Authors:
Steven Newhouse, Microsoft
Andrew Grimshaw, University of Virginia

7 October, 2008

Independent Software Vendors (ISV) Remote Computing Usage Primer

Status of This Document

This document provides information to the Independent Software Vendor (ISV) community as to how specifications from the Open Grid Forum and other standard setting organizations (SSO) can be used to support Remote Computing scenarios. It does not define any standards or technical recommendations. Distribution is unlimited.

Copyright Notice

Trademark

Abstract

This document shows how specifications, that have been developed within and external to the Open Grid Forum, can be used to enable desktop access to distributed computing resources. Increasingly, Independent Software Vendors (ISVs) are developing applications that have two components – a graphical client that resides on the client and a compute intensive server component that can run on a 'back-end' compute cluster. Integrating these two components within networks which frequently have firewalls and NATs, across different operating systems and software environments, provides many challenges. A standards based interface to this environment would provide ISVs with a solid foundation upon which to build their own applications.

We identify and describe a set of standards and specification that have been developed to facilitate access to distributed computing resources. We illustrate how these specifications could enable access to distributed computing resources through five scenarios. These range from simple job submission to a compute cluster where the client and the cluster have a common file system, to a client that stages files to and from the compute cluster while having bi-directional interaction between the application running on the compute cluster and the remote client.

By illustrating how these specifications can be applied to these scenarios we identify requirements for both the middleware providers and the ISVs – recognizing that we are providing advice rather than rigid prescriptive solutions. Finally, we identify some open issues for discussion and feedback by the community relating to resource selection and security.

Contents

1 Introduction

Many users would like to integrate their desktop ISV provided applications residing on their client workstations with 'back-end' compute clusters in a uniform or standard manner. This may include both connected and disconnected client workstations, and may involve file movement between the client workstations, the compute back-end, and third party data servers. In addition there are issues arising from firewalls, NATS, and other networking impediments that exist in the real world. In other words, the client workstation, and/or compute backend, may not have globally addressable IP addresses.

This document shows how specifications, that have been developed within and external to the Open Grid Forum, can be used to enable desktop access to distributed computing resources. As such it provides suggestions as to how these specifications can be used, rather than a rigid prescriptive solution. We identify a series of typical computing environments with distributed computing and file storage resources, and some typical usage scenarios within such environments. For these scenarios we identify the relevant specifications and how they could be used. Specifically, the execution of an application that may include file staging to and from the computing resource to the client workstation as an alternative to a common file system, and which may in addition include two-way interaction with the running application from the client workstation through network firewalls. Finally, we identify a set of issues where we feel we need further discussion with the community in order to reach a clear recommendation for implementation and adoption.

The generic physical infrastructure is described below:

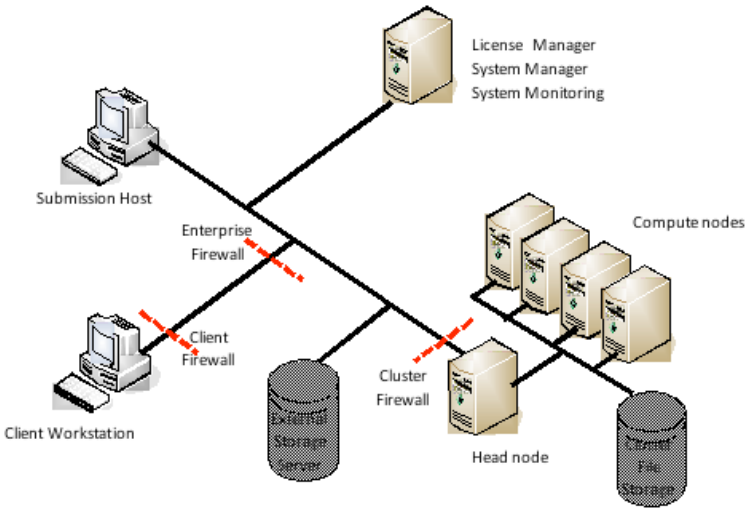


Figure 1: Generic Physical Infrastructure

1.1 Actors

Here we define the important characteristics of the actors/players in the diagrams.

- Client workstation: The machine used by the end-user to submit, monitor and review their results. A client machine can be considered to be mobile, frequently dis-connected from the network and not permanently sharing a file system with the enterprise network or the computing resource.
- Submission host: A workstation from which the user can submit and monitor jobs to the compute cluster. Generally a machine that is permanently connected to the enterprise network and able to access the corporate and cluster file systems.
- The back-end compute cluster, consisting of:

  - Head node: This is the primary contact point for submission hosts to submit work to the cluster. Possibly the only part of the compute cluster visible to other network clients.
  - Compute nodes: The compute cluster managed by the head node. These nodes may only be accessible from the Head node.
  - A “cluster” file system that is common between the head node and compute nodes and may be accessible from the submission host, but not the client workstation.
- External Storage Server: A storage resource that is accessible through authenticable network protocols (e.g. http, scp, ftp, ..) from outside the enterprise network (the client workstation) and within the enterprise network (from the submission host) and from the cluster’s head node. This is not necessarily a global file storage system.
- Network Router: Linking the workstation, submission hosts and head nodes.
- Firewall(s): One of many protecting the head node, client workstation and submission hosts.

1.2 Scenarios

We briefly identify the scenarios that we will be considering in this document. The complexity of the network environment is the main differential in these increasingly more complex scenarios. The basic scenario in each remains the execution of an application on a compute cluster from a desktop environment.

1. Execution of an application on a cluster from a submission host directly using a proprietary distributed resource manager.
2. Execution of an application on a cluster from a submission host using a standardized API.
3. Execution of an application on a cluster from a submission host using a web service protocol and a shared file system.
4. Execution of an application on a cluster from a client workstation using a web service protocol with file staging instead of a shared file system (the main difference from scenario 3).
5. Execution of an application on a cluster from a client workstation using a web service protocol with two way interaction between the client workstation and the running application.

For each scenario we identify the environment, the pre-requisites, a high-level view of the interaction between the actors, and the benefits gained from the scenario. These scenarios are summarized below:

	Scenario				
Feature	1	2	3	4	5
Application uses proprietary client API	Yes	No	No	No	No
Application requires DRM's client libraries to be installed	Yes	Yes	No	No	No
A web service protocol links the client and the cluster	No	No	Yes	Yes	Yes
Client and cluster require common file system	Yes	Yes	Yes	No	No
A file transfer protocol links the client and the cluster	No	No	No	Yes	Yes
Explicit file transfer from the client to enable disconnected operation	No	No	No	Yes	Yes
Interactive bi-directional communication between the client and the application running on the cluster	No	No	No	No	Yes

Figure 2: Supported Features within a Scenario

Scenarios 3, 4 & 5 share common features through the use of a web service to support job submission. The features in scenarios 4 & 5 can be combined – they are shown separately here for simplicity.

1.3 Additional constraints

The goal in this scenario is to provide a 'good' user experience when the client workstation is in 'difficult' networking environments. Specifically:

- Many computational applications have an element of interaction with the compute nodes. In such a scenario the compute nodes need to be able to 'connect back' to the client workstation to provide interim results.
- Any solution must leverage the existing security mechanisms available on the client workstation and available within the enterprise. Bespoke security solutions should not be embedded within the ISV application.

1.4 What we are not worrying about right now

In these scenarios we are making realistic simplifications to provide a still useful solution. In particular:

- We are not concerning ourselves with the process by which a client application running on the client workstation finds a compute cluster. We assume that such a resource is 'well known' within an organization and such a resource may encapsulate other resources… but if it does such behavior is not visible to the client application.
- We are not concerned about accounting for use of the application within the scenario. The cluster's management system will frequently have an accounting capability but this is not exposed in these scenarios.
- We are not concerned about License Managers. An application will frequently require the presence of a License Manager and its execution on the cluster may be delayed until

appropriate licenses are available. In such situations this can be reflected in the state exposed by the cluster management system.

2 Specification Pool

In enabling the interaction between the client and the application installed on the cluster we will use web service and related specifications developed within the Open Grid Forum and other standards bodies. These are described briefly below – further information and technical details of the specification can be found elsewhere.

2.1 WS-Addressing

The WS-Addressing specification defines an extensible XML data structure called the *endpoint reference* (EPR) that serves to encapsulate the information needed by a client to message a service. The EPR includes such data as a network protocol address, an extensible metadata section to convey arbitrary suggestions such as security policies, and an opaque section for session/resource identifiers, etc.

2.2 Security specifications and profiles

The Web Services Security (WS-Security) family of specifications defines a general-purpose mechanism for associating security credentials with message content which is then used to construct a set of OGF specific profiles for encoding popular token types (e.g., X.509, Kerberos, SAML, and Username-token credentials). The WS-Security Core specification also defines the application of XML Encryption and XML Digital Signature to provide end-to-end messaging integrity and confidentiality without the support of the underlying communication protocol. In order to achieve real-world interoperability, the WS-I BSP (Basic Security Profile) provides guidance on the use of WS-Security and its associated security token formats to resolve nuances and ambiguities between communicating implementations intending to leverage common security mechanisms.

Two profiles have been defined in the WS-Security space that may come into play: WS-Secure Addressing (GFD 131) and WS-Secure Communication (GFD 132). WS-Secure Addressing is a profile on WS-Addressing and WS-Security Policy that addresses the secure binding of metadata such as keys, type, identity, and security policy into WS-Addressing endpoint references (EPRs). For example, what authentication mechanisms does an endpoint support or require. WS-Secure Communication further profiles commonly used security mechanisms defined in the WS-I Basic Security Profile.

2.3 Job Submission Description Language (JSDL) (GFD 56)

JSDL is an XML-based schema for describing applications, the resources required for the application (e.g., memory, number and sped of CPU's, etc.), files to stage-in before the application executes, files to stage-out upon completion, the command line string to be executed, etc. JSDL defines terms in the job description space and encourages definition of new terms.

2.4 JSDL Single Process Multiple Data Application Extension (GFD 115)

The SPMD Application extension defines a number of additions to the JSDL Application element to support the definition of parallel applications. It re-uses a number of elements already defined by other Application extensions, for example, to specify the path to the executable, the working directory and so on. It adds support for specifying the number of processes, the number of threads per process and also how many processes to execute per host. The type of parallel environment that the application requires can also be identified, for example, the type of MPI required for execution.

2.5 JSDL Parameter Sweep Extension (Working group draft)

The Parameter Sweep extension defines how the values of one or more of the elements (parameters) in a JSDL document may be changed to produce a new JSDL document. This extension therefore defines a collection of jobs in a single document made up of a base JSDL document and a sweep definition.

The sweep definition allows for changing a single or multiple parameters at the same time and supports arrays of values for each parameter. Loops can be defined and nesting of sweep loops is supported. Therefore a potentially huge number of jobs can be encoded in a single document.

This specification is still in progress within the JSDL-WG and it is expected to be finalized during the summer of 2008.

2.6 Basic Execution Service (BES) (GFD 108)

The BES specification defines interfaces for creating, monitoring, and controlling computational entities such as UNIX or Windows processes, or parallel programs—what we call *activities*. Clients define activities using JSDL. A BES implementation executes each activity that it accepts. A BES resource may represent a single computer; a cluster managed through a resource manager such as Load Sharing Facility (LSF), Sun Grid Engine (SGE), Portable Batch System (PBS), Windows HPC Server 2008 (HPCS 2008) or Condor; or even another BES implementation.

2.7 HPCP-Application Extension (GFD 111)

The HPC Profile Application Extension specification describes additions to the JSDL's Application element to support the definition of applications for execution on HPC resources. Specifically, it allows an execution task to be given a name, to specify the path to the executable and any specific working directory, any arguments that need to be passed to the executable, any environment variables needed for the executable, to specify the standard input/error/output file paths, and to specify the user identity the task should be run under if it is different from the user identity of the user submitting the task.

2.8 HPC Basic Profile (HPCBP) (GFD 114)

The HPCBP is a profile on BES and JSDL which uses the HPCP Application Extension to support a minimal set of capabilities to satisfy particular use cases around the submission of HPC applications to a cluster resource. The set of operations does not include data staging, delegation or application deployment. As there was not an agreed upon security model at the time the document was developed, the profile includes username/token and X.509 token credential profiles from WS-I BSP (WS-I Basic Security Profile) for authentication. The profile was written during the summer and early Fall of 2006, with an interoperability demonstration at SC'06 in Tampa with a dozen different implementations from around the world.

2.9 File Staging Extension to the HPC Basic Profile (GFD 135)

The JSDL specification supports a 'DataStaging' element that allows a remote source or target data to be mapped to a local copy. If a source data URI is specified the data is copied from the remote source to the local destination before the execution task commences. If a target data URI is specified the data is copied from the local source to the remote destination after the task has been executed. The user is able to define the behavior if the data already exists locally on the machine and if the data is to be removed once the activity is over.

The HPCBP File Staging Extension uses the JSDL model and defines its behavior for common file movement protocols – http(s), ftp(s) and scp – and specifies how credentials specific to each data action can be provided to enable access to secured resources. These credentials can be different for each data source/sink and different from the credentials used to access the computational resource. A compliant implementation must support one of WS-Security's Username Token element or the X.509 Certificate Token element encodings.

The state model used within the HPCBP Service can be extended by the implementation to reflect if the task is staging data in or out as part of its execution phase.

Although the HPCBP File Staging Extension only defines a small core set of protocols an implementation can support additional mechanisms such as GridFTP, DMI (Data Movement Interface), e-mail, RNS (Resource Namespace Service) for data movement and naming.

2.10 ByteIO (GFD 87)

ByteIO provides POSIX-like read and write operations on sequences of bytes, there are two variations of this interface. In the RandomByteIO interface, the offset, number of bytes, and data buffers are passed to the operations. In the Streamable ByteIO interface, the operations do not take the offset.

2.11 GLUE (In public comment)

The GLUE 2.0 specification provides a defined model for describing cluster resources, and more broadly resources that may be shared between different communities through defined service interfaces.

2.12 Distributed Resource Management Application API (DRMAA) (GFD 22)

The Distributed Resource Management Application API (DRMAA) has been an OGF standard since 2004. It provides an API around the submission and monitoring of jobs into local distributed resource management systems. See www.drmaa.org for more details.

2.13 Resource Namespace Service (RNS) (GFD 101)

While EPRs are convenient for applications to manipulate they can easily exceed hundreds of characters in length making them awkward for humans to use. Further, the EPR namespace usually does not represent relationships between EPR's. To address these short-comings and make Grids more human friendly the RNS (OGF: Resource Namespaces Service) provides a hierarchical directory structure that maps string paths to EPRs much as a Unix directory maps string paths to *inodes*. For example, suppose I have the RNS path "/biology/databases/Sequences/pir21.a". RNS can be used to map the human readable path to a WS-Addressing EPR. The EPR can be used to directly access the resource.

3 Scenarios

In this section we will describe in detail the main motivating HTC and HPC scenarios for ISVs illustrating how OGF specifications can be used to provide an interoperable infrastructure. The scenarios will observe the requirements for:

- The client workstation to have an IP address that can be accessed by the compute cluster in order for the running application to access a service on the client workstation.
- The client workstation to be available throughout the job's complete life-cycle (i.e. submission, queuing and execution).

For each scenario we will describe the:

- Environment: How the application in the scenario communicates with the software.
- Pre-requisites: The software components/packages that need to be installed on the system.
- Interactions: The ordered interactions between the components in the system.
- Benefits: The advantages offered by this scenario for the end-user and systems administrator.

3.1 Direct Job Submission using a proprietary Command Line Interface

3.1.1 Environment

This scenario assumes a shared file system between the submission host and the compute cluster. This is frequently a requirement of the scheduling software and common place within many Enterprise deployments. Many applications invoke shell scripts that can be customized to a particular job management software, e.g.,

qsub job_script

3.1.2 Pre-requisites

In addition to the operating system on the submission host and across the cluster the following software needs to be installed and configured:

- Distributed Resource Management software on the submission host and the cluster
- The application on the submission host and the compute cluster

3.1.3 Interactions

1. Interaction between the local user with the client application to formulate the problem that needs to be solved on the compute cluster.
2. Identification and generation of any configuration or data files needed by the application when it is executed on the remote cluster.
3. Use of the DRMs proprietary interface (either through a programmatic API or by directly invoking the command line tools) to submit, monitor and manage a job.
4. Display or analysis of any output files by the application or the user.

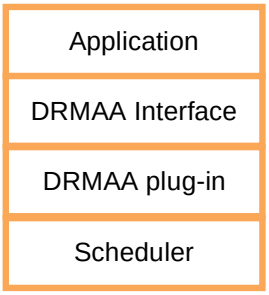
3.1.4 Benefits

This scenario represents a viable deployment and realistic scenario for ISV deployments that are completely contained within the Enterprise. The client needs to stay connected to the network during this scenario in order to provide access to any local files.

3.2 Direct Job Submission using a standardized API

3.2.1 Environment

This scenario assumes a shared file system between the submission host and the compute cluster. This is a commong feature of job scheduler deployments within an Enterprise, although it is rarely a requirement driven by the chosen job scheduler. In addition, a library provided by the scheduling vendor, is required to bridge the DRMAA (Distributed Resource Management Applications API) operations used within the ISV application to the locally installed scheduler.



3.2.2 Pre-requisites

In addition to the operating system on the submission host and across the cluster the following software needs to be installed and configured:

- Distributed Resource Management software (the job scheduler) on the submission host and the cluster
- The application on the submission host and the compute cluster
- Dynamic library that implements the DRMAA specification for the locally installed DRM software needs to be installed on the client.

3.2.3 Interactions

1. Interaction between the local user with the client application to formulate the problem that needs to be solved on the compute cluster.
2. Identification and generation of any configuration or data files needed by the application when it is executed on the remote cluster.
3. Use of DRMAA's programmatic interface to submit, monitor and manage a job.
4. Display or analysis of any output files by the application or the user.

3.2.4 Benefits

This scenario represents a viable deployment and realistic scenario for ISV deployments that are completely contained within the Enterprise. For the ISV it provides a programmatic interface that is independent of the deployed scheduling software which means that once implemented the application will run with any scheduling software compliant with the DRMAA specification. Higher level APIs such as SAGA (Simple API for Grid Applications) can also be consumed by the ISV developer and these may internally use DRMAA (or other APIs) to access the scheduler infrastructure. This scenario moves the burden of providing a platform specific integration with the job scheduler form the Application ISV to the Scheduling ISV. The Application ISV works to the DRMAA interface while the Scheduler ISV provides a plug-in that links the DRMAA interface to the native scheduling interface.

3.3 Direct Job Submission through a Web Service

3.3.1 Environment

This scenario assumes a shared file system between the submission host and the compute cluster. The application ISV makes use of a web service protocol and interface to communicate with a web service endpoint that provides access to the compute cluster. The main specification used to define this behavior is the HPC Basic Profile specification (and its dependent specifications – BES, JSDL and HPCP Application specifications). This web service may be accessed by the application directly consuming the web service interface or by using an intermediate library.

3.3.2 Pre-requisites

In addition to the operating system on the submission host and across the cluster the following software needs to be installed and configured:

- The application on the submission host and the compute cluster.
- Web service client environment installed on the client or integrated into the application on the submission host.
- Web service hosting environment and service installed on the Enterprise network so that it can submit, monitor and manage jobs within the compute cluster.

- The web service on the cluster needs to be accessible from the client.

3.3.3 Interactions

1. Interaction between the local user with the client application to formulate the problem that needs to be solved on the compute cluster.
2. Identification and generation of any configuration or data files needed by the application when it is executed on the remote resource.
3. Formulation of the JSDL document using:
    a. HPC Profile Application if a single processor job
    b. JSDL-SPMD if a multi-processor job
    c. JSDL-Parameter Sweep specification if a parameter sweep job
4. Submission of the JSDL document to the cluster's HPCBP endpoint.
5. Execution of the application.
6. Display or analysis of any output files by the application or the user.

3.3.4 Benefits

This scenario represents a viable deployment which requires a minimum of dependent software on the client. The client side web service stack can be fully contained within the application instead of a having to deploy a mixture of DRM specific software and an interface/wrapper around the DRM software. A web service is needed on the Enterprise network, accessed from the client, for submitting, monitoring and managing the jobs within the DRM. The result is a simpler client that can provide more predictable interactions with its environment for the application ISV.

3.4 Direct Job Submission through a Web Service using File Staging

3.4.1 Environment

This scenario assumes no shared file system between the client workstation and the compute cluster. The application ISV makes use of a web service protocol and interface to communicate with a web service endpoint that provides access to the compute cluster. The main specification used to define this behavior is the HPC Basic Profile specification (and its dependent specifications – BES, JSDL and HPCP Application specifications) and the HPCBP File Staging extension. This web service may be accessed by the application directly consuming the web service interface or by using an intermediate library. In addition, any files needed by the application during execution are explicitly transferred to the cluster, through the external storage server,

3.4.2 Pre-requisites

In addition to the operating system on the client workstation and across the cluster the following software needs to be installed and configured:

- The application on the client workstation and the compute cluster.
- Web service client environment installed on the client or integrated into the application on the client workstation.
- Web service hosting environment and service installed on the Enterprise network so that it can submit, monitor and manage jobs within the compute cluster.
- External Storage Server that supports at least one file movement protocol service that is supported by file movement protocol clients on both the client workstation and the compute cluster.

3.4.3 Interactions

1. Interaction between the local user with the client application to formulate the problem that needs to be solved on the compute cluster.

2. Identification and generation of any configuration or data files needed by the application when it is executed on the remote resource.
3. Formulation of the JSDL document using:
    a. HPC Profile Application if a single processor job
    b. JSDL-SPMD if a multi-processor job
    c. JSDL-Parameter Sweep specification if a parameter sweep job

    and the HPCBP File Staging extension to specify the transfer of any dependent input and/or output files from external storage server to the compute cluster using one of the supported protocols.
4. Uploading of the files to the designated external storage server if required by the client using a protocol supported by the client and external storage server.
5. Submission of the JSDL document to the cluster's HPCBP endpoint.

The client workstation may disconnect from the network at this point with the next steps taking place asynchronously on the cluster. These are driven by the middleware on the compute cluster through the local file system.

6. Downloading of the files to the cluster from the external storage server.
7. Execution of the application.
8. Uploading of any output files to the designated external storage server.

The client workstation either:

a) Has remained connected to the network and recognizes through polling that the activity is complete.
b) Reconnects to the network and realizes that the activities it has dispatched are complete.

Execution on the client workstation continues:

9. Retrieval of the output files from the designated external storage server to the client workstation.
10. Notification (locally) to the user that the job is complete.

3.4.4 Benefits

This scenario allows a client workstation to disconnect from the network once the job has been submitted and to reconnect to download any output files once the job is complete. No DRM specific software needs to be deployed on the client workstation. The client side web service stack can be fully contained within the application instead of a having to deploy a mixture of DRM specific software and an interface/wrapper around the DRM software. A web service is needed on the Enterprise network, accessed from the client, for submitting, monitoring and managing the jobs within the DRM.

3.5 Direct Job Submission through a Web Service with run-time interaction

3.5.1 Environment

This scenario assumes no shared file system between the client workstation and the compute cluster. The application ISV makes use of a web service protocol and interface to communicate with a web service endpoint that provides access to the compute cluster. The main specification used to define this behavior is the HPC Basic Profile specification (and its dependent specifications – BES, JSDL and HPCP Application specifications). This web service may be accessed by the application directly consuming the web service interface or by using an intermediate library. Two-way interaction between the client workstation and the application running on the compute cluster takes place through an intermediary ByteIO service that could be running on the compute cluster's head node.

3.5.2 Pre-requisites

In addition to the operating system on the client workstation and across the cluster the following software needs to be installed and configured:

- The application on the client workstation and the compute cluster.
- Web service client environment installed on the client or integrated into the application on the client workstation.
- Web service hosting environment and service installed on the Enterprise network so that it can submit, monitor and manage jobs within the compute cluster.
- ByteIO service that is accessible to both the client workstation and the compute nodes in the compute cluster.

3.5.3 Interactions

1. Interaction between the local user with the client application to formulate the problem that needs to be solved on the cluster resource.
2. Identification and generation of any configuration or data files needed by the application when it is executed on the remote resource.
3. Initialization of a ByteIO endpoint on the cluster's head node.
4. Formulation of the JSDL document with the ByteIO endpoint:
    a. HPC Profile Application if a single processor job
    b. JSDL-SPMD if a multi-processor job
    c. JSDL-Parameter Sweep specification if a parameter sweep job
5. Submission of the JSDL document to the cluster's HPCBP endpoint.

Activity on the compute cluster and on the client workstation takes place concurrently. A two-way channel is established between the client and the running application allowing the client workstation to 'steer' the application and the application to deliver intermediate results back to the client.

On the compute cluster	On the client workstation
6. Initialization of the application.	
7. Delivery of interim results to the ByteIO endpoint on the head node.	Retrieve interim results from the ByteIO endpoint and act on them.
8. Completion of the application.	Poll the activity until it is complete.

Execution on the client workstation continues:

9. Notification (locally) to the user that the job is complete.

3.5.4 Benefits

This scenario allows a client workstation to both retrieve interim results from a running application and to send control commands to the running application. This requires no shared file system and only a ByteIO service that is visible to both the client workstation and the running application. No DRM specific software needs to be deployed on the client workstation. The client side web service stack can be fully contained within the application instead of a having to deploy a mixture of DRM specific software and an interface/wrapper around the DRM software. A web service is needed on the Enterprise network, accessed from the client, for submitting, monitoring and managing the jobs within the DRM.

There is no requirement that the client workstation is directly addressable by the cluster or a need for a shared file system. The application needs to be modified to deliver interim output to the ByteIO endpoint. The client is able to operate behind firewall with NAT yet the running application is able to deliver results back to the client.

4 Implementation Architecture

4.1 Client Workstation

The administrator of the client machine will need to install any client components of the application software and any client components of the middleware software needed to support interactions from client to other components in the system. The middleware components may include:

- JSDL & HPC Profile Application Extension – the schema for defining the job (execution task and any data movement) to be executed remotely.
- HPC Basic Profile – the web service client protocol communicating with the remote cluster resource.
- Data Movement – the client protocol(s) that will be used to move any files/data to/from the client to the External Storage service.
- ByteIO – the web service client protocol needed to send/receive information to/from the ByteIO service on the head node.

In addition, the client workstation environment will need to identify the local credentials that will be used by the client to identify themselves to the cluster resource. If no local credential is available that can be used to identify the client with the cluster's head node than an alternative credential will have to be obtained. Once the credential has been identified it will be passed through the WS-Security protocol to the remote services.

4.2 External Storage Server

The external storage server provides a location, accessible to both the client workstation and the compute nodes for the exchange of data – primarily files in these scenarios. It will need to support at least one of the Data Movement services supported by both the client workstation and application (compute node) environments.

4.3 Compute Cluster's Head Node

The cluster administrator needs to perform a one off installation and configuration of the server side components of the 'middleware' software – the bits that glue the application, the operating system and the cluster management software together. The services on the head node (in addition to those needed by the local resource manager) may include:

- HPC Basic Profile Service – a web service to submit and manage jobs on the cluster which will include parsing the JSDL document submitted to the service.
- ByteIO Service – a web service that provides a 'rendezvous' point for the client workstation and the application to exchange data while the application is running. This is ONLY needed if the application has to deliver interim results to the client workstation and the compute nodes are not able to access the client workstation directly.

It is assumed that the base operating system and cluster management software (e.g. PBS, LSF, Windows HPCS 2008, SGE, etc) are already installed and working. The middleware is the non-application specific software necessary to connect the client to the cluster scheduler and the application. Software from different applications will also need to be installed on cluster.

4.4 Applications Running on the Compute Cluster's Compute Node(s)

The application will need to support:

- ByteIO – the web service client protocol needed to send/receive information from the application during its execution. This is ONLY needed if the application has to deliver interim results to the client workstation. If the running application is not able to access the client workstation directly, it can use the ByteIO service on the head node as a rendezvous' point.
- Data Movement – the client protocol necessary to move the data from/to the External Storage server. This may be embedded in the application or implemented as part of the cluster's execution infrastructure, e.g. HTTP, HTTPS, FTP, GridFTP, etc.

4.5 Validating the Deployed Software Environment

Several verification steps can be identified on both the cluster and the client machine before attempting any of these scenarios:

- Run the application on the cluster from the head node or submission host (without any use of the 'middleware'):
  - On a single processor.
  - On multiple processors as part of a parallel (MPI) job.
- Upload a file from the head node to the external storage server.
- Download a file from the external storage server to the head node.
- Upload a file from a compute node to the external storage server.
  - This may not be supported by the cluster if compute nodes have no external network visibility.
- Download a file from the external storage server to a compute node.
  - This may not be supported by the cluster if compute nodes have no external network visibility.
- Upload a file from the client machine to the external storage server.
- Download a file from the external storage server to the client machine.
- That there is network connectivity from the client machine to the cluster.
- Upload a file from the client machine to the cluster's head node.
- Download a file from the cluster's head node to the client machine.
- The credentials provided by the user on the client machine can access the middleware services on the head node.
- The credentials provided by the user on the client machine can access and submit a job through the service on the head node to the cluster.

Running through these tests, both on installation and during use, provides the means to self-configure the client to adapt to its current network environment, and the means to trouble shoot any failures.

5 Advice to ISVs

This document provides a systematic approach, using established and emerging web service standards, to common usage scenarios in a distributed computing environment. The availability of this infrastructure (from platform providers) would allow ISVs to concentrate on the domain specific functionality that they provide through the application, rather than the inherent complexity of moving bytes around complicated firewalled networks.

Implementing the client aspects of these web service standards is relatively straightforward – through the availability of open-source toolkits or commercially supported sample code and frameworks.

The projects/products in the following table use a variety of open and closed source toolkits across several different operating systems (e.g. Java & C Apache Axis, Windows Communication Framework, gSOAP, etc.) and with either community or commercial support. The interoperability experience gained by the OGF community with these varied technologies has already been established through the experience documents provided in support of these standards and integrated into the software provided by each product or project.

The following table records the support (as of May 2008) of the specifications discussed in this document. All of these projects use WS-Addressing when required and elements of WS-Security for authentication, transport or message level security. The JSDL extensions for SPMD and Parameter Sweep applications are in their early stages of adoption. The HPC Basic Profile includes support for the HPC Profile Application extension. Support for the DRMAA specification is recorded on http://www.drmaa.org.

	Specifications						
Project or Product	OGSA-BSP 2.0	JSDL	OGSA-BES	HPCBP	File Staging	OGSA-ByteIO	RNS
Globus	No	Yes	Yes	Yes	No	No	No
UNICORE 6	No	Yes	Yes	Yes	No	Yes	?
USMT (Fujitsu)	Will	Yes	Yes	Yes	No	Yes	?
HPCS 2008 (Microsoft)	No	Yes	Yes	Yes	Yes	No	No
Genesis II	Will	Yes	Yes	Yes	Yes	Yes	Yes
GridSAM (OMII-UK)	No	Yes	Yes	Yes	Yes	No	No
Crown	No	Yes	Yes	Yes	No	No	No
BES++ (Platform)	No	Yes	Yes	Yes	Yes	No	No
NAREGI	No	Yes*	No	No	No	No	?
Gfarm	No	No	No	No	No	Will	No
gLite	No	Prototype	Prototype	Prototype	No	Prototype	?
ARC1 (NorduGrid)	Not planned	Yes	Yes	Yes	No	Yes	No

Figure 3: Specification support in selected projects and products

* Provides support for JSDL SPMD

\+ Provides support for JSDL Parameter Sweep

6 Advice to Platform Providers

A platform provider in this context is a supplier of the middleware that will be utilized by the ISV application to initiate work on the cluster resource. The platform provider may be a an open

source platform provider that integrates with the cluster's job scheduling software, or provider of job scheduling software that also supports these middleware specifications.

Support of these specifications enables the use by an ISV application of distributed computing infrastructure in a systematic, standardized and interoperable manner that is not currently possible within many common network environments.

7 Open Issues

7.1 How to select a resource for Direct Job Submission

In the scenarios described in Section 3, it was known in advance which HPCBP resource was to be used. Frequently, an enterprise may have many different HPCBP resources each representing different physical computing clusters. Each HPCBP resource has a unique WS-Addressing endpoint reference (EPR) that may be used as the target of the Web Service interaction as described in Sections 3.3, 3.4 and 3.5.. The challenge is to select an HPCBP resource for the client to use. There are several ways this may be accomplished. Four approaches are described here: querying an information service, use of an RNS path name, use of a meta-scheduling HPCBP resource, and use of OGSA-RSS services.

7.1.1 Querying an Information Service

An information service is queried using resource description terms, e.g., OperatingSystem=Windows, PhysicalMemory>2GB and a set of resources matching these requirements is returned in an XML document. The GLUE specification provides one example of a rich schema for describing distributed computing and storage capabilities. Currently, the HPCBP (through the underlying Basic Execution Service) supports a limited basic schema for describing the resources contained within an endpoint. Each XML document contains relevant HPCBP and the EPR of the corresponding HPCBP resource that satisfies the query. The client selects one of the HPCBP endpoints, and the scenario proceeds as described previously.

In this approach, resources matching a user specified query are returned to the client where a decision is made as to which resource is to be used.

7.1.2 Traversing a RNS path

Different HPCBP resources are placed into a directory structure served by a Resource Namespace Service (RNS). A RNS provides a hierarchical namespace (i.e. a directory structure) that maps path names to EPRs. For example, an RNS directory "/clusters/USNationalCenters" could include entries "TACC", "IU", and "NCSA" that point to HPCBP resources representing clusters at TACC, IU, and NCSA respectively. The client software could let a user browse the RNS namespace and select the cluster they wish to use, e.g., "/clusters/USNationalCenters/TACC". Once an HPCBP endpoint has been selected the scenario proceeds as described previously.

In this approach the user is able to browse a set of resources, organized hierarchically, to select a resource that is to be used.

7.1.3 Meta-scheduling HPCBP resource

In this scenario another HPCBP resource acts as a proxy that encapsulates a number of other HPCBP resources. This "meta-scheduling" HPCBP resource receives client requests and internally selects a compatible HPCBP resource by some internal policy and delegates the job to the selected HPCBP resource. From the clients' perspective the interaction is as if the meta-scheduling HPCBP resource is doing the job on its own. Note that security delegation may become an issue, though if the HPC-BP File Staging extensions are being used data access credentials may be carried in the JSDL.

In this approach the user submits their job to a resource which makes a decision internally as to which resource to route the job to.

7.1.4 Resource Selection Services

The OGSA Execution Management Services section of the OGSA Architecture document describes an execution architecture that includes “job managers”, execution services (e.g., HPCBP), information services, and resource selection services (RSS). Resource selection services are responsible for taking job descriptions such as might be found in a JSDL document, determining a candidate set of execution services, and selecting the best (by some optimization function that may only generate “good” choices) resource on which to execute the job. The OGSA-RSS working group has developed a proposed interface.

This approach allows a user to define their resource requirements and for the RSS to provide an ordered list of one or more selected HPCBP resources.

7.1.5 Benefits

The benefits of these approaches are that the client is not tied to one particular resource. Each time a resource is needed an appropriate resource can be selected from those that are currently available to the user. Instead of the client needing to know of a specific ‘well-known’ compute cluster it needs to know of a ‘well-known’ source of information that contains details of the compute clusters that it can go use.

7.2 Interacting with the Web Service Infrastructure

An open issue is how the application developer will interact with the web service infrastructure. Will they interact through a programmatic API, either generated by directly consuming the web service interface using a proprietary service stack or through a portable interface that could be implemented by any service stack? Alternatively, the application could invoke a set of command line tools provided by a software supplier that encapsulates the interaction with the web service.

7.3 Integration with Existing Security Infrastructures

Any web service infrastructure that is deployed has to integrate with existing service infrastructures – deploying a new security infrastructure to support a particular application or web service is not an option. Therefore, what are the ‘native’ security mechanisms that are currently being seen in deployed environments and how are these stored and how can they be discovered transparently by the web service frameworks? How are these security tokens used for authentication and authorization checks on the compute cluster, the external storage server and can they be integrated into the file transfer protocols (e.g. ftp/scp/http/…)?

8 Security Considerations

The scenarios described in this document provide some significant security challenges – mobile clients, high value computing resources, and simulation data with potentially high commercial value. In addition, the middleware that needs to be deployed to enable these scenarios MUSTintegrate with the existing security (and software) infrastructures.

Many of the web service toolkits mentioned in this document utilize elements of the WS-Security specification. The use of this specification is further qualified by the broader web service community by the WS-I BSP (Basic Security Profile) and the grid community by the OGSA-BSP 2.0. Individual specifications in this document have been very specific in defining their supported security models. The HPC Basic Profile and File Staging Extension uses WS-Security’s Username and Password token and X.509 Certificate token profiles.

9 Author Information

Steven Newhouse
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052
USA
steven.newhouse@microsoft.com

Andrew Grimshaw
University of Virginia
Charlottesville, VA
USA
grimshaw@virginia.edu

10 Contributors & Acknowledgements

We gratefully acknowledge the contributions and discussions made to this specification by Narfi Stefansson and other members of the UVA Workshop and the OGSA Working Group.

We would like to thank the people who took the time to read and comment on earlier drafts. Their comments were valuable in helping us improve the readability and accuracy of this document.

11 Full Copyright Notice

12 Intellectual Property Statement

The OGF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights which may cover technology that may be required to practice this recommendation. Please address the information to the OGF Executive Director (see contact information at OGF website).