indicate the likelihood of seeing a specific amount of surface motion within a specified period of time. PSHA is used (among other things) to determine the placement and design of buildings and other structures. To date, PSHA curves have been calculated using simplistic empirically-derived attenuation relationship to describe the relationship between a rupture and the surface motion. However, advances in a variety of geophysical models related to earthquakes have for the first time made it possible to calculate these models based on physics-based earthquake wave propagation simulations.

The requirements of new calculations go well beyond the computational and storage capacities of the SCEC member institutions, thus significant fractions of the calculations must be outsourced to high-end facilities such as the TeraGrid[3] in order to be completed in an acceptable time frame.

The PSHA calculations are quite complex, and involve many different processing steps, models and data sources. While all of these components could be re-hosted on the large-scale computational resources, this imposes a significant burden on the application community and greatly limits the flexibility in which computational sites may be used. A more effective approach is to integrate the external resources into the SCEC Grid, essentially augmenting the SCEC VO's resources and capabilities with the additional resources and services required to operate the VO.

Overlaying a VO-specific structure on the deployed Grid infrastructure can achieve a number of goals:

- Integration of external services with those owned by the community. This provides a uniform view of the community resources to the community members and to the applications.
- Removal of the "noise" of resources and services that is not relevant to the community.
- Deployment of domain-specific services that are part of the community infrastructure. This simplifies the application development and supports the enforcement of community policies. Additionally, deploying domain-specific services such as schedulers can potentially improve the throughput of the applications.

In this paper we describe how we constructed a SCEC-specific environment optimized to the calculation of PSHA by combining the infrastructure provided by the TeraGrid and the SCEC resources. We outline how we built the VO-based environment and provide results from running PSHA calculations in this environment. In this environment the workflows can be executed end-to-end the derived data can be automatically staged-out to long term storage, and provenance information can be recorded. First, we describe PSHA calculations and their computational requirements (Section 2). Next we detail the approach we took in building up the SCEC VO environment (Section 3). In Section 4 we describe the results obtained from running PSHA calculations on a variety of resources. We show results of running approximately 250,000 jobs corresponding to 1.8 CPU years over a period of 23 days. These jobs produced and registered approximately 100,000 files. Finally, we conclude with related work and future directions.

2. CyberShake and its Requirements

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. These seismic simulations are leading toward a physics-based, model-oriented approach to earthquake science with the eventual goal of transforming seismology into a predictive science with forecasting capabilities similar to those available today in the areas of climate modeling and weather forecasting.

To characterize the earthquake hazards in a region, seismologists and engineers utilize the Probabilistic Seismic Hazard Analysis technique. PSHA attempts to quantify the peak ground motions from all possible earthquakes that might affect a particular site (geographic area) and to establish the probabilities that the site will experience a given ground motion level over a particular time frame. PSHA information is used by city planners and building engineers and is often the basis for building codes in a region.

Up to now PSHA techniques have not fully integrated the recent advances in earthquake simulation capabilities. Probabilistic seismic hazard curves have been calculated using empirically-based attenuation relationships that represent relatively simple analytical models based on the regression of observed data. However, it is widely believed that significant improvements in PSHA will rely on the use of more physics-based waveform modeling. The goal of the CyberShake project is to bring the physics-based modeling for PSHA calculation into the mainstream.

For each site of interest (geographical area) a CyberShake calculation consists of the following steps:

1. Generation of Strain Green Tensors (SGTs)—this is an MPI-based finite-difference simulation that can run over a period of days using 144 or 288 processors and produces as much as 10TB of data.
2. Calculation of synthetic seismograms from SGT data calculated in Step 1 for each likely earthquake in the region. Depending on the rupture forecasts, there can as many as 35,000 and 5,000 on average of such calculations. Seismogram calculation runs on a single processor, and each calculation for each rupture can run for as little as a few minutes and as long as 3 days.
3. Calculation of spectral acceleration. This processes the seismograms into some peak intensity measure type of interest, in our case spectral acceleration. The number of these tasks corresponds to the number of tasks in step 2. However, these are short running tasks—on the order of minutes
4. Generation of hazard curves which combine the spectral acceleration values into a probabilistic hazard curve based on their likelihood of occurring..

This series of operations will produce a single PSHA curve which describes hazards at a particular site. In order to produce hazard maps for the Ventura and Los Angeles counties in California, at least 1,000 and up to 10,000 sites need to be calculated. Thus, CyberShake requires significant amounts of large high-performance computing resources in order to support the highly parallel, MPI-based SGT code. There is also a need for a high-throughput computing platform for the second and third phases of the analysis in which a large number of processing jobs must be submitted and monitored across a distributed computing environment.

Up to now SCEC relied mostly on its own resources-a high performance cluster at USC to perform the calculations. Although the cluster is large, over 1,800 processors [7], a single user such as SCEC can have access to only small fractions of the resources

at any one time. Additional burden is placed on the resources by the TeraByte-scale data sets. Thus, it is essential for SCEC to draw upon external resources in order to support the desired computations.

A third requirement imposed by the CyberShake is a significant file-oriented, data management capability. The post-processing phase (steps 2 and 3 above) is data intensive generating tens or hundreds of thousands of files. All files must be tracked, and managed, and annotated with metadata.

In the next Section we describe how we provided an environment for CyberShake that seamlessly integrated SCEC and outside resources and supported the automated management of both computation and data: from resource provisioning, to resource scheduling; from data registration to provenance recording.

3. Approach

3.1. Problem Representation

CyberShake can be represented as an ordered set of tasks described as a workflow and shown in Figure 1. Our initial calculations were limited to a small number of sites so as to focus on scientific validation of the results, and consequently the workflows were limited to seismogram calculation with SGT calculations being done ahead of time. With validation complete, we need to run several hundred sites, and the workflow is being augmented to include SGTs. Although the current workflow conceptually consists of only two main steps (steps 2 and 3 in Section 2), we plan to expand the workflow description to include all of the analysis. Each geographical site which is the target of the CyberShake analysis consists of tens of workflows each composed of between 11 and 1,000 of the two-step analysis components. In this paper we present results from running CyberShake for two sites: Pasadena and the University of Southern California (USC). The Pasadena site contained a total of 33 workflows and the USC site 26 workflows. The cumulative number of synthetic seismograms and spectral acceleration generation tasks for each site was on the order of 50,000.

CyberShake workflows were expressed in abstract terms in that they described only the analysis that needed to take place and the data needed in abstract terms without indicating either the execution locations or the actual storage location of the data. The VO we designed and implemented took care of provisioning the necessary resources, mapping the abstract workflow onto the VO resources, executing the necessary tasks and cataloging the results and provenance information.

3.2. SCEC-centric Grid

Executing large-scale, long running applications on today's Grid systems is a challenge as the resources are often shared across many users and applications. From the point of view of the application flexibility, it is important to provide a view that allows for the design of the analysis in terms independent of the underlying resources with the software used to manage VO resources performing the mapping onto the resources and the management of the execution.

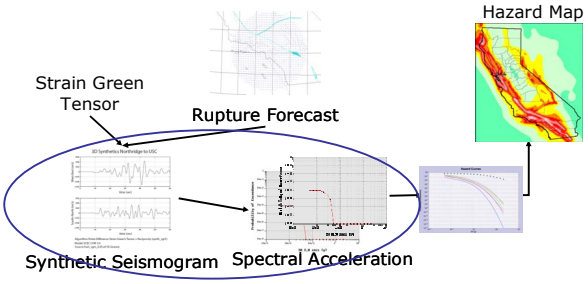

Figure 1: The CyberShake Workflow.

Figure 2 depicts the infrastructure we designed to support the execution of SCEC workflows. The infrastructure relies on heterogeneous resources drawn from the SCEC and TeraGrid resources: both storage and compute. The information about the resources available to the VO and other services such as data movement, replica management, and job scheduling services are maintained in the *VO Service Catalog*.

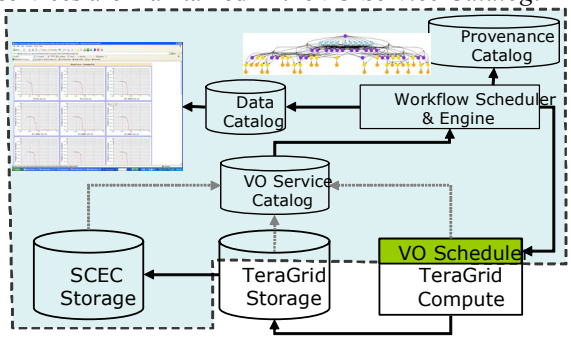

Figure 2: The SCEC Virtual Organization.

To support the VO operation, we deployed a VO-specific workflow scheduler and engine. This scheduler uses a workflow refinement system to augment workflow tasks with data management and catalog operations and maps tasks to specific resources dynamically provisioned into the VO. The workflow scheduler uses a VO specific execution engine based on Condor DAGMan [8] to execute and manage the mapped tasks. The tasks are sent for execution to a VO scheduler that manages the VO resources. As a result of the workflow execution, the newly created data products are stored in the Data Catalogs. The provenance information related to the data generation

and the workflow execution itself are stored in the *Provenance Catalog*.

In order to design and deploy the system we relied on existing Grid technologies. In some cases we integrated some of the components to automate the process of workflow management. We describe the software involved and the integration process below. We also designed and implemented SCEC-specific components that provided feedback to the scientists on the progress of the analysis (in the top left of Figure 2 we see a web-based display that generated and updated the hazard curves in real time.)

SCEC Catalogs

In this Section we describe the catalogs that were used by the SCEC VO and their implementation.

VO Service Catalog

The VO Service Catalog maintains information about the resources available to SCEC. Among the information included are the resources names, the jobmanagers [9] that allow for remote job submission, the GridFTP[10] servers that can be used to stage data in and out of the resource, the Replica Location Service (RLS)[11] which is used to maintain information about the data location, and other information. The VO Service Catalog, which is implemented by the Pegasus site (resource) catalog [13] is usually populated using the Globus Monitoring and Discovery Service (MDS) [14] or in absence of that service a static file can be provided. Having dynamic MDS information allows the Workflow Scheduler to make job placement decisions based on the current system conditions. In absence of this information static scheduling is used.

Data Catalog

The function of the Data Catalog is to store information about the newly created data products. Regardless of where the data was created, whether on the TeraGrid or on the SCEC resources, the data is moved back automatically to the SCEC permanent storage. Thus the Data Catalog needs to maintain information about the content of the data files (metadata) and locations of the files. In the SCEC VO the function of the Data Catalog is played by two systems, the Metadata Catalog Service (MCS) [15] and the Replica Location Service. MCS contains information about application-specific metadata attributes such as the site name, rupture information, and many others. MCS identifies the application files using a location independent logical filename. The RLS maintains mappings between the logical filenames and the physical locations of the files.

Provenance Catalog

Metadata is not the only information needed to ascertain the quality of the data. When the data is generated as the result of a variety of processing, it is important to know exactly how the data was obtained, which input data was used, where the computations have taken place and any other relevant information. This provenance is being stored in the Provenance Tracking Catalog (PTC) [16]. In addition to the provenance information, the catalog also maintains performance information related to the execution of the workflow tasks, such as runtime, memory used, etc. The results presented in this paper have been obtained via mining of the PTC.

Workflow Refinement and Execution

As we mentioned in Section 3.1 the initial workflow is represented in an abstract way, independent of the execution and data resources. The role of the workflow refinement is to transform the abstract workflow into an executable form. This transformation is performed based on the available compute resources and the location of the data.

As we mentioned in Section 2, our CyberShake workflows require 10TB of input data and some of the computations can run over a period of several days. In order to simplify the process of application level scheduling of VO-specific tasks, and isolate the VO from the underlying resource management policies, we provisioned resources into the VO, allocating them ahead of time and turning them over to the VO-level scheduler for mapping of the workflow. Thus in our work we included the resource provisioning process as part of the overall workflow management.

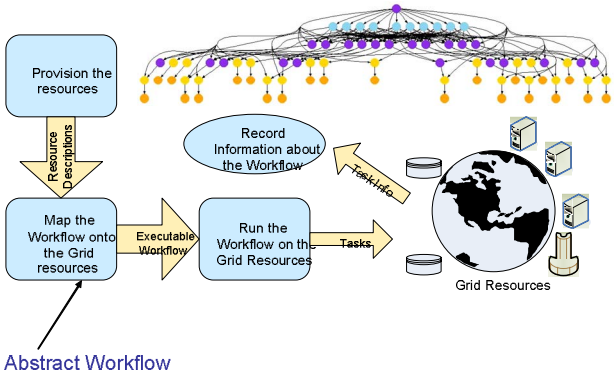

Figure 3: Workflow refinement, execution and resource provisioning.

Figure 3 depicts the end-to-end workflow management process. First, the resources are provisioned making them exclusively available to the VO to schedule and use in accordance to its policy. The act of provisioning isolates the members of the VO from the details of the local resource management, and impact of other users [17]. Based on the resource

descriptions obtained through the provisioning process and published into the VO Service Catalog the abstract workflow is mapped onto these resources. The resulting executable workflow is given to the workflow engine for execution on the specified resources. After the successful execution, the data, metadata and provenance information are stored in the VO data catalogs. The top right side of Figure 3 shows a small executable workflow. Later in this section we describe the executable workflow in more detail.

Workflow Provisioning

We employed two methods of provisioning resources: through advance reservations and using Condor Glide-ins [18]. In the first case we were granted reservations for a certain number of processors for a given amount of time. Along with the reservations we also obtained special queues to use for the scheduling. In the case of Condor Glide-ins we submitted (via Condor-G) as many as 100 Glide-ins at any one time to the regular TeraGrid queues. The Glide-ins were essentially place holders. When they were executed on the target system, they reported back to the condor pool that scheduled them, effectively becoming part of the initial condor pool. This dynamic pool of resources that can be composed of heterogeneous resources could then be used by Pegasus as a target for the workflow mapping process. We have to note that the resources obtained via Glide-ins had the same limitations as the jobs scheduled through the same remote queues. For example if a queue on the TeraGrid had a maximum wall time of 1 day, the Glide-ins and thus the dynamically provisioned resources were available for a maximum of 1 day.

To maintain uniformity in the system, we also scheduled Glide-ins to the reserved resources. As a result the workflow management system mapped the abstract workflows to a heterogeneous pool of resources, managed by the VO-scheduler as described below. The information about the resource types and availability were published in the VO Service Catalog.

Workflow Refinement

Once the resources were provisioned and made available via a dynamic Condor pool, the workflow mapping and scheduling process could begin. Currently the interaction between the provisioning process and the workflow refinement and execution is not automated and is the subject of future work.

We used Pegasus [19-22] to perform the mapping of the abstract CyberShake workflows onto the available resources. Pegasus performs a series of workflow refinement operations that enable the individual application components to run on a variety of resources. To conduct the refinement Pegasus consults the VO Service Catalog to find what resources are available and what their characteristics are. Given this set of target resources, Pegasus determines where the workflow components can run and uses simple scheduling heuristics (random, round-robin or min-min) to select the desired resources. In addition to the execution site, Pegasus also selects which data (possibly replicated) to transfer to the execution site. It consults the Replica Location Service to find the locations of the data. Obviously priority is given to the data co-located with the computational resources. Once the resources are identified, Pegasus generates a workflow that can be executed by a workflow execution system such as Condor's DAGMan [8]. The executable workflow includes directives for staging data in and out of computations, for the setup of the remote execution system such as isolated execution directories and the necessary environmental variables. When Pegasus targets DAGMan as a workflow execution engine is writes out the executable workflow in the form of Condor submit files.

Workflow Execution

The executable workflow is given to Condor DAGMan for execution. DAGMan follows the dependencies in the workflow and releases jobs into the local Condor queue as their dependencies are satisfied. As jobs progress through the queue they are being scheduled to the pool of provisioned resources (by the VO-Scheduler—here Condor). Since the resources in the pool are heterogeneous and distributed geographically Pegasus indicates resource preferences (for example the NCSA TeraGrid) by providing appropriate Class-Ads [23] in the Condor submit files. In cases of failures we used the DAGMan retry mechanism to retry a failed workflow node and the Rescue DAGs generated by DAGMan when it could no longer continue with the workflow tasks. The rescue DAG contained the jobs that remained to be executed.

VO Scheduler

The VO Scheduler manages the pool of resources that are available to the VO. The In our case the scheduler is implemented as a Condor scheduler that is responsible for a Condor pool. As we mentioned before we use provisioning via Condor Glide-ins to provision resources from the TeraGrid. Once the Glide-ins start running, they report back to the VO Scheduler and are thus made part of the pool of resources managed by the scheduler. The workflow refiner uses this pool and the scheduler as the target for mapping the workflows. Since the resources were provisioned ahead of time, when a job in the workflow is submitted, it can start running immediately without incurring any further overheads thus reducing the overall runtime of the workflow[24].

Provenance Tracking

When dealing with large amounts of data it is essential to keep track of how the data was generated and where. Additional information also encompasses performance information. In this work we used a code wrapper named *kickstart* to manage the local execution of the application codes—launching them, capturing the exit status and runtime information. Kickstart is one of the tools in the Virtual Data System [25]. The information provided by *kickstart* was then automatically stored in the Provenance Tracking Catalog and was mined to produce the results shown in the next section.

4. Results

In the subsequent results, we used the following set up. The resources were provisioned from the TeraGrid nodes at NCSA and SDSC using Glide-ins and via reservations. We obtained reservations for 11 nodes at SDSC starting on November 2nd 2005 for a duration of 9.5 days. Additional 125 nodes were reserved at NCSA starting on November 3rd for a period of 7 days.

We provisioned as many as 100 TeraGrid nodes into the VO resource pool. The abstract workflows were submitted for refinement and execution to a SCEC host. The execution resources were drawn from the VO resource pool. Although we had the capability of running individual workflows across multiple sites, because of the size of the input data sets, running a single workflow at more than one site was not efficient.

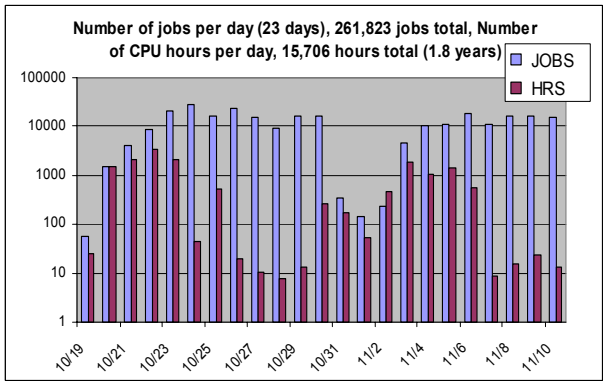

Figure 4: Number of jobs run per day on the TeraGrid resources.

Figure 4 shows the distribution of jobs over the 23 days of our initial computations that were used for validation purposes. The fist bar shows the total number of jobs run on a particular day and the second bar shows the total number of CPU hours that the jobs took. During this analysis run we calculated the hazard curves for two SCEC sites: Pasadena and USC. As can be seen in the figure, the initial jobs were of long duration (the number of jobs is proportional to the number of hours the jobs ran). As time progressed we see that the number of jobs was disproportionately greater than the runtime of the jobs (towards 10/25/05). This shows that the execution of the data management jobs such as the data stage-out and data registration jobs are a dominant factor. These jobs are numerous but short running. Then, (towards 11/1/05) we see again that the number of jobs is proportional to their runtime. At this point the application jobs (with long execution times) are running again. This also indicates the ramp up for the calculations for the second SCEC site—USC. Towards the end of the graph, we see again the short running data management tasks. In summary, over the period of 23 days, the system executed 261,823 jobs whose cumulative runtime corresponded to 1.8CPU years. This throughput is possible only because we were able to provision resources ahead of time and minimize the overheads one normally incurs in a queue management system such as PBS or LSF.

Figure 5 gives the breakdown of the jobs run. The first two bars refer to the application codes. There were over 87,000 application jobs run. There were significantly more data management tasks performed than application jobs. Approximately 170,500 tasks transferred the final and intermediate results to the SCEC storage systems and registered these data in the Replica Location Service.

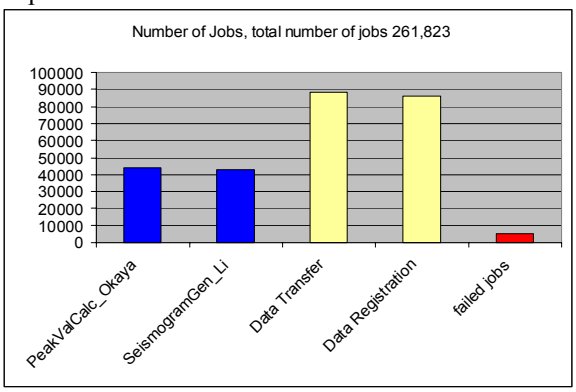

Figure 5: Type of jobs run.

There were also almost 5,300 failed jobs. These failures were due mostly to overloading of the GridFTP servers (too many simultaneous requests) and due to RLS server time outs. Both kinds of failures were dealt with by automatically retrying the task execution (using the Condor retry feature) and/or by submitting the rescue DAG. Figure 6 shows another view of the data. The graph indicates the number of seismogram jobs (the longest running application jobs) that ran for a given time interval. We can see that there were mostly short running jobs. Approximately 35,000 jobs ran for less than 10 CPU minutes. 7,300 jobs ran

between 20 and 220 CPU minutes. There were also 2 jobs that ran close to 3 CPU days.

Figure 7 shows the distribution of jobs across the execution sites. The first bar shows the total number of jobs at a given site and the second bar indicates the cumulative amount of time (in CPU days) the jobs ran at the sites. We can see that most of the jobs were run locally (on the SCEC submit machine) and at NCSA. Although approximately 100,000 jobs ran locally, their runtime was short as these were replica registration jobs. The application tasks ran both at NCSA and SDSC TeraGrid sites.

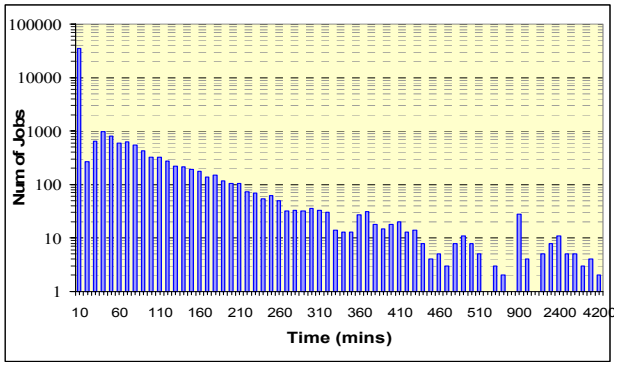

Figure 6: The distribution of seismogram jobs.

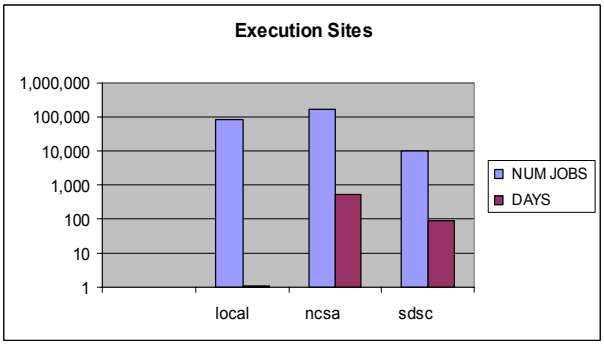

Figure 7: Number of jobs run on the TeraGrid and local sites. The graph also shows the cumulative runtime of the jobs on the sites (in CPU days).

Our results are not solely confined to runtimes on the TeraGrid. As part of the collaboration between the application and computer scientists we were able to pinpoint the inefficiencies in the application code. As a result we have recently been able to improve the data storage requirements by reducing the amount of space needed for the SGTs from 10TB to 250GB! This was done by selectively saving SGT data for only fault surfaces rather than the full SGT simulation volume. We also saw an improvement from 24,000 to 11,000 and then all the way down to 560 CPU hours in runtime for the post processing jobs by reducing the need to read large amounts of non-relevant SGT data.

5. Related Work

There are two main areas of related work. One deals with managing large-scale workflows in the Grid environment and the other with developing VO-based services. The work described in this paper bridges these two categories.

Kepler [26] is a data-driven workflow system where the user can author data processing steps as a network of predefined workflow components called 'actors' and use 'directors' for describing execution models. Triana [27] allows the user to specify execution behavior easily by supporting an abstract layer for Grid computing called GAP, a subset of the GridLab GAT[28]. Taverna [29] is an authoring and execution tool from the myGrid project [30]. myGrid exploits semantic web technology to support data intensive bioinformatics experiments in a grid environment. ICENI [31] provides a component based Grid middleware. Users can construct an abstract workflow from a set of workflow components and the system generates a concrete workflow using its scheduler. GridAnt [32] is a client-side workflow system that assists users to express and control execution sequences and test Grid services. GridFlow [33] provides user portal and services of both global grid workflow management and local grid sub-workflow scheduling. Unicore [34] provides a programming environment where users can design and execute job flows with advanced flow controls. Gridshell [35] transparently incorporates useful distributed and grid computing concepts into the UNIX shell login environment.

Our work differs from others in that our workflow system presents a more comprehensive grid-based environment for scientific workflows. It integrates resource provisioning and comprehensive data, metadata and provenance tracking capabilities with workflow management. This type of integration is not present in other systems.

In terms of the development of VO-based services, some of the most closely related work are the Open Science Grid[36] Edge Services [37]. These services are deployed on the boundary of a private/public network on a particular resource. The services can include scheduling and data management services that can manage the execution environment for the applications. Although there are plans for implementation, edge services are not currently being used by applications. On the other hand our VO-based services are used on a daily basis by the SCEC scientist to perform the CyberShake analysis.

6. Conclusions and Future Work

In this paper we described the VO and workflow-based approach to running larger-scale SCEC applications on the Grid. We drew upon SCEC and TeraGrid resources to deliver the needed computing and storage capacities. From the point of view of the SCEC application, the execution environment presented under the umbrella of a VO was treated uniformly. Using our approach we have automated a significant fraction of the CyberShake analysis and started on the path of reaching the scientific goal of producing a detailed hazard map of the Los Angeles area. The solution is not solely tied to SCEC but can be employed by any number of applications.

Much work still needs to be done. In particular we are expanding the CyberShake workflow to include the initial SGT calculations. This will only increase the requirements on the workflow management system. We also plan to smooth the transition between the provisioning and workflow mapping phases. As we continue to improve the system, we are proceeding in running the newly improved application codes with the aim of reaching the scientific goals.

7. Acknowledgments

This work was support by the SCEC Community Modeling Environment Project which is funded 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 supported in part by the Southern California Earthquake Center (SCEC). SCEC is funded by NSF Cooperative Agreement EAR-0106924 and USGS Cooperative Agreement 02HQAG0008. The SCEC contribution number for this paper is 952. Pegasus is being distributed as part of the GriPhyN Virtual Data System (vds.isi.edu). The authors would like to thank the administrators of the TeraGrid and the USC HPC cluster for the help with reservations and for the use of their resources.

8. References

[1] I. Foster et al., "The Grid2003 Production Grid: Principles and Practice," HPDC 2004.

[2] "Enabling Grids for eScience in Europe," http://public.eu-egee.org, 2005.

[3] TeraGrid www.teragrid.org, 2005.

[4] I. Foster, et al, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," IJHPCA, vol. 15, pp. 200-222, 2001.

[5] "Southern California Earthquake Center (SCEC)," 2004 http://www.scec.org/

[6] P. Maechling, et al. "Simplifying Construction of Complex Workflows for Non-Expert Users of the Southern California Earthquake Center Community Modelling Environment," *ACM SIGMOD Record, special issue on Scientific Workflows.*, 2005.

[7] K. Durkin, "USC Supercomputer Breaks Barrier," 2005 www.usc.edu/uscnews/stories/11846.html

[8] DAGMAN,www.cs.wisc.edu/condor, 2005.

[9] Globus Project, "Globus Resource Allocation Manager (GRAM)," 2002.

[10] B. Allcock, et al. "The Globus Striped GridFTP Framework and Server," SC'2005.

[11] A. Chervenak, et al. "Giggle: A Framework for Constructing Scalable Replica Location Services.," SC2002.

[12] A. Chervenak, et al "Performance and Scalability of a Replica Location Service," HPDC, 2004.

[13] "Site Catalog Administration," 2005 vds.uchicago.edu/vds/doc/userguide/

[14] K. Czajkowski, et al, "Grid Information Services for Distributed Resource Sharing," HPDC, 2001.

[15] G. Singh, et al. "A Metadata Catalog Service for Data Intensive Applications," SC, 2003.

[16] I. Foster, et al. "Chimera: A Virtual Data System for Representing, Querying, and Automating Data Derivation," SSDBM 2002.

[17] K. Czajkowski, et al, "Agreement-Based Resource Management," *Proc. of the IEEE*, vol. 93, 2005.

[18] "Condor website," http://www.cs.wisc.edu/condor/

[19] E. Deelman, et al "Workflow Management in GriPhyN," in *Grid Resource Mgmt*, 2003.

[20] E. Deelman, et al, "Pegasus: a Framework for Mapping Complex Scientific Workflows onto Distributed Systems," *Scientific Prog.*, (13) 3 2005.

[21] E. Deelman, et al "Mapping Abstract Complex Workflows onto Grid Environments," *Journal of Grid Computing*, vol. 1, pp. 25-39, 2003.

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

[23] R. Raman, et al "Policy Driven Heterogeneous Resource Co-Allocation with Gangmatching," HPDC 2003.

[24] G. Singh, et al "Optimizing Grid-Based Workflow Execution," *Journal of Grid Computing*.

[25] "Virtual Data System," 2006 http://vds.isi.edu

[26] I. Altintas, et al "Kepler: An Extensible System for Design and Execution of Scientific Workflows," SSDBM, 2004.

[27] I. Taylor, et al. "Distributed P2P Computing within Triana: A Galaxy Visualization Test Case.," IPDPS 2003.

[28] E. Seidel, et al "Gridlab: A Grid Application Toolkit and Testbed," GCS vol18, 2002.

[29] T. Oinn, et al, "Taverna: a tool for the composition and enactment of bioinformatics workflows.," *Bioinformatics* vol. 20, 2004.

[30] R. D. Stevens, et al "myGrid: personalised bioinformatics on the information grid," *Bioinformatics*, vol. 19, 2003.

[31] S. Newhouse., "ICENI: An integrated Grid Middleware to enable e-Science.," The 2nd Annual RealityGrid Workshop., 2004.

[32] G. v. Laszewski, et al "GridAnt – Client Side Grid Workflow Management with Ant," 2003 www-unix.globus.org/cog/projects/gridant/gridant-whitepaper.pdf

[33] J. Cao, et al "GridFlow: WorkFlow Management for Grid Computing," CCGRID, 2003.

[34] UNICORE, "Uniform Access to Computing Resources," http://www.unicore.org, 2002.

[35] E. Walker et al, "Orchestrating and Coordinating HPDC, 2004.

[36] "Open Science Grid," www.opensciencegrid.org, 2005.

[37] K. Keahey, "Edge Services Framework for OSG," OSG Technical document OSG-doc-167-v1, 2005.