Integrating Grid Services into the Cray XT4 Environment

Shreyas Cholia *and* **Hwa-Chun Wendy Lin**,
National Energy Research Scientific Computing Center at
Lawrence Berkeley National Laboratory

ABSTRACT: *The 38640 core Cray XT4 "Franklin" system at the National Energy Research Scientific Computing Center (NERSC) is a massively parallel resource available to Department of Energy researchers that also provides on-demand grid computing to the Open Science Grid. The integration of grid services on Franklin presented various challenges, including fundamental differences between the interactive and compute nodes, a stripped down compute-node operating system without dynamic library support, a shared-root environment and idiosyncratic application launching. In our work, we describe how we resolved these challenges on a running, general-purpose production system to provide on-demand compute, storage, accounting and monitoring services through generic grid interfaces that mask the underlying system-specific details for the end user.*

KEYWORDS: Cray, XT4, grid, Globus, distributed computing, OSG, science gateways, MPI, PBS

1. Introduction

High performance computing (HPC) is becoming increasingly parallel. With clock speeds flattening out, and power consumption playing a major role in CPU design, multi-core and many-core technologies are seen as the most efficient way to increase overall performance and scalabilty. While grid computing originally evolved from a serial model of computing, it has become increasingly important for grids to be able to take advantage of highly parallel resources in order to maximize resource utilization, while maintaining the benefits of a distributed, on-demand service. Because of its unique architecture, the Cray XT4 system presents a special set of challenges when it comes to integrating grid services and tools with the underlying environment. NERSC has successfully integrated on-demand grid services based on the Open Science Grid (OSG) software stack on its 38640 core (38128 compute cores) Cray XT4 system. In this white-paper, we discuss the challenges presented by this environment and our solution for creating fully functional OSG compute and storage elements on this system. We pay special attention to security, job management, storage, accounting and reporting services for the grid, while using the principal of least privilege to set up the software stack. The end result is a parallel OSG computing platform that can be transparently accessed through generic grid software. This allows users to access the underlying HPC resources without needing detailed knowledge of the Cray XT4 architecture, thus increasing overall usability through transparent, service-oriented, cross-platform interfaces.

The NERSC Cray XT4 system, named Franklin, is a massively parallel compute system available for scientific research. The system is capable of providing 356 TFlop/sec of computational power through its 9532 quad-core 2.3 GHz AMD-Opteron processors. It also provides access to a 436TB Lustre parallel filesystem. Franklin is located at the NERSC supercomputing center at Lawrence Berkeley National Laboratory, and is available to scientific researchers under the umbrella of the Office of Science in the U.S. Department Of Energy.

2. What is Grid Computing?

Grid computing provides the ability to share and aggregate heterogeneous, distributed computational capabilities and deliver them as a service. According to Ian Foster, a grid is a system that *"coordinates resources that are not subject to centralized control, using standard, open, general-purpose protocols and interfaces, to deliver nontrivial qualities of service"*. This idea can be explained by the following principle: a uniform set of software interfaces to access non-uniform and physically distributed compute and storage resources. In practice, grid computing does not make any sense unless the underlying resources (compute systems, data storage systems) are integrated into a larger whole i.e. a working grid that coordinates resources and users.

There are several flavors of grids, but, for the most part, scientific computing grids seem to have converged on a common interoperable infrastructure as seen in the OSG, TeraGrid, Earth Systems Grid and EGEE to name a few. Most of these grids use some flavor of the Globus middleware stack or equivalent to provide a common layer of services that expose the underlying resources

Grid computing is well equipped to deal with serial and "embarrassingly parallel" tasks (computational jobs that can be broken up into parallel tasks that require little to no communication between these tasks). This model has been exploited by certain scientific applications, particularly in the high-energy physics community. There is little interaction between nodes running a subtask, and each task can run to completion, without consideration for other nodes in the system. However, there has been an increasing need for access to grid based parallel computers – systems that can be provisioned on-demand based on specific needs, while still providing the benefits of a highly parallel resource with a fast interconnect. This allows users to farm out a tightly coupled set of computational interactions to an appropriately tightly coupled system that matches the job requirements.

Additionally, user data is often distributed across multiple centers and may need to be moved across the grid, to an appropriate storage resource that is local to the computational tasks. There needs to be a set of standard high-performance interfaces to access and transfer data from multiple locations. This points to a data-grid where users can move data across resources transparently.

3. Open Science Grid

The Open Science Grid (OSG) is a distributed computing infrastructure for large-scale scientific research, built and operated by a consortium of universities, national laboratories, scientific collaborations and software developers. Researchers from many fields, including astrophysics, bioinformatics, weather and climate modeling, computer science, medical imaging, nanotechnology and physics use the OSG infrastructure to advance their research.

Sites can make their resources available to the OSG by installing a pre-defined service stack that is made available as part of the Virtual Data Toolkit (VDT). This includes the Globus software stack for the core computational and data grid functionality, and a set of supporting software for coordinating the resources in the grid. The Globus Toolkit includes services that support the following operations:

1. GSI user and host authentication and authorization
2. Globus Job submission and management through GRAM (GT2 and GT4)
3. File storage and transfer through Globus GridFTP

Additionally a site must be able to advertise resource descriptions and availability, and report on usage accounting information back to the central OSG infrastructure. In order to do this the site must run a set of services:

1. CEMon for resource descriptions
2. RSV probes for resource availability
3. Gratia probes for accounting information

As the OSG broadens in scope parallel computing becomes increasingly important as a means to achieve scalable performance. Newer science communities including ones in bioinformatics and climate science have a stake in the OSG, but also have parallel HPC requirements for their jobs. The NERSC Franklin system plays a key role in fulfilling the parallel computing needs of the OSG.

Opening up the NERSC Cray XT4 system to the OSG allowed us to deliver the power and scale of a massively parallel HPC system to whole new set of scientists. This has created an opportunity to enable new science by allowing scientists from around the world to run their jobs in this manner through the OSG.

However, there are some peculiarities in the XT4 architecture that make this quite a challenge. The key behind grid computing is the idea of a standard, uniform interface that can be used to access a wide variety of underlying platforms. This means that any system specific details must be hidden from the user. The user only needs to define the job or workflow or data transfer operation through a common interface, and to be able to query the resources or jobs. The grid middleware must then abstract all system specific details, and should be able to interface with the underlying environment to be able to manage these requests.

In the rest of this paper we describe how we set up and configured the OSG and Cray software to navigate the complexities of the XT4, so that the system could be available to grid users in a transparent fashion. This paper is primarily focused on OSG, but much of it applies to any Globus based grid

4. Franklin Cray XT4

The following table describes the current Franklin Cray XT4 system configuration and computing environment (Table 1.):

Number of compute nodes	9,532
Processor cores per node	4
Number of compute processor cores	38,128
Processor Core type	Opteron 2.3 GHz Quad Core
Physical memory per compute node	8 GB
Memory usable by applications per node	7.38 GB
Number of login nodes	10
Switch Interconnect	SeaStar2
Usable disk space	436 TB
Batch system	PBS Torque/Moab
Service Node Operating System	SuSE SLES10 SP1 Linux
Compute Node Operating System	Compute Node Linux
Communication Layer	Portals
Compute Node Application Launcher	ALPS utility (aprun)

Table 1. Franklin Configuration

There are a few things worth noting about the above configuration:

1. The system is partitioned into service nodes and compute nodes. The service nodes handle interactive user-logins, batch system management, and I/O. The compute nodes run user jobs.
2. The compute nodes do not have the same environment as the interactive or service nodes. While the interactive nodes have a full-featured derivative of SuSE Linux, the compute nodes run a stripped down operating system called Compute Node Linux (CNL). This allows for increased scalability and performance on the compute nodes, but also means that any executables that are staged on the service nodes, must be precompiled as static binaries, since there is no formal support for dynamically loaded libraries.
3. We use a version of OpenPBS called Torque, along with the Moab job scheduler, to manage the batch system. From the standpoint of the grid software, this is equivalent to any standard PBS system, and we will simply refer to it as PBS in the rest of this paper. The PBS batch system runs on a separate set of service nodes called the MOM nodes that handle job processing.
4. Service nodes run diskless and make use of an NFS filesystem exported from the System Database node for non-root common files.
5. Application launching is done through a special utility called “aprun” instead of the more common mpirun/mpiexec tools because of the unique nature of the compute nodes.

5. NERSC Franklin Grid Configuration

Because of the idiosyncrasies described in the previous section, configuring grid software on a running production system, takes on additional complexity. In this section we describe the overall approach to configuring grid software on Franklin.

Designated Grid Node

We designate one of the interactive nodes as the grid node. All grid services run on this node. Control connections for grid traffic go through this grid node. We install the OSG Compute and Storage Element software on the grid node. This is the point of contact for the OSG with respect to routing jobs/data and collection of site information. We also ensure that xinetd is installed on the node for service startup.

DNS Aliasing

We use a DNS alias for the grid node in the OSG software configuration. This allows us to transparently switch to an alternate node in case of failure, by simply pointing the DNS alias to a new, pre-configured backup grid node. For example, the Franklin grid node is referred to by its DNS Alias (franklingrid.nersc.gov) by any OSG consumers. This is currently mapped to a specific node (nid00256-eth0.nersc.gov) in DNS. In case of a failure, we simply change the DNS mapping for franklingrid.nersc.gov to the failover node, but since the grid clients only use the generic alias, this change is essentially transparent.

Note: All GSI host certificates need to be issued for the actual DNS hostname (not the generic alias). However, this should be transparent to the clients who will only to connect to the alias.

Install on Shared NFS Filesystem

Service nodes run diskless with a preconfigured boot image and a shared root filesystem, which makes it non-trivial to make configuration changes. Configurations are defined for different classes of nodes (login node, network node, dvs node etc.) with a specific login node set up as the grid node.

We avoid installing software on the shared-root filesystem as much as possible. Instead we install it on the common filesystem called /usr/common, which is NFS imported from the sdb (system database) node. When necessary, we create symbolic links in the shared-root pointing to /usr/common for the software installation and configuration. For node specific configuration, we create per node links in the common filesystem. In practice this means:

- installing grid software in a common filesystem as a non-root administrative user
  eg. /globus
- setting up per-node directories in the common filesystem for node-specific files
  eg./globus/gridsecurity/nid1234/hostcert.pem
- creating symbolic links for the grid node to point to appropriate locations
  eg. /etc/grid-security→/globus/grid-security-nid1234, /etc/grid-security/certificates→/globus/certificates
- The only exception for this is service software that needs to be run out of init or xinetd.

Install as non-root user

We try to install most of the OSG software as a non-root administrative user (such as "globus"). This allows us to maintain a clear privilege separation. This will prevent us from running the automated startup installers for xinetd and init. These must be configured manually, by copying the appropriate files from the OSG software bundle. Where possible, servers and cron jobs are run as a non-root user.

Before we discuss how we set up specific components of the grid infrastructure, it may be helpful to go over the usage model for grid software.

6. Usage Model

From a user's standpoint the grid infrastructure could allow her to submit a job or move data to multiple locations, each with very different underlying architectures, while using the same front-end interface. Alternately, the user could query the grid for a list of resources and select her target resources based on the job and data characteristics.

In our model (Figure 1.) we envision that users will access grid resources through gateway portals where they can define jobs and select target resources through web interfaces. This is then translated into the appropriate Globus job specification by the web portal and sent to the target site. At the target site the job then gets translated through the local job manager to the batch system.

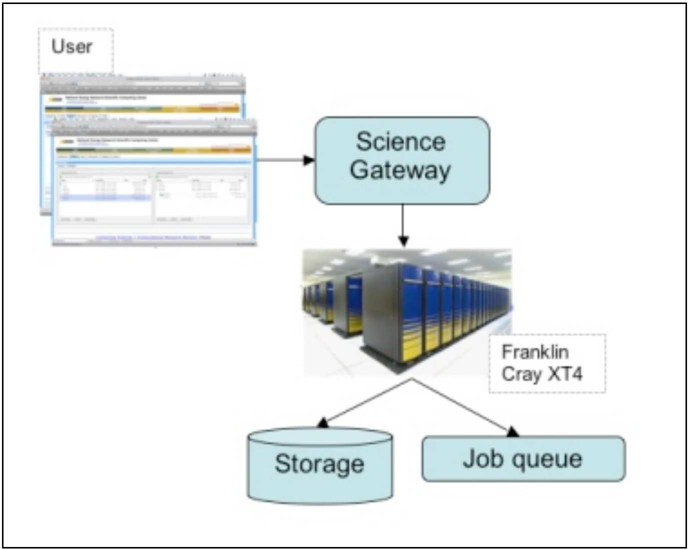

Figure 1. Submitting a job through a web portal

Alternately, more experienced users, may wish to use a grid job management framework like Condor-G to write their own generic job descriptions which can be submitted to target resources. Example 1. shows a simple Condor-G job description for a 4 way MPI job.

```
Universe = grid
Executable = test_dir/test_application
transfer_executable = false
grid_resource=gt2 franklingrid.nersc.gov/jobmanager-pbs
```

```
globus_rsl = (jobType=mpi) (count=4)
output = test.out
error = test.err
log = test.log
Queue
```

Example 1. Condor-G job example

The Condor-G client will convert this into Globus RSL before submission to the jobmanager. A simplified version of the RSL is shown in Example 2.

```
& (count=4)
(jobtype=mpi)
(directory=test_dir)
(executable=test_dir/test_application)
(stdout=x-gass-
cache://$(GLOBUS_GRAM_JOB_CONTACT)stdout
anExtraTag)
(stderr=x-gass-
cache://$(GLOBUS_GRAM_JOB_CONTACT)stderr
anExtraTag)
```

Example 2. Globus RSL example

Condor-G also handles the complexities of staging files in and out, as well as dealing with intermediate files and caches. Simple jobs can also be submitted directly through the Globus RSL, but since this becomes unwieldy for more complex jobs, Condor-G is the preferred interface for job submission on the OSG.

So, in order to run a 1000-way parallel job on another grid resource, the user would run an ldap query for a resource with the appropriate backend hardware, and submit the job to the given resource. All they would have to change in the above example would be the target resource and the level of parallelism.

Compare this with a specification for the same job on Franklin's local PBS batch system (Example 3.).

```
#PBS -l mppwidth=4
#PBS -l mppnppn=4
#PBS -e test.err
#PBS –o test.out
cd $PBS_O_WORKDIR
aprun -n 4 -N 4 ./a.out
```

Example 3. PBS job description

There are several system specific details that would ideally be hidden from the end user. Many of these details such as processor count or job launcher syntax are generally constant for a given system and are best left for the system administrators to determine. Moreover different parallel machines may be configured with very different underlying batch processing systems. Respecifying the above job to run on a loadleveler or SGE queue would require a completely different set of directives that essentially accomplish the same task. By using a single grid job specification, you can now run the same parallel job on multiple systems, without having to rewrite the job for each system.

Life Cycle of a Grid Job

The following describes the lifecycle of a grid job in the Condor-G/Globus framework on Franklin:

1. User defines job in condor or Globus RSL on client
2. Authenticate to Globus gatekeeper on Franklin grid node
3. Upon successful authentication, stage in files if needed
    a. Authenticate to Franklin gridFTP server
    b. Use globus-url-copy to stage files from client to gridFTP server
4. Upon successful stage-in, submit job to PBS job manager
5. Jobmanager converts Globus RSL into PBS job submission script
6. Jobmanager issues qsub
    a. Submit job to PBS
    b. Get PBS Job ID to keep track of job
7. Return a Globus endpoint URL for the client to query
8. Client will query endpoint
    a. Server listens on endpoint
    b. Returns Job status
9. Server periodically queries PBS for job status
10. Upon job completion server will change status to DONE
11. Client will pull any files that need to be staged out

7. OSG Software Configuration

This section describes how we configure the OSG CE software stack for the Cray XT4. While we provide some basic background information, it is assumed that the reader will have some familiarity with the Globus toolkit and the VDT/OSG service stack. More details on how to configure an OSG compute element can be found here: https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/WebHome
We cover the following software components here:

1. Overall Installation
2. GSI Authentication and Authorization
3. Globus Job Submission
4. File transfer through GridFTP

5. Site Monitoring and Availability through CEMon and RSV
6. Job Usage Accounting with Gratia

7.1 Overall Installation

The OSG software is installed from the VDT toolkit using an installation management tool called "pacman". The basic installation works as follows:

1. Download and install latest OSG release (SLES Compute Element and PBS components) from VDT using pacman
2. Run osg configuration script (configure-osg.py) for site specific details.
3. Run vdt-control tool to install specific services

We go through the default installation procedure, but must run step 3 manually because of the root separation constraints described earlier. Steps 1 and 2 are run as a non-root administrative user and will create a complete software installation in the desired location. We manually copy the OSG services, init and xinetd entries as the root user into the grid node configuration.

7.2 Authentication and Authorization:

The OSG grid infrastructure at NERSC uses a model based on PKI X.509 grid certificates and the Grid Security Interface (GSI) to handle authentication and authorization functions. This enables a single sign-on (SSO) type system, where the user is in possession of a single identity certificate, and can authenticate to multiple sites using the same certificate. SSO technology is crucial for access to multiple sites since managing multiple credentials for each resource does not scale well on the grid. This model assumes the presence of certain dynamically generated files to provide authorization and authentication information.

Authentication:

A set of trusted certificate authorities (CAs) and their associated certificate revocation lists are downloaded periodically from a central repository at NERSC (the contents of which are updated from their authoritative sources as determined by the OSG). The NERSC grid infrastructure will trust a user that presents a certificate issued by one of these trusted CAs. A user is authenticated if he/she presents a valid (signed, unrevoked and unexpired) certificate from a trusted CA.

Authorization:

An authorization map file (grid-mapfile), containing mappings from certificate subject Distinguished Names to local accounts is updated periodically from an LDAP database at NERSC. Once a user has been authenticated (has a valid certificate) they need to have a valid mapping in this file to be authorized to use the resource as a specific user. If no valid mapping exists, the request to access the resource is denied.

Both these steps rely on a set of files that are being constantly repopulated with current information. The grid infrastructure expects these files to be in a specific location on the root filesystem. However, since it is not practical to update files in the root filesystem, and since these files will be shared across nodes, we create a symbolic link pointing to a shared writable filesystem. This allows us to keep these files updated, and enables sharing these files across the entire system.

Note: The OSG software also has support for a more dynamic authentication and authorization service in the form of GUMS. While we do not use this service at NERSC because of incompatibilities with our site account management infrastructure, it significantly simplifies the above process by replacing these files with an external service that provides authentication and authorization assertions.

7.3 Globus Gatekeeper and Jobmanager:

Globus jobs are described using a generic job specification language called the Resource Specification Language (RSL). A job specified in Globus RSL can be submitted to any Globus GRAM gatekeeper, irrespective of the backend compute platform or batch system.

The Globus jobmanager is the interface between the generic Globus RSL and the underlying batch system. It will translate RSL directives into batch system specific directives. The jobmanager also supports running fork commands directly on the grid node.

Franklin uses PBS to manage its job queues. Any batch job that comes in as Globus RSL must be translated into a PBS job script. The default Globus PBS jobmanager is designed to work with a very simple PBS installation, but does not handle the specifics of the Cray XT4 very well. We had to make the following modifications to the pbs.pm jobmanager file to get this to work on Franklin:

1. Replace MPI launcher comands (mpirun / mpiexec) with Cray's aprun.
2. Default to MPI jobs.
3. Modify nodefile semantics, so that node selection is automatically handled by PBS.
4. Change "cluster" variable for single jobs, to differentiate from MPI jobs.
5. Support for mppwidth and mppnppn directives in PBS job file.
6. Identify the job executable and launch this directly through aprun, instead of wrapping this in a job script.

Additionally the system administrators have the ability to include default environment variables and PBS settings by making minor modifications to the pbs.pm file. For example, one may wish to redirect grid jobs to a specific queue, or adjust the number of processors that are available to a job on a given node.

For fork jobs (simple jobs that run immediately on the grid node), we simply use the default Globus Fork jobmanager. Globus also supports other batch systems – LSF, Loadleveler, Condor, SGE. Similar modifications would be required if these were managing the batch queue.

7.4 GridFTP:

GridFTP provides a high performance file transfer layer for moving data in and out of Franklin. In order to take full advantage of GridFTP, we need to be able to tune buffer sizes and set up striped transfers. GridFTP needs to be able to make both incoming and outgoing connections to work optimally, particularly when performing striped transfers.

To do this we create a hole in the network firewall and reserve a range of ports for GridFTP to be able to make and accept connections in both directions. We configure the GridFTP server to use this port range through an environment variable.

```
GLOBUS_TCP_PORT_RANGE=<START>,<END>
```

Here <START> and <END> define the beginning and end of the open port range. We then rely on more sophisticated network monitoring techniques, such as the BRO intrusion detection system, to ensure that the open ports are not being abused by malicious entities.

Additionally we configure GridFTP to use the generic DNS alias for the host, so that users have a standard endpoint to connect to, irrespective of underlying node changes

```
GLOBUS_HOSTNAME=franklingrid.nersc.gov
```

In the context of grid jobs, the Franklin GridFTP server provides an interface to stage-in input data and stage-out the results and output files back to the client. Stage-In and Stage-Out are handled directly by a GridFTP server on the interactive grid node, and are not typically managed by the batch system.

7.5 Monitoring and Availability Reporting:

In order for a system to integrate well with the grid, it must advertise details about its system configuration and availability. This allows users to make queries based on desired system characteristics and to schedule jobs on appropriate resources. For example a user may be interested in a parallel resource with a certain number of cores or a particular flavor of MPI. Monitoring and availability software plays a key role in matching users and jobs with the appropriate resources, thus optimizing grid usage.

In the OSG CEMon provides this functionality. CEMon runs as a Java web service in an Apache Tomcat container that queries the local resources and reports back to a central OSG collector in two formats - ReSS (for condor matchmaking services) and BDII (for WLCG compatibility). The CEMon information is based on the Generic Information Provider GLUE schema which includes several attributes considered important to grid users when trying to select resources.

In order to support MPI and parallel jobs we need to extend the GLUE schema to publish additional attributes that are necessary to describe a parallel platform. Specifically, we publish the following extended attributes:

- MPIInterconnect
- MPIVendor
- MPICompilerLocation

We update the Glue Schema template file (gip/etc/GlueCluster.template) with these attributes to make sure they get published.

CEMon is expected to run on the main interactive grid node for the cluster. By default CEMon will query this node for system characteristics and report back to the OSG CEMon collector.

However, on the Franklin system, much of this information will be inaccurate for the compute node where the user's job is expected to run. For example the interactive node runs a version of SLES 10 Linux, while the compute node runs a very different operating system - CNL – with a stripped down Linux kernel and much more restrictive functionality.

CEMon provides a mechanism to override this information, and to publish additional attributes in the form of two files:

- add-attributres.conf – publish additional attributes (See example 4.)
- alter-attributes.conf – modify values for existing attributes (See example 5.)
-

```
# Cray MPICH2
dn:          GlueSoftwareLocalID=MPICH2-pgi_2.1.4HD,
GlueSubClusterUniqueID=franklingrid.nersc.gov,
GlueClusterUniqueID=franklingrid.nersc.gov,mds-vo-
name=local,o=grid
objectClass: GlueClusterTop
objectClass: GlueSoftware
objectClass: GlueKey
objectClass: GlueSchemaVersion
GlueHostApplicationSoftwareRunTimeEnvironment:
pgi_2.1.41HD
GlueSoftwareLocalID: MPICH2-pgi_2.1.41HD
GlueSoftwareName: Cray MPICH2 PGI
    GlueSoftwareVersion: 2.1.41HD
GlueSoftwareInstalledRoot: /opt/cray/xt-asyncpe/2.0
GlueSoftwareModuleName: PrgEnv-pgi
GlueSoftwareEnvironmentSetup: module load PrgEnv-
pgi
GlueChunkKey:
GlueSubClusterUniqueID=franklingrid.nersc.gov
GlueSchemaVersionMajor: 1
GlueSchemaVersionMinor: 3
```

Example 4. Add-attributes.conf for Franklin

```
dn:       GlueSubClusterUniqueID=franklingrid.nersc.gov,
GlueClusterUniqueID=franklingrid.nersc.gov,mds-vo-
name=local,o=grid
GlueHostApplicationSoftwareRunTimeEnvironment:
pgi_2.1.41HD
MPIVendor: MPICH2
MPIInterconnect: CStar
MPICompilerLocation: /opt/cray/xt-asyncpe/2.0/bin/cc
```

Example 5. Alter-attributes.conf for Franklin

Use of the above files, allows us to accurately describe the underlying system architecture in the published CEMon records.

7.6 Accounting:

While the PBS accounting records provide a comprehensive view of all jobs run, we need to be able to separate any grid jobs from the local jobs. The grid jobs need to be reported back to the centralized grid accounting collector (Gratia), so that usage can be tracked for a given user or Virtual Organization (VO) on the grid.

The Gratia probe is a python script that runs periodically (as a cron job) on the CE node, collecting this PBS accounting information and reporting it back to the OSG. However, since PBS is configured to run on a separate service node, we need to periodically move the accounting information back to the grid node. For our purposes we found that a nightly copy of the accounting records from the PBS node to the grid node, was sufficient. This also gets around a couple of additional difficulties:

1. Timing issues of dealing with live records that may be in the process of being written – this is addressed by only reporting on the previous day's file.
2. Permissions problems with the accounting records (typically only readable by root) - we copy the accounting file over so that it is accessible to the Gratia administrative user (such as "globus")

We need to parse through the PBS records and then filter them based on

1. Designated OSG users
2. Gatekeeper logs for grid specific jobs

In order to filter on designated OSG users we look for a file called "osg-user-vo-map.txt". This file will contain a list of current OSG users and the VOs to which they belong. This allows us to limit reporting to OSG users.

The OSG software stack makes some minor modifications to the Globus Gatekeeper, to keep track of grid jobs for Gratia reporting. Before sending information back to the Gratia collector, job information is reconciled with the gatekeeper logs so that we only report on jobs submitted via the grid interfaces (and not locally submitted jobs).

The above configuration changes, provide a high-level description of the changes needed to create a production OSG compute/storage element on the Franklin Cray XT4. Readers are encouraged to contact the authors of this paper for additional configuration and setup details.

8. Open Issues and Future Work

We have identified the following as open issues and areas for future work in our grid access model:

- Static Binaries – user jobs must directly invoke the application binary because the compute nodes only support statically compiled binaries. These binaries must be precompiled with static library linking. This also implies that compute node jobs cannot be wrapped in a script. We expect that dynamic library support would address this issue.
- Support for grid portals – we are currently creating science gateway portals that will encapsulate grid job submission, thus hiding most of the Globus RSL details from the user.

- Insufficient MPI support in OSG – MPI extensions are not a standard part of the GLUE schema. These attributes will need to be made permanent so that MPI support is part of the standard OSG configuration.
- Support for external login node – create an external Franklin login node for grid access. This will simplify the root node configuration, and make the system more manageable.
- Shared project accounts – investigating the feasibility of grid VO based project accounts to share jobs and data, while maintaining individual traceability.

Conclusion

Using the Open Science Grid interfaces we have successfully created transparent user-friendly interfaces into the NERSC Franklin Cray XT4 system. This has allowed scientists from a wide range of fields to run parallel HPC jobs across multiple OSG sites, without having to deal with the complexities of the individual system. For instance, this has enabled the Franklin system to be used for grid-based weather forecasting models and fusion research simulations. Along with the advent of web gateways, we expect this model to become increasingly common as a user-friendly method to access HPC resources.

Acknowledgments

This work was supported by the Director, Office of Science, Office of Advanced Scientific Computing Research of the U.S. Department of Energy under Contract No. DE-AC02-05CH11231.

About the Authors

Shreyas Cholia is a software engineer at the NERSC. He is primarily responsible for managing the grid infrastructure at NERSC, in conjunction with the Open Science Grid. He has been involved with various grid projects since 2002. Prior to his work at NERSC, Shreyas was a developer and consultant for IBM with the HPSS project. Email: scholia@lbl.gov

Hwa-Chun Wendy Lin is a Systems Engineer at NERSC and the Backup Analyst of Franklin. She also helps with the system side of the grid work. Before joining NERSC, Wendy worked for Purdue University. She was a member of the Purdue TeraGrid team, made the IBM SP resource available to TeraGrid users, and helped build the nanoHUB science gateway. E-mail: hclin@lbl.gov.

The Authors can be reached at Lawrence Berkeley National Laboratory, 1 Cyclotron Road, MS 943-256, Berkeley 94720, USA.

References

[1] "What is the Grid? A Three Point Checklist". Ian Foster. Argonne National Laboratory & University of Chicago. July 20, 2002
[2] "About Franklin". From http://www.nersc.gov/nusers/systems/franklin/about.php
[3] "Notes for Networks and Parallel Processing". Aaron Harwood. Melbourne School of Engineering. 2004. From http://www.cs.mu.oz.au/498/notes/node40.html
[4] "Introduction to Parallel Computing". Blaise Barney, Lawrence Livermore National Laboratory. From https://computing.llnl.gov/tutorials/parallel_comp/
[5] "Open Science Grid Admin Setup MPI". From https://twiki.grid.iu.edu/bin/view/Sandbox/AdminSetupMPI
[6] "Generic Information Providers". From https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/GenericInformationProviders
[7] "Open Science Grid 1.0.0 Release Documentation". From https://twiki.grid.iu.edu/bin/view/ReleaseDocumentation/
[8] "BRO Intrusion Detection System". From http://www.bro-ids.org/
[9] "Pacman 3". From http://atlas.bu.edu/~youssef/pacman/