NORDUGRID-TECH-13
4/2/2005

ARC Compute Resource Management Interface

Technical Description, DRAFT!!!

NorduGrid Collaboration*

Refers to ARC release series 0.4 and up

*Comments to: nordugrid-discuss@nordugrid.org

1 Introduction

The document describes the available methods and channels for clients to interact with Grid-enabled computing resources running the ARC middleware. Client interactions with data management systems (e.g. data indexing services, grid storages) and resource indices (resource registries) are out of the scope of this document.

2 ARC Computing Resource

what is assumed about a computing resource, what services are required, what are the open channels, etc. what are the capabilities of a CR,....

An ARC computing resource is capable of

- accepting job submission requests of clients and 'forwarding' those to the local batch system
- acting actively in the stage-in process of grid job, collecting input data
- altering the grid job's execution in the local batch system
- providing information about itself
- providing information about grid jobs running on the resource
- providing access to the grid job's session directory
- assisting in the stage-out process of the grid jobs

3 ARC grid job

what is a grid job in an ARC system, what can be done with a job, etc..

3.1 Formulating a grid job request: XRSL ARC job description language

XRSL is used to:

- describe the stage-in process
- specify requirements with respect to computing resource: XRSL attributes are matched against the information system attributes during the brokering process.
- describe process to be executed
- describe evironment/preinstalled software required
- describe the stage-out process

description of the ARC specific attributes, ...

4 ARC Clients

Clients in ARC are individual agents, everything can be a client which is capable of speaking the language of ARC grid resources. Clients interact with different type of grid components, such as computing resource, data and resource indexing services. This document focuses only on the computing resource interaction.

The client tasks within ARC are the following

- interpret the user's job request formulated in XRSL: a sort of client - user interaction
- find Grid resources by contacting the resource index services: resource discovery
- obtain information about discovered grid resources: information collection
- select the best suitable resource by comparing the collected information and the user's job request: brokering
- submit a job to the selected grid computing resource: job submission
- find input data by contacting data indexing services
- handle the input data required by the grid job: grid data stage-in
- interaction with active grid jobs (kill, delete, resubmit, etc.): job management
- checking job-related information: job status monitoring
- handle the job's output data: grid data stage-out

The next sections expand those client tasks where computing resources are involved. Please notice that client tasks of XRSL interpretation, resource discovery, resource selection (or brokering) do not involve interaction with computing resources.

5 Client - Computing resource interactions

This section gives a detailed overview of those client tasks where the client interacts with the computing resource. Figure **??** shows a schematic view of the different interactions between ARC client and Computing Resource. Also if information about behavior of Computing Resource frontend is provided.

5.1 Information collection

the interface to the local ldap information tree, what sort of info is collected, resource representation via nordugrid schema.

5.2 job submission

Job submission is done by contacting computing resource frontend through GridFTP interface. Job control commands are mapped to FTP commands and information exchange is done by uploading/downloading files. Each job is represented by a virtual directory and job management is performed by applying FTP comamnds to that directory.

For job submission client obtains jobs ID(name of directory, CWD command is used) and uploads job's description.

If there are any files to be used by job on a machine where client runs, it uploads them to provided directory

5.3 job management

Job cancelation and cleaning is done by applying DELE and RMD commands to job's directory.

Full content of a directory in which local job is being executed is fully accessible through same interface (read-only during execution). Also information about job's processing (status, debug information, local attributes) is available through a set of virtual directories and files. This makes it possible to fully analyze job's general and application specific state at any time.

5.4 job status monitoring

job monitoring via ldap, job states, job info schema

5.5 input data management

Input data is analyzed by the client (ngsub) and properties of input files are extracted (if possible) to be used in brokering.

Files residing on user's machine are uploaded to the job's directory and job's description is modified to provide information about their properties (currently size and checksum).

Stage-in of the other input files is handled on server side by downloading them into job's directory. Retries and caching are supported.

5.6 output data management

Results produced by job are either delivered to specified location or temporarily stored in job's directory (typically for 24 hours) as specified in job's description. Results not specified in job's description are erased immediately after execution finished.

6 Next Generation ARC interface

Here we present our future plans, which is basically our list of requirements for job managemnet interfce capable of handling both grid-enabled and legacy jobs. Most significant requirements are:

1. Interface must be complete. Job control should not be split among different interfaces and services. In a worst case one action should require no more than one interface. And preferably all actions should be grouped together in same interface.

2. Interface must be expandable without necessity to redefine it. Hence it should have well balanced set of initially defined actions and effective way to define and negotiate new ones. Alternatively every new version must support all actions and attributes of old one in a transparent way.

3. Interface must be backend independent. Interface should not depend on a computing backend. It should be no separate interface for fork, for pbs, for condor, etc. Capabalities of computing resources should be represented in general enough form. Actual capabilitioes could be negotiated or discovered during job submission. But it should be no predefined set of capabilities because one can never be sure how much LRMS mutates tomorrow.

4. Both push and pull modes must be supported. That means both capabilities to push job into service and to pull job form service/storage (like centralised broker) must be provided. Or even better that should be the same interface. This would allow different models of Grid to be built still conforming to same standards.

5. Job description language (if not integrated into interface) must be rich enough to support complex logical constructs (conditions, simple variables, possibly logical and arithmetical expressions).

6. Minimal set of operations to be supported: 6.1 Request for basic descriptions of computing resource 6.2 Negotiation of possibility to run job (lightweight, check if job can run) 6.3 Submission 6.4 Cancelation 6.5 Destruction 6.7 Access to all information related to job - from status to every data produced by job

6.8 Modification of job's attributes during processing (credentials, data access, job description, etc.). 6.9 Restart of job after modifications ...

References

[1] NorduGrid project. `http://www.nordugrid.org`