Experiences Using GlideinWMS and the Corral Frontend Across Cyberinfrastructures

Mats Rynge, Gideon Juve, Gaurang Mehta, Ewa Deelman
Information Sciences Institute
University of Southern California
Marina del Rey, CA, U.S.A.
{rynge, gideon, gmehta, deelman}@isi.edu

G. Bruce Berriman
Infrared Processing and Analysis Center
California Institute of Technology
Pasadena, CA, U.S.A.
gbb@ipac.caltech.edu

Krista Larson, Burt Holzman
Fermi National Accelerator Laboratory
Batavia, IL, U.S.A.
{klarson1, burt}@fnal.gov

Scott Callaghan
Southern California Earthquake Center
University of Southern California
Los Angeles, CA, U.S.A.
scottcal@usc.edu

Igor Sfiligoi, Frank Würthwein
Department of Physics
University of California, San Diego
San Diego, CA, U.S.A.
{isfiligoi, fkw}@ucsd.edu

***Abstract*: Even with Grid technologies, the main mode of access for the current High Performance Computing and High Throughput Computing infrastructures today is logging in via ssh. This mode of access locks scientists to particular machines as it is difficult to move the codes and environments between hosts. In this paper we show how switching the resource access mode to a Condor glidein-based overlay can bring together computational resources from multiple cyberinfrastructures. This approach provides scientists with a computational infrastructure anchored around the familiar environment of the desktop computer. Additionally, the approach enhances the reliability of applications and workflows by automatically rerouting jobs to functioning infrastructures. Two different science applications were used to demonstrate applicability, one from the field of astronomy and the other one from earth sciences. We demonstrate that a desktop computer is viable as a submit host and central manager for these kind of glidein overlays. However, issues of ease of use and security need to be considered.**

Keywords: Condor, glidein, GlideinWMS, Peridograms, CyberShake, Workflow, Corral, OSG, TeraGrid

I. INTRODUCTION

The computational requirements for many individual researchers and small research teams are usually much higher than what a desktop computer can provide, but much smaller than the requirements for major collaborations such as the global physics projects associated with the Large Hadron Collider [1]. When a researcher has reached the conclusion that more resources are required than are available locally, a common path forward is to turn to either department/campus resources, or to national cyberinfrastructures such as the Open Science Grid [2] or the TeraGrid [3], or a mix of these resource providers. The traditional model for accessing such resources is via a head node attached to the resource. In the case of Open Science Grid, there is no direct login access to the resources, but it is common to use a managed submit node, which, like a head node, is not local to the researcher. The interface to the resources usually consists of ssh/scp. To run a workload, the researcher has to copy input data sets to the resource, log in to a head node, set up an environment, and submit jobs to the local job scheduler. If the researcher wants to add or move to another resource, the steps have to be repeated for each additional resource.

In this paper we show how rather than moving code and data to remote resources, a researcher can have remote resources, departmental/campus clusters, and national cyberinfrastructures resources register to a local desktop computer in order for these resources to appear as a single large virtual resource, with the desktop computer serving as central manager and submit host. The advantages of using the desktop computer as the submit host are that the researcher is already familiar with the local environment, and the desktop computer is also potentially where the input data exists and the outputs should be stored. The reason for using resources from multiple cyberinfrastructures is flexibility. When everything is working fine, the scientific application will have more resources to run on and increase the overall application performance; but more importantly, running across multiple cyberinfrastructures protects the scientist from downtimes, technical problems, and allocation issues by automatically routing the jobs onto the functioning infrastructures.

Bringing these resources together is done using Condor glideins [4]. The idea of glideins, or more generally pilot jobs, and the concept of simultaneous computational overlays on top of distributed Grid and Cloud resources are not new [5][6]. In this paper we show how glidein technology, specifically as leveraged by GlideinWMS [7] with a new frontend named Corral, and the Pegasus Workflow Management System [8], are not only useful for large experiments but also small research teams and individual researchers with few local computational resources.

The paper is structured in the following way. First, we describe the cyberinfrastructures used for this demonstration, followed by an introduction to the existing capabilities of GlideinWMS, and the new capabilities that Corral introduces. We also describe two different scientific applications used in this paper, one from the field of astronomy and the other from earth sciences. We conclude with a discussion of lessons learned, related work, and conclusions.

II. Infrastructure differences between local resources, Open Science Grid and TeraGrid

Researchers have a choice of computational infrastructures, each with their own characteristics and requirements. In the following work, we used a Condor pool provided by a campus department, and multiple resources from the two major cyberinfrastructures in the U.S.: the Open Science Grid (OSG) and the TeraGrid. These national cyberinfrastructures share a large set of services and software stacks, but handle users, priorities and jobs differently. Ultimately, researchers only care about getting their science done, and that is what makes the glidein overlay model so useful; it provides a uniform view of all available resources, so that the barrier to using the various computing resources is lowered.

There are two key differences between the TeraGrid and the OSG: one is how they view their users (individuals versus communities), and another is how they allocate resources to the users. The Open Science Grid is based on virtual organizations (VOs) [9]), which are organizations for people with a common goal, potentially spanning administrative domains. In OSG, there are VOs to support projects, campus grids, domain sciences, and a few for users who do not fit in the larger VOs. Users can be members of one or more VOs, and resources are in the same way owned and operated by one or more VOs. Resource owners can (and most do) allow VOs other than the owner of the resource to use the resource on an opportunistic basis. For example, the Engagement VO, which helps new users get started, does not provide any computational resources. Instead, Engagement users get unused cycles from other VOs. The resource owners are free to set conditions on the acquisition of these cycles and almost all give opportunistic jobs lower priority than the owner VO's jobs. Some resources are configured to preempt opportunistic jobs if a higher priority job comes along. Additionally, with opportunistic use it is impossible to tell beforehand how many resources will be available for a run.

TeraGrid has neither the concept of VOs nor opportunistic use. Administratively, the whole TeraGrid is one VO, and scientists become members by submitting allocation proposals. Once accepted, the awarded service units can be used to "pay" for time on one of the TeraGrid resources. The allocation does not provide guaranteed access or turnaround times for jobs. The user is subject to resource priority policies and the current state of the job queue at the resource, and if the queue time is long, there is not much the user can do other than wait.

Another noteworthy difference is that generally, Open Science Grid and TeraGrid are built for different kinds of workloads. The Open Science Grid has mostly high throughput computing (HTC) jobs, which are usually serial jobs or small (in terms the level of parallelism) parallel codes. Most jobs on the TeraGrid are high performance computing (HPC) parallel codes, and the TeraGrid resources are designed and built with fast interconnects optimized for low latency communication between tasks in the same job. However, the HTC/HPC distinction of the two infrastructures is not strict. You can run parallel jobs on OSG and serial jobs on TeraGrid and as we will see in the CyberShake discussion later in the paper, for workflows with a mix of parallel and serial jobs, you can schedule the parallel jobs on the TeraGrid and the serial jobs on OSG. However, coordinating resources across the two cyberinfrastructures is challenging partly due to different submission interfaces, but most importantly because of different preferences set by the schedulers. In order to facilitate seamless access to these kinds of resources, we have been developing a system, GlideinWMS, which we describe next.

III. GlideinWMS

Glideins is the technique for on-demand dynamic provisioning of Condor job slots [4]. Some services of the Condor pool, such as the *collector*, the *schedd* and *negotiator* (job submit/management daemons) are statically deployed with a fixed and accessible network location. The *startd* daemon provisions the job slots of the Condor system. A glidein is fundamentally the *startd* daemon and configuration packaged into a job and sent to a remote compute resource. Once the glidein starts, the *startd* daemon registers with the collector, and the result is a Condor pool with the ability to dynamically grow and shrink. Glidein-based frameworks such as GlideinWMS manage the glidein provisioning and decide how and when to grow or shrink the pool based on the current workload demand.

GlideinWMS was initially developed to meet the needs of the CMS (Compact Muon Solenoid) experiment [10] at the Large Hadron Collider (LHC) at CERN. It has been in production across the Worldwide LHC Computing Grid (WLCG), with a large deployment operated by the Open Science Grid. GlideinWMS has been used in production with more than 15,000 concurrently running jobs; the CMS use alone totals over 29 million CPU hours over the last two years. GlideinWMS has a distributed design, with the main components being a frontend and a factory. The frontend queries the Condor job queue (where the user submits their jobs) and determines if the pool needs to grow. If an increase is desired, the frontend then sends a request to the factory to provision more glideins. The factory uses standard Grid interfaces to start glideins on remote resources. The experiments in this paper used a factory installed on a testbed, but for most production users, the factory component is

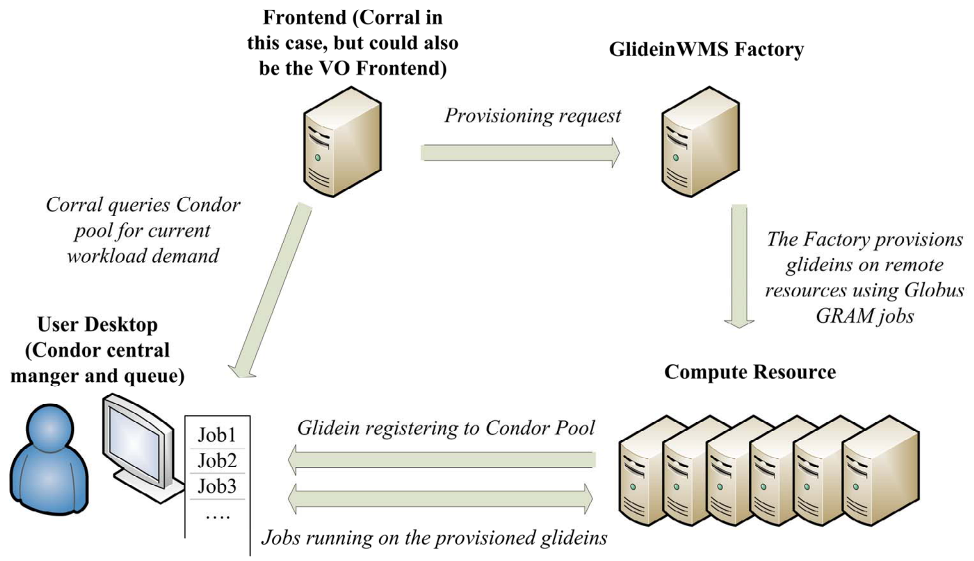
Frontend (Corral in this case, but could also be the VO Frontend)
GlideinWMS Factory
Provisioning request
Corral queries Condor pool for current workload demand
The Factory provisions glideins on remote resources using Globus GRAM jobs
User Desktop (Condor central manger and queue)
Compute Resource
Job1
Job2
Job3
….
Glidein registering to Condor Pool
Jobs running on the provisioned glideins

Figure 1. GlideinWMS with Corral Frontend

provided as software as a Service (SaaS), hosted at University of California, San Diego.

GlideinWMS comes with two frontends, the VO Frontend and Corral. The VO Frontend has been the main frontend for GlideinWMS since the beginning of the project. It was principally developed for use on OSG. The VO Frontend has VO-specific features, such as the ability to reuse glideins for users in the same VO, and provides a scalable system supporting a large number of users in production. Current users include the particle physics experiments CMS [10], CDF [11], and DZero [12], the nuclear experiment GlueX [13], the gravitational wave experiment LIGO [14], the structural biology research communities SBGrid [15] and NEBioGrid [16], the nanotechnology research community NanoHUB [17], the campus grid communities of the University of Nebraska (HCC) [18], University of Wisconsin (GLOW) [19] and University of California San Diego (UCSDGrid), the Northwest Indiana Computational Grid (NWICG) [20], and the OSG Engagement VO [21].

IV. Corral – A New GlideinWMS Frontend

Corral [22] was initially developed as a standalone glidein provisioning system and was extensively used by Pegasus users who needed glideins, either due to short running jobs or a mixed HTC/HPC workload (see CyberShake section below). In 2009, Corral was repurposed as a GlideinWMS frontend. This allowed Corral to leverage the GlideinWMS Factory functionality and scalability, but at the same time remain a simple frontend for use by individual users.

Compared with the VO Frontend, the Corral Frontend lacks the concept of VOs. This is most clearly apparent in the X.509 credential handling. The VO Frontend often submits the glideins with a service (as opposed to personal) certificate, thus allowing the glideins to be reused between members of the same VO. The Corral Frontend uses the individual's personal credential to submit the glideins, and the glideins in turn are locked to the user requesting them. The glideins are configured to not be sharable because that would use a personal certificate in a way, which is not allowed by computing infrastructures. This constraint makes the Corral Frontend less efficient in the multi-user setup, but enables the use of GlideinWMS outside a VO-centric environment. Most importantly, this flexibility allows Corral to acquire a mix of resources with different user/group mappings when running across cyberinfrastructure.

The Corral Frontend also has some new features, which are not yet implemented in the VO Frontend. One is the *multislot request*, which groups a set of glidein requests into a single bigger request. For example, if the current application or workflow needs 512 glideins, without the multislot functionality, the frontend would send the request to the Factory, which would then submit 512 individual Globus GRAM jobs to the resource to start the 512 glideins. Since many large HPC resources have limits on how many jobs a user can have in the remote job scheduler's queue, the multiple single core requests would not be successful. Sometimes HPC resources put a limit on the number of running jobs; sometimes on the number of total jobs in the queue. There are also resources which give higher priority to jobs with larger core count requests. With multislot, the frontend submits only a single request to the Factory for 512 glideins. The Factory would then submit 1 Globus GRAM job to the resource, configured to request 512 cores, and start up the glideins accordingly. By carefully selecting the size of multislot requests, a user can ensure that the glidein pool is still dynamic without exceeding the remote queue limits.

V. Example Applications

The following experiments demonstrate the use and applicability of GlideinWMS/Corral running two different applications across the national cyberinfrastructure. The first application is from the Southern California Earthquake Center [23], and the second from the NASA Exoplanet Science Institute, Infrared Processing and Analysis Center [24]. Both applications are workflow-based and managed by the Pegasus Workflow Management System (WMS).

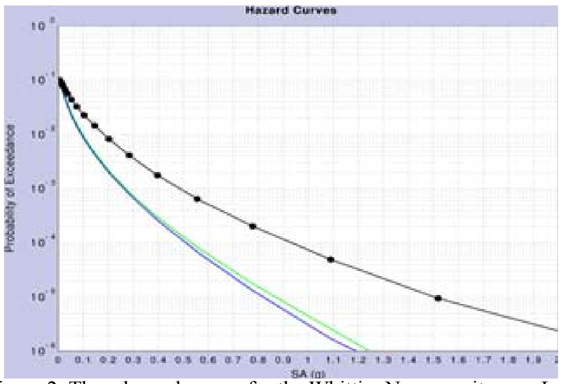

Figure 2. Three hazard curves for the Whittier Narrows site near Los Angeles. The black line is the hazard curve calculated with the physics-based CyberShake simulation. The blue and green lines are hazard curves calculated using two common attenuation relations

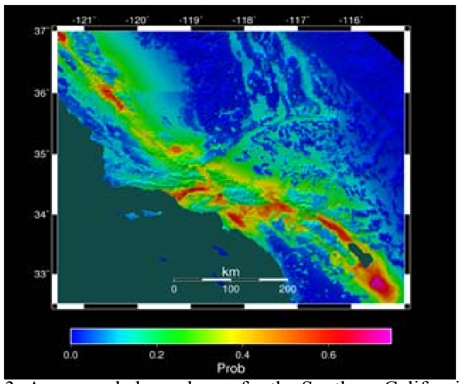

Figure 3. An example hazard map for the Southern California area generated using attenuation relationships. Colors show the probability that ground motions will exceed 0.1 g (acceleration due to gravity) from 2007 to 2057

The researcher's desktop was used as a submit host and Condor central manager. It had a standard Condor install with two local configuration changes. The first change is to enable GSI (Grid Security Infrastructure) X.509 authentication and authorization. This is needed as GlideinWMS uses GSI for communication between the glideins and the central manager. The second configuration change added 10 Condor slave collectors, which are additional instances of the Condor *collector* daemon (a default install only has one). The glideins are configured to register with the slave collectors, and then the slave collectors forward the glidein entries to the main collector. This is necessary due to the longer latencies associated with establishing trusted GSI connections with a singly-threaded Condor collector. By having 10 slave collectors we can have a large number of glideins register to randomly chosen slave collectors at the same time without affecting the performance of the main collector. Smaller setups can use the main collector directly, while bigger setups require a larger number of slave collectors. The firewall on the desktop needs to accommodate a range of incoming ports, and because of this, even though the changes are not major, we strongly suggest that the desktop machine should be configured and maintained by professional IT staff, where possible.

The resulting overlay environment looks and behaves almost just like any other Condor pool, and it is equally important to consider both the heterogeneity of the base operating systems and software stacks.

A. Southern California Earthquake Center / CyberShake

The Southern California Earthquake Center (SCEC)[23] performs earthquake system science research to improve the understanding of earthquakes and reduce their risk. One way to quantify seismic hazard for a location is via probabilistic seismic hazard analysis (PSHA). PSHA provides a technique for estimating the probability that earthquake ground motions at a location of interest will exceed some intensity measure (IM), such as peak ground velocity or peak ground acceleration, over a given time period. These kinds of estimates are useful for civic planners, building engineers, and insurance agencies, and as the basis for building codes, influence billions of dollars of construction each year.

PSHA estimates are delivered by hazard curves that relate ground motion to probability of exceedance for a site of interest (Figure 2). A set of hazard curves for a region can be interpolated to produce a hazard map for an area, holding either probability or ground motion constant and plotting the other as a function of geographic location (Figure 3).

The CyberShake computational platform uses a physics-based approach to produce PSHA hazard curves [25]. CyberShake must consider a set of possible large earthquakes that could influence the site. About 415,000 different earthquakes must be considered for a typical site.

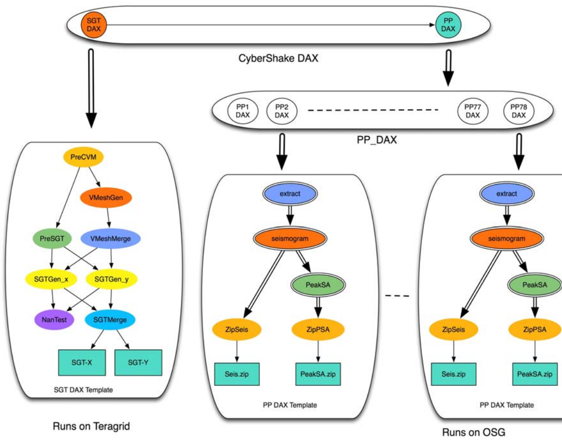

Figure 4. CyberShake Workflow

CyberShake has extensive computational requirements, which are detailed in [26]. Essentially, CyberShake performs two wave propagation simulations, which are parallel jobs, followed by approximately 840,000 short-running serial jobs. In total, calculating a single hazard curve requires about 14,000 CPU-hours and generates about 750 GB of data. Figure 4 shows the CyberShake workflow managed by Pegasus. The workflow contains two sub-workflows, the SGT workflow and the Post Processing (PP) workflow. The SGT workflow consists of several MPI codes calculating wave propagation (SGT files). The PP workflow contains 78 sub-workflows each with approximately 12,000 jobs. Each PP workflow takes the input SGT files and generates seismograms and peak spectral acceleration values as output.

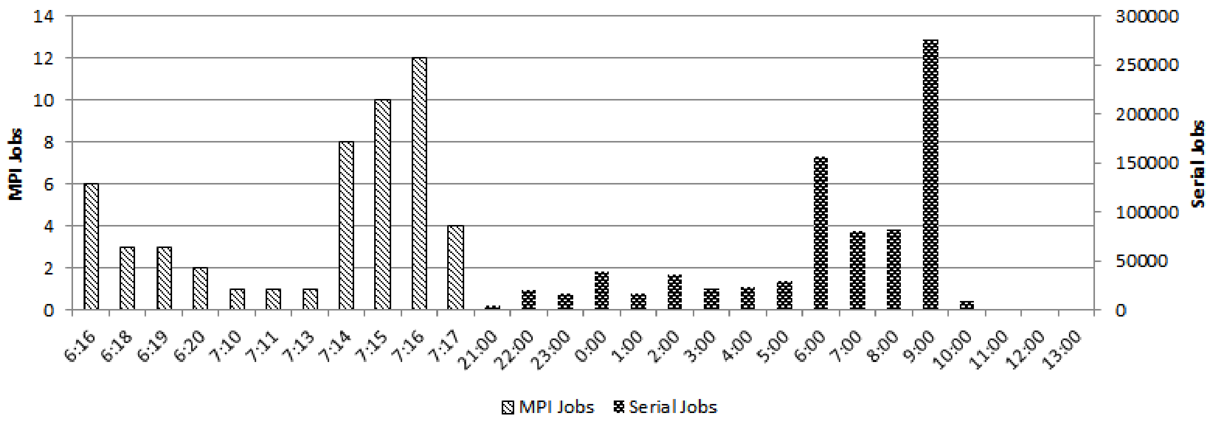

Figure 5. CyberShake workflow. Parallel and serial jobs. (Note the difference in scales (left and right axis) for the parallel and serial jobs.)

In our experiment, Pegasus ran the SGT workflow on the TeraGrid site Ranger, and the PP workflow on the OSG site Firefly. Figure 5 shows the number of parallel jobs finishing on Ranger on the left side. On the right side, it shows the number of serial jobs finishing on Firefly. Pegasus submitted the parallel jobs directly to the Globus GRAM interface, while the PP sub workflows were submitted by Pegasus to the resources provisioned by GlideinWMS on Firefly. The 20 GB of SGT output files were automatically staged by Pegasus from Ranger to Firefly. The final outputs consisted of 800,000 files, which were compressed and automatically transferred back from Firefly to the scientist's desktop for further analysis.

B. High-Performance Periodogram Calculations in the Search for Exoplanets

Another example of using GlideinWMS to leverage the national cyberinfrastructure is the application that processes Kepler data. The Kepler satellite[27], launched on 06 March 2009, is a NASA mission that uses high-precision photometry to search for transiting exoplanets around main sequence stars. The French mission Convection Rotation and Planetary Transits (CoRoT)[28], launched in late 2006, has similar goals. Kepler's primary mission is to determine the frequency of Earth-sized planets around other stars. In May 2009, it began a photometric transit survey of 170,000 stars in a 105 square degree area in Cygnus. The photometric transit survey has a nominal mission lifetime of 3.5 years. As of May 2011, the Kepler mission has released 381,665 light curves of 175,934 stars; these light curves contain measurements made over 418 days, with between 500 and 50,000 epochs per light curve. The Kepler team has identified 16 new planets [29], and 1,235 stars that may harbor planets [30]. They based their findings on periodic variations in the light output of the host stars as the planet transits in front of them. These impressive results do, however, only scratch the surface of the scientific content of the Kepler dataset. What is needed is an analysis of all the stars to identify periodic signals in their light curves as a starting point in finding new exoplanets.

Identifying periodic signals requires calculations of periodograms, which reveal the power in periodic signals present in time-series data and estimates of their significance. However, periodograms are computationally intensive, and the volume of data generated by Kepler demands high-performance processing. The NASA Star and Exoplanet Database (NStED) has developed a periodogram service [31] written in C, to take advantage of the "brute force" nature of periodograms and achieve the required performance. The processing of each frequency sampled in a periodogram is performed independently of all other frequencies, and so periodogram calculations are easily performed in parallel on a machine cluster by simply dividing the frequencies among the available cores. In practice, the processing is managed by a simple front-end job manager that splits the processing across all available cores, and then combines the results. The code itself returns the periodogram, a table of periodicities and their significance, light curves phased to the periodicities and plots of the periodograms and light curves.

The need for parallelization is shown in Table I, which shows the processing times on a single Dell 1950 processor for three algorithms supported by the service. These algorithms are:

- **Lomb-Scargle (L-S).** Supports unevenly sampled data. Most useful for looking for sinusoidal-like variations, such as the radial velocity wobble of a star induced by an orbiting planet. [32][33]
- **Box Least Squares (BLS).** Optimized to identify "box"-like signals in time series data. Most useful for looking for transiting planets. [34]
- **Plavchan.** Binless phase-dispersion minimization algorithm. It identifies periods with coherent phased light curves (i.e., least "dispersed"). There is no assumption about the underlying shape of the periodic signal. [35]

Table 1. Processing times for Periodogram algorithms on a Dell 1950 server, with 2 X 2.5 GHz quad-core CPU's with 8 GB memory, running Red Hat Linux 5.3

# Data Points	**L-S**	**BLS**	**Plavchan**	**# Periods Sampled**
1,000	25 s	<15 s	50 s	100,000
10,000	5 min	2 min	14 min	100,000
100,000	40 min	15 min	2 hr	100,000
420,000	9 hr	3 hr	41 hr	420,000

The scientific goal is to generate an atlas of periodograms of the public Kepler data, computed with all three algorithms for maximal science value. The atlas will be served through NStED, along with a catalog of the highest-probability periodicities culled from the atlas. End-users will be able to search the atlas, browse periodograms and phased light curves, identify stars for further study, and refine the periodogram calculations as needed.

Because there were not enough resources available behind the periodogram service to do a full run of the current Kepler dataset, a decision was made to leverage national cyberinfrastructure, and a workflow generator was developed to create the abstract workflow that is given to Pegasus. The generator estimates the runtime of each light curve based on the number of data points, the algorithm being used, and the parameters supplied for the algorithm. Each computational task is categorized according to its estimated runtime as being: **1) very fast** - sub-second run-time, **2) fast** - run-time in seconds, or **3) slow** - run-time about an hour.

Based on our experiences with different application workflows [36], we decided to cluster the workflow tasks in order to reduce both the scheduling overhead and the total number of Condor jobs. The majority of the experiments used a simple numeric clustering scheme, where *very fast* tasks were clustered into groups of 100 tasks per job, *fast* tasks were clustered into groups of 20 tasks per job and *slow* tasks were not clustered at all. We specified a target runtime for the jobs of 60 minutes and clustered the tasks based on their estimated runtime using the simple first-fit bin packing algorithm. A cluster containing a number of tasks is treated as a single job by the workflow execution system and Condor. We ran all 3 algorithms and with the total number of jobs being 8,264.

Input and output data for the workflow reside on the submit host. Condor file I/O was used to automatically stage input files before executing jobs on remote resources and to retrieve output files after each job finishes. To ease the management of input and output files and improve transfer times, the inputs was pre-compressed using gzip and uncompressed on the remote resource before executing the computation. The outputs generated were compressed on the remote resource before being staged back to the submit host. Compressing the data resulted in bandwidth savings of more than 80 percent. The size of the input data set was 61 GB. The size of the output data set was 790 GB

Figure 6 shows the total number of glideins registered during the experiment. For more than half of the duration there were about 1,000 glideins, which was above expectations. The number of glideins submitted to TeraGrid was limited to 256 as we were about to run out of allocation. On OSG, we provisioned as many cores as were available on resources at the University of Nebraska-Lincoln and the University of California San Diego. The breakdown on where the glideins executed can be seen in Figure 7. During the run, there were 1384 job restarts. Given that the total number of jobs was 8264, this could be seen as an excessive number of restarts. However, the restarts were mostly from glideins disappearing from the system, and the job restarts were automatic without the user

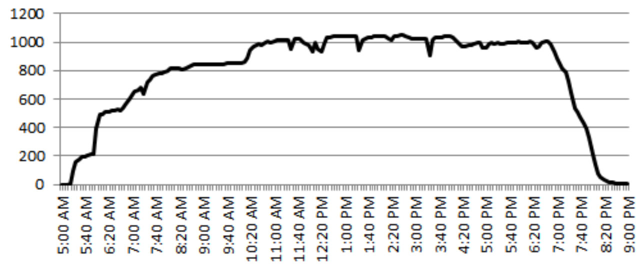
Figure 6. Total number of glideins during Periodogram run

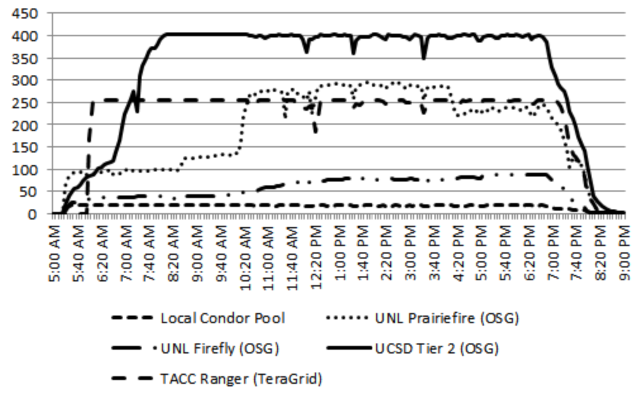

Figure 7. Number of glideins per resource for the Periodogram run

even knowing about it, so the overall usability was not affected. The main reason glideins disappeared was preemption on the opportunistic resources. Most of the preemption happened at the UNL Prairiefire and UCSD Tier 2 resources as seen in the dips of Figure 7. The dip affecting several of the resources around noon can be attributed to an automatic backup job on the desktop machine, using up a large amount of both CPU and bandwidth resources – a good example of problems, which can show up when using desktop computers as submit / central managers. In this case the jobs were short enough that job restarts were not a major problem. For longer jobs, one solution for minimizing the amount of lost work caused by these restarts would be to enable Condor checkpointing.

VI. Lessons Learned

The systems in both experiments did an excellent job shielding the scientists from hitting resource limits and transient problems. None of the experiments required user intervention at runtime, and it was not until logs were analyzed that problems were noticed. Some of the logged issues, such as glideins being preempted due to higher priority workload, should be excluded when analyzing the overall performance, as preemption is an inherent feature of an opportunistic system. A more interesting concern is the fact that a desktop machine and associated infrastructure (such as networking), are not set up to be as scalable and reliable as production servers usually are. This was noticed in part when the desktop backup job affected registered glideins during the periodogram run, but also the fact that the output dataset of that run was almost 800 GB, which is a large percentage of the available disk on a typical desktop machine. One solution to this particular problem would be to configure Pegasus to stage data out directly from the execution site to a dedicated storage system. In general, modern desktops provide enough scalability and reliability to run medium-sized workflows, and for many scientists, the convenience and availability of the desktop still makes the setup an attractive solution. However, for larger workflows or larger communities, a hosted infrastructure such as the one provided by glideinWMS with the VO Frontend is more appropriate.

Another lesson learned is that the overlay created by glideinWMS does not protect the scientist from different environments and software versions. This is no different from running in any other heterogeneous Condor pool. When using a single cyberinfrastructure, the problem is not as obvious as when running across different cyberinfrastructures. In the case of the periodogram run, the periodogram code was wrapped in a python wrapper, which required python 2.4. The mix of python versions on the targeted resources meant we had to add a wrapper script to select the correct version of installed python.

From a scientist's point of view, it can be difficult to know the exact behavior of the code, and how that behavior matches the default glidein configuration. An example of this occurred during the CyberShake run, for which some jobs required more than 2 GB of RAM per core, which was the default provisioned by the glideinWMS system at that time. This is an example of an issue that would be visible to the scientist only when a job fails and the relevant logs are analyzed. Once the cause of failure was detected, the solution was simple: configure the glideinWMS factory to request more memory per core.

Finally, it is interesting to note that one higher-level concern that scientists have with a dynamic provisioning model is that it is hard to estimate run time and completion time. Not only does this matter when hard deadlines are present, but also in day-to-day operations. GlideinWMS provides a set of graphing and querying tools to aid the scientists in understanding how the system is provisioning resources over time. When running without historical data, such as the first couple of times a workflow is executed, it is difficult to envision resource requirements and how the dynamic provisioner will react to those requirements. One solution may be to run a smaller, representative workflow in order to collect these initial statistics or to use a system like Stampede [37] to gather statistics at runtime.

VII. Related Work

There are several systems that provide functionality similar to Corral/glideinWMS. These systems include generic pilot job systems, workload management systems, pilot job application frameworks, desktop grid systems, and meta-schedulers.

Condor_glidein [38] is a command-line tool distributed with the Condor system that is used to add grid resources to an existing Condor pool using the glidein technique. Condor_glidein is simple to use, but requires users to manage resource provisioning manually and does not support dynamic provisioning. *MyCluster* [39] creates personal clusters using several different resource managers including Condor. Like Corral/glideinWMS, MyCluster enables individual researchers to easily provision and manage large pools of distributed resources from a simple desktop interface. The main advantages of the Corral/glideinWMS approach are the dynamic provisioning and scalability features provided by glideinWMS.

Several different pilot-based workload management systems have been developed to manage data processing jobs for LHC experiments. These systems include DIRAC [40], used by LHCb; PanDA [41][42], used by ATLAS; and AliEn [43] used by ALICE. All of these systems perform VO-level functions similar to glideinWMS. The main difference between glideinWMS and these systems is that glideinWMS is built on top of a generic high-throughput computing system, Condor, while the other systems use custom, VO-specific schedulers, policies, and data management facilities. The use of Condor makes glideinWMS a more general-purpose, VO-agnostic system. In addition, the combination of Corral and glideinWMS enables high-performance, application-level scheduling appropriate for individual researchers that are not part of a VO.

There are also application frameworks that utilize the pilot job concept to provide distributed task execution. DIANE [44] is a master-worker framework that uses pilot jobs to start worker processes on distributed resources, and BigJob [45] is a pilot job framework for distributed applications that is built on top of the SAGA API. Both of these systems require users to modify their applications to fit the system's framework. In contrast, our approach is based on a general-purpose batch system, Condor, which lets application jobs run unmodified.

Desktop grid and volunteer computing systems such as BOINC [46], and XtremWeb [47]. These systems are designed to support very large numbers of relatively low-power, unstable computational resources. They are also oriented towards idle cycle harvesting. As a result, they focus more on issues of trust, partial network connectivity, community building, and integrity of results.

Our solution is also similar to meta-scheduling solutions such as GridWay [48], which focuses primarily on fault tolerance and resource selection. The primary difference between Corral/glideinWMS and a pure meta-scheduling approach is that the use of glideins enables resources to be reserved for short periods and reused for multiple jobs.

VIII. Conclusion

To conclude, we found that desktop machines work very well as central managers for glidein-based infrastructures and for submitting and managing small to medium-sized workflows. When running across cyberinfrastructures, the provided glidein overlay shields the scientist from most, but not all, differences of the infrastructures, and enables the scientist to concentrate on the science instead of where available resources might be available.

IX. Acknowledgments

Corral and GlideinWMS are supported in part by NSF under grant OCI-0943725. This research was done using resources provided by the Open Science Grid, which is supported by the NSF and the DOE's Office of Science. This research was also supported in part by the National Science Foundation through TeraGrid resources under grant number

TG-CCR1000018. G. B. Berriman. is supported by the NASA Exoplanet Science Institute at the Infrared Processing and Analysis Center, operated by the California Institute of Technology in coordination with the Jet Propulsion Laboratory (JPL).

X. References

[1] "CERN - The Large Hadron Collider." [Online]. Available: http://public.web.cern.ch/public/en/lhc/lhc-en.html.

[2] R. Pordes et al., "The open science grid," *Journal of Physics: Conference Series*, vol. 78, no. 1, p. 012057, 2007.

[3] C. Catlett and et. al., "TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications.," in *HPC and Grids in Action*, IOS Press, 2007.

[4] J. Frey, T. Tannenbaum, M. Livny, I. Foster, and S. Tuecke, "Condor-G: a computation management agent for multi-institutional grids," in *High Performance Distributed Computing, 2001. Proceedings. 10th IEEE International Symposium on*, 2001, pp. 55-63.

[5] G. Juve and E. Deelman, "Resource Provisioning Options for Large-Scale Scientific Workflows," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, 2008, pp. 608-613.

[6] E. Walker, "Continuous adaptation for high performance throughput computing across distributed clusters," in *Cluster Computing, 2008 IEEE International Conference on*, 2008, pp. 369-375.

[7] I. Sfiligoi, D. C. Bradley, B. Holzman, P. Mhashilkar, S. Padhi, and F. Wurthwein, "The Pilot Way to Grid Resources Using glideinWMS," in *Computer Science and Information Engineering, 2009 WRI World Congress on*, 2009, vol. 2, pp. 428-432.

[8] E. Deelman et al., "Pegasus: a framework for mapping complex scientific workflows onto distributed systems," *Scientific Programming Journal*, vol. 13, pp. 219–237, 2005.

[9] I. Foster, C. Kesselman, and S. Tuecke, "The Anatomy of the Grid: Enabling Scalable Virtual Organizations," *Int. J. High Perform. Comput. Appl.*, vol. 15, no. 3, pp. 200-222, 2001.

[10] "CMS Experiment." [Online]. Available: http://cms.web.cern.ch/cms/.

[11] "The Collider Detector at Fermilab." [Online]. Available: http://www-cdf.fnal.gov/.

[12] "The DZero Experiment." [Online]. Available: http://www-d0.fnal.gov/.

[13] "The GlueX Experiment." [Online]. Available: http://gluex.org/.

[14] P. J. Sutton, "Searching for gravitational waves with LIGO," *Journal of Physics: Conference Series*, vol. 110, no. 6, p. 062024, 2008.

[15] "SBGrid: Structural Biology Grid." [Online]. Available: http://sbgrid.org/.

[16] "NEBioGrid Virtual Organization." [Online]. Available: http://nebiogrid.org/.

[17] G. Klimeck, M. McLennan, M. S. Lundstrom, and G. B. Adams, "nanoHUB.org - Online Simulation and More Materials for Semiconductors and Nanoelectronics in Education and Research," in *Nanotechnology, 2008. NANO '08. 8th IEEE Conference on*, 2008, pp. 401-404.

[18] "Holland Computing Center," *Holland Computing Center - University of Nebraska-Lincoln*. [Online]. Available: http://hcc.unl.edu/main/index.php.

[19] "GLOW." [Online]. Available: http://www.cs.wisc.edu/condor/glow/.

[20] "Northwest Indiana Computational Grid - Home." [Online]. Available: http://www.nwicgrid.org/.

[21] "OSG Engagement VO." [Online]. Available: https://twiki.grid.iu.edu/bin/view/Engagement/WebHome.

[22] G. Juve, E. Deelman, K. Vahi, and G. Mehta, "Experiences with resource provisioning for scientific workflows using Corral," *Sci. Program.*, vol. 18, no. 2, pp. 77-92, 2010.

[23] "Southern California Earthquake Center." [Online]. Available: http://www.scec.org/.

[24] "Infrared Processing and Analysis Center." [Online]. Available: http://www.ipac.caltech.edu/.

[25] R. Graves et al., "Physics Based Probabilistic Seismic Hazard Calculations for Southern California," presented at the 14th World Conference on Earthquake Engineering, Beijing, China, 2008.

[26] S. Callaghan et al., "Reducing Time-to-Solution Using Distributed High-Throughput Mega-Workflows - Experiences from SCEC CyberShake," in *eScience, 2008. eScience '08. IEEE Fourth International Conference on*, 2008, pp. 151-158.

[27] "Kepler." [Online]. Available: http://kepler.nasa.gov/.

[28] "COROT." [Online]. Available: http://www.esa.int/esaMI/COROT/index.html.

[29] "Kepler Discoveries." [Online]. Available: http://kepler.nasa.gov/Mission/discoveries/.

[30] "Kepler: Planet Candidates." [Online]. Available: http://kepler.nasa.gov/Mission/discoveries/candidates/.

[31] "NStED Periodogram Service." [Online]. Available: http://nsted.ipac.caltech.edu/periodogram/cgi-bin/Periodogram/nph-simpleupload.

[32] K. D. Horne and Baliunas, "A Prescription for Period Analysis of Unevenly Sampled Time Series," *ApJ*, no. 302, p. 757, 1986.

[33] J. D. Scargle, "Studies in astronomical time series, II. Statistical aspects of spectral analysis of unevenly spaced data," *ApJ*, no. 263, p. 835, 1982.

[34] G. Kovacs, S. Zucker, and T. Mazeh, "A box-fitting algorithm in the search for periodic transits," *A&A*, vol. 391, no. 1, pp. 369-377, 2002.

[35] P. Plavchan, M. Jura, J. D. Kirkpatrick, R. M. Cutri, and S. C. Gallagher, "Near-Infrared Variability in the 2MASS Calibration Fields: A Search for Planetary Transit Candidates," *The Astrophysical Journal Supplement Series*, vol. 175, no. 1, p. 191, 2008.

[36] G. Singh et al., "Workflow task clustering for best effort systems with Pegasus.," in *Mardi Gras Conference'08*, 2008, pp. -1–1.

[37] S. Taghrid et al, "Online Fault and Anomaly Detection for Large-Scale Scientific Workflows," in *13th IEEE International Conference on High Performance Computing and Communications*, 2011

[38] "condor_glidein." [Online]. Available: http://www.cs.wisc.edu/condor/manual/latest/5_4Glidein.html.

[39] E. Walker, J. P. Gardner, V. Litvin, and E. Turner, "Creating Personal Adaptive Clusters for Managing Scientific Jobs in a Distributed Computing Environment," 2006.

[40] A. Tsaregorodtsev et al., "DIRAC: a community grid solution," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062048, 2008.

[41] P. Nilsson, "Experience from a pilot based system for ATLAS," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062038, 2008.

[42] T. Maeno, "PanDA: distributed production and distributed analysis system for ATLAS," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062036, 2008.

[43] S. Bagnasco et al., "AliEn: ALICE environment on the GRID," *Journal of Physics: Conference Series*, vol. 119, no. 6, p. 062012, Jul. 2008.

[44] J. T. Moscicki, "DIANE - Distributed Analysis Environment for GRID-enabled Simulation and Analysis of Physics Data," presented at the IEEE Nuclear Science Symposium, 2003.

[45] A. Luckow, L. Lacinski, and S. Jha, "SAGA BigJob: An Extensible and Interoperable Pilot-Job Abstraction for Distributed Applications and Systems," presented at the 10th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, 2010.

[46] D. P. Anderson, "BOINC: A System for Public-Resource Computing and Storage," 2004, pp. 4-10.

[47] C. Germain, V. Néri, G. Fedak, and F. Cappello, "XtremWeb: Building an Experimental Platform for Global Computing," 2000.

[48] E. Huedo, R. S. Montero, and I. M. Llorente, "A framework for adaptive execution in grids," *Software: Practice and Experience*, vol. 34, no. 7, pp. 631-651, 2004.