Platform™

TECHNICAL WHITEPAPER

Open source metascheduling for Virtual Organizations with the Community Scheduler Framework (CSF)

The Structure of a Virtual Organization

Grid computing is becoming a common approach for enabling applications to run in large-scale distributed computing environments. As stated by Ian Foster et al in *The Anatomy of the Grid* [Anatomy] "The real and specific problem that underlies the Grid concept is *coordinated resource sharing and problem solving in dynamic, multi-institutional virtual organizations*." Furthermore, "This sharing is, necessarily, highly controlled, with resource providers and consumers defining clearly and carefully just what is shared, who is allowed to share, and the conditions under which sharing occurs. A set of individuals and/or institutions defined by such sharing rules form what we call a *virtual organization* (VO)"

A VO is composed of different physical resources (e.g. compute, storage, network, and software resources) that reside in separate administrative domains, within different organizations. Organizations will often have different policies governing the use of their resources, with different resource management interfaces.

A VO provides a single interface for end users to gain access to underlying resources, regardless of the interface being used by the actual physical resources.

The VO represents a higher level of structure on top of heterogeneous lower-level resources, therefore the purpose of grid middleware is to present resources as uniformly accessible. End-users are not concerned with the individual usage policies of the physical resources, and are able to interact with a VO from a single interface.

Grid Services and OGSI-Agreement

Interoperability is the key to a grid infrastructure. For a higher-level community scheduler to function with lower-level resource managers, there must be agreement on how these entities will interact with each other, since the functions and interfaces of the many lower-level resource managers are quite different from each other. Standards are required to meet interoperability requirements and resource manager services must use standard semantics, to ensure that the behaviour of a resource manager is predictable to a community scheduler. The Global Grid Forum (GGF) is an organization that works on defining the standards and best practices, which guide the technology development for building a grid infrastructure.

More recently, there has been a convergence of the grid standards defined by GGF, with the protocols and mechanisms defined by the Web services community. The Open Grid Services Architecture (OGSA) [OGSA] has been proposed as an enabling infrastructure for applications and services on a grid. The OGSA defines *Grid services* as common activities of a distributed computing system within a Web services model for defining interfaces and functions. The OGSA model represents all entities on a grid (resources, applications, jobs, agreements, data, etc) as Grid services.

While OGSA activities remain focused on the architecture of grid systems, the Open Grid Services Infrastructure (OGSI) [OGSI] is a specification that defines mechanisms for managing services on the grid. The specification defines the base level interfaces and behaviors for managing Grid services and the way that they interact with each other. In addition, a number of "core" interfaces are defined which correspond to some of the most common low-level constructs found in distributed computing systems. For example, OGSI defines a type of core Grid service that is used to manage the lifetime of the services and provide access to their service state. Other interfaces include the mechanism for name resolution, event notification, and service aggregation. Multiple low-level interfaces are composed together to build higher-level Grid services.

OGSI defines a necessary set of Grid services needed to deploy a production grid infrastructure. Higher-level services built on top of the OGSI services are required to map between a VO and underlying resource managers. OGSI defines the vocabulary, and higher-level specifications will define the grammar needed for a meaningful discussion. Working groups within the GGF are working to define higher-level Grid service interfaces required to build a grid infrastructure. The OGSA documentation defines a roadmap that helps guide which services are considered essential for computing on a grid. These include services for security, messaging, data access, resource management and scheduling.

Grid environments based on OGSA are composed of many different, interacting Grid services. Each of these services has different policies to manage underlying resources. In order to deal with the complexities of large collections of these services, there need to be mechanisms for Grid service management and the allocation of resources for applications. A Grid service interface called *OGSI-Agreement is being proposed at the GGF as a protocol and interface for managing Grid services*. The OGSI-Agreement [AGSM] describes the Agreement-based Grid Service Management model as "…the ability to create Grid services and adjust their policies and behaviors based on organizational goals and application requirements." This model is composed of a set of OGSI-compliant portTypes that allow clients to negotiate with management services to manage Grid services or other legacy applications. In other words, if a user wants to submit a compute job to run on a cluster, the Grid service client contacts a job management service, and negotiates an agreement which ensures that the user's job has access to cpus, memory and storage space.

The OGSI-Agreement is based on Agreement services, which represent an ongoing relationship between an agreement provider and an agreement initiator and defines the behavior of a delivered service to a consumer.

Agreements are negotiated with new Agreement services through a Grid service that implements the AgreementFactory interface. A client invokes the AgreementFactory service using creation parameters that contain the requested terms. Some CreationParameters are required while others are subject to counter offers. If the agreement terms are not acceptable to the service provider, the creation operation returns a fault. Otherwise, a new Agreement service instance is created, which encapsulates the negotiated terms in its service state, or an AgreementOffer service instance is created, which represents a number of alternative offers for clients to choose. The instantiation of an Agreement service indicates that the agreement has been accepted.

Consider the example of job submission to a resource manager. A client contacts a job management service, which implements the AgreementFactory interface with creation parameters that state, "this job requires a software license for application X, and requires 8 cpus, and 4G of RAM". If the job management service couldn't provide the necessary software license the agreement terms are rejected. If all of the terms are met, an Agreement service instance representing the job resources is created. If the terms of the original creation parameters can't be met, however the job management service could supply either 4 cpus and 4G of RAM, or 8 cpus and 2G of RAM, an AgreementOffer with the two alternatives would be offered. The client then chooses the best alternative because cpus and memory were terms subject to the counter offers.

Defining terms to represent different types of resources available within a VO, community schedulers are written to negotiate with resource managers for resources on behalf of the user community. An example set of terms for making advanced reservations are listed in Appendix 1.

Community Schedulers

Grid middleware provides *coordinated* access to the underlying resources of a VO, regardless of their physical location or access mechanisms. When an application utilizing a grid makes use of more than one physical resource during its execution, a grid middleware system maps the resource requirements to the multiple physical resources that are available to run that application. *Metaschedulers* or *community schedulers* are key to making the VO's resources easily accessible to end-users, by automatically matching the requirements of a grid application with the available resources while staying within the conditions that the VO has specified with the underlying resource managers.

A community scheduler serves each VO. End-users submit their grid applications to the community scheduler, which in turn matches the resource requirements of the jobs with the underlying physical resources through interaction with local resource managers. The resources provided by the local resource managers can be part of multiple VO's, with local policies reflecting the nature of the agreements negotiated by the different user communities for access to the physical resources. The local resource manager becomes a service provider, and a community scheduler consumes services based on Service Level Agreements (SLA's).

Figure 1 illustrates two VO's making use of three different resource managers that are shared between one another. VO "A" has access to physical organization 1 and physical organization 2. VO "B" has access to physical organization 3 and only part of physical organization 2.

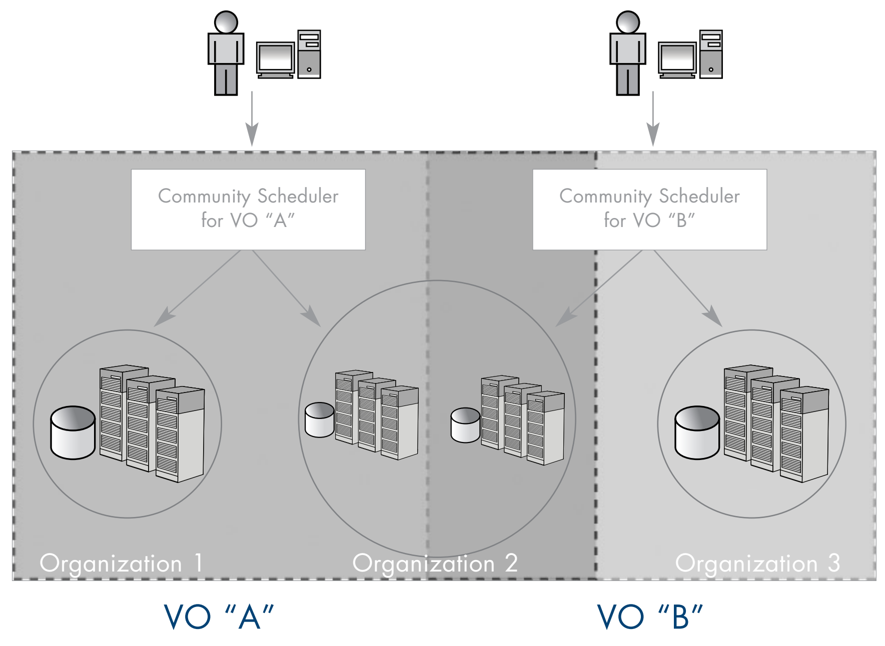

Community Scheduler for VO "A"
Community Scheduler for VO "B"
Organization 1
Organization 2
Organization 3
VO "A"
VO "B"

Figure 1: Mapping of VO's to physical organizations

This is an example of the role of a community scheduler. An end-user wants to reserve time to use resources from two different organizations: a compute resource for data analysis and a visualization "cave" for rendering the results of the computing in real time. If resources are geographically separated, and under the control of different resource managers, end-users need to make various reservations including cpus for computation, the visualization cave, bandwidth between the two sites, and storage at both sites for temporary staging results. Currently, this is done manually. Some resource managers provide an interface for reserving resources, but more often this is done on the phone with administrators located at various sites.

With grid middleware and community schedulers an end-user could submit a request for resources to the community scheduler, which describes the resources needed in a standard description language. Using the knowledge of resource managers a community scheduler automatically interacts with the various organizations available to meet the requirements of the job. This process is much easier for an end-user since they only have to deal with one scheduler, and less prone to accidental conflict, because local resource managers control the allocation of resources. If a community scheduler couldn't meet the needs of the VO with the current resources available, an agreement with an "on-demand" computing center could be reached in order to meet the time requirements of the end-user.

Introducing the Community Scheduler Framework (CSF)

The Community Scheduler Framework (CSF) is an open-source implementation of a number of Grid services, which together perform the functions of a grid metascheduler or community scheduler. CSF provides basic capabilities for scheduling and can be used as a development toolkit for implementing community schedulers.

Platform Computing, in consultation with the Globus Project, has contributed the open source metascheduler called CSF to the Globus Toolkit Version 3.0 (GT3) OGSI container architecture. The CSF classes can be extended to provide more domain specific community schedulers and support many different kinds of grid deployment models. Examples of grid level scheduling algorithms include scheduling across multiple clusters within a VO, co-scheduling across multiple resource managers, scheduling based on SLA's, and economic scheduling models.

By making use of the open source CSF, grid scheduling implementations ensure that they interact with resource managers using standard interfaces such as the OGSI-Agreement, without needing to know all of the underlying details of the specification or having to implement the protocol themselves.

Figure 2 illustrates the components of CSF.

The Metascheduler

The services implemented as part of CSF provide a metascheduler infrastructure. The purpose of the metascheduler is to allow end-users to interact with underlying resource managers in a system independent fashion, by defining the protocols that interact with resource managers. In addition, the metascheduler is a high-level abstraction for some of the concepts that are key to resource management including a "job", a "resource reservation", and a "scheduler".

The higher-level Grid service interfaces are described below:

- Job service – creates, monitors and controls compute jobs
- Reservation service – guarantees resources are available for running a job
- Global Information Service – allows for the propagation of information between resource managers and the metascheduler
- Queuing service – provides a service where administrators can customize and define scheduling policies at the VO level, and/or at the different resource manager levels
- Resource Manager Adapter Service (RM Adapter) – provides a Grid service interface which bridges the Grid service protocol and resource managers (e.g. Platform LSF or Altair PBS)

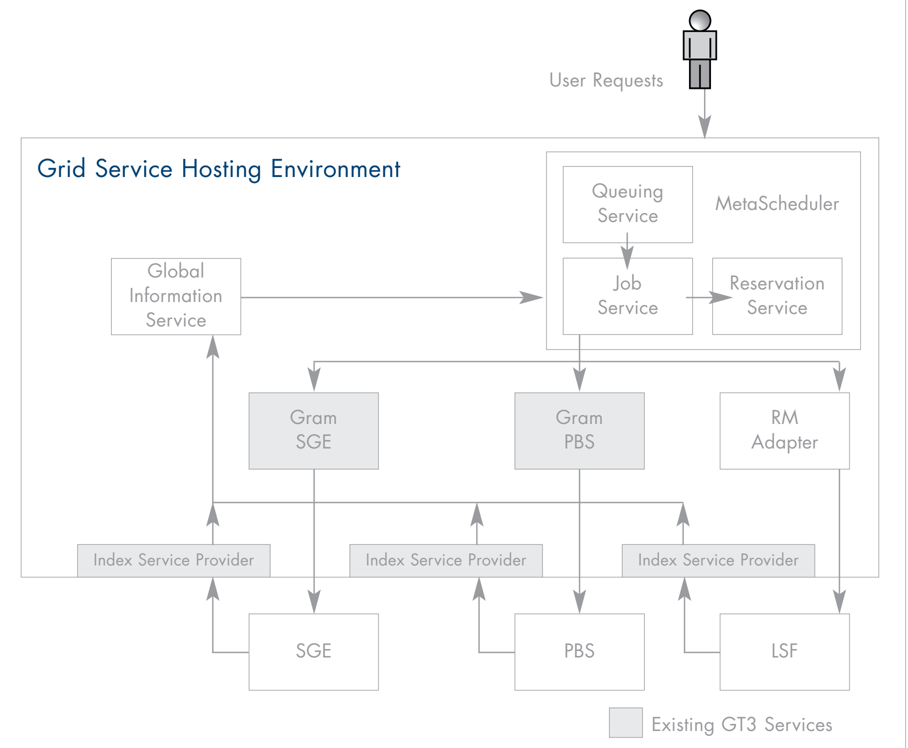
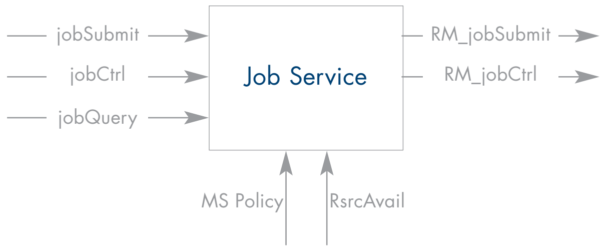


Figure 2: CSF architecture

JOB SERVICE



A Job Service provides an interface for placing jobs on a resource manager and interacting with the job once it has been dispatched to the resource manager. The Job Service provides basic matchmaking capabilities between the requirements of the job and the underlying resource manager for running the job. More advanced Job Services take into account more advanced job characteristics such as interactive execution, parallel jobs across resource managers, and jobs with requirements based on SLAs.

The interfaces provided by the Job Service include:

- jobSubmit – user job submission requests. A job is submitted with resource requirements in RSL (Resource Specification Language) syntax, with a reservation id in order to bind the job to a specific resource reservation. Otherwise a job is submitted with the name of a specific resource manager on which to run the job.

- jobCtrl – controls a job after it has been instantiated. This would include the ability to suspend/resume, checkpoint, and kill a job.

- jobQuery – query the status of a job. This includes information such as the status of a job, job runtime, and resource usage.

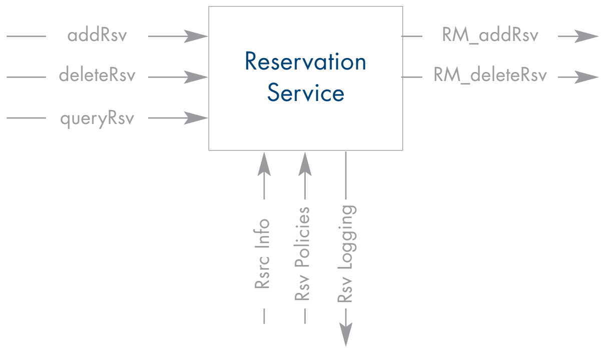
The Job Service uses information such as policies, which are defined at the metascheduler level, and resource information about available resource managers, queues, host, and job statuses as provided by the Global Information Service. The Job Service uses the Resource Manager Adapter (RM Adapter) to submit jobs to the underlying resource manager to control running jobs.

The Job Service also makes use of the Global Information Service to store it's own state information. The information stored includes job submission information, job status, and a list of jobs that a particular Job Service instance can manage, which provides the ability for the Job Service to recover from any faults.

In the current version of the CSF, the Job Service accepts client requests in the form of the Globus RSL (Resource Specification Language). In the future, job resource requirements and descriptions will be defined in a job definition language currently being defined by the GGF.

A given Job Service instance can actually manage multiple user jobs. If this model is in use, there is a Job Service Dispatcher component that is responsible for matching user job submission requests to running Job Service instances. The Job Service Dispatcher creates a Job Service instance on behalf of the user if there is no applicable Job Service available for the user.

RESERVATION SERVICE



The Reservation service allows end-users or a Job Service to reserve resources under the control of a resource manager to guarantee their availability to run a job. This service allows reservations for any type of resource (e.g. hosts, software licenses, or network bandwidth). Reservations can be specific (e.g. provide access to host "A" from noon to 5pm), or more general (e.g. provide access to 16 Linux cpus on Sunday). Once a reservation is made, a Job Service sends a job to a resource manager that is associated to a provided reservation. Some of the policy decisions made by the Reservation Service for Platform LSF include the identification of who can make reservations (e.g. Platform LSF administrators), how many hosts a particular user or user group can reserve at a time, when reservations can be made (e.g. blackout periods), and what types of hosts can be reserved.

The Reservation service provides the following interfaces:

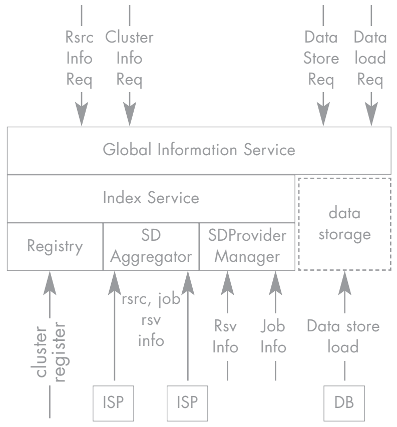
- addRsv – request a new reservation with a particular resource requirement, starting at a specific time for a given duration

- deleteRsv – remove a reservation

- queryRsv – retreive the details of a particular reservation

The Reservation Service makes use of information about the existing resource managers and policies that are defined at the metascheduler level and will make use of a logging service to log reservations. The Reservation Service uses the RM Adapter interface to make and delete existing reservations.

The Reservation service implements the AgreementFactoryType as defined in the OGSI-Agreement specification so that a client can make reservations based on agreement terms supplied. Reservations can also be specified using RSL.



GLOBAL INFORMATION SERVICE

The Global Information Service provides a repository for information required to operate the metascheduler. It is built upon GT3's Index Service, with some extensions. The GT3 Index Service aggregates various lower-level Grid services in order to replicate service data between Grid services and to create a dynamic data-generating and indexing node. The components used in the Global Information Service include:

- Registry components - provides static information about installed services such as service name, location and service ACLs. Each Platform LSF-based cluster (through the RM Adapter) registers itself in the ClusterRegistry, enabling Job and Reservation services to find available resource managers.

- ServiceDataProviderManager - manages the acquisition of dynamic data into the Global Information Service via external programs. GT3 uses the Resource Information Provider Service (RIPS) to collect job and resource information from each cluster in the system. This includes aggregate information about the individual resource managers (e.g. how many jobs in the system, average job turnaround time, what host types are available, etc). In addition, it aggregates more detailed information about individual jobs, queues, reservations and hosts.

- ServiceDataAggregator - aggregates the service data of other services and provides notification mechanisms. This is used in the Global Information Service to aggregate information about Job Service instances (e.g. submission, job status, execution, Job Service instance, etc), and about Reservation Service instances (e.g. requests and individual reservation details).

The Global Information Service provides a consistent interface for persisting state information for other services implemented as part of CSF.

QUEUING SERVICE

The Queuing Service provides scheduling capabilities to the metascheduler. With a set of policies defined at the VO level, a Queuing service maps jobs to resource managers based on defined policies. For example, a Queuing Service implementing a Fairshare policy ensures that all users within the VO receive reasonable turnaround time on their jobs, as opposed to being starved by other users' jobs ahead of them in the queue.

The CSF implementation of the Queuing Service implements a plug-in scheduling framework. Schedulers are java classes that implement the schedPlugin interface, which is composed of four methods. The two main methods used during the scheduling cycle are schedOrder and schedMatch. A list of jobs queued in the system are passed to schedOrder, which order the list depending on which jobs have precedence based on the implemented scheduling algorithm. The schedMatch method matches jobs with the resource managers (either RM Adapter resource managers or GRAM job factories) that are available to the scheduler. The job-to-RM mappings is called a "job decision".

If there are multiple scheduling policies in the Queuing Service, the different schedulers will be called in order based on the job list and on the job decisions, allowing the effects of one scheduling plug-in to be combined with the effects of another. For example, two scheduling plug-ins are provided as part of the CSF Queuing service including a FCFS queue and a job throttle.

The FCFS scheduler orders the list of jobs based on their submission time to the Queuing service. During the matching phase, jobs are either mapped to an RM, based on a provided cluster name or reservation id, otherwise the FCFS scheduler finds an RM, based on a round-robin choice from the available RMs.

The job throttling scheduler doesn't reorder the list of jobs from the FCFS scheduler. It's match phase ensures that too many jobs are not sent to the same RM at the same time, so that:

a) an RM is not overwhelmed, and

b) a job is not queued at an RM which may not run the job for awhile, when another RM may be available sooner to run the job.

The architecture of the Queuing service easily allows 3rd party developers to add new grid schedulers into the CSF, to support the needs of their VO.

RM ADAPTER

The RM Adapter is an interface for communicating with underlying resource managers. The primary purpose of the RM Adapter is to provide a bridge between the OGSI-compliant grid clients (such as the metascheduler) and legacy resource managers that do not have a native Grid service interface. Translations at the RM Adapter level include mapping from the grid to the resource manager's security model, mapping the RSL to the resource manager's resource requirement syntax, and translating between SOAP/XML request/response to the resource manager's request/response protocol.

The current implementation of CSF supports the GRAM protocol (from Globus Toolkit Version 2.x and earlier) to access the services of the resource managers, which do not support the RM Adapter interface. GRAM does not support any reservation interfaces, so this limits the metascheduler to only use the Job service when communicating using this interface. GRAM only serves one resource manager, whereas the RM Adapter can be configured to service multiple resource managers at the same time.

CSF and Platform Products

Platform's products (including Platform LSF and Platform Multiculster), leverage the CSF to ensure OGSA compliance, and a framework for implementing metascheduling.

Platform LSF

CSF, through the RM Adapter, provides an alternate interface method to Platform LSF in addition to the regular Platform LSF commands and protocols. CSF exposes the RM services using OGSA compliant services, and therefore Platform LSF is OGSA-compliant via the metascheduler interfaces, and directly through the RM Adapter.

Since the OGSI specification is based on Web services, this allows organizations to access the services of Platform LSF via standard Web services interfaces that doesn't require an OGSA-compliant client.

The Grid services interface to Platform LSF enables organizations to use more generic interfaces for Platform LSF, without needing to write to proprietary API's and protocols. This is useful for portal writers, and ISVs who are developing applications to run on a grid, because it is simpler to integrate with standard interfaces than with many different proprietary ones.

The integration between Platform LSF and CSF is a new interface called the Grid Gateway. The Grid Gateway acts as a bridge between Platform LSF's proprietary protocols and CSF's Grid service interface. A new scheduling plug-in for Platform LSF scheduler (the metascheduler plug-in) makes the decision regarding which particular Platform LSF jobs should be forwarded to the metascheduler. This decision is based on information from the Global Information Service, provided by the Grid Gateway. When a job is sent to the metascheduler, the mbatchd will dispatch the job through the Grid Gateway, and receive the job status information from the Grid Gateway. The Grid Gateway uses the Job and Reservation services from the CSF.

There are four use cases supported by this new architecture, which illustrate how users use Platform LSF interfaces including, bsub, brsvadd, and bkill, or use CSF directly.

1. Job Forwarding by a user – a user identifies a resource manager to send a job to, by querying the GIS using commands such as grminfo and ghostinfo. The user submits a job with the directive in the bsub command that the metascheduler plug-in will forward this job to the requested RM. This does not guarantee the resources will be available to run the job when it reaches the RM.

2. Job Forwarding by Platform LSF – a user submits a job without specifying a resource manager. Platform LSF through the metascheduler plug-in makes the decision whether a job should be forwarded to another RM, based on the job's resource requirements. This does not guarantee that the resources will be available when the job is forwarded.

3. Forwarding with user created advance reservation – a user finds an RM to run their job, and makes a reservation with the RM using the Platform LSF reservation mechanism. If the reservation is successful, the user submits a job to Platform LSF with the reservation id. The metascheduler plug-in then forwards the job to the target RM with the reservation id.

4. Forwarding with LSF created advance reservation – a user submits the job without specifying a resource manager. The metascheduler plug-in chooses a remote RM based on the job resource requirements, and makes a reservation on behalf of the user. The job is then forwarded to the target RM using the created reservation.

Platform MultiCluster

Platform MultiCluster allows multiple clusters using Platform LSF to communicate by forwarding work between clusters transparently to end-users. Platform MultiCluster has superior performance in terms of scalability and fault tolerance.

CSF enhances Platform MultiCluster by providing a framework for scheduling amongst multiple resource managers, :

1) Enables Platform MultiCluster to communicate with non-Platform LSF-based resource managers through the use of the CSF for job forwarding

2) Ensures Platform MultiCluster installations are OGSA-compliant when accessed through CSF

3) Platform MultiCluster enables Platform LSF users to take advantage of hierarchical scheduling by forwarding jobs to CSF

The next generation of Platform MultiCluster will make use of CSF and the scheduling policies and algorithms that plug into it along with other custom and 3rd party scheduling algorithms developed by ISVs and end-users. Moreover, jobs submitted through CSF take advantage of the additional functionality in Platform MultiCluster, for example - resource leasing, which allows hosts in one cluster to be loaned to another cluster.

A migration strategy for current Platform MultiCluster customers to CSF-enabled Platform MultiCluster installations, has been developed based on the architecture shown in Figure 3.

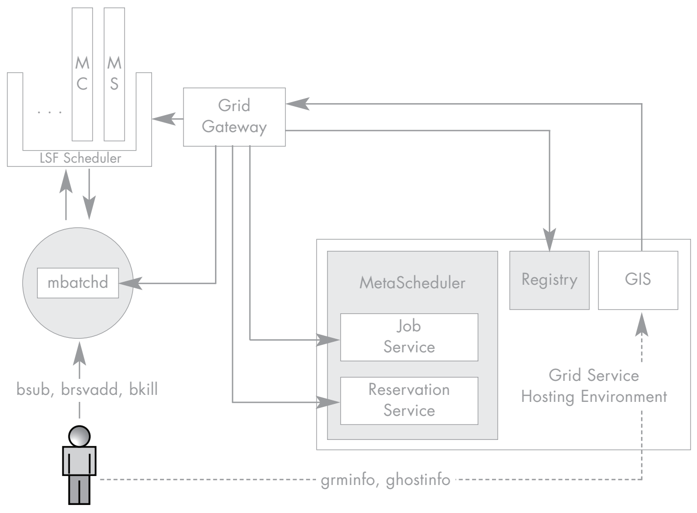


Figure 3: CSF-enable Platform MultiCluster

Summary

The Community Scheduler Framework (CSF) is an open source add-on to the Globus Toolkit Version 3.0 for the development of community schedulers. Community schedulers are commonly referred to as metaschedulers and accept user requests to run jobs, and map them to the resources, which have been made available within the user's Virtual Organization (VO).

CSF implements a number of Grid services, which together provide the functionality of a metascheduler, including services such as a "job", "advance reservation" and "queue". These services make use of the OGSI-Agreement specification as a negotiation protocol for negotiating the use of resources through a resource manager. CSF allows users and ISV's to plug into grid scheduling algorithms developed, enabling VOs to write schedulers to implement policies.

CSF will be leveraged in the next generation of Platform Computing's products, providing OGSA compliance, and a framework for implementing metascheduling. Platform MultiCluster will provide metascheduing functionality to communicate with both Platform LSF and non-Platform LSF-based resource managers.

Appendix 1

This is an example set of agreement terms for making an advanced reservation with Platform LSF.

```
<?xml version="1.0" encoding="UTF-8"?>
<rsl:rsl  xmlns:rsl="http://www.globus.org/namespaces/2003/04/rsl"

xmlns:metascheduler="http://com.platform.metascheduler/2003/05/rsl/metasched-
uler"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
        xmlns:ogsi="http://www.gridforum.org/namespaces/2003/03/OGSI"
          xsi:schemaLocation="
            http://www.globus.org/namespaces/2003/04/rsl
            /usr/local/gt3/schema/base/gram/rsl.xsd
            http://com.platform.metascheduler/2003/05/rsl/metascheduler
            /usr/local/gt3/schema/metascheduler/metascheduler.xsd
          http://www.gridforum.org/namespaces/2003/03/OGSI
          /usr/local/gt3/schema/ogsi/ogsi.xsd"
          >

<!--
  You need to modify to absolute path for schema locations, such as:
          o rsl.xsd
          o metascheduler.xsd
-->

  <metascheduler:request>
    <metascheduler:reservation>

      <!-- Resource Requirment in LSF RES_REQ format -->
      <metascheduler:rsrc>
      <rsl:string>
        <rsl:stringElement value="type==LINUX86"/>
      </rsl:string>
      </metascheduler:rsrc>

      <!-- Hosts delimited by space -->
      <metascheduler:hosts>
      <rsl:string>
        <rsl:stringElement value="plato"/>
      </rsl:string>
      </metascheduler:hosts>

      <!-- Name of user using this reservation -->
      <metascheduler:user>
      <rsl:string>
        <rsl:stringElement value="csmith"/>
      </rsl:string>
      </metascheduler:user>

      <!-- Number of CPU request for this reservation -->
      <metascheduler:number>
      <rsl:long value="1"/>
      </metascheduler:number>

      <!-- Start time -->
      <metascheduler:beginTime>
        <ogsi:ExtendedDateTimeType  value="2003-07-23T20:00:00.000Z"/>
      </metascheduler:beginTime>

      <!-- End time -->
      <metascheduler:endTime>
        <ogsi:ExtendedDateTimeType  value="2003-07-23T24:00:00.000Z"/>
      </metascheduler:endTime>

    </metascheduler:reservation>
  </metascheduler:request>
</rsl:rsl>
```

References

[Anatomy] *The Anatomy of the Grid: Enabling Scalable Virtual Organizations*, I. Foster, C. Kesselman, S. Tuecke, Authors. International Journal of High Performance Computing Applications, 15 (3). 200-222. 2001. Available at http://www.globus.org/research/papers/anatomy.pdf

[OGSA] *The Open Grid Services Architecture Platform*, I. Foster, D. Gannon, Editors. Global Grid Forum. draft-ggf-ogsa-platform-2

[OGSI] *Open Grid Services Infrastructure (OGSI) Version 1.0*, S. Tuecke, K. Czajkowski, I. Foster, J. Frey, S. Graham, C. Kesselman, T. Maquire, T. Sandholm, D. Snelling, P. Vanderbilt, Editors. Global Grid Forum. draft-ggf-ogsi-gridservice-29

[AGSM] *Agreement-based Grid Service Management (OGSI-Agreement), Version 0*, K. Czajkowski, A. Dan, J. Rofrano, S. Tuecke, Editors. Global Grid Forum. draft-ggf-czajkowski-agreement-00

Platform™

About Platform Computing

Platform Computing delivers intelligent, practical enterprise grid software and services that allow organizations to plan, build, run and manage grids by optimizing IT resources. Through our proven process and methodology, we link IT to core business objectives, and help our customers enhance service capabilities, reduce costs and improve business performance. With industry-leading partnerships and a strong commitment to standards, we are at the forefront of grid software development, propelling over 1,600 clients around the world toward powerful insights that create real, tangible business value. For more information www.platform.com.

World Headquarters	US	Europe	Asia
Platform Computing Inc.	Boston: 781.685.4966	London: +44 (0) 1256.370.500	Beijing: +86 1062 381125
3760 14th Avenue	Detroit: 248.359.7820	Dusseldorf: +49 (0) 2102 61039.0	Hong Kong: 852.2869.5687
Markham, Ontario	Maryland: 410.290.0105	Paris: +33 (0) 1 34 65 51 60	Tokyo: +813 5326-3105
L3R 3T7 Canada	Newport Beach: 949.798.5654		Osaka: +81 6 6225 1163
Tel: 905.948.8448	New York: 212.672.1770		Seoul: +82 2 5811410
Fax: 905.948.9975	San Jose: 408.392.4900		
Toll-free tel: 877.528.3676			
info@platform.com	For more information, visit www.platform.com/contactus/index.asp		

AUGUST 2003