TeraGrid's Integrated Information Service

Lee Liming[1], John-Paul Navarro[1], Eric Blau[1], Jason Brechin[3], Charlie Catlett[1], Maytal Dahan[5], Diana Diehl[4], Rion Dooley[5], Michael Dwyer[4], Kate Ericson[4], Ian Foster[1], Ed Hanna[6], David L. Hart[4], Chris Jordan[5], Rob Light[6], Stuart Martin[1], John McGee[7], Laura Pearlman[2], Jason Reilly[7], Tom Scavo[3], Michael Shapiro[3], Shava Smallen[4], Warren Smith[5], Nancy Wilkins-Diehr[4]

[1]The University of Chicago/Argonne National Laboratory, [2]Information Sciences Institute, University of Southern California, [3]National Center for Supercomputing Applications, [4]San Diego Supercomputer Center, [5]Texas Advanced Computing Center, [6]Pittsburg Supercomputing Center, [7]Renaissance Computing Institute

{ liming, navarro, blau, smartin }@mcs.anl.gov, { catlett, foster }@anl.gov, laura@isi.edu,
{ brechin, mshapiro, trscavo }@ncsa.uiuc.edu, { mcgee, jdr0887 }@renci.org,
{ dhart, diehl, kericson, mdwyer, ssmallen, wilkinsn }@sdsc.edu,
{ ctjordan, dooley, maytal, wsmith }@tacc.utexas.edu, { ehanna, light }@psc.edu

Abstract – **The NSF TeraGrid project has designed and constructed a federated integrated information service (IIS) to serve its capability publishing and discovery needs. This service has also proven helpful in automating TeraGrid's operational activities. We describe the requirements that motivated this work; IIS's system architecture, information architecture, and information content; processes that IIS currently supports; and how various layers of the system architecture are being used. We also review motivating use cases that have not yet been satisfied by IIS and outline approaches for future work.**

Keywords – *High-performance computing systems, distributed systems, information services, information federation, service registries, directory services, schema, TeraGrid.*

I. Introduction

As operators of the NSF-sponsored TeraGrid, we have a critical need to describe, track, and publish a diverse set of hardware and software capabilities offered by independently operated resource providers. TeraGrid's federated nature complicates this task in a way that is not uncommon in other large distributed systems.

A centralized capability database has several disadvantages: (1) it makes content owners depend on someone else's information system, (2) databases typically have rigid schemas, (3) databases easily get out of sync with reality, and (4) merging local information from federation participants into a central database can be challenging. To address TeraGrid's requirements, we designed and deployed an integrated information service (IIS) with a federated model for publishing, aggregating, and accessing capability information. Section II describes this motivation and background in more detail.

Quite different from a central database approach, IIS's architecture is similar to the World Wide Web indexing model, in which content is published locally by many sources and separately indexed by agents such as Google. Section III describes IIS's system architecture, the software components we assembled to implement IIS's distributed architecture.

Another key aspect of IIS's design is an agile approach to content and formats. This approach has allowed us to be highly responsive to changing user requirements, both from information publishers and consumers. Section IV describes IIS's information architecture: how we model TeraGrid capabilities as data, and how that data is organized.

IIS has significantly improved our ability to automate key pieces of TeraGrid's operations and to provide users and user applications with up-to-date and useful information about TeraGrid and its capabilities. Although we have encountered some scaling limitations, IIS's overall system design provides high reliability and scalability. Section V describes how TeraGrid users and operations currently use IIS to enhance their work. Section VI summarizes the challenges that IIS has successfully addressed, and suggests areas for future work.

II. Describing An HPC Resource Federation

Our creation of TeraGrid's IIS was motivated by a need to describe TeraGrid's federated high-performance computing (HPC), storage, and visualization resource capabilities. Federation membership changes over time, as do the capabilities offered by members. Above all, our primary sponsor, NSF, requires that our federation present itself to users as a coherent service, and not as a collection of separate services. This combination of requirements motivated development of an information service and architecture that together can accurately and coherently collect, track, and present the diverse federated system capabilities.

A. National-Scale HPC Federation

TeraGrid was commissioned in 2001 by NSF [1] as a single Teraflop/s research facility—the Distributed Terascale Facility, or DTF [2]—with four locations and common system architecture, a high-performance wide-area network, and distributed computing tools such as Globus middleware. By the end of the three-year construction, NSF had added seven HPC systems with unique architectures and applications and charged the expanded TeraGrid program with supplying coordinated, comprehensive, and production-quality HPC services to support general U.S. academic research [3]. TeraGrid thus became a federation of many independently operated HPC centers, each serving a variety of user needs and focused on specialized types of HPC applications.

When commissioned in 2004, this federation presented itself to the academic community as a full-service suite of HPC services with a single, unified process for applying for access and obtaining support. It proved significantly more difficult, however, to provide a consistent user environment across a dozen HPC platforms aimed at different application types.

B. Consistent User Environment

Our initial strategy for a consistent user environment was the Common TeraGrid Software Stack (CTSS-1): a specific list of software components that all TeraGrid system had to install and make available to users. By providing the same components on all TeraGrid systems, we would reduce our users' needs for specialized training and orientation to new systems and would encourage users to use a wider variety of TeraGrid's resources.

This approach worked moderately well for the DTF--four systems with common system architecture--but not when we tripled the number of systems and incorporated several new architectures aimed at new application types. Key problems included: (1) many CTSS components were poor matches for some new system architectures, (2) some CTSS components required considerable porting effort to more exotic platforms, (3) some components required expensive licenses that couldn't be imposed on newer TeraGrid members, (4) the CTSS definition lacked explicit use cases or user scenarios informing users and operators of targeted functionality, and (5) the periodic need to upgrade individual CTSS components impact users in unexpected ways when upgrading the entire stack as occurred between CTSS-1, CTSS-2, and CTSS-3.

Responding to these issues, in 2006 we introduced a new strategy for consistent user environments. The Coordinated TeraGrid Software and Services (CTSS-4) was based on a capability model, in which the common user environment was defined in terms of user capabilities, not software components. Each user capability had an associated set of use cases enabled by the capability and a recommended implementation of software components. Most importantly, all but one of these capabilities were optional and each resource could choose the capabilities suited for its system architecture and target uses and users. In addition, capabilities could be developed, deployed, and upgraded independently of each other.

The capability model has proven to be significantly more workable for a distributed federation of resource providers and diverse resources. Both users and resource providers appreciate a system definition based on modular user capabilities instead of software components. It made sense that not all capabilities were appropriate for all HPC platforms. The model introduced a new set of challenges, however, most notably: (1) the need to enable autonomous resource provider decisions about which capabilities to offer on their resources, (2) the need to document those capability choices, (3) the need to document technical details necessary for users to discover and use available capabilities, and (4) the need to maintain this information accurately as both resources and capabilities evolved over time asynchronously.

C. Federated Information Service

Our shift from the software stack model to the capability model made it necessary to create a TeraGrid-wide capability information service. This need was not new. From the beginning, TeraGrid had envisioned an information service that could be used for automatic resource selection (i.e., meta-scheduling or resource brokering), workflow configuration, and other advanced computing purposes [4][5]. Indeed, TeraGrid already had specialized central databases. However, keeping these databases up-to-date was a challenge in the face of a federated membership. Our new pressing need was to tell our users which TeraGrid coordinated capabilities were available on each TeraGrid system. This information had to be highly reliable and current, and it had to be available in time for the release of CTSS-4.

The most fundamental requirements for TeraGrid's information service were as follows. (1) Information in the system must be more current than user documentation, which was known to lag the system status by weeks or months in some cases. (2) All federation members must have control of the information about their own resources. Our experience with central databases was that keeping a database in sync with reality was a challenge. (3) Capability descriptions must include technical details specific to each capability. Though a small set of standard fields could satisfy many user needs, we soon found users demanding more details than could be expressed in standardized fields. (4) Authenticated access is required for some information (e.g. individual job information)—but is infeasible for information that had to be public. (5) A variety of access methods and formats must be supported. New information consumers often had unique integration requirements, requiring us to support a variety of protocols (e.g., WS/SOAP, WS/REST) and data formats (e.g., HTML, XML, CSV, JSON).

III. System Architecture

We believe that there are two key elements to the success of TeraGrid's IIS. Section IV discusses IIS's flexible information architecture and the content published via IIS. Here, we present IIS's system architecture: the distributed collection of components, services, and interfaces and how their interactions achieve our high-level goals of an authoritative, accurate and reliable capability information service. We also discuss the security and operational reliability requirements; and the requirements of information consumers and discovery clients for high-performance, robust, and simple information access.

A. Distributed Design

TeraGrid resource and service providers do not have exclusive relationships with TeraGrid. They also maintain local users and resources and they may participate in other federated systems such as the Open Science Grid [6]. This was a key consideration in our basic IIS design. In particular, we realized that an information service that could leverage information already provided for other federations and that could easily be re-used in other federations would be more likely to be kept up-to-date than one that involved maintaining a separate copy of the data solely for TeraGrid. In short, resource and service providers should be able to maintain a single local set of descriptive data and be able to publish that data in any of the federations to which they belong.

TeraGrid resource and service providers operate services for their users, and are in the best position to accurately describe them. We recognized that in our IIS, publishing information about services should be the privilege and responsibility of the service provider, accomplished via a simple local information service. TeraGrid should be responsible for aggregating information from its constituent members to enable collaboration-wide discovery.

A distributed publisher/aggregator model also improves the availability, reliability, and serviceability of aggregation services. These benefits are described more fully in the section below on Aggregation and Caching Design.

B. High-Level Components and Interfaces

We chose the Globus MDS4 [7][8] and WebMDS tools [9] for the initial IIS implementation because they offered a robust, secure, and flexible framework for distributed information publishing, aggregation, indexing, and access. As our implementation evolved it became clear that MDS4 could not satisfy all of our information persistence and aggregation requirements and that WebMDS and WS/SOAP could not satisfy all of our client requirements, particularly the need for lightweight, agile interfaces. While leveraging the strengths of MDS4 [10] and WebMDS, we addressed these limitations by adding a custom aggregation and caching system and REST interfaces based on Apache 2. This multi-tool strategy has given us the flexibility to meet existing functional requirements and to adapt to new requirements.

Our system architecture calls for two types of information services. Each service provider within the federation operates a local information service that publishes its local capabilities. These local registries are not expected to handle heavy client loads, nor do they need to be available all of the time. They do, however, support authentication in order to offer restricted access to some data. A central information index aggregates and re-publishes this information for use by others. The central index is engineered for high availability, in part using redundant servers at multiple locations in a "hot standby" configuration. Figure 1 illustrates this high-level architecture. On the left are multiple local service provider information services, each of which publishes content to TeraGrid-wide indices (of which there are several for redundancy). On the right are clients, each of which accesses the central index via a dynamic name service entry (info.teragrid.org) that can rapidly and transparently redirect clients to a backup index in the event of a failure.

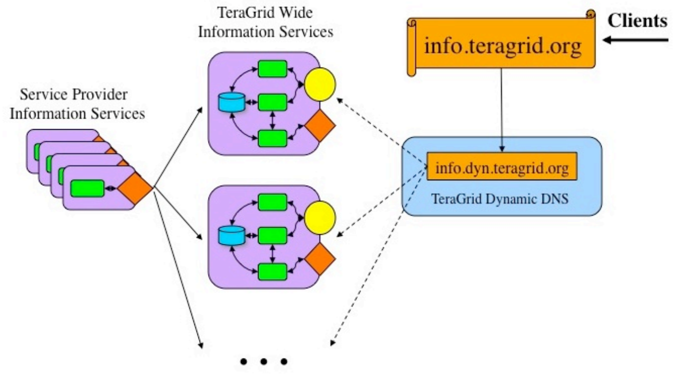


Figure 1. High-level IIS components

C. Local Capability Registries

Most local capability registries are implemented as Globus MDS4 Index Services. These MDS4 services publish XML documents produced by information-gathering plug-ins (known as "information providers" in MDS4 terminology). MDS4 services authenticate to upstream TeraGrid-wide aggregation servers using GSI credentials [11], ensuring secure and authoritative information aggregation. TeraGrid-wide aggregation servers restrict registration and aggregation to known local information services.

Local capability registries may also be implemented as REST-compatible web services, so long as they are able to generate XML documents using the recognized schemas. When a REST service is used for publishing, central information services are configured to pull information from local services.

When we create software installers that implement TeraGrid capabilities, we include IIS registration files that describe the capabilities and associated components. These files are in simple "keyword = value" format making them easy to maintain. We also provide registration file translators to generate TeraGrid standard XML documents. CTSS-4 installation instructions include the steps to add this data to the local capability registry, By tightly coupling capability publishing with capability deployment, we have made it easy for service providers to keep their service descriptions synchronized with their actual deployments.

If a service provider has some or all of the data they need to describe their capabilities in local systems, they can write scripts to generate or update registration files, or can bypass registration files and generate TeraGrid standard XML documents directly.

D. Aggregation Design

IIS's central indices are based on the MDS4 Index Service. The Index Service is designed to participate in a hierarchy of information services: local MDS instances register their local information with higher-level instances that aggregate the data from lower-level instances. This built-in mechanism uses a soft-state (automatically expiring) registration pattern in order to prune out-of-date information if a lower-level registry disappears without explicitly de-registering [4]. This mechanism was designed with system monitoring in mind, where the goal is to track the current status of every system element. IIS initially used this aggregation framework, but we soon found that it was not a good match for our capability registry requirements.

Like other services, local service registries are subject to outages. These outages may be planned or unplanned and may be quite short (a few minutes) or quite long. It is not uncommon for a TeraGrid resource provider's network or machine room to be offline for several days at a time. Using the built-in aggregation mechanism, if a local MDS instance fails to renew its registration during a service or network outage, the information it had been reporting will not persist in the central aggregation service. Also, the MDS4 index service stores its aggregated data in memory, so if the central index resets for any reason while a capability registry is unavailable, the data from that registry will not reappear in the central index until the registry returns. Both of these outcomes are unacceptable. It is vital that the information that describes services persist in TeraGrid-wide views even when the local registry is not currently available.

Our solution was to implement an external aggregation mechanism on the central indices. Local service registries use MDS4 mechanisms to register their existence and contact information with the central indices. Once registered in the central index, a set of scripts on the central servers—run periodically by the cron mechanism—queries these registries and stores valid results in persistent files. An information provider plug-in for the

MDS4 Index Service assembles the data from the cached files for central service publishing. This on-disk content caching satisfies our persistence requirements.

E. High-availability Engineering

Central information services are often the first contact point for users and services discovering TeraGrid's capabilities. We designed IIS to provide continuous, high availability (99.5% uptime) service for client-side uses.

To achieve high availability, all local information services register to two redundant central index services. These services operate independently of each other at separate physical facilities in a primary and hot-standby configuration. Both contain the latest aggregate data from all service provider registries. Using TeraGrid's "dyn.teragrid.org" dynamic DNS domain we can redirect the info.teragrid.org hostname from one server to the other within 15-minutes—the time it takes our dynamic DNS updates to propagate through all Internet DNS caches. This design helped us meet the requirement for highly available information.

To ensure the synchronization of the central indices' software environments, we automate non-intrusive server comparison, allowing us to easily propagate changes between peer servers. We also swap primary and standby servers on a routine basis to ensure that the hot-standby server is always ready to assume primary duty. Through these processes, we have provided better than 99.5% availability for more than a year despite hardware failures, network outages, power outages, and host system OS and hardware upgrades. It has also made the operational process of switching servers—whether due to failure or other reasons—a routine event.

F. Client Interfaces

Aggregated information is published using simple REST interfaces [12][13], HTTP WebMDS interfaces, and MDS4's web-service SOAP interfaces. All of these interfaces access the same aggregated information.

We chose REST interfaces as our primary and preferred information-publishing interface because it eliminates the need for special client tools or libraries, because the interface is simple and easy-to-use, because the interface style is familiar to a large number of Web developers, because it is amenable to engineering techniques for high scalability and performance, and because there is significant interest amongst discovery client developers.

The WebMDS interface accesses MDS4 native XML content, can convert that content into a variety of formats using XSLT transforms, and then serves the results via standard HTTP to client programs or web browsers. The MDS4 WS/SOAP interface provides a wider range of functionality, including subscription and notification features and authenticated access for restricted data.

To support interactive users and scripted discovery, we also developed a command line discovery client (written in Perl for wide portability) called 'tginfo' [14]. It can access IIS from any computer with Internet connectivity.

IV. INFORMATION ARCHITECTURE

The second key element to the success of TeraGrid's IIS is the flexible information architecture and the published content. This section describes that content -- the data entities and relationships that we use to describe TeraGrid's characteristics. This information architecture achieves important goals independent of the IIS system architecture.

A. The Generic Capability

Wikipedia defines Information Architecture as "the art of expressing a model or concept of information used in activities that require explicit details of complex systems." [15]. TeraGrid is a complex system of many compute, storage, network hardware, software, and service components. Describing it is a challenge because there are many ways to do it, and many perspectives on what characteristics need to be described.

Rather than describing a complex infrastructure by describing hardware and software components and their interactions, we designed a user-focused information architecture and introduced the generic "capability" architectural element. Capabilities are collections of software, services, and information that enable specific user activities, or use cases. For example, most TeraGrid systems allow users to locally submit batch jobs. To enable job submission from anywhere inside or outside TeraGrid, we defined the Remote Computation capability. An HPC system that offers the Remote Computation capability provides remote job submission via one or more of the interfaces documented in the Remote Computation capability definition.

Since capabilities are abstractions, we were able to define a small set of universal elements and attributes (see Figure 2) that we felt were important to users and operators. These entities and attributes identify a particular instantiation of a capability by describing the resource the capability is deployed on, the institution that operates that resource, and the capability name and version. We provide a short capability description, a capability class, current and target support levels, a URL where the capability status can be verified, and the institution name and contact location that provides support for the capability.

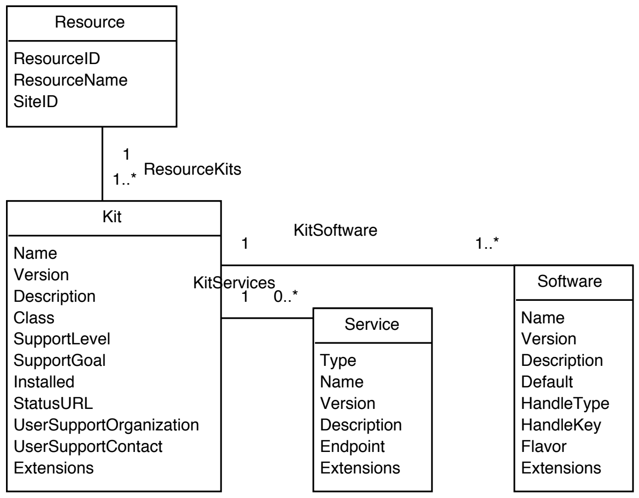


Figure 2. Capability entities and attributes

The capability kit is the information architecture element that binds all TeraGrid IIS information. When specific capabilities need to advertise additional information, they publish separate schemas that are referenced in their capability registration *Extensions*. Additional extension information may apply to the entire capability kit, or an individual capability service or software component. When clients need to discover capability information, they first look for capabilities that support use cases of interest, then examine the common attributes to find where the capability is available, it's operational status, support information, and included software and service components.

We note that these entities and attributes do not use the terms TeraGrid or CTSS. We use this structure to describe capabilities in the extended TeraGrid community that are not directly deployed, coordinated, or supported by TeraGrid. TeraGrid elements are identified by the suffix "teragrid.org" in the SiteID, ResourceID, or capability kit name.

The native IIS data format is XML. By defining extensions elements in TeraGrid XML schemas we can aggregate and publish arbitrary additional information with little or no development.

B. TeraGrid Capabilities

CTSS-4 was introduced in 2007 with eight capabilities that covered the original CTSS-3 components. In less than two years the number of capabilities has doubled. Initially, IIS covered CTSS capabilities that were coordinated across compute and storage resources. Now, IIS also describes local and gateway capabilities that are not coordinated. The diversity of the capabilities continues to grow significantly. IIS allows users and members of the TeraGrid federation to maintain a coherent and up-to-date view of our capabilities.

Below we list production or near-production CTSS-4 capabilities. This list highlights the broad scope of TeraGrid capabilities and demonstrates how the capability kit abstraction is used in practice. All capabilities publish standard information, and some publish custom information. All published information is discoverable through IIS.

Basic Capabilities - TeraGrid Core Integration [16] allows local capability publishing in IIS. Remote Login enables access to a TeraGrid standard command shell using local or GSI credentials and supports TeraGrid's single sign-on capability.

Computation Capabilities - Remote Computation allows computation tasks to be submitted and managed remotely. Science Gateway Support enables science gateways to submit jobs using a single credential per gateway with embedded user attributes so that resource providers can track and manage end user access and restrict the capabilities of gateway accounts. Advance Reservation allows users to establish a reserved time at which they will be guaranteed access to a service. Co-scheduling allows users to establish a time when they may use multiple services at the same time. Application Development & Runtime Support enables code development and execution by registering local programming languages, tools, and libraries and by providing a baseline set of runtime tools. Science Workflow Support manages large numbers of parallel tasks and workflows for users. Parallel Application Support enables MPI-based parallel applications by registering local MPI tools and libraries. Distributed Parallel Application Support enables MPI-based parallel applications to be built and executed using multiple compute systems.

Data and Storage Capabilities - Data Movement provides the ability to transfer files to/from a storage system, including mass storage. Data Management allows users to manage large collections of data and meta-data in mass storage. Data Visualization provides a standard set of data analysis and visualization tools. Wide-area GPFS and Wide-area Lustre provide access to shared high-performance global file systems.

Uncoordinated Capabilities - Local HPC Software allows local HPC software tools and libraries to be registered in IIS. Science Gateway Applications allows scientific applications that are currently available via science gateways to be registered in IIS. User Profiles allows the TeraGrid User Portal and authenticated science gateways to obtain user profile, resource, and allocation information.

C. Queue and Scheduler Load Information

The TeraGrid User Portal needs dynamic TeraGrid resource load and job information for display in the System Monitor. The System Monitor has always been one of the most prominent features of the user portal and must have reliable and accurate information. Prior to IIS, the portal team ran custom cron jobs from personal accounts on RP resources to gather information. With over twenty resources, this approach requires significant support. Using IIS, we were able to integrate load and queue publishing into RP local information services, have the information published and aggregated securely, and have the user portal and other consumers access it from IIS.

D. Resource Descriptions

TeraGrid's Core Services 2.0 [17] planning effort recognized the need for a definitive source of current and historical TeraGrid resource descriptions to address the challenges of resource federation. This central Resource Description Repository (RDR) [18], currently under development, will model resources as implemented by various central TeraGrid services, including the allocations and accounting system, Resource Catalog, and User Portal. Beyond current needs, the RDR was envisioned to support more complex resource models and enable more complex functions within TeraGrid's central services.

As currently deployed, IIS does not fully satisfy RDR requirements for maintaining historical information and tracking changes to certain attributes over time, for example when the number of nodes or processor speed in a resource is upgraded.

Resource attributes for RDR were defined in consultation with the maintainers of existing central services that stored resource information—the TeraGrid Central Database (TGCDB), POPS (for allocation requests), the Resource Catalog—as well as the maintainers of information already published via IIS. The descriptive attributes for resources were initially broken down into two major sets: a "common" set that applies to all resources, and a "compute" set that applies only to a computational resource. The common set includes such fields as the organization, resource name, and resource type. The compute set encompasses attributes related to the hardware and operational environment of the HPC compute systems.

To ensure rapid adoption and validate values critical to the functioning of TeraGrid's core services, the RDR will facilitate data entry of resource information, store the history of changes to this resource information, and provide an interface to enable RPs to create valid IIS XML data files. A RESTful web service [12] will allow RPs to automatically or manually download their resource information from the RDR to publish using IIS.

The goal is that all consumers of RDR information, including TeraGrid's central services, will access resource descriptions from IIS. As RPs become more familiar and comfortable with IIS, we will enable RPs to publish resource data directly to IIS for ingestion into the RDR.

E. Science Gateway Information

Science gateways provide scientists and engineers with additional capabilities and collaboration opportunities that are tailored to specific research communities. Gateways are customized applications (often web portals) that make use of TeraGrid computation and storage services on behalf of their users.

Science gateways use IIS both as content consumers and as content publishers. Gateways frequently use TeraGrid's remote computation, remote login, data management, and data movement capabilities. When using these capabilities, gateways need to be able to look up details such as the service endpoints, the local software available on a system, the current load average, or the job queue length. But gateways also publish data in IIS. The following examples demonstrate how science gateways are both publishers and consumers of IIS information.

The gateway registry is used to maintain metadata about over thirty gateways from diverse science disciplines. We help scientists and engineers discover gateways that may be of use to them by maintaining and publishing gateway information through various channels, including IIS.

Most gateways develop and deploy web services that can be invoked by a user from their work process pipeline via third-party clients, like Taverna [19], or by directly coding against the web service. While these web services have proven invaluable for broadening access to TeraGrid, scientists are often unaware of the availability of gateway capabilities. The Gateway Application Web Service Registry (GAWSR) will raise awareness of the scientific applications made available via gateways by defining a standard schema (GAWSR-XSD) for describing gateway services and by aggregating the information from many gateways to facilitate discovery. This mechanism will offer gateway developers a re-usable way to publish their gateways' capabilities and will offer discovery clients a standard schema and content to search. By aggregating the GAWSR metadata in IIS, the challenging task of finding these web service endpoints can be accomplished by a single IIS search. It may even be possible to invoke the web services based on the metadata available in the GAWSR. TeraGrid's education, outreach, and training will also benefit by providing a searchable store of scientific resources.

Since science gateways may serve hundreds or thousands of end users, gateways typically submit job requests under a community credential: a single security credential that is used for all requests from the gateway. For each request, the gateway embeds information about the end user in the community credential using the GridShib SAML Tools [20], which leverage Security Assertion Markup Language (SAML) [21] to communicate information to resource providers. This frees resource providers from having to maintain long lists of end gateway users while still giving them the ability to know their end users for the purposes of auditing and incident response. To prevent abuse and auditing errors, resource providers verify that the request is from a gateway and that the asserted end user identity is valid for the gateway that is reporting it. To this end, IIS provides a scalable and secure way to aggregate information from various data sources and to produce a security configuration file (in SAML Metadata format [22]) for TeraGrid resource providers to configure their WS GRAM service with GridShib for GT [20] in the Science Gateway Support capability. In this way, a resource provider verifies that the end user identity reported in a gateway request is valid.

F. Advanced Scheduling Information

During the past year, TeraGrid has deployed several metascheduling tools that enable users to submit jobs to TeraGrid as a whole, instead of specific systems. A metascheduler chooses where jobs should execute based on the requirements and preferences of each job and typically also manages the execution of jobs on the systems to which they are matched. Several tools are being deployed and while some (e.g., Moab [23] and MCP [24]) gather information themselves or require only a minimum amount of information, other tools (e.g., Condor-G [25], GridWay [26], and Swift [27]) require externally provided information. TeraGrid is deploying information gatherers to support this second class of metaschedulers.

The primary user of the gathered information is, at present, Condor-G matchmaking [28]. A Condor-G user submits a job to Condor and Condor then uses Globus [29] mechanisms to execute the job on a remote computer system. A Condor-G user specifies an exact system to execute the job on so Condor-G adds only fault tolerance and job tracking functionality atop Globus. A user of Condor-G with matchmaking specifies constraints and preferences for the system that their job will be run on. Condor-G selects a system that satisfies the constraints and optimizes the preferences and then manages the execution of the job on that system. A secondary goal of this information gathering is to be able to support other metaschedulers, such as Gridway, or custom resource selection tools deployed by TeraGrid science gateways. Fortunately, the set of information that metascheduling tools need is relatively consistent. This information includes hardware information (nodes, CPUs, memory, file systems), software information (operating system, available Grid services), and scheduling information (queues, queue limits, load). There is some variation and some tools, such as Condor-G, allow full customization of the information that is provided about systems.

To provide information to metaschedulers, we had to identify what information they needed, select or create a schema for this information, create information gatherers, and integrate the gathering of this information into IIS's information architecture. We determine what information to gather by examining the needs of several metascheduler implementations and deployments. We were familiar with the Grid Laboratory Uniform Environment (GLUE) schema for describing Grid sites and version 2 of the GLUE schema was under development in the Open Grid Forum (OGF) [30]. Version 2 has recently been released as a recommendation [31]. We chose GLUE 2 because it met our initial needs, allows for extension, and will allow us to interoperate with other grids.

We initially implemented information-gathering scripts that gather scheduling-related information from the LoadLeveler, LSF, PBS, and SGE batch scheduling systems and produce XML in an early version of the GLUE 2 schema. Resource providers publish this GLUE 2 XML document via local information services. TeraGrid wide information servers periodically aggregate across all resource providers. We note that while MDS4 supports periodically executing information provider scripts, the execution time of our GLUE 2 information-gathering scripts are long enough that MDS4 frequently times out waiting for them. We therefore generate GLUE 2 XML documents asynchronously.

GLUE 2 XML documents describing many of TeraGrid's member systems are now available in IIS. This information can be retrieved and processed for incorporation into metaschedulers. For example, using IIS information we can produce Condor class ads of TeraGrid systems and queues. These ads can be viewed on many TeraGrid systems by simply by executing 'condor_status.'

G. Local HPC Software

Most TeraGrid resource providers have locally supported software not coordinated, documented, or supported by TeraGrid as a whole. Resource providers have a variety of local catalogs and several use the HPC Software Catalog [32] application developed and supported by NCSA. TeraGrid users are interested in all available software; they are less concerned about whether specific software is considered TeraGrid- or RP-supported.

To enable users to discover all available software, in 2008 TeraGrid started developing a new Local HPC Software publishing capability using IIS. IIS will aggregate local RP software information, and combine it with CTSS-4 software information, and eventually Science Gateway application registry information to enable TeraGrid users to discover all available software, including how to access the software and obtain support.

H. Accounting and Allocations

The TGCDB is the cornerstone of TeraGrid's allocations, accounting, and user identity infrastructure. Thus far, TeraGrid has provided direct, custom interfaces to subsets of TGCDB information destined for users (e.g., the TeraGrid User Portal and the tgusage command-line utility) and for staff (via online monitoring interfaces). As we make more TGCDB information available to outside services, we are using IIS as a general-purpose mechanism for publishing information, particularly for programmatic purposes. The first TGCDB information to be published via IIS was resource identification and cross-reference data, providing a rudimentary linkage between other resource information being published via IIS and the information known to TeraGrid's central services. Attributes published by the RDR (described above) will supersede this early resource information.

More recently and more representative of the potential capabilities, TGCDB began publishing information about the resources available to a given allocated project (i.e., the set of resource authorizations associated with a given research group). This service was initiated to serve both TeraGrid science gateways and TeraGrid's scheduling activities. In both cases, the objective is to enable a service to discover the resources on which a gateway or an individual user (who provides a valid project identifier) should be able to run.

A new User Profile Service will provide a RESTful interface for users, portals, and gateways to consume accounting and allocation information from TGCDB. Authenticated users can discover usage information, resource allocations, and colleague activity and account status. This fills a need in the overall TeraGrid architecture for a way for gateways to quickly discover user-appropriate information. It also opens the door for gateway developers to more effectively integrate value-added services with the existing TeraGrid resources.

V. LEVERAGING IIS

IIS was motivated by the need to provide authoritative and up-to-date information to users about the capabilities available within TeraGrid. We present here how user-facing systems, such as user documentation and the user portal, are leveraging IIS to achieve TeraGrid's goal of presenting users with accurate and up-to-date capability information. We also summarize how our TeraGrid's operational practices have significantly benefited from both the IIS architecture and information content. Finally, we review how emerging capabilities leverage TeraGrid's IIS to publish capability specific information, and how this approach is making a significant impact on capability discovery by both users and software systems.

A. TeraGrid User Documentation

As the diversity and heterogeneity of TeraGrid resources grew, so did the demands for exceptions and special cases in documentation. While resources diverged from the original configurations, their documentation requirements also became nonstandard and unique. The ability to keep user information accurate and relevant was a growing challenge. In response to this, TeraGrid User Support documentation began exposing dynamic data for resource-specific information.

Beginning with CTSS-4, users can now look up the availability of TeraGrid capabilities in any of four common ways. Users may view tables showing the capabilities, software components, and service interfaces offered by specific TeraGrid resources. Users can also search for TeraGrid resources that support specific software or services, specific CTSS capabilities (by name), or specific use cases. This approach provides a fuller view of the capabilities available on TeraGrid through a user-friendly interface integrated into familiar documentation Web pages.

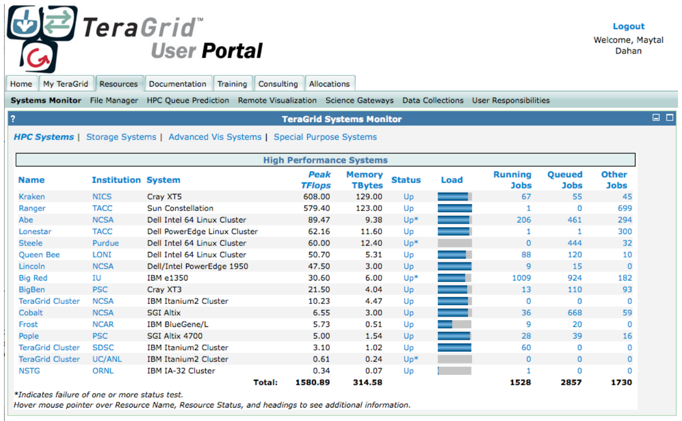
Throughout TeraGrid's user documentation, we provide many lists of resources, services, and configuration details that grow and change frequently. Before IIS, these lists were often out of date because the system changed more rapidly than the documentation could be updated. Now, user documentation obtains live data for many of these lists from IIS, and the content is always current. This documentation is used between 300 and 600 times per month. (TeraGrid has roughly 2,000 active users.)

Another important topic covered by TeraGrid's user documentation is the availability of system-specific software that is not coordinated with other TeraGrid systems. The HPC Software Catalog was designed as a searchable repository of all non-CTSS software on TeraGrid resources, organized by site and resource. This feature is currently being redesigned to be more inclusive and to integrate software from CTSS as well as local non-CTSS collections and Science Gateways into a single, seamless, searchable repository. Since users typically do not care where the software they want to use ultimately comes from or by whom it is supported, it makes sense to combine all of these assets into a single collection and present the results together. This new design also leverages IIS, making the information available to all services.

For users, this new search interface greatly improves their ability to identify optimal resources by helping them find combinations of compiler, workflow, job execution and data movement software in a single resource, or combinations of resources that can support their specific job, scientific data handling, and workflow software needs.

B. TeraGrid User Portal

One of the most visited pages on the TeraGrid User Portal (TGUP) [33] is the System Monitor. The System Monitor provides basic descriptive information (name, institution, type, etc.), resource attributes (number of processors, peak performance, total memory, etc.), plus current status, load, and jobs queued. Thus, the System Monitor presents both dynamic and static information in an interface that helps users decide which machines are appropriate for their computational work and which are less heavily loaded and more likely to execute jobs sooner. Figure 3 is a snapshot of the System Monitor.



High Performance Systems

Name	Institution	System	Peak TFlops	Memory TBytes	Status	Load	Running Jobs	Queued Jobs	Other Jobs
Kraken	NICS	Cray XT5	608.00	129.00	Up		67	55	45
Ranger	TACC	Sun Constellation	579.40	123.00	Up		1	0	699
Abe	NCSA	Dell Intel 64 Linux Cluster	89.47	9.38	Up*		206	461	294
Lonestar	TACC	Dell PowerEdge Linux Cluster	62.16	11.60	Up		1	1	300
Steele	Purdue	Dell Intel 64 Linux Cluster	60.00	12.40	Up*		0	444	32
Queen Bee	LONI	Dell Intel 64 Linux Cluster	50.70	5.31	Up		88	120	10
Lincoln	NCSA	Dell/Intel PowerEdge 1950	47.50	3.00	Up		9	15	0
Big Red	IU	IBM e1350	30.60	6.00	Up*		1009	924	182
BigBen	PSC	Cray XT3	21.50	4.04	Up		13	110	93
TeraGrid Cluster	NCSA	IBM Itanium2 Cluster	10.23	4.47	Up		0	0	0
Cobalt	NCSA	SGI Altix	6.55	3.00	Up		36	668	59
Frost	NCAR	IBM BlueGene/L	5.73	0.51	Up		9	20	0
Pople	PSC	SGI Altix 4700	5.00	1.54	Up		28	39	16
TeraGrid Cluster	SDSC	IBM Itanium2 Cluster	3.10	1.02	Up		60	0	0
TeraGrid Cluster	UC/ANL	IBM Itanium2 Cluster	0.61	0.24	Up*		0	0	0
NSTG	ORNL	IBM IA-32 Cluster	0.34	0.07	Up		1	0	0
		Total:	1580.89	314.58			1528	2857	1730

*Indicates failure of one or more status test.
Hover mouse pointer over Resource Name, Resource Status, and headings to see additional information.

Figure 3. User portal system monitor

The system monitor is based on the GridPort Information Repository (GPIR) [34]. GPIR provides a place to store information about Grid resources that is readily accessible via web services to the portal. As described in IV.C the first version of TGUP relied on custom scripts running from developer account and sending information to a GPIR Ingester web service. This design was a challenge to support. The System Monitor was improved by incorporating publishing scripts in resource provider information services, having IIS aggregate that information, and having the user portal use a single provider script that accesses IIS aggregated data and sends it to the GPIR service.

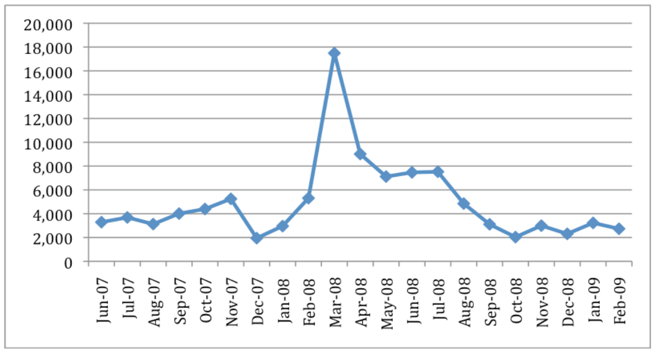


Figure 4. User portal system monitor usage: accesses per month

Figure 4 shows TeraGrid User Portal system monitor usage. The system monitor usage is consistently a highly used feature of the user portal with peaks during March 2008 due to the release of the new Ranger system, which generated attention to TeraGrid and the User Portal. The reduced frequency in the system monitor beginning in August 2008 is due to the release of the new system monitor, which uses JavaScript to display data and thus does not require a server refresh for each user request.

Future work on TGUP will include expanding the IIS services used within the portal. We expect to include metascheduling services, resource information, local software services, and more as user requirements harden.

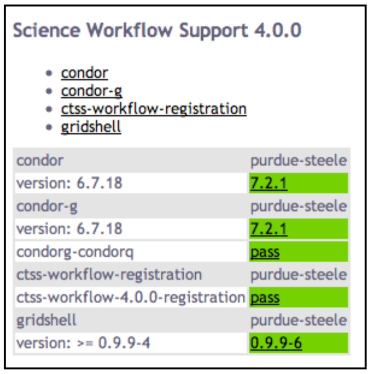
C. Operations Verification and Validation

As section III describes, resource providers are in the best position to describe accurately the capabilities they provide. However, it is equally important to have external verification of the published information to ensure the highest quality of service. In TeraGrid, a Grid monitoring tool called Inca performs the external verification of registered capabilities.

Inca detects infrastructure problems by executing automated, user-level testing of Grid software and services [35]. Originally designed for TeraGrid, Inca has been monitoring TeraGrid since 2003 and is also used in other large-scale global Grid projects including ARCS [36], DEISA [37], and NGS [38]. Today, Inca improves the reliability of Grid software and services by detecting user-level failures and providing detailed information about its tests and their execution to aid in debugging problems. Grid managers can use Inca to identify failure trends and verify that resource providers fulfill operations requirements. System administrators and users may use Inca to debug and resolve user account and environment issues.

Inca tests capabilities published in IIS with a SupportLevel of *testing* or *production*. Predefined tests are then deployed to monitor the individual software and services provided by a CTSS capability. When errors of critical services are detected, email notification is sent to the resource provider. Today, Inca executes 2,483 tests on TeraGrid resources. Typically, lightweight tests of critical services run more frequently while heavier-weight tests run every once or twice a day.

Inca provides many web status page views from detailed test information to summary views and historical reports. All Inca data is published as XML and is either translated to HTML using XSLT or graphed using JFreeChart in order to generate particular web status page views. Because Inca and IIS both use XML, it was straightforward to provide analogous REST interfaces to view Inca status information for resource capabilities. For example, the figure on the left shows the status of a workflow capability (in HTML format) on Purdue's Steele machine using a REST URL. TGUP and the LEAD science gateway display resource status information accessed via Inca's REST interfaces.



Science Workflow Support 4.0.0

- condor
- condor-g
- ctss-workflow-registration
- gridshell

condor	purdue-steele
version: 6.7.18	7.2.1
condor-g	purdue-steele
version: 6.7.18	7.2.1
condorg-condorq	pass
ctss-workflow-registration	purdue-steele
ctss-workflow-4.0.0-registration	pass
gridshell	purdue-steele
version: >= 0.9.9-4	0.9.9-6

D. Expanding IIS adoption

Although initially driven by the need to advertise CTSS-4 capabilities offered by resource providers, we have seen considerable uptake among TeraGrid working groups and service providers interested in leveraging IIS to address broader publishing, discovery, and streamlined operations use cases. The following are some examples.

There is growing interest in further leveraging IIS to streamline TeraGrid capability verification and validation using the Inca system. By formally defining a test repository and mapping tests to capability kits and components, we believe an automated system for testing a federation is achievable.

TeraGrid's data area has identified extended GridFTP service information that enables data movement clients to make more optimal data movement decisions. They also used IIS to configure the Speedpage data movement testing framework [39].

The recently formed TeraGrid Quality Assurance working group is looking at leveraging IIS to register and discover components of a QA test bed and to register information about the QA tests that a particular capability must satisfy.

An Open Science Grid-to-TeraGrid gateway is now using IIS to discover where it can submit jobs. Other TeraGrid science gateways are considering using IIS to automate capability discovery and resource selection.

Figure 5 shows monthly aggregated information service usage information. We are missing WS/REST usage data from November and December '08. In addition, these numbers exclude search engine scans and internal accesses (where queries to one service result in internal queries to other services). Since the user portal, user documentation, Inca system, and other persistent systems often use cached IIS content, these graphs do not reflect total IIS information accesses.

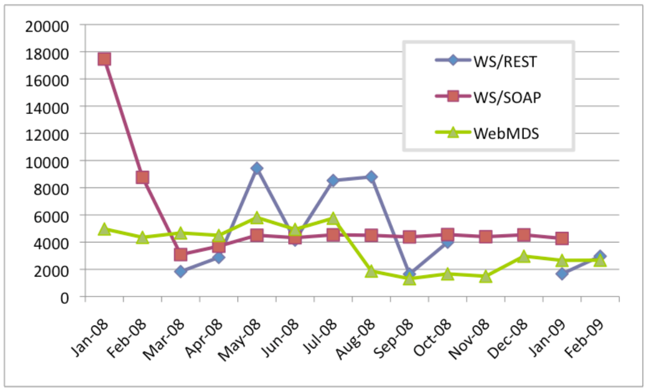


Figure 5. Aggregated information services queries. Some WS/REST data are missing during the Nov-Dec 08 period

VI. Conclusion And Future Work

TeraGrid's IIS system and information architecture have proven to be an effective means to communicate capability information from resource and service providers to TeraGrid users who need to discover and use those capabilities. It has also provided significant benefits for automating key aspects of TeraGrid's operations and user support activities.

IIS's two key design elements are its flexible information architecture and its federated system architecture. The information architecture permits TeraGrid to describe diverse capabilities and adapt quickly to new discovery needs. The system architecture allows us to gather information from diverse service providers, not limited in any way to those that have been traditionally part of TeraGrid. The IIS architecture seems ideally suited for federating resources and for simplifying and streamlining how federated resources are discovered and operated.

Our future system architecture and implementation work is focused on: (1) improving local service registry implementations and local information quality assurance tools, (2) improving and scaling our custom aggregation and caching system and exploring new aggregation, caching, and publishing frameworks to satisfy existing and new requirements, (3) evaluating historical information tracking requirements, (4) improving discovery interfaces to further enable Web 2.0 and emerging collaborative technologies, and (5) improving the local information service software distribution, perhaps by providing virtual machine images.

Our future information architecture work is focused on: (1) formally describing TeraGrid-operated capabilities such as our web sites, portals, accounting, allocations, and user management systems, (2) expanding our generic capability schemas to support more Grid and Cloud computing use cases, and (3) expanding the generic capability schemas to publish use cases, access policies, and testing/QA information.

We believe the challenges faced by TeraGrid that motivated IIS are not unique to TeraGrid. Our outreach activities are focused on: (1) reaching out to non-TeraGrid service providers (e.g., science gateway developers, external compute and storage providers) to register their services in TeraGrid's IIS to spread awareness among TeraGrid users, (2) encouraging other federations to deploy their own IIS-like information services, and (3) working with standards groups and other federated communities to encourage interoperability among our information services.

VII. Acknowledgements

This work is supported by NSF award OCI-0503697 to TeraGrid's Grid Integration Group (GIG), and in part by the U.S. Dept. of Energy under contract DE-AC02-06CH11357. The TeraGrid User Portal is funded by the NSF and includes input and participation from TeraGrid Partner Institutions. We thank TeraGrid users for their suggestions, feedback, and support in making TeraGrid a highly used, productive service for the national community. We also acknowledge the NSF-funded Open Grid Computing Environments (OGCE) project for the technology contributions to the User Portal project.

We thank Mike D'Arcy, Jennifer Schopf, Neill Miller, and Robert Miller for their contributions to the design and implementation of MDS4 services on TeraGrid.

References

[1] Catlett, C. and others, TeraGrid: Analysis of Organization, System Architecture, and Middleware Enabling New Types of Applications, in High Performance Computing and Grids in Action. 2007.

[2] NSF Program Synopsis, URL: http://www.nsf.gov/funding/pgm_summ.jsp?pims_id=5456

[3] NSF Fact Sheet "From Supercomputing to the TeraGrid," April 19, 2006.

[4] Czajkowski, K., et al., Grid Information Services for Distributed Resource Sharing, in 10th IEEE International Symposium on High Performance Distributed Computing. 2001, IEEE Computer Society Press. p. 181-184.

[5] Fitzgerald, S., et al., A Directory Service for Configuring High-Performance Distributed Computations, in 6th IEEE Symposium on High-Performance Distributed Computing. 1997. p. 365-375.

[6] Pordes, R., et al., The Open Science Grid, in Scientific Discovery through Advanced Computing (SciDAC) Conference. 2007

[7] J.M. Schopf, L. Pearlman, N. Miller, C. Kesselman, I. Foster, M. D'Arcy, and A. Chervenak, "Monitoring the grid with the Globus Toolkit MDS4," Journal of Physics: Conference Series, vol. 46, 2006, pp. 521-525.

[8] MDS4, URL: http://www.globus.org/toolkit/docs/4.0/info/

[9] http://www.globus.org/toolkit/docs/4.0/info/webmds/WSMDSWebMDSFacts.html

[10] J. M. Schopf, et al., Monitoring and Discovery in a Web Services Framework: Functionality and Performance of Globus Toolkit MDS4, in Argonne National Laboratory, Technical Report #ANL/MCS-P1315-0106, 2006.

[11] Foster, I., et al., A Security Architecture for Computational Grids, in 5th ACM Conference on Computer and Communications Security. 1998. p. 83-91.

[12] http://www.xfront.com/REST-Web-Services.html

[13] http://en.wikipedia.org/wiki/Representational_State_Transfer

[14] The tginfo command line tool, URL: http://info.teragrid.org/tginfo/

[15] http://en.wikipedia.org/wiki/Information_architecture

[16] "CTSS 4 TeraGrid Core Integration Capabilities - TeraGrid Wiki."

[17] Core Services 2.0 Working Group. "Core Services 2.0: An Integrated TeraGrid Vision." *Core Services 2.0.* TeraGrid. 2007. http://www.teragridforum.org/mediawiki/index.php?title=Core_Services_2.0 (accessed March 10, 2009).

[18] Hart, David, Ed Hanna, Rob Light, and JP Navarro. "Simplifying TeraGrid Resource Integration with the Resource Description Repository." Submitted to *TG'09,* 2009.

[19] Oinn, T., et al., Taverna: A tool for the composition and enactment of bioinformatics workflows Bioinformatics Journal, 2004. 20(17): p. 3045-3054.

[20] T. Scavo and V. Welch. A Grid Authorization Model for Science Gateways. International Workshop on Grid Computing Environments, 2007. See http://library.rit.edu/oajournals/index.php/gce/article/view/99

[21] OASIS Security Services (SAML) Technical Committee. See http://www.oasis-open.org/committees/tc_home.php?wg_abbrev=security

[22] S. Cantor et al. Metadata for the OASIS Security Assertion Markup Language (SAML) V2.0. OASIS Standard, March 2005. Document ID saml-metadata-2.0-os. See http://docs.oasis-open.org/security/saml/v2.0/saml-metadata-2.0-os.pdf

[23] Moab Workload Manager User's Manual. http://www.clusterresources.com/products/mwm/docs/moabusers.shtml

[24] Master Control Program (MCP). http://www.sdsc.edu/us/tools/mcp.html

[25] .Frey, J., et al., Condor-G: A Computation Management Agent for Multi-Institutional Grids. Cluster Computing, 2002. 5(3): p. 237-246.

[26] Eduardo Huedo and Rub{\a'e}n S. Montero and Ignacio M. Llorente. The GridWay Framework for Adaptive Scheduling and Execution on Grids. Scalable Computing - Practice and Experience. 6(3):1-8, 2005.

[27] Zhao, Y., et al., Swift: Fast, Reliable, Loosely Coupled Parallel Computation, in 1st IEEE International Workshop on Scientific Workflows. 2007. p. 199-206.

[28] Rajesh Raman, Miron Livny, and Marvin Solomon, Matchmaking: Distributed Resource Management for High Throughput Computing. Proceedings of the Seventh IEEE International Symposium on High Performance Distributed Computing, July 28-31, 1998, Chicago, IL.

[29] Ian Foster. Globus Toolkit Version 4: Software for Service-Oriented Systems. Proceedings of the IFIP International Conference on Network and Parallel Computing, Springer-Verlag LNCS 3779, pp 2-13, 2006.

[30] Open Grid Forum (OGF). URL: http://www.ogf.org/

[31] Sergio Andreozzi, et al. GLUE Specification v. 2.0. The Open Grid Forum, GFD-R-P.147, 2009.

[32] http://hpcsoftware.ncsa.uiuc.edu/Software/user/index.php

[33] M. Dahan, E. Roberts, J. Boisseau, "TeraGrid User Portal v1.0: Architecture, Design, and Technologies", Grid Computing Environments Workshop, November 2006.

[34] GridPort Information Repository, URL: http://www.collab-ogce.org/ogce/index.php/GPIR

[35] Smallen, S., Ericson, K., Hayes, J., and Olschanowsky, C. 2007. User-level grid monitoring with Inca 2. In Proceedings of the 2007 Workshop on Grid Monitoring (Monterey, California, USA, June 25 - 25, 2007). GMW '07. ACM, New York, NY, 29-38.

[36] The Australian Research Collaboration Service Web Page, http://www.arcs.org.au/.

[37] The Distributed European Infrastructure for Supercomputing Applications Web Page, http://www.deisa.eu.

[38] The National Grid Service Web Page, http://www.grid-support.ac.uk/.

[39] Speedpage, URL: http://speedpage.psc.edu/