WG Internal
OGSA-DMI

Michel Drescher, Fujitsu

6 November 2006

Unicore Data Transfer Use Cases

Status of This Document

This document provides information to the Grid community regarding use cases for data transfer in a Unicore based Grid. It does not define any standards or technical recommendations. Distribution is unlimited.

Abstract

This document provides information about the Unicore Data Transfer Use Cases, and their implementation on a high level.
To understand the Unicore Use Case, the document presents a bird eye's view on the Unicore architecture in the first part of the document. The second part presents the principal Unicore Use Cases and how they have been implemented.

Contents

1. Introduction to Unicore

Unicore is a job execution centric Grid Middleware. Everything is modelled as a Job or a Task in Unicore to accomplish the work, and is submitted for execution to an appropriate site.

1.1 Unicore Topology

In a Unicore Grid, each participating site is called “Usite” which exposes its computing resources to the Unicore Grid. The single point of entry into a Usite is a Unicore Gateway at that site. Within a Usite, a site may partition its computing resources into one or many Vsites, being defined as a set of computing resources maintaining a shared filespace. Each Vsite is maintained and run by a server called “Network Job Supervisor (NJS)”. The NJS takes part of the shared filespace in that managed Vsite. The NJS in turn manages one or more “Target System Interfaces” that in turn front one or more actual computing resources (e.g. a cluster). TSIs are responsible for resource reporting (discovering the resources, e.g. RAM or type and speed of CPUs).

However, common practise is that a TSI fronts a single computing resource, and a NJS manages one TSI.

Before execution, the Gateway authenticates and authorises incoming submissions and passes them to the appropriate Vsite (that is, the Vsite’s NJS takes over control of that submission).

1.2 Job model

A single atomic work unit in Unicore can be called a “Task”. For example, compiling one (or many) source code files into object files is called “CompileTask”. An invocation of Povray, or any other executable, is called “ExecuteTask” and is atomic in that sense as well.

Tasks can be grouped into a Unicore “Job”. As a Job is also a Task, a Job can contain sub Jobs. Coordination of Task execution is done using “Dependencies” which are also part of a Job. A Unicore Job is associated with a Vsite, and as such, under control of the Vsite’s NJS at execution time. This way, a user is able to create hierarchical trees of Unicore Jobs. The root Job, the one Unicore Job that does not have a parent Unicore Job, is submitted to its target Vsite and executed. If the NJS at that Vsite encounters a sub Job, it submits that sub Job to that sub Job’s target Vsite on the user’s behalf, thanks to Explicit Trust Delegation.

NJSs are allowed to submit Unicore Jobs to other Vsites independently of any user-submitted Unicore Jobs. NJSs utilise this functionality to poll status and outcome of sub Jobs, transfer files and discover resources.

Not all Tasks are actually execution Tasks (e.g. compiling source code, executing an application) that execute at a target system (e.g. a Linux cluster). Other Tasks are management Tasks (e.g. controlling Jobs and Tasks, fetching results, discovering available resources, etc.), or miscellaneous Tasks.

The Job model in Unicore is built upon the Java Object model. That is, the foundations are laid using abstract Java classes, which are then sub-classed by more concrete classes that can be instantiated. Hence each Task and Job in Unicore (in fact, every object in the Unicore Job model) has an id that is unique in time and space. This id is generated and assigned at Job creation time (i.e. at the client side) instead of the job execution time. This is an important requirement to enable data transfer to a Uspace of a Job that did not execute yet.

Unicore jobs are executed using a user id (i.e. a user’s login name on the targeted TSI). Each task in a Unicore Job is hence executed using the same user id.

File Transfer, or in the domain of OGSA-DMI data movement, is also modelled as Jobs and Tasks that are submitted to a site.

1.3 File Handling in Unicore

Unicore enforces a strict separation between files that are used during Task or Job execution, and those that are not.

During its lifetime, a Unicore Job has an exclusive “Uspace” attached to it. The Uspace is effectively a directory residing in the shared filespace of a Vsite. Files residing in such Uspace are under control of the executing Job.

1.3.1 Portfolios

Within an Uspace, files are never referenced directly in Unicore Jobs, except when incarnated to the TSI. Instead, Tasks in a Unicore Job pass “Portfolio”s around, which are a placeholder for one or many files (or directories) in the Uspace. For example, a CompileTask in Unicore takes a Portfolio as input (the files to compile) and generates a Portfolio as output (the compiled object files). Importantly, one file can be referenced by more than one Portfolio, enabling dynamic regrouping of associated file sets.

A Portfolio’s contents (i.e. the list of files and directories) is determined at runtime. Through the lifetime of a Portfolio its contents can change. The mapping of contents to the Portfolio is handled in the Uspace, not in the Portfolio.

Some Tasks in Unicore show deterministic behaviour in terms of Portfolio contents. For example, a CompileTask creates as many object files as source code files were fed in. The incarnation of such a CompileTask then takes care of a proper association of files in the Uspace with the CompileTask’s output Portfolio.

However, for some Tasks in Unicore, this is impossible as their incarnation is essentially non-deterministic in terms of file handling. For example, a “UserTask” executes a user-provided executable, which principally is able to create any amount of output files. In such cases, after such a Task finishes execution, there exist no association of the Task’s output Portfolio and the files that Task has created in the Uspace. To make such files available to Unicore, the managing NJS needs to get those files on its records. This is accomplished using a special Task in Unicore called “MakePortfolio”.

Usually, files and directories in a Uspace are created by Tasks that are inside the associated Unicore Job. However, files and Portfolios can sometimes be created by external processes, for example Tasks that are outside the associated Unicore Job, or asynchronous file transfers. (In fact, several use cases described below make use of this scenario.) In such cases a semaphore is necessary to synchronise such activities with activities that take place in the receiving Unicore Job’s Uspace. Unicore solves this by placing the contents of such Portfolios temporarily outside the Uspace until the receiving Unicore Job declares that it is now ready to process the received data. This temporal external storage is performed hidden from both the sending external Task, and the receiving internal Task. To declare that a Task is ready to process a Portfolio, that Task may be preceded by a “DeclarePortfolio” Task to ensure that the correct contents of the Portfolio is available: If there is no “hidden” contents, the DeclarePortfolio Task simply does nothing. Otherwise, it takes the hidden Portfolio contents and moves it into the Uspace and updates necessary references, so that the following Task accesses the most current Portfolio contents.

Any file that resides outside of a Uspace of any Unicore Vsite is defined to reside in a “Xspace” which is completely out of control of Unicore. However, Unicore provides several means to move files into and out of Uspaces and between Uspaces, whether remote or at the same Vsite.

2. Unicore file transfer use cases

2.1 Import from user‘s workstation

When the user submits a Unicore Job, necessary files may reside on the user’s workstation. Instead of transferring the files out of bounds (prior to the job submission), the user wishes to transfer the necessary files along with the job submission.

The user has two choices to solve this: Smaller files can be incorporated directly into the Unicore Job as binary contents of a specialised Task. Executing this task will create files in the Uspace, which immediately afterwards need to be revealed as a Portfolio to be of any further use (using the “MakePortfolio” Task).

Alternatively, the user streams the files as a ZipFile utilising the “Unicore Protocol Layer (UPL)” immediately after the corresponding Unicore Job has been submitted using the same UPL connection. The files in the ZipFile correspond to a “DeclarePortfolio” Task in the Unicore Job that has been submitted just before. This technique is quite similar to Internet messages with MIME or DIME attachments, and happens transparent to the user.

2.2 Export to user’s workstation

Frequently, users wish to access a Unicore Job’s result directly on their workstation. A user sends a “RetrieveOutcome” command to the Vsite (the NJS, really) indicating to fetch the outcome of a Unicore Task or Job. If the Unicore Job or Task has finished already, the outcome includes any generated files that are explicitly attached to the outcome.

The NJS sends a “RetrieveOutcomeReply” indicating whether result files are streamed afterwards. This is very similar to clients streaming files to import them from their workstation when submitting a Unicore Job. The result files, if any, are organised in a ZipFile. The files are grouped into directories where each directory corresponds to a Portfolio in the RetrieveOutcomeReply sent earlier.

2.3 Import from Xspace to Uspace within one Vsite

The user may need to process files that are outside the Uspace of a Unicore Job. They may reside on a storage server attached to the Vsite, or in or within certain well-known locations (such as the user’s home directory, the root directory, or a special spool directory) within the Vsite’s shared filesystem.

Unicore provides the specialised “ImportTask” to import files from such locations. The exact source location is modelled as a resource to the ImportTask, and the incarnation of the ImportTask is specific to that resource.

The outcome of the ImportTask is a Portfolio ready for further use in a Unicore Job.

2.4 Export from Uspace to Xspace within one Vsite

The user may want to copy files from the Uspace to another location in the Xspace. This is symmetric to the previous use case.

Unicore provides the specialised Export Task to copy files from the Uspace to the given external location. The target location is given as a resource.

The outcome is a list of strings indicating the directories and files that have been exported.

2.5 File transfer within a Uspace

The user may want to make available a set of output files of a Task for another Task as input files, all within the same Unicore Job.

Unicore allows the user to reference the output Portfolio of the predecessor Task as input Portfolio of the successor Task.

Depending on the nature of the predecessor Task, the user may need to insert a “MakePortfolio” Task between the predecessor and successor Task so that the NJS can track and manage the files. Technically, this use case does not involve any data transfer operation at all.

2.6 File transfer between Uspaces within a Vsite

The user may want to make available a set of output files of a Task for another Task as input files. The predecessor Task and the successor Task are part of different Unicore Jobs that address the same Vsite.

Unicore allows the user to either pull or push a set of necessary files into the target Uspace. Unicore provides the Tasks “GetPortfolio” and “PutPortfolio”, respectively.

For pushing a set of files to a target Uspace, the PutPortfolio Task creates the target Uspace if it does not exist. Files may be overwritten if configured to do so. For pulling a set of files in a Portfolio, the user may configure the Task to overwrite any existing files in the target Uspace.

2.7 File transfer between Uspaces in different Vsites

The user may want to transfer files between Unicore Jobs that execute on two different Vsites. The Vsites may or may not be hosted at the same Usite.

Unicore provides the “Transfer” Task, an idiomatic implementation that defaults if there is no alternative implementation present (i.e. utilising a leased line between the Usites that host the two Vsites). “Transfer” makes use of the fact that NJSs are allowed to submit independent Unicore Jobs to any Vsite in the Grid (given that the target Vsite trusts the emitting NJS). “Transfer” utilises the implementations of two other use cases as follows:

1. **Import from user’s workstation**
   In this case, “user’s workstation” is actually the Uspace of the Unicore Job that contained the “Transfer” Task. In fact, from the perspective of the target Vsite it does not matter whether the data sourced from a user’s workstation, another remote Vsite or anywhere else, given that the submitted Unicore Job is correctly authenticated and authorised.
2. **File transfer between Uspaces within a Vsite**
   A the containing Unicore Job will get its own Uspace, the files in this Uspace need to be moved to their final destination. This is done using the implementation of this use case.

2.8 File transfer between Xspaces in different Vsites

The user may want to transfer files between external storage spaces attached to Unicore Vsites at two different Usites.

Unicore currently does not provide a default implementation for this use case (which is possibly a definition hole). However, it is possible with current Unicore means to construct a functional implementation of this use case (which is in fact what idiomatic implementations do).
A possible default idiomatic implementation may be provided as follows:

1. **Import from Xspace to Uspace within one Vsite**
2. **File transfer between Uspaces in different Vsites**
   If the idiomatic implementation would be used the following sub Tasks would be executed:
   a. Import from user's workstation
      (treating the Uspace as the user's workstation for this use case)
   b. File transfer between Uspaces within a Vsite
3. **Export from Uspace to Xpsace within one Vsite**

3. Security Considerations

This document provides an informational overview on the Unicore Architecture regarding data transfer. Hence many aspects of Unicore are left out, such as security aspects, and this document shall not be used as a reference document on the complete Architecure of Unicore. Nor should it be used as a guideline on how to set up and run a Unicore Grid.

4. Intellectual Property Statement

The OGF takes no position regarding the validity or scope of any intellectual property or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; neither does it represent that it has made any effort to identify any such rights. Copies of claims of rights made available for publication and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the OGF Secretariat.

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.

5. Disclaimer

This document and the information contained herein is provided on an "As Is" basis and the OGF disclaims all warranties, express or implied, including but not limited to any warranty that the use of the information herein will not infringe any rights or any implied warranties of merchantability or fitness for a particular purpose.

6. Full Copyright Notice

Appendix A. Further remarks

The idiomatic implementation as outlined for file transfers between Xspaces in different Vsites acts as a good example for a typical Unicore Job that contains sub Jobs:

1. The **root Unicore Job**
   This Job contains three Tasks, and two Dependencies (a --> b) and (b --> c)
   a. **ImportTask**
   b. **Transfer**
      This Task is actually a sub Unicore Job comprising two Tasks and one Dependency (i --> ii). In this example, the data transferred via UPL as described above.
      i. **DeclarePortfolio**
      ii. **PutPortfolio**
   c. **ExportTask**