caGrid Transfer Service

Shannon Hastings

hastings@bmi.osu.edu

1/17/2007

Contents

Motivation:

Many user groups in caGrid have expressed the need to transfer large data files in the grid without paying the penalty of serialization or deserialization or having to have the entire data in core memory. Early on in the project we created support for utilizing GridFTP for solving these usage scenarios. Several issues with this current approach have left us searching for another solution. Some of the issues are as follows:

- GridFTP server is not cross platform.
- GridFTP requires a globus-C installation.
- GridFTP has to be extended to be able to make authorization callouts to a java based middleware such as caGrid.
- GridFTP installation and configuration is quite advanced for our non power user community.
- Globus does not support SOAP attachments.

Requirements:

In order to better serve our user group we have come up with the following requirements for an alternative non-grid high performance delivery mechanism:

- Cross platform.
- One click/command install with no required configuration.
- Work within the same web application container as caGrid is deployed.
- Utilize GSI sockets for securely transporting the data.
- No deserialization or serialization required on server or client.
- No minimum requirement for core memory.

Architecture:

The architecture of the caGrid Transfer Service is simple yet powerful. It is comprised of the following 4 main components:

1. Transfer Service
2. Transfer Service Helper
3. Transfer WebApp
4. Transfer Client Helper

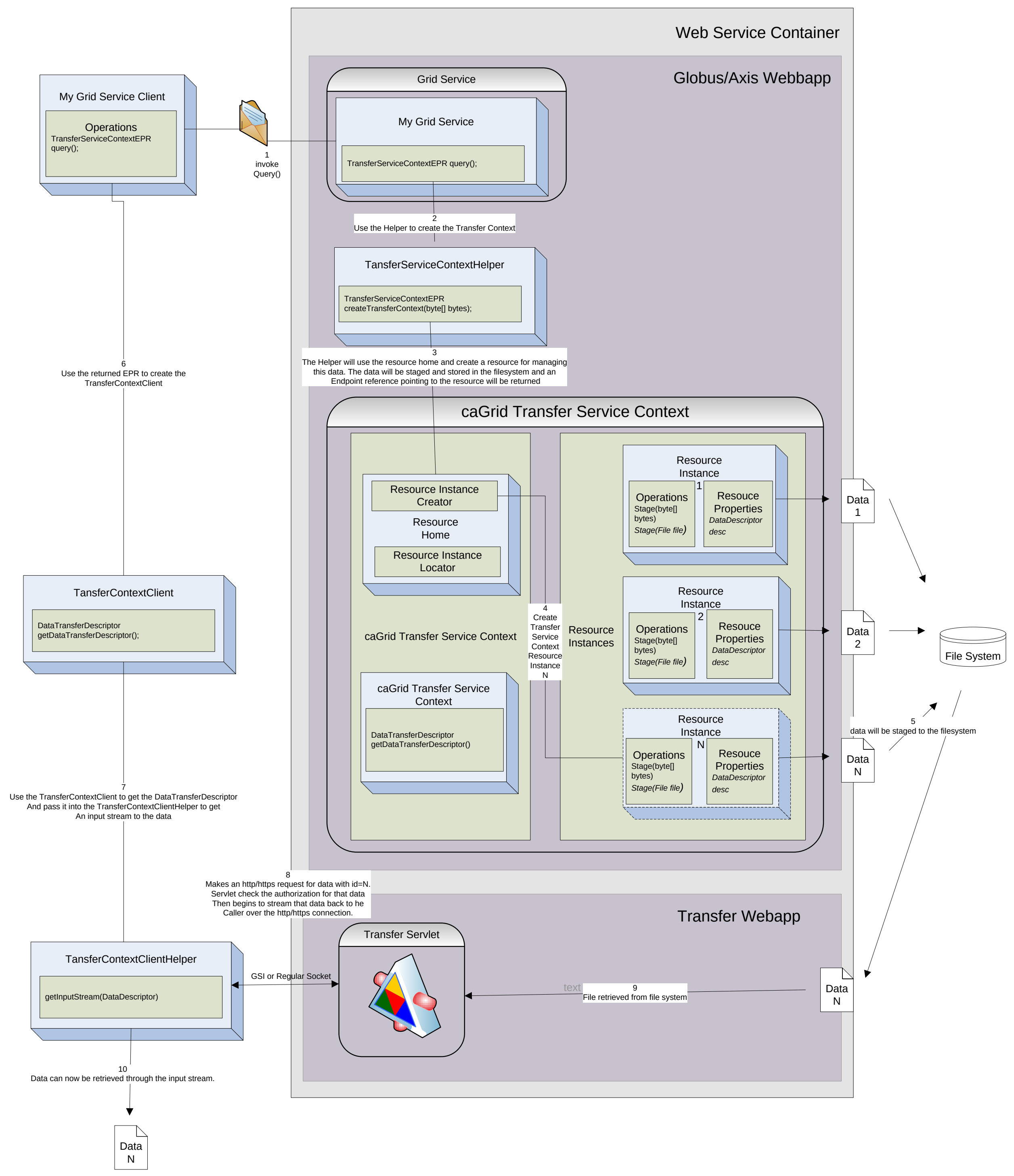
Web Service Container
Globus/Axis Webbapp
Grid Service
My Grid Service
TransferServiceContextEPR query();
My Grid Service Client
Operations
TransferServiceContextEPR query();
1 invoke Query()
2 Use the Helper to create the Transfer Context
TansferServiceContextHelper
TransferServiceContextEPR createTransferContext(byte[] bytes);
3 The Helper will use the resource home and create a resource for managing this data. The data will be staged and stored in the filesystem and an Endpoint reference pointing to the resource will be returned
6 Use the returned EPR to create the TransferContextClient
caGrid Transfer Service Context
Resource Instance Creator
Resource Home
Resource Instance Locator
caGrid Transfer Service Context
caGrid Transfer Service Context
DataTransferDescriptor getDataTransferDescriptor()
4 Create Transfer Service Context Resource Instance N
Resource Instances
Resource Instance 1
Operations Stage(byte[] bytes) Stage(File file)
Resouce Properties DataDescriptor desc
Resource Instance 2
Operations Stage(byte[] bytes) Stage(File file)
Resouce Properties DataDescriptor desc
Resource Instance N
Operations Stage(byte[] bytes) Stage(File file)
Resouce Properties DataDescriptor desc
Data 1
Data 2
Data N
File System
5 data will be staged to the filesystem
TansferContextClient
DataTransferDescriptor getDataTransferDescriptor();
7 Use the TransferContextClient to get the DataTransferDescriptor And pass it into the TransferContextClientHelper to get An input stream to the data
8 Makes an http/https request for data with id=N. Servlet check the authorization for that data Then begins to stream that data back to he Caller over the http/https connection.
Transfer Webapp
Transfer Servlet
TansferContextClientHelper
getInputStream(DataDescriptor)
GSI or Regular Socket
9 File retrieved from file system
text
Data N
10 Data can now be retrieved through the input stream.
Data N

Figure 1: Overall Architecture

Transfer Service:

The transfer service is a WSRF based grid service that is responsible for creating resources which represent the data to be held and transferred. This service utilizes the WS-ResourceFramework in order to create unique stateful resource instances for each data item desired to be transferred.   The user of this service is intended to be another service running in the same container which has a large data item that the service does not wish to transmit over soap in the envelope as an XML element. The user will be able to pass either a pointer to a file or the data itself to the TransferServiceHelper. This Helper will create an Instance of the TransferServiceContext resource, via its ResourceHome, which will be responsible for storing this data and persisting a pointer to this data until the user's client either picks this data up, or it is destroyed. The data will be stored as a file on the file system where it will sit until it is transferred and/or destroyed. The resource is also persistent, meaning it stores the information needed to operate, so that it can survive a container or system restart. This information will be persisted as ResourceProperty on the service of the type DataStorageDescriptor.

```
<xs:complexType name="DataStorageDescriptor">
      <xs:sequence>
            <xs:element type="string" name="location"></xs:element>
            <xs:element type="string" name="userDN"></xs:element>
            <xs:element type="tns:DataDescriptor"
name="descriptor"></xs:element>
</xs:sequence>
```

If this service is deployed into a secure container it will use the caller's identity to protect the data. It will write the callers DN and the location of the file into its persisted data so that this data can be used by the TransferSerlvlet to validate the retriever of the data has the appropriate identification to retrieve the data item.

The service will have a public method called getTransferDataDescriptor() which apon invoking will return a TransferDataDescriptor object to the user. This object will contain the URL required to retrieve this data over http or https.

```
  <xs:complexType name="DataTransferDescriptor">
      <xs:sequence>
            <xs:element type="string" name="url"></xs:element>
            <xs:element type="tns:DataDescriptor"
name="descriptor"></xs:element>
      </xs:sequence>
  </xs:complexType>
```

Transfer Service Helper:

The Transfer Service Helper is an API to be used on the server side that is used to create the TransferServiceContext resource for the data item to be transferred. It will use the ResourceHome of the TransferServiceContext to create an instance of a TransferServiceContextResource which will maintain the user identification and the file location of the data to be transferred. The Transfer Service Helper has several *createTransferContext* methods which can be used to create a Transfer resource. Currently it can take in either a byte array, input stream, or file. Each of these methods will return a TransferServiceContextReference which contains the EPR to the resource. The user will then be able to use this EPR to get the DataTransferDescriptor or the resource to be used for retrieval.

```
<element name="TransferServiceContextReference">
  <complexType>
    <sequence >
      <element ref="wsa:EndpointReference"></element>
    </sequence>
  </complexType>
</element>
```

Transfer Webapp:

The Transfer Webapp is responsible for delivering the data to the consumer over an HTTP or GSI based HTTPS connection.   This is written as a java servlet and deployed as a webapp into the same container that globus has been delplоyed. If the globus is deployed to this container in a non secure mode this container will be using basic http sockets and security is not enforced, i.e. anyone with the URL to the data item will be able to retrieve it. If the container is running securely than having this servlet in a webapp that is deployed to the same container enables this servlet to communicate over the same secure sockets, for example by using the same *connector* in Tomcat, which is used by Axis/Globus. Using this same *connector* enables https connections using the GSI secure sockets to be used to invoke the java servlet. This secure connection will contain the credentials of the caller enabling the Transfer Webapp to compare the caller's identity to the identity of the resource they are requesting. The Transfer Webapp will read the persistence file, the DataDescriptor, for this resource from the Globus container and compare the id attribute with the servlet callers DN. If they match then the caller has the same credentials as the creator of the data item and the data will then be streamed back to the caller. If not the connection will be dropped and the data will remain protected on the server.

Transfer Client Helper:

The Transfer Client Helper is an API to be used on the client for retrieving the data item which was created by the grid service and being held by the servlet waiting to deliver it. If the container is secure than the user will call the:

InputStream getDataStream(DataTransferDescriptor desc, GSSCredential credentials)

operation and else the user will call the:

InputStream getDataStream(DataTransferDescriptor desc).

This call will create the appropriate socket connection to the url provided in the DataTransferDescriptor which points to the data item to be transferred. If this connection is opened properly and the user is athorized the InputStream to the data will be returned. This InputStream will then be able to be read by the user to obtain the data.

User Guide

Transfer Service Deployment

In order to use the caGrid Transfer Service you must deploy it to the grid container. The caGrid Transfer service will currently only run in a Tomcat container. In order to deploy this service to Tomcat make sure that you first have globus deployed to the Tomcat container and security configuration completed if desired (http://www.globus.org/toolkit/docs/4.0/common/javawscore/admin-index.html#javawscore-admin-tomcat-deploying). This can be accomplished by the caGrid Installer or can be done manually by following the instructions provided by Globus.

Once the container is set up and ready you will be able to deploy the caGrid Transfer Service. You will need to acquire the caGrid 1.2 release. Once this is installed you can then deploy the service. This is accomplished by changing directory to the caGrid Transfer project:

From the top of the caGrid release:

```
cd projects/transfer
```

And then deploy to Tomcat:

```
ant deployTomcat
```

Once these steps have been completed the container will need to be restarted.

Utilizing the Transfer Service

In order to make your service utilize the transfer service you will need to add the `caGrid_Transfer` extension to your service when you are creating it. This can be done in the *Advanced* tab of the creation panel in Introduce. There is one API that you will need to familiarize yourself with. This API, `TransferServiceHelper`, is the gateway to staging data in the service that is to be retrieved later by a client. The API is as follows:

```
package org.cagrid.transfer.context.service.helper;

public class TransferServiceHelper {

    public static org.cagrid.transfer.context.stubs.types.TransferServiceContextReference
createTransferContext(File file, DataDescriptor dd) throws RemoteException {

…

    }

    public static org.cagrid.transfer.context.stubs.types.TransferServiceContextReference
createTransferContext(byte[] data, DataDescriptor dd) throws RemoteException {
```

```
…

    }

    public static org.cagrid.transfer.context.stubs.types.TransferServiceContextReference
createTransferContext(InputStream is, DataDescriptor dd) throws RemoteException {

…

    }

    public static org.cagrid.transfer.context.stubs.types.TransferServiceContextReference
createTransferContext(DataDescriptor dd) throws RemoteException {

…

    }

}
```

The first three methods can be used to stage data and return a `TransferServiceContextReference` which contains an `EndPointReference` that can be used on the client to retrieve the data. The last method is used to create a resource which the client can upload the data to and the service can then pick up locally.

Download

On the client side, if a method from your service utilized the `TransferServiceHelper` to stage data locally and then return the type `TransferServiceContextReference` then the `TransferServiceContextClient` and `TransferClientHelper` can be used to retrieve the data on the client. Below is a client example of how this can be executed in an insecure environment:

```
HelloWorldClient client = new HelloWorldClient(args[1]);
//create transfer is a method that staged some data and returned the Reference
TransferServiceContextReference ref = client.createTransfer();
//create a client that enables me to talk to my transfer resource
TransferServiceContextClient tclient = new
TransferServiceContextClient(ref.getEndpointReference());
//use the TransferClientHelper to get an InputStream to the data
InputStream stream = TransferClientHelper.getData(tclient.getDataTransferDescriptor());
```

Again, in a secure environment:

```
HelloWorldClient client = new HelloWorldClient(args[1],GlobusCredential.getDefaultCredential());
//create transfer is a method that staged some data and returned the Reference
TransferServiceContextReference ref = client.createTransfer();
//create a client that enables me to talk to my transfer resource
TransferServiceContextClient tclient = new
TransferServiceContextClient(ref.getEndpointReference(),GlobusCredential.getDefaultCredential());
//use the TransferClientHelper to get an InputStream to the data
InputStream stream =
TransferClientHelper.getData(tclient.getDataTransferDescriptor(),GlobusCredential.getDefaultCrede
ntial());
```

Upload

The caGrid Transfer service can also be used to upload data from client to service. The service, as in the download case, must provide a method which returns an TransferServiceContextReference. Thay can do this by using the TransferServiceHelper API, except, in this case of download they will call the createTransferContext(DataDescriptor ddd) operation. This operation does not take any data in, as the intent is that the data is staged in by something else, however, it does generate the resource and reserve a url where that data can be written and then eventually stored on the local machine.   Below is an example of how the client is to use the TransferServiceContextClient and TransferClientHelper in order to upload data to grid service so that it can be used:

```
HelloWorldClient client = new HelloWorldClient(args[1],GlobusCredential.getDefaultCredential());
//createTransfer on the service creates a resource for me to upload to
TransferServiceContextReference ref = client.createTransfer();
//use the EPR from the reference to create a client to talk to my resource
TransferServiceContextClient tclient = new
TransferServiceContextClient(ref.getEndpointReference());
//use the helper to get the output stream that I can upload my data with
OutputStream stream = TransferClientHelper.putData(tclient.getDataTransferDescriptor());
stream.write(new String("New upload test").getBytes());
//be sure to close the stream when I am done adding the data to it.
stream.close();
//tell the resource that the data has been uploaded.
tclient.staged();
```

Again, in a secure environment:

```
HelloWorldClient client = new HelloWorldClient(args[1],GlobusCredential.getDefaultCredential());
//createTransfer on the service creates a resource for me to upload to
TransferServiceContextReference ref = client.createTransfer();
//use the EPR from the reference to create a client to talk to my resource
TransferServiceContextClient tclient = new
TransferServiceContextClient(ref.getEndpointReference(),GlobusCredential.getDefaultCredential());
//use the helper to get the output stream that I can upload my data with
OutputStream stream =
TransferClientHelper.putData(tclient.getDataTransferDescriptor(),GlobusCredential.getDefaultCrede
ntial());
stream.write(new String("New upload test").getBytes());
//be sure to close the stream when I am done adding the data to it.
stream.close();
//tell the resource that the data has been uploaded.
tclient.staged();
```