Grid Standards and APIs

Introduction

This page will provide a high level overview of different grid standards for the service layer of each system identified as a site that Ergo will need to support for running jobs. For the service layer of each system, a brief overview will be provided and for more detailed information, follow the links to the different standards mentioned or the overview page for the entire architecture.

Unicore

Service Layer

More information about the Unicore service layer can be found here: http://www.unicore.eu/unicore/architecture/service-layer.php

UNICORE/X

- UNICORE/X server is a WSRF compliant web service, it provides access to storage resources, file transfer services and job submission and management services.
    - UNICORE has a proprietary set of web service interfaces called UNICORE Atomic Services (UAS). It offers a complete, integrated set of basic services to higher level services, clients and users
    - Standardised set of interfaces based on open, common standards (OGSA-*) is available in UNICORE 6 in addition to UAS. Currently implemented standards are OGSA-BES and HPC-P, used for the creation, monitoring, and control of jobs
- UNICORE Job definition is compliant with the JSDL (+ JSDL HPC extensions) standard. For file transfers (site to site and client to site), a variety of protocols are available (HTTPS, OGSA Byte-IO, UFTP, GridFTP and UDT)
- For user authorization, UNICORE/X can use a variety of attribute resources such as XUUDB, UVOS or flat files. The actual authorization engine uses XACML 2.0 policies. Full X.509 certificates are used as a base line. Proxy certificates are optionally supported in UNICORE 6 (e.g. to use GridFTP)
- UNICORE's internal execution management engine and link to the system layer is called XNJS

IDB

- The Incarnation Database (IDB) is used during job incarnation, i.e. the mapping of the abstract job description (in JSDL) to the concrete job description for a specific resource (a shell script). Information about available applications and resource characteristics has to be defined in this database.

XUUDB

- XUUDB is the user database and provides mapping from X.509 certificates to the actual users' logins and roles. The XUUDB component is a web service in itself, so it can be used from multiple UNICORE installations.

UVOS

- UNICORE Virtual Organization Service (UVOS) is an alternative to the XUUDB. This virtual organization (VO) service can be used for user authorization. It uses the SAML standard and offers a wide variety of features.

Registry

- Global service registry where different services can register once they are started. A single service registry is necessary to build-up and operate a distributed UNICORE infrastructure. This service registry is contacted by the clients in order to "connect to the grid". Like UNICORE/X, the service registry runs in UNICORE's WS-RF hosting environment.

CIS

- The Common Information Service (CIS) is the information service of UNICORE 6. It gathers both static and dynamic information from all connected XNJS, which are then displayed either in raw XML or human-readable text form. As longitude and latitude information is also stored, Google maps views allows a geographical representation of the Grid infrastructure.
- CIS is based on the GLUE 2.0 standard from the Open Grid Forum.

Workflow Engine

- The workflow support in UNICORE 6 is implemented as a two layered architecture consisting of a workflow engine and the service orchestrator layer. The workflow engine deals with high level workflow execution.
- The workflow features of UNICORE 6 can be used from the graphical UNICORE Rich Client (URC) and from the command line with UCC.

Service Orchestrator

- The Service Orchestrator layer is responsible for executing individual tasks in a workflow, handling job execution and monitoring on the grid. Different brokering strategies are implemented to find the best suited resource for each workflow step. Other brokering strategies can be plugged in.

Clients

- UCC - Command line client
  - It might be possible to use DataWolf (formerly Cyberintegrator) to wrap the command line client commands as tools
  - Download: http://www.unicore.eu/download/unicore6/
- UNICORE Rich Client
  - Eclipse-based UNICORE Rich Client (URC) offers the full set of functionality to the users in a graphical representation.
  - Uses GridBeans for interacting with Grid Services
  - Download: http://www.unicore.eu/download/unicore6/
- HiLA - High Level API for Grid Applications
  - This is a layer of abstraction for use with web clients, but it might provide a way to either run jobs directly from the current Ergo architecture or for building DataWolf (Cyberintegrator) Java tools to run jobs
  - Download: http://www.unicore.eu/download/unicore6/

Unicore - XSEDE

https://www.xsede.org/software/unicore

XSEDE

Software

- Globus Toolkit
  - OGSA
  - OGSI
  - WSRF
  - Job Submission Description Language JSDL
  - DRMAA
  - WS-Management
  - WS-BaseNotification
  - SOAP
  - Web Service Description Language (WSDL)
  - Grid Security Infrastructure (GSI)

KISTI HPC Open API

Genesis II

Genesis II is an open source, standards based grid middleware designed to support high-throughput computing by the University of Virginia Department of Computer Science and the Virginia Center for Grid Research. Genesis II follows grid standards from the Open Grid Forum (OGF), W3C, and OASIS, including many from the Open Grid Services Architecture (OGSA) and the Web Services Resource Framework (WSRF). Genesis II adheres to parts or all of OGSA Basic Execution Services (BES), Resource Naming Service (RNS), OGSA ByteIO, WS-Security, WS-Naming, WS-Trust, and Job Submission Description Language (JSDL). Genesis II supports X.509 certificates and username/password (based on WS-Security and OGF Basic Security Profile).

Main wiki: http://genesis2.virginia.edu/wiki/

Clients

The primary access seems to be through a client using the graphical user interface or the terminal.

- Genesis II Commandline client
  - run commands in GUI or terminal
  - It might be possible to use DataWolf (formerly Cyberintegrator) to wrap command line client commands as tools
  - Reference: http://genesis2.virginia.edu/wiki/Main/GenesisIICommands

General Standards

- Open Grid Forum (OGF)
- World Wide Web Consortium (W3C)
- Advancing Open Standards for the Information Society (OASIS)
- IETF
- Open Grid Services Architecture (OGSA) - a distributed interaction and computing architecture based around services, assuring interoperability on hetergenous systems so that different types of resources can communicate and share information. OGSA is based on several other web service technologies such as the Web Services Description Language (WSDL) and the Simple Object Access Protocol (SOAP), but it aims to be largely independent of transport-level handling of data.
- Web Services Resource Framework (WS-RF 1.2)
- Open Grid Services Infrastructure (OGSI)
- Job Specification Description Language 1.0 - describes the requirements of computational jobs for submission to resources, particularly in Grid Environments, though not restricted to the Grid.
  - Reference
  - XML based language
  - Adopted by several software platforms - compatible with UNICORE 6, Genesis II, Windows HPC Server 2008, etc
  - Not intended for human consumption

Grid Software

- Globus Toolkit - an open source software toolkit used for building grids that follows OGSA standards
- Simple API for Grid Applications (SAGA) - an abstractions-based suite of well-defined capabilities that are architected for scalable, interoperable and sustainable approaches to support science on a range of high-performance and distributed computing systems
  - http://saga-project.org
  - jSAGA - http://grid.in2p3.fr/jsaga/

Questions / Notes

1. What are the target resources to be considered? We should pick one or more current and/or upcoming HPC systems that ERGO must be able to submit to so we can determine if a common standard is used across the systems (e.g. do multiple target systems have client libraries supporting JSDL).
2. Can we write HPCExecutors for each system and put common operations inside a parent class? For example, if all systems support JSDL, but use a different client library for job submission, the concrete implementation could call the client libraries
3. How does Taverna submit jobs to Unicore? There is a plugin, but which client library to submit the JSDL?