rvGAHP – Push-Based Job Submission using Reverse SSH Connections

Scott Callaghan
University of Southern California
Los Angeles, California
scottcal@usc.edu

Gideon Juve
USC Information Sciences Institute
Marina Del Rey, California
gideon@isi.edu

Karan Vahi
USC Information Sciences Institute
Marina Del Rey, California
vahi@isi.edu

Philip J. Maechling
University of Southern California
Los Angeles, California
maechlin@usc.edu

Thomas H. Jordan
University of Southern California
Los Angeles, California
tjordan@usc.edu

Ewa Deelman
USC Information Sciences Institute
Marina Del Rey, California
deelman@isi.edu

ABSTRACT

Computational science researchers running large-scale scientific workflow applications often want to run their workflows on the largest available compute systems to improve time to solution. Workflow tools used in distributed, heterogeneous, high performance computing environments typically rely on either a push-based or a pull-based approach for resource provisioning from these compute systems. However, many large clusters have moved to two-factor authentication for job submission, making traditional automated push-based job submission impossible. On the other hand, pull-based approaches such as pilot jobs may lead to increased complexity and a reduction in node-hour efficiency. In this paper, we describe a new, efficient approach based on HTCondor-G called reverse GAHP (rvGAHP) that allows us to push jobs using reverse SSH submissions with better efficiency than pull-based methods. We successfully used this approach to perform a large probabilistic seismic hazard analysis study using SCEC's CyberShake workflow in March 2017 on the Titan Cray XK7 hybrid system at Oak Ridge National Laboratory.

CCS CONCEPTS

• **Computer systems organization** → **Grid computing**; Client-server architectures; • **Computing methodologies** → *Distributed computing methodologies*; • **Applied computing** → *Earth and atmospheric sciences*;

KEYWORDS

scientific workflows, remote job submission, resource provisioning, seismic hazard analysis

ACM Reference Format:
Scott Callaghan, Gideon Juve, Karan Vahi, Philip J. Maechling, Thomas H. Jordan, and Ewa Deelman. 2017. rvGAHP – Push-Based Job Submission using Reverse SSH Connections. In *WORKS'17: WORKS'17: 12th Workshop on Workflows in Support of Large-Scale Science, November 12–17, 2017, Denver, CO, USA*. ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3150994.3151003

WORKS'17, November 12–17, 2017, Denver, CO, USA

ACM ISBN 978-1-4503-5129-4/17/11…$15.00
https://doi.org/10.1145/3150994.3151003

1 INTRODUCTION

Modern scientific applications typically require the execution of multiple codes, may include computational and data dependencies, and contain varied computational models ranging from a bag of tasks to a monolithic parallel code. These applications are often complex software suites with substantial computational and data requirements, especially as simulations migrate from an individual researcher's desktop to departmental, university, and national cluster resources.

To assist with execution, many applications have turned to scientific workflow tools [9, 17]. These tools provide features, such as automation, staging of data, resource provisioning, and management and capture of metadata and provenance, which enable scientific workflow applications to be run more efficiently and be easily reproducible. As a natural consequence of the streamlining workflow tools provide, application developers look to increase the size of their simulations in order to address cutting-edge science questions. Running at larger scales necessarily requires support for scientific workflows on larger systems.

Typically, workflow tools perform resource provisioning using either push-based requests, which originate from the workflow scheduler, such as over SSH [20] or GRAM [12], or pull-based approaches [14] [4], which originate from the remote resource. However, both of these approaches have limitations. Some clusters, for security reasons, do not permit SSH keys or X.509 proxy authentication. Pull-based approaches often result in increased overhead and complexity, since resource provisioning may not match demand and jobs may have heterogeneous computational requirements.

To address these concerns, we have developed a new technique, reverse GAHP (rvGAHP). rvGAHP is an implementation of GAHP (Grid ASCII Helper Protocol), a text-based protocol created by HTCondor [18]. It enables communication between a client and a grid or cloud service, which enables push-based resource provisioning even on systems without support for automated authentication methods. Using this technique, the Southern California Earthquake Center (SCEC) [3] was able to perform a month-long series of simulations using CyberShake, a seismic hazard analysis scientific

workflow application, on the Titan [2] Cray XK7 supercomputer at Oak Ridge National Laboratory.

The rest of the paper is organized as follows. Section 2 provides an overview of the SCEC CyberShake science application, its computing requirements, and the workflow tools it uses. Section 3 describes the two approaches used in the past by CyberShake to provision resources on remote supercomputing clusters, and their limitations on Titan. Section 4 explains our new approach, rvGAHP, that allows us to push jobs to remote clusters that require two factor authentication using Condor-G by employing reverse SSH connections. Section 5 describes the results and our experiences of using rvGAHP for a CyberShake study performed in March 2017, utilizing resources from Titan and NCSA Blue Waters [1]. Limitations are presented in Section 6, related work is discussed in Section 7, and the paper is concluded in Section 8. 5g

2 APPLICATION DESCRIPTION

2.1 Science Overview

Seismologists quantify the seismic hazard for a location using probabilistic seismic hazard analysis (PSHA) [6]. PSHA is a technique for estimating the probability that earthquake ground motions at a location of interest will exceed a given level of some intensity measure (IM), such as peak ground velocity or peak ground acceleration, over a certain time period. PSHA results are useful for civic planners, building engineers, and insurance agencies, and impact billions of dollars a year in construction through building codes.

PSHA estimates are communicated through hazard curves, which relate ground motion intensities to probability of exceedance for a single site of interest. Figure 1 shows a representative hazard curve for downtown Los Angeles. From this curve, we could determine that this site has a 2 percent chance in 50 years of experiencing 0.4g of acceleration, a common probability level for building engineering. A set of hazard curves for geographically distributed locations can be interpolated to produce a hazard map for a region by holding either probability or ground motion constant and plotting the other as a function of location (Figure 2).

PSHA estimates are often produced using ground motion prediction equations (GMPEs). These empirically-derived attenuation relationships are based on historical data and often include site-specific parameters to capture local velocity structure, which may amplify or dampen ground motions. Although computationally inexpensive, attenuation relationships have difficulty capturing complex physical effects, such as basin amplification and rupture directivity, which may have large impacts on ground motions. To accurately include these effects, simulation-based approaches are needed. Figure 1 compares the CyberShake simulation-based results (red triangles) to four common GMPEs (dashed lines).

CyberShake [13], developed by SCEC, performs three-dimensional physics-based deterministic PSHA. After initial input files are generated, a velocity mesh is produced and populated with material properties. This mesh is input to an anelastic wave propagation code which generates a wavefield of Strain Green Tensors (SGTs). Next, seismic reciprocity is performed using the SGTs to simulate about 500,000 earthquakes per site of interest, generating seismograms for each event [21]. The resulting seismograms are processed

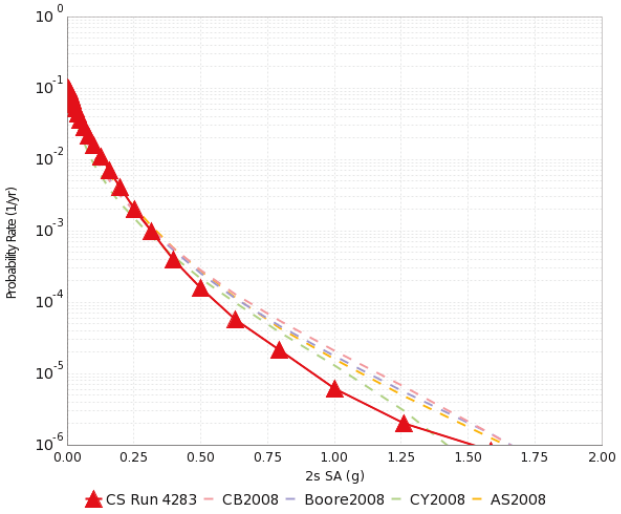

Figure 1: CyberShake hazard curve for downtown Los Angeles (solid red curve) compared with hazard curves at the same site from four Next Generation Attenuation empirical GMPEs (dashed lines).

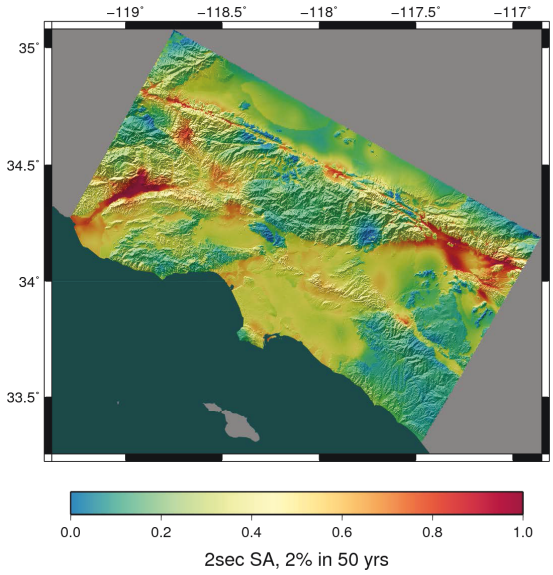

Figure 2: CyberShake hazard map for Southern California, showing the spectral accelerations at a 2-second period exceeded with a probability of 2% in 50 years.

to obtain intensity measures, which are combined with the individual earthquake probabilities from an earthquake rupture forecast, UCERF2 [15], to obtain a hazard curve. Individual PSHA results from hundreds of locations are interpolated to produce a hazard map.

Table 1: CyberShake Computational Requirements (per site)

Stage	Code Name	Code Type	Output Data	Titan Node Hours
Mesh Creation	UCVM	MPI CPU	120 GB	90
SGT Generation	AWP-ODC-SGT	MPI GPU	1500 GB	1490
Seismogram Synthesis	DirectSynth	MPI CPU	30 GB	2440
Other stages		Sequential	<1 GB	10
Total			1650 GB	4030

By using a physics-based approach like CyberShake, we are able to include 3D effects such as rupture directivity and basin amplification which lead to much of the hazard variability seen in Figure 2.

Optimizing and automating CyberShake calculations is key to enable simulations like those required for Figure 2 to be performed in a reasonable makespan (end-to-end runtime of a workflow).

2.2 Technical Overview

CyberShake requires significant computational resources to perform PSHA. An overview of the major software components and their computational and data requirements for generating PSHA results for a single site, using OLCF's Titan, is given in Table 1.

The CyberShake workflow is illustrated in Figure 3. Conceptually, we divide the CyberShake workflow into two parts: SGT generation and post-processing. In the SGT generation part, input files are constructed and passed to the mesh generator, UCVM[16]. This MPI code typically runs on 100 nodes and creates a seismic velocity mesh of 5–10 billion points. Next, a serial code is run to produce another set of input files, which, combined with the mesh, are used by the SGT generator, AWP-ODC-SGT. AWP-ODC-SGT is a modified version of AWP-ODC, an extreme-scale anelastic wave propagation code developed within the SCEC community [7]. This code shows speedup of a factor of 6 on GPUs for CyberShake simulations and is typically run on 100-800 GPU nodes, depending on problem size. AWP-ODC-SGT produces the SGTs needed by the post-processing.

The post-processing part of the workflow is dominated by the seismogram synthesis code DirectSynth. DirectSynth is an MPI job, which typically runs on about 200 nodes. It uses a master-worker paradigm, with the master handing out tasks to workers, and data handlers sending SGT data via MPI messages to workers upon request. This job produces seismograms and other intensity measure files as output, which are staged back to SCEC storage for database insertion, generation of hazard curves and maps, and archiving.

Overall, calculating PSHA results for a single location requires approximately 4000 node-hours and 1650 GB of data. To produce a hazard map requires PSHA results for 200-400 sites, which can result in a makespan of several weeks.

2.3 Workflow Tools

The heterogeneous computational requirements, data management needs, and automation required by CyberShake calculations have led us to use scientific workflow tools; specifically, a software stack, which includes Pegasus-WMS, HTCondor, and the Globus Toolkit.

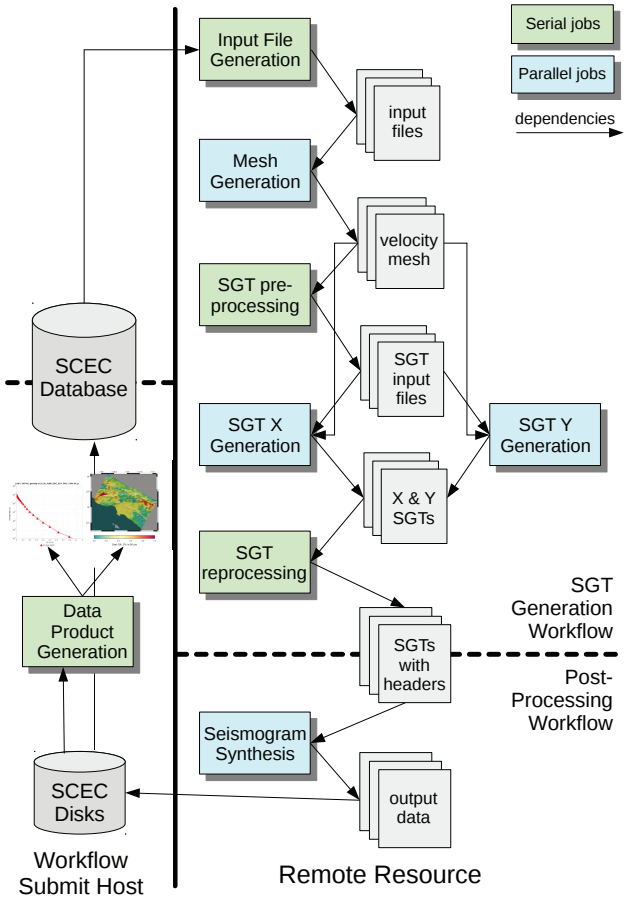

Figure 3: Schematic overview of the CyberShake SGT and post-processing workflows.

Pegasus-WMS [10] is a workflow management system, which enables users to create a high-level (abstract) description of their workflow, capturing the files, tasks, and dependencies between them. Pegasus augments this abstract workflow with metadata capture and data transfer jobs, performs graph level optimizations for improved performance and efficiency, and plans it for execution on specific target resources, generating a Directed Acyclic Graph (DAG) and job description files designed for use with HTCondor. HTCondor [19] is a workload management system, which is used by CyberShake to orchestrate workflow execution, manage real-time

dependencies, and provide error handling. In particular, we heavily use HTCondor's Directed Acyclic Graph Manager (DAGMan), which, given a DAG generated by Pegasus, can queue and execute tasks in the DAG while respecting dependencies. To submit jobs to remote resources, we use the HTCondor GridManager to submit jobs to Globus. The Globus Toolkit [11] enables communication over the grid. For CyberShake we rely especially on two components: the GRAM protocol for communication between the workflow submission host on a SCEC server and the remote execution system, and GridFTP for file transfer between systems. Since migrating to a workflow-based approach using these workflow tools in 2007, CyberShake has been successfully run for more than 100 million core-hours on nine different clusters.

To meet SCEC's science goals of improving and expanding physics-based PSHA, we targeted Titan at Oak Ridge National Laboratory, the 4th-ranked system on the June 2017 TOP500 list, as a system for CyberShake execution. Given the excellent performance of AWP-ODC-SGT on GPUs, Titan with over 18,000 GPU nodes is a logical fit for CyberShake. However, this naturally requires migration of the CyberShake scientific workflows to Titan. Due to security reasons, Titan does not permit users to use SSH keys, X.509 proxies, or any alternative method for authentication that is not two-factor. This, in turn, complicates many methods for performing automated resource provisioning and job submission. In the next section, we will discuss the limitations of currently existing resource provisioning approaches for scientific workflow applications on systems like Titan.

3 LIMITATIONS OF EXISTING SOLUTIONS

Workflow tool strategies for automated job submission can be roughly divided into two classes: 1) Resource requests which originate from the workflow submission host (push-based), and 2) requests which originate from the remote resource (pull-based).

3.1 Push-based

In push-based approaches, requests for resources are made on-demand from the runtime manager component of the workflow tool. As workflow tasks become eligible to run, resource requests are submitted to the remote system, which start up and execute the task. This is illustrated in Figure 4.

For example, in the CyberShake software stack, when jobs become eligible in the HTCondor queue on the workflow submission host, the HTCondor GridManager daemon submits the job to the remote Globus GRAM frontend, where it translates the Globus Resource Specification Language (RSL) [8] description of the job and requested resources into a scheduler submit script, and submits it to the scheduler on the remote resource.

Push-based approaches are an efficient way to acquire resources, since compute time is only consumed while workflow jobs are running. They can incur queuing delays, since the job is only submitted to the remote queue once there is work to do and must wait on the scheduler to run. Automated authentication from the workflow submission host to the remote system is required. Typically, this can be performed using approaches such as SSH keys or X.509 proxies, which do not require a human-in-the-loop. However, two-factor authentication is becoming more common, and some systems may not permit these kind of automated one-factor authentication approaches, Titan among them.

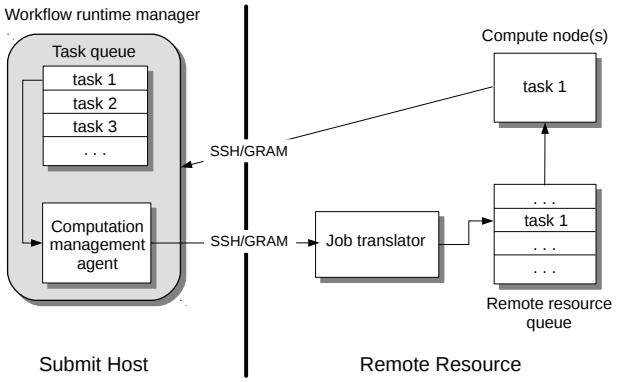

Figure 4: The push approach for resource provisioning.

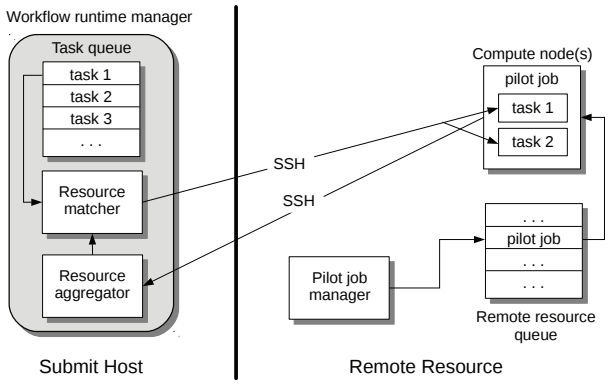

Figure 5: The pull approach for resource provisioning.

3.2 Pull-based

In pull-based approaches, requests for resources originate from the remote system itself. Typically, a long-running process on the remote system submits jobs to the remote scheduler, either periodically or in response to demand on the workflow submission host. Once these jobs start, a process notifies the workflow submission host that resources are available for scheduling. This is illustrated in Figure 5.

For example, in the CyberShake software stack, a long-running daemon running on the remote resource login node queries the HTCondor DAGMan workflow job queue and, when it finds eligible-to-run jobs, submits pilot jobs to the remote resource of a size and length commensurate with the workflow jobs waiting to run. When the pilot jobs start up, HTCondor processes call back to the HTCondor collector on the workflow submission host and advertise themselves. The HTCondor negotiator matches these available resources on the remote cluster with the eligible-to-run jobs on the submit host, and the workflow jobs begin execution on the remote resource pilot jobs. Other examples of pull-based

approaches are PanDA [14] and Work Queue [4] that dial home to a central server to retrieve the jobs that need to run.

Since the long-running daemon only needs to be started once, and all connections originate from the remote system, pull-based approaches have the advantage of not requiring automated authentication to the remote system. They do however require authentication to the workflow submission host. They offer additional flexibility, since multiple workflow jobs can be mapped to a single pilot job. However, since the resource request and the workflow job queue are separated, there is the potential for extra overhead to be introduced. Pilot job runtimes and sizes must be selected to account for the length of the workflow jobs, but since workflow jobs may end before the pilot jobs, or not require all the resources provisioned by the pilot job, resources may be consumed without accomplishing any workflow work.

In 2015, SCEC performed a CyberShake study (Study 15.4), using Titan for the SGT generation part of the workflow, and NCSA Blue Waters, with its support for push-based resource provisioning, for the post-processing part of the workflow. To enable CyberShake workflows to run on Titan, we used a pull-based pilot job approach like the one outlined above. We ran a long-running pilot job daemon on the head node, which periodically checked the HTCondor queue on the submit host to determine when new jobs were eligible to run. If new jobs were found, the pilot job daemon submitted requests for pilot jobs to the Titan scheduler. Although the SGT generation part of the CyberShake workflows is dominated computationally by AWP-ODC-SGT, it only accounts for about 25% of the SGT workflow makespan. Therefore, requesting a pilot job wide enough to run AWP-ODC-SGT, but long enough to accommodate the other tasks, would result in ~75% of the computational resources sitting idle. To avoid this, we submitted requests for four pilot jobs of varying sizes, with dependencies between them enforced via qsub (the queue submission command on Titan). Workflow jobs were matched with their corresponding pilot jobs using the PILOT_JOB_TYPE string. Overall, this approach was effective in accomplishing the CyberShake study, but with high overhead. By comparing the node-hours consumed by workflow jobs (runtime times nodes) to the node-hours charged on Titan, we discovered our resource utilization – the percentage of node-hours burned that were actually used for performing workflow work, as opposed to overhead – was only 68%. Because of these inefficiencies, we were not able to perform as much of the study on Titan as originally planned. In our experience, it is challenging to use pull-based resource provisioning efficiently for workflows with heterogeneous tasks.

4 REVERSE GAHP - rvGAHP

As described in the previous section, the pull-based approach is not ideal for running large scale workflows with heterogeneous tasks because of the possible mismatch between the resources acquired by the pilot jobs, and the resources required by the workflow jobs that are ready to run at a particular time on the workflow submit host. In turn this mismatch can reduce the efficiency of resource usage. The push-based approach has the drawback of queuing delays, but that is often preferable over a pull-based approach when running workflows that consume hundreds of thousands of CPU hours for which compute efficiency is of paramount concern.

For the CyberShake SGT workflows, the heterogeneous nature of the workflow jobs mean that pilot jobs cannot be submitted before workflow jobs are queued without wasting large quantities of resources. For example, if an 800-node pilot job is queued and begins execution before an AWP-ODC-SGT job is in the HTCondor queue, most of those nodes will sit idle, since other kinds of workflow jobs are not wide enough to fill it. To avoid running pilot jobs with no work assigned, our solution for CyberShake Study 15.4 was to only submit pilot jobs when there were idle workflow jobs in the HTCondor queue - but this negates the queuing advantage of pilot jobs. Given that, we wanted to design a push-based solution which could run on systems like Titan.

In the push-based approach we rely on HTCondor-G to push jobs to a remote resource using either GRAM or via SSH. HTCondor-G runs on the workflow submit host, and launches a HTCondor-G GridManager process whenever it detects a job in the local HTCondor queue that needs to be pushed to a remote resource. If there are no running jobs in the queue, HTCondor-G automatically shuts down the daemon. The GridManager process submits jobs either via

(1) Globus GRAM - connects to a Globus GRAM front end on the remote cluster using X.509 authentication
(2) SSH - GridManager opens a SSH tunnel [20] to the remote cluster and starts a remote BLAHP daemon, and then submits jobs to the remote BLAHP daemon.

In both approaches, the daemon picks up credentials and uses them without any user intervention. However, this is not possible for systems like Titan that require two-factor token based authentication, where the token is a time dependent RSA token that has to entered by the user when the connection is opened by HTCondor GridManager. In order to get around these technical limitations and avoid standing up a listening service on Titan headnode such as GRAM, we implemented a new version of remote_gahp for HTCondor, called rvGAHP. rvGAHP does not require listening services on the remote resource. Instead, it uses a "reverse" SSH connection from the remote resource to the submit host to establish communication between the local GridManager running on the workflow submit host and the GAHP process running on the remote resource. The key benefit of this approach is that it enables remote job submission without requiring the remote resource to:

(1) run services that accept incoming network connections
(2) accept SSH connections without two-factor authentication (e.g. RSA tokens), which are sometimes prohibited by security policy.

Our solution is illustrated in Figure 6.

The rvGAHP server running on the remote resource uses SSH to establish a secure connection to the submit host. The SSH connection starts a proxy process that listens on a UNIX domain socket for requests. This SSH connection remains open all the time, even when there are no jobs running. If the SSH session gets disconnected, the server immediately reestablishes the connection. When a remote GAHP job is submitted, the GridManager launches a client process to communicate with the GAHP servers. The client connects to the proxy through the local UNIX socket and sends the name of the GAHP to start (batch_gahp for job submission or condor_ft-gahp for file transfer). The proxy forwards the request to the server,

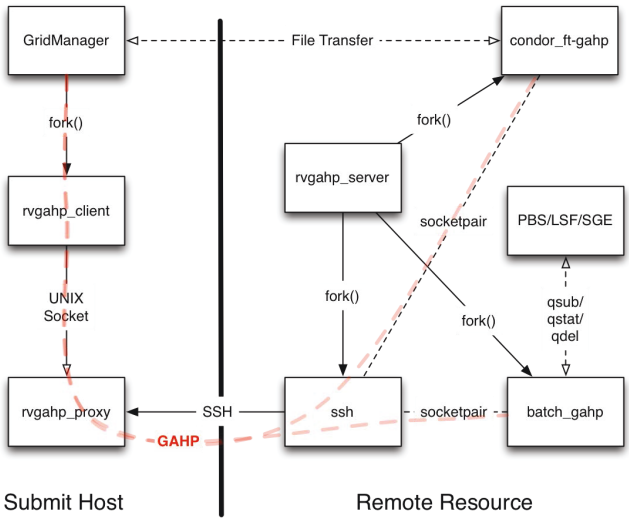

Figure 6: rvGAHP connection setup

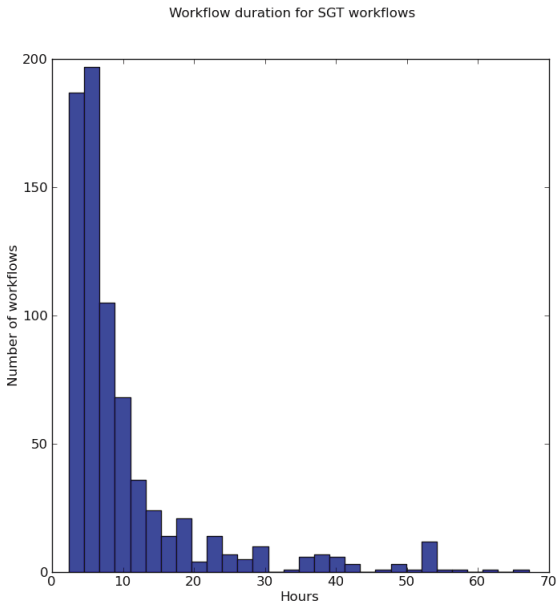

Figure 7: Histogram showing runtime of SGT-only CyberShake workflows on Titan, using rvGAHP.

which forks the appropriate GAHP and connects it to the SSH process using a socketpair. Once this is done, the server immediately establishes another SSH connection to the submit host, to prepare for additional jobs. The client copies its stdin from the GridManager to the proxy, and data from the proxy to stdout and back to the GridManager. The proxy passes data to the SSH process, and the SSH process passes data to the GAHP. Once all connections are established, the job execution proceeds. When the GridManager is done, the GAHP servers exit and the connections are torn down. As with HTCondor jobs in general, the job can request an arbitrary number of nodes and runtime, and specify any other parameters supported by the underlying job description language (in our case, Globus RSL) and the remote system scheduler.

Since rvGAHP is a push based approach, each workflow job is submitted as a separate job in the remote resource queue. This tight coupling leads to much improved resource utilization as compared to a pilot job-based approach, where the provisioning of the nodes from the remote resource is decoupled from the scheduling of the workflow jobs. To enable rvGAHP, users must install the server on the remote system in user space, the client and proxy on the submit host, and make a few modifications to the HTCondor configuration files. It is designed to be usable by anyone already using HTCondor.

5 APPLICATION STUDY

In March 2017, SCEC conducted a CyberShake study on Titan and Blue Waters. In order to run end-to-end CyberShake workflows with all processing stages on Titan, we deployed rvGAHP on Titan. The rvGAHP server daemon ran on a Titan login node and connected to the workflow submission host, running on a SCEC server. The daemon had to be restarted manually whenever the login node was rebooted, but this only happened four times in the course of the 31-day study.

Using rvGAHP, 13,334 jobs were submitted to the Titan queue, which consumed a total of 450,000 node-hours. These jobs ranged from 10 minutes to 9 hours and from 1 to 240 nodes. On average, 6 jobs were running on Titan on 669 nodes throughout the entire length of the study. All SGT workflows were run on Titan, along with 17% of the post-processing workflows. Figure 7 plots the runtime of the 736 SGT-only workflows, and Figure 8 plots the runtime of the 147 end-to-end (both SGT and post-processing) workflows. We can see that the runtimes of the workflows and the SGT component vary significantly, which caused poor resource utilization for the pull-based approach. Being able to use Titan extensively, combined with the push-based rvGAHP approach enabled SCEC to reach its science goal, and improved our resource utilization to 97%, representing a savings of 130,000 node-hours over the previous pilot job approach. The average delay per job (sum of workflow tools overhead and remote queue time) [5] dropped from 9.2 hours to 0.6 hours, indicating that for CyberShake the rvGAHP approach was much better at acquiring timely resources than pilot jobs.

6 LIMITATIONS

Our rvGAHP approach does have limitations. The long-running daemon must run somewhere on the remote resource, so the remote resource must have policies and hardware that permit this, and the workflow submission host must allow single-factor authentication. Since a new GAHP process and SSH connection is initiated per workflow job, workflows which submit large numbers of jobs could encounter scaling issues, though many systems enforce a limit on the maximum number of jobs in the queue. Additionally, we encountered a complication when transferring X.509 proxies. Even though rvGAHP spawns the HTCondor file transfer GAHP on the remote cluster, and that allows us to retrieve the job stdout and stderr on job completion back on the submit host, we noticed that if the job proxy was set to be transferred with the job using the

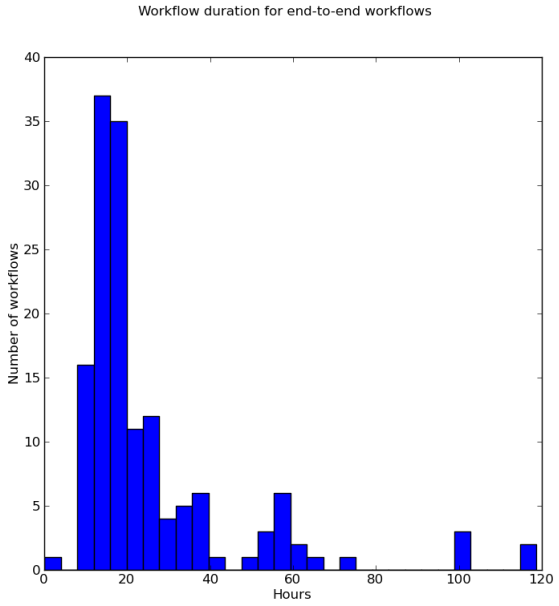

Figure 8: Histogram showing runtime of end-to-end (both SGT and post-processing) CyberShake workflows on Titan, using rvGAHP.

x509userproxy HTCondor submit script attribute, the job submit on the remote end failed and the job was never queued on Titan. We reverted to transferring the proxy via globus-url-copy to resolve the issue.

7 RELATED WORK

Our work on rvGAHP is most closely related to the BOSCO work [20] that allows users to grab opportunistic resources using SSH, using HTCondor-G. Our work has benefited from the changes made to HTCondor-G and GridManager to enable SSH job submissions to remote clusters. The key contribution and differentiation in our work is how the underlying SSH connections are setup. This allows us to submit jobs to remote resources that require two-factor authentication, which cannot be achieved with BOSCO for reasons outlined earlier in the paper. The Atlas project has deployed PanDA [14] on OLCF Titan and uses a pull-based approach for submitting jobs to Titan. The PanDA approach is similar to the approach that we deployed for CyberShake (described in 3.2) for Study 15.4, and as such is not an efficient option for CyberShake, suffering from the same efficiency issues that we saw with pilot jobs. Our motivation for moving to rvGAHP was driven by a desire to achieve higher node utilization of our allocation, and the characteristics of our workload, which made it a poor fit for pull-based approaches.

8 CONCLUSION

We have presented a new approach called rvGAHP, which extends the traditional SSH push-based approach to include remote resources that require two factor authentication, previously a barrier to push-based techniques. rvGAHP was successfully deployed for a large 31-day CyberShake study conducted in March–April 2017. Using this new approach, we were able to improve our utilization of node hours from 68% to 97% compared to our previous pull-based approach using pilot jobs.

While the focus of this paper is on running CyberShake on Titan, other similar heterogeneous workloads are common in the scientific community, as there is often a need for smaller-scale pre- and post-processing surrounding one or more large parallel jobs where the majority of computational work is performed. The advantages provided by rvGAHP over current solutions – increased efficiency over pull-based approaches and access to systems currently off-limits to push-based job submission – are broadly applicable to workflows with heterogeneous resource requirements. Use of rvGAHP will continue to keep the largest open-science systems accessible to scientific workflows, encouraging application developers to scale up their workflows and tackle novel scientific research problems.

9 ACKNOWLEDGEMENTS

CyberShake workflow research was supported by the National Science Foundation (NSF) under the OAC SI2-SSI grant #1148493, the OAC SI2-SSI grant #1450451, and EAR grant #1226343. This research was supported by the Southern California Earthquake Center (SCEC Contribution No. 7540). SCEC is funded by NSF Cooperative Agreement EAR-1033462 & USGS Cooperative Agreement G12AC20038.

Computational support was provided by the Innovative and Novel Computational Impact on Theory and Experiment (INCITE) program. This research used resources at the Oak Ridge Leadership Computing Facility, a DOE Office of Science User Facility supported under Contract DE-AC05-00OR22725. This research is part of the Blue Waters sustained-petascale computing project, which is supported by the NSF (awards OCI-0725070 and ACI-1238993) and the state of Illinois. Blue Waters is a joint effort of the University of Illinois at Urbana-Champaign and its National Center for Supercomputing Applications. Computational support was provided by NSF Petascale Computing Resource Allocations (PRAC) awards OAC-0832698, ACI-1440085, and OAC-1713792. Additional computational support was provided by the Center for High Performance Computing at the University of Southern California.

Pegasus is currently funded by the National Science Foundation(NSF) under the OAC SI2-SSI #grant 1664162. Previously, NSF has funded Pegasus under OCI SDCI program grant #0722019 and OCI SI2-SSI program grant #1148515. This work was also supported by NSF CIF21 DIBBs program grant #1443047.

REFERENCES

[1] [n. d.]. Blue Waters: Sustained Petascale Computing. ([n. d.]). https://bluewaters.ncsa.illinois.edu/
[2] [n. d.]. Introducing Titan: Advancing Era of Accelerated Computing. ([n. d.]). https://www.olcf.ornl.gov/titan/
[3] [n. d.]. Southern California Earthquake Center. ([n. d.]). http://www.scec.org
[4] Peter Bui, Dinesh Rajan, Badi Abdul-Wahid, Jesus Izaguirre, and Douglas Thain. 2011. Work queue+ python: A framework for scalable scientific ensemble applications. In *Workshop on python for high performance and scientific computing at sc11*.
[5] Scott Callaghan, Philip Maechling, Patrick Small, Kevin Milner, Gideon Juve, Thomas H. Jordan, Ewa Deelman, Gaurang Mehta, Karan Vahi, Dan Gunter, Keith Beattie, and Christopher Brooks. 2011. Metrics for Heterogeneous Scientific Workflows: A Case Study of an Earthquake Science Application. *International Journal of High Performance Computing Applications* 24, 3 (August 2011), 274–285.
[6] C. Allin Cornell. 1968. Engineering seismic risk analysis. *Bulletin of the Seismological Society of America* 58, 5 (1968), 1583–1606.

[7] Yifeng Cui, Efecan Poyraz, Jun Zhou, Scott Callaghan, Phil Maechling, Thomas H. Jordan, Liwen Shih, and Po Chen. 2013. Accelerating CyberShake Calculations on XE6/XK7 Platforms of Blue Waters. In *Proceedings of Extreme Scaling Workshop 2013*.

[8] Karl Czajkowski, Ian T. Foster, Nicholas T. Karonis, Carl Kesselman, Stuart Martin, Warren Smith, and Steven Tuecke. 1998. A Resource Management Architecture for Metacomputing Systems. In *Proceedings of the Workshop on Job Scheduling Strategies for Parallel Processing*.

[9] Ewa Deelman, Dennis Gannon, Matthew Shields, and Ian Taylor. 2008. Workflows and e-Science: An Overview of Workflow System Features and Capabilities. *Future Generation Computer Systems* (July 2008). http://dx.doi.org/10.1016/j.future.2008.06.012

[10] Ewa Deelman, Karan Vahi, Gideon Juve, Mats Rynge, Scott Callaghan, Philip J Maechling, Rajiv Mayani, Weiwei Chen, Rafael Ferreira da Silva, Miron Livny, and Kent Wenger. 2015. Pegasus: a Workflow Management System for Science Automation. *Future Generation Computer Systems* 46 (2015), 17–35. https://doi.org/10.1016/j.future.2014.10.008 Funding Acknowledgements: NSF ACI SDCI 0722019, NSF ACI SI2-SSI 1148515 and NSF OCI-1053575.

[11] Ian Foster, Carl Kesselman, and Steven Tuecke. 2001. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International Journal of High Performance Computing Applications* 15, 3 (Aug. 2001), 200–222. https://doi.org/10.1177/109434200101500302

[12] James Frey, Todd Tannenbaum, Miron Livny, Ian Foster, and Steven Tuecke. 2001. Condor-G: A Computation Management Agent for Multi-Institutional Grids. In *Proceedings of the 10th IEEE International Symposium on High Performance Distributed Computing (HPDC '01)*. IEEE Computer Society, Washington, DC, USA, 55–. http://dl.acm.org/citation.cfm?id=874077.876491

[13] Robert Graves, Thomas Jordan, Scott Callaghan, Ewa Deelman, Edward Field, Gideon Juve, Carl Kesselman, Philip Maechling, Gaurang Mehta, Kevin Milner, David Okaya, Patrick Small, and Karan Vahi. 2011. CyberShake: A Physics-Based Seismic Hazard Model for Southern California. *Pure and Applied Geophysics* 168, 3-4 (2011), 367–381. https://doi.org/10.1007/s00024-010-0161-6

[14] Alexei Klimentov, A. Vaniachine, Kaushik De, Torre J. Wenaus, Sergey Panitkin, Dantong Yu, Gergely V. Záruba, and M. Titov. 2012. Abstract: PanDA: Next Generation Workload Management and Analysis System for Big Data. In *2012 SC Companion: High Performance Computing, Networking Storage and Analysis, Salt Lake City, UT, USA, November 10-16, 2012*. 1521–1522. https://doi.org/10.1109/SC.Companion.2012.301

[15] 2007 Working Group on California Earthquake Probabilities. 2008. The Uniform California Earthquake Rupture Forecast, Version 2. (2008). https://pubs.usgs.gov/of/2007/1437/

[16] Patrick Small, David Gill, Philip J. Maechling, Ricardo Taborda, Scott Callaghan, Thomas H. Jordan, Kim B. Olsen, Geoffrey P. Ely, and Christine Goulet. 2017. The SCEC Unified Community Velocity Model Software Framework. *Seismological Research Letters* 88, 5 (September 2017). https://doi.org/10.1785/0220170082

[17] Ian J. Taylor, Ewa Deelman, Dennis B. Gannon, and Matthew Shields. 2006. *Workflows for e-Science: Scientific Workflows for Grids*. Springer Publishing Company, Incorporated.

[18] Douglas Thain, Todd Tannenbaum, and Miron Livny. 2005. Distributed computing in practice: the Condor experience. *Concurrency - Practice and Experience* 17, 2-4 (2005), 323–356.

[19] Douglas Thain, Todd Tannenbaum, and Miron Livny. 2005. Distributed Computing in Practice: The Condor Experience. *Concurrency and Computation: Practice and Experience* 17, 2-4 (Feb. 2005), 323–356. https://doi.org/10.1002/cpe.v17:2/4

[20] Derek Weitzel, I Sfiligoi, B Bockelman, J Frey, Frank Wuerthwein, D Fraser, and D Swanson. 2014. Accessing opportunistic resources with bosco. In *Journal of Physics: Conference Series*, Vol. 513. IOP Publishing, 032105.

[21] Li Zhao, Po Chen, and Thomas H. Jordan. 2006. Strain Green's Tensors, Reciprocity, and Their Applications to Seismic Source and Structure Studies. *Bulletin of the Seismological Society of America* 96, 5 (October 2006), 1753–1763. https://doi.org/10.1785/0120050253