GT 4.0 RFT

GT 4.0 RFT

Table of Contents

Chapter 1. GT 4.0 Development Release Notes for RFT

Component Overview

The Reliable Transfer Service (RFT) Service implementation in 4.0 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and deletion of files and directories using GridFTP. The service also provides an interface to control various transfer parameters over GridFTP control channel like TCP buffer size, parallel streams, DCAU etc. The user creates a RFT resource by submitting a Transfer Request (consists of a set of third-party gridftp transfers) to RFT Factory service. The resource is created after the user is properly authorized and authenticated. RFT service implementation exposes operations to control and manage the transfers (the resource). The resource the user created exposes the state of the transfer as a resource property to which the user can either subscribe for changes or poll for the changes in state periodically using standard WS-RF command line clients and other resource properties.

Feature Summary

Features new in release 4.0

- No new features since 3.9.5 release

Supported Features

- Delete files : Delete a set of files/directories on a GridFTP server.
- Exponential Backoff: Configurable exponential back off before a failed transfer is retried
- Transfer All or None: If this option is set and one of the transfers in the request fails RFT will stop transferring the remainder of the request and delete the files that were already transferred successfully.
- Transfer Permissions : File permissions are restored at the destination once the file is transffered successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports MLST command.
- Configurable number of concurrent transfers per container and per request.
- Better Error reporting and Faults.
- Database purge of the request and transfers after life time expiration.
- Cumulative (aggregate) Resource Properties on the factory provide some statistical information.
- One status Resource Property for the entire transfer.
- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- Third-party directory,file transfers and deletes
- Data channel authentication (DCAU)
- NoTPT Option
- Different subject names for source and destination GridFTP servers for authorization mechanism.
- Support for binary/ascii type of transfers
- Configurable number of retries for failed transfers per request.
- Block Size in bytes.

Deprecated Features

- None

Bug Fixes

- Bug 2749[1]
- Bug 2724[2]
- Bug 2683[3]
- Bug 2662[4]
- Bug 2703[5]
- Bug 2847[6]
- Bug 2826[7]
- Bug 2312[8]
- Bug 2879[9]
- Bug 2930[10]
- Bug 2935[11]
- Bug 2852[12]
- Bug 2986[13]
- Bug 3017[14]
- Bug 2984[15]
- Bug 2965[16]
- Bug 2666[17]
- Bug 2927[18]
- Bug 3072[19]
- Bug 2916[20]
- Bug 2721[21]
- Bug 2999[22]
- Bug 3110[23]
- Bug 3091[24]
- Bug 3130[25]
- Bug 2914[26]
- Bug 3115[27]
- Bug 2956[28]

Known Problems

Does not compile with JDK 1.3.1

The configured maximum allowed active transfers constraint is not enforced. For more details please look at the bug report.[29]

Technology Dependencies

RFT depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization

- Delegation Service
- Service Groups
- MDS useful RP

RFT depends on the following 3rd party software:

- PostgreSQL 7.1 version or later. Not tested with 8.0 yet.

Tested Platforms

- Linux

Backward Compatibility Summary

Protocol changes since GT version 3.2

- Added All or None option, maximum attempts, finishBy to transfer request
- Not backwards compatible with OGSI version

API changes since GT version 3.2

- None

Exception changes since GT version 3.2

- None

Schema changes since GT version 3.2

- WSDL changes to work with new Java WS Core

For More Information

Click here[30] for more information about this component.

Notes

1. http://bugzilla.globus.org/globus/show_bug.cgi?id=2749
2. http://bugzilla.globus.org/globus/show_bug.cgi?id=2724
3. http://bugzilla.globus.org/globus/show_bug.cgi?id=2683
4. http://bugzilla.globus.org/globus/show_bug.cgi?id=2662
5. http://bugzilla.globus.org/globus/show_bug.cgi?id=2703
6. http://bugzilla.globus.org/globus/show_bug.cgi?id=2847
7. http://bugzilla.globus.org/globus/show_bug.cgi?id=2826
8. http://bugzilla.globus.org/globus/show_bug.cgi?id=2312
9. http://bugzilla.globus.org/globus/show_bug.cgi?id=2879
10. http://bugzilla.globus.org/globus/show_bug.cgi?id=2930
11. http://bugzilla.globus.org/globus/show_bug.cgi?id=2935
12. http://bugzilla.globus.org/globus/show_bug.cgi?id=2852
13. http://bugzilla.globus.org/globus/show_bug.cgi?id=2986
14. http://bugzilla.globus.org/globus/show_bug.cgi?id=3017
15. http://bugzilla.globus.org/globus/show_bug.cgi?id=2984

16. http://bugzilla.globus.org/globus/show_bug.cgi?id=2965
17. http://bugzilla.globus.org/globus/show_bug.cgi?id=2666
18. http://bugzilla.globus.org/globus/show_bug.cgi?id=2927
19. http://bugzilla.globus.org/globus/show_bug.cgi?id=3072
20. http://bugzilla.globus.org/globus/show_bug.cgi?id=2916
21. http://bugzilla.globus.org/globus/show_bug.cgi?id=2721
22. http://bugzilla.globus.org/globus/show_bug.cgi?id=2999
23. http://bugzilla.globus.org/globus/show_bug.cgi?id=3110
24. http://bugzilla.globus.org/globus/show_bug.cgi?id=3091
25. http://bugzilla.globus.org/globus/show_bug.cgi?id=3130
26. http://bugzilla.globus.org/globus/show_bug.cgi?id=2914
27. http://bugzilla.globus.org/globus/show_bug.cgi?id=3115
28. http://bugzilla.globus.org/globus/show_bug.cgi?id=2956
29. http://bugzilla.globus.org/globus/show_bug.cgi?id=3121
30. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/index.html

Chapter 2. GT 4.0 RFT : System Administrator's Guide

Introduction

This guide contains advanced configuration information for system administrators working with RFT. It provides references to information on procedures typically performed by system administrators, including installation, configuring, deploying, and testing the installation.

This information is in addition to the basic installation instructions in the GT 4.0 System Administrator's Guide[1].

RFT is used to perform third-party transfers across GridFTP servers. It uses a database to store its state periodically so the transfers can be recovered from any failures. RFT uses standard grid security mechanisms for authorization and authentication of the users. In order to effectively use RFT you should have installed and configured a database with RFT database schemas and have the necessary security infrastructure in place to perform a 3rd party transfer.

Building and Installing

RFT is built and installed as part of a default GT 4.0 installation. For basic installation instructions, see the GT 4.0 System Administrator's Guide[2]. No extra installation steps are required for this component.

The following are specialized instructions for advanced developers who want to deploy latest code from CVS:

Table 2-1. Build RFT from CVS:

1	Configure your CVSROOT to point to globus cvs location.
2	Run: `cvs co ws-transfer`
3	Run: `cd ws-transfer/reliable`
4	Set GLOBUS_LOCATION to point to your globus installation.
5	Run: `ant deploy`

Configuring

Required configuration: configuring the PostgreSQL database

PostgreSQL (Version 7.1 or greater) needs to be installed and configured for RFT to work. You can either use the packages which came with your operating system (RPMs, DEBs, ...) or build from source. We used PostgreSQL version 7.3.2 for our testing and the following instructions are good for the same.

1. Install Postgresql. Instructions on how to install/configure postgresql can be found here[3].
2. Configure the postmaster daemon so that it accepts TCP connections. This can be done by adding -o "-i" switch to postmaster script (This can be init.d script found in /etc/init.d/postgresql or /var/lib/ depending on how you

installed postgresql). Follow the intstructions here [4] to start the postmaster with -i option.

3. Now you need to set security on the database you are about to create. You can do it by following the steps below:

   `sudo vi /var/lib/pgsql/data/pg_hba.conf` and append the following line to the file:

   ```
   host rftDatabase "username" "host-ip" 255.255.255.255 trust
   ```

   ```
   sudo /etc/init.d/postgresql restart
   ```

4. You will now need to create a postgresql user that would connect to the database. This is usually the account under which the container is running. You can create a postgresql user by running the following command: `su postgres createuser globus` . If you get the following error : `psql: could not connect to server: No such file or directory Is the server running locally and accepting connections on Unix domain socket "/tmp/.s.PGSQL.5432"?` This generally means that 1. either your postmaster is not started with -i option or 2. you did'nt restart the postmaster after above mentioned step
5. To create the database that is used for RFT, run (as user globus): `createdb rftDatabase`
6. To populate the RFT database with appropriate schemas, run: `psql -d rftDatabase -f $GLOBUS_LOCATION/share/globus_wsrf_rft/rft_schema.sql` Now that you have created a database to store RFT's state, the following steps configure RFT to find the database:
7. Open `$GLOBUS_LOCATION/etc/globus_wsrf_rft/jndi-config.xml`
8. Find the `dbConfiguration` section under `ReliableFileTransferService <service>` section.
9. Change the `connectionString` to point to the machine on which you installed Postgres and name of the database you used in step 2. If you installed Postgres on the same machine as your Globus install, the default should work fine for you.
10. Change the `userName` to the name of the user who owns/created the database and do the same for the password. (It also depends on how you configured your database.)
11. Don't worry about the other parameters in that section. The defaults should work fine for now.
12. Edit the configuration section under `ReliableFileTransferService`. There are two values that can be edited in this section.
13.
    - `backOff` : Time in seconds you want RFT to backoff before a failed transfer is retried by RFT. Default should work fine for now.
    - `maxActiveAllowed`: This is the number of transfers the container can do at given point. Default should be fine for now.

Deploying

[information about deploying the component into various containers/environments]

Testing

You need to checkout the tests from CVS because RFT tests are not included in the installer. Please follow these steps to run RFT unit tests:

Table 2-2. RFT Testing

1	set `$GLOBUS_LOCATION` to point to your Globus install.
2	Start a gridftp server on the machine you are running the tests on default port. This can be done by running: `$GLOBUS_LOCATION/sbin/globus-gridftp-server -p 2811 &`
3	Start the container with RFT deployed in it.
4	Edit $GLOBUS_LOCATION/globus_wsrf_rft_test/test.properties. Put in appropriate values for properties like authzValue(self or host), HOST - host ip of container, PORT- port on which container is listening, sourceHost,destinationsHost - hostnames of gridftp servers. The default values will work fine if you are running the tests with a standard stand-alone container.
5	The *.xfr files in $GLOBUS_LOCATION/share/globus_wsrf_rft_test/ are the transfer files that will be used in the tests. Again the default values work fine if you followed the instructions so far.
6	Run the following command which will run all the rft unit tests: `ant -Dtests.jar=$GLOBUS_LOCATION/lib/globus_wsrf_rf -f share/globus_wsrf_rft_test/runtests.xml`
7	Run the following command to generate the test-reports in html form: `ant -f share/globus_wsrf_rft_test/runtests.xml generateTestReport`

Security Considerations

Permissions of service configuration files

The service configuration files such as `jndi-config.xml` or `server-config.wsdd` (located under `etc/<gar>/` directory) contains private information such as database passwords and username. Ensure that these configuration files are only readable by the user that is running the container.

The deployment process automatically sets the permissions of `jndi-config.xml` and `server-config.wsdd` files as user readable only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

Access of information stored in the database

RFT stores the transfer request in a database. Proper security measures need to be taken to protect the access of the data by granting/revoking appropriate permissions on tables that are created for RFT use and other steps that are appropriate and consistent with site specific security measures.

Permissions of persistent data

RFT uses subscription persistence API from GT4 core to store all of its subscription data under the `~/.globus/pe rsisted` directory. Ensure that the entire `~/.globus/persisted` directory is only readable by the user running the container.

Troubleshooting

Problem: If RFT is not configured properly to talk to a PostgreSQL database, you will see this message displayed on the console when you start the container :

```
"Error creating RFT Home: Failed to connect to database ...
Until this is corrected all RFT request will fail and all GRAM jobs that re-
quire staging will fail".
```

Solution: Usual mistake is Postmaster is not accepting TCP connections which means that you must restart Postmaster with -i option (see the Section called *Required configuration: configuring the PostgreSQL database*).

Problem: Make RFT print more verbose error messages:

Solution: Edit $GLOBUS_LOCATION/container-log4j.properties and add following line to it: `log4j.category.org.globus.transfer=DEBUG` . For more verbosity add `log4j.category.org.globus.ftp=DEBUG` which will print out Gridftp messages too.

Usage statistics collection by the Globus Alliance

The following usage statistics are sent by default in a UDP packet at the end of life time of each RFT Resource (or when a RFT resource is destroyed).

- Total number of files transferred by RFT since RFT is installed
- Total number of bytes transferred by RFT since RFT is installed
- Total number of files transferred in this RFT Resource
- Total number of bytes transferred in this RFT Resource
- Creation time of this RFT Resource
- Factory Start Time

We have made a concerted effort to collect only data that is not too intrusive or private, and yet still provides us with information that will help improve the GRAM component. Nevertheless, if you wish to disable this feature, please see the Java WS Core System Adminstrator guide section on Usage Statistics Configuration[5] for instructions.

Also, please see our policy statement [6] on the collection of usage statistics.

Notes

1. ../../../admin/docbook/
2. ../../../admin/docbook/
3. http://www.postgresql.org/docs/manuals/

4. http://www.postgresql.org/docs/7.4/static/postmaster-start.html
5. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTarget
6. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/Usage_Stats.html

Chapter 3. GT 4.0 RFT : User's Guide

Introduction

RFT Service implementation in 3.9.5 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and to delete files using GridFTP. The user creates a RFT resource by submitting a list of URL pairs of files that need to be transferred/deleted to RFT Factory service. The user also specifies the time to live for the resource the user is creating to a GT 3.9.5 Container in which RFT is deployed and configured. The resource is created after the user is properly authorized and authenticated. RFT service implementation exposes operations to control and manage the transfers (the resource). The operations exposed by both RFT factory and RFT service are briefly described below. The resource the user created also exposes the state of the transfer as a resource property to which the user can either subscribe for changes or poll for the changes in state periodically using standard command line clients.

Command-line tools

Reference I, *Commandline Reference*

Graphical user interfaces

There is no GUI for RFT service in this release.

Troubleshooting

- Always have a valid proxy before using command line RFT clients.
- Make sure to provide suitable options to the client especially for the Termination time so that the resource does not get destroyed before finishing the transfers.

Usage statistics collection by the Globus Alliance

The following usage statistics are sent by default in a UDP packet at the end of life time of each RFT Resource (or when a RFT resource is destroyed).

- Total number of files transferred by RFT since RFT is installed
- Total number of bytes transferred by RFT since RFT is installed
- Total number of files transferred in this RFT Resource
- Total number of bytes transferred in this RFT Resource
- Creation time of this RFT Resource
- Factory Start Time

We have made a concerted effort to collect only data that is not too intrusive or private, and yet still provides us with information that will help improve the GRAM component. Nevertheless, if you wish to disable this feature, please see the Java WS Core System Adminstrator guide section on Usage Statistics Configuration[1] for instructions.

Also, please see our policy statement [2] on the collection of usage statistics.

Notes

1. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/common/javawscore/admin-index.html#s-javawscore-Interface_Config_Frag-usageStatisticsTarget
2. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/Usage_Stats.html

Chapter 4. GT 4.0 RFT : Developer's Guide

Introduction

RFT Service implementation in GT 4.0 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and to delete files using GridFTP. The user creates a RFT resource by submitting a list of URL pairs of files that need to be transferred/deleted to RFT Factory service. The user also specifies the time to live for the resource the user is creating to a GT 4.0 container in which RFT is deployed and configured. The resource is created after the user is properly authorized and authenticated. RFT service implementation exposes operations to control and manage the transfers (the resource). The operations exposed by both RFT factory and RFT service are briefly described below. The resource the user created also exposes the state of the transfer as a resource property to which the user can either subscribe for changes or poll for the changes in state periodically using standard command line clients.

Architecture and design overview

A design doc can be found here[1].

Public Interface

The semantics and syntax of the APIs and WSDL for the component, along with descriptions of domain-specific structured interface data, can be found in the public interface guide[2].

Usage scenarios

- Scenario 1: You have a large dataset and you want to make sure all the files are transferred. RFT is used to reliably transfer the data.
- Scenario 2: RFT is also used to perform staging operations in GRAM and to remove files and directories on a gridftp server.

Tutorials

There are no tutorials available at this point.

Feature summary

Features new in release 4.0

- No new features since 3.9.5 release

Supported Features

- Delete files : Delete a set of files/directories on a GridFTP server.
- Exponential Backoff: Configurable exponential back off before a failed transfer is retried
- Transfer All or None: If this option is set and one of the transfers in the request fails RFT will stop transferring the remainder of the request and delete the files that were already transferred successfully.
- Transfer Permissions : File permissions are restored at the destination once the file is transffered successfully. This can be configured to throw a fatal error or

a transient error depending on whether the GridFTP server supports MLST command.

- Configurable number of concurrent transfers per container and per request.
- Better Error reporting and Faults.
- Database purge of the request and transfers after life time expiration.
- Cumulative (aggregate) Resource Properties on the factory provide some statistical information.
- One status Resource Property for the entire transfer.
- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- Third-party directory,file transfers and deletes
- Data channel authentication (DCAU)
- NoTPT Option
- Different subject names for source and destination GridFTP servers for authorization mechanism.
- Support for binary/ascii type of transfers
- Configurable number of retries for failed transfers per request.
- Block Size in bytes.

Deprecated Features

- None

Tested platforms

Tested platforms for RFT:

- Linux

- Fedora Core 1 i686
- Fedora Core 3 i686
- RedHat 7.3 i686
- RedHat 9 x86
- Debian Sarge x86
- Debian 3.1 i686

Backward compatibility summary

Protocol changes since GT version 3.2

- Added All or None option, maximum attempts, finishBy to transfer request
- Not backwards compatible with OGSI version

API changes since GT version 3.2

- None

Exception changes since GT version 3.2

- None

Schema changes since GT version 3.2

- WSDL changes to work with new Java WS Core

Technology dependencies

RFT depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization
- Delegation Service
- Service Groups
- MDS useful RP

RFT depends on the following 3rd party software:

- PostgreSQL 7.1 version or later. Not tested with 8.0 yet.

Security considerations

Permissions of service configuration files

The service configuration files such as `jndi-config.xml` or `server-config.wsdd` (located under `etc/<gar>/` directory) contains private information such as database passwords and username. Ensure that these configuration files are only readable by the user that is running the container.

The deployment process automatically sets the permissions of `jndi-config.xml` and `server-config.wsdd` files as user readable only. However, this might not work correctly on all platforms and this does not apply to any other configuration files.

Access of information stored in the database

RFT stores the transfer request in a database. Proper security measures need to be taken to protect the access of the data by granting/revoking appropriate permissions on tables that are created for RFT use and other steps that are appropriate and consistent with site specific security measures.

Permissions of persistent data

RFT uses subscription persistence API from GT4 core to store all of its subscription data under the `~/.globus/pe rsisted` directory. Ensure that the entire `~/.globus/persisted` directory is only readable by the user running the container.

Debugging

A standard way to debug RFT is to make container print out more verbose error messages. You can do that by doing the following steps:

Edit $GLOBUS_LOCATION/container-log4j.properties and add following line to it: `log4j.category.org.globus.transfer=DEBUG` . For more verbosity add `log4j.category.org.globus.ftp=DEBUG` which will print out Gridftp messages too.

Troubleshooting

Database configuration is the most complicated and important part of RFT setup. You can find more instructions on troubleshooting at the Section called *Troubleshooting* in Chapter 2.

Related Documentation

- Lessons learned producing an OGSI compliant Reliable File Transfer Service (pdf)[3]
- Reliable Data Transport: A Critical Service for the Grid (pdf)[4]

Notes

1. Protocol_overview.doc
2. RFT_Public_Interfaces.html
3. http://www-unix.mcs.anl.gov/%7Ekeahey/DBGS/DBGS_files/dbgs_papers/allcock.pdf
4. http://www.doc.ic.ac.uk/%7Esjn5/GGF/GGF11/BGBS-Allcock.pdf

Chapter 5. GT 4.0 Component Fact Sheet: Reliable File Transfer (RFT)

Brief component overview

The Reliable Transfer Service (RFT) Service implementation in 4.0 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and deletion of files and directories using GridFTP. The service also provides an interface to control various transfer parameters over GridFTP control channel like TCP buffer size, parallel streams, DCAU etc. The user creates a RFT resource by submitting a Transfer Request (consists of a set of third-party gridftp transfers) to RFT Factory service. The resource is created after the user is properly authorized and authenticated. RFT service implementation exposes operations to control and manage the transfers (the resource). The resource the user created exposes the state of the transfer as a resource property to which the user can either subscribe for changes or poll for the changes in state periodically using standard WS-RF command line clients and other resource properties.

Summary of features

Features new in release 4.0

- No new features since 3.9.5 release

Supported Features

- Delete files : Delete a set of files/directories on a GridFTP server.
- Exponential Backoff: Configurable exponential back off before a failed transfer is retried
- Transfer All or None: If this option is set and one of the transfers in the request fails RFT will stop transferring the remainder of the request and delete the files that were already transferred successfully.
- Transfer Permissions : File permissions are restored at the destination once the file is transffered successfully. This can be configured to throw a fatal error or a transient error depending on whether the GridFTP server supports MLST command.
- Configurable number of concurrent transfers per container and per request.
- Better Error reporting and Faults.
- Database purge of the request and transfers after life time expiration.
- Cumulative (aggregate) Resource Properties on the factory provide some statistical information.
- One status Resource Property for the entire transfer.
- Recursive directory transfers and deletes.
- Parallel streams
- TCP Buffer Size
- Third-party directory,file transfers and deletes
- Data channel authentication (DCAU)
- NoTPT Option
- Different subject names for source and destination GridFTP servers for authorization mechanism.
- Support for binary/ascii type of transfers
- Configurable number of retries for failed transfers per request.
- Block Size in bytes.

Deprecated Features

- None

Usability summary

Usability improvements for RFT:

- The command-line RFT clients print out more detailed and accurate fault information for all commonly occuring errors.

Backward compatibility summary

Protocol changes since GT version 3.2

- Added All or None option, maximum attempts, finishBy to transfer request
- Not backwards compatible with OGSI version

API changes since GT version 3.2

- None

Exception changes since GT version 3.2

- None

Schema changes since GT version 3.2

- WSDL changes to work with new Java WS Core

Technology dependencies

RFT depends on the following GT components:

- Java WS Core
- WS Authentication and Authorization
- Delegation Service
- Service Groups
- MDS useful RP

RFT depends on the following 3rd party software:

- PostgreSQL 7.1 version or later. Not tested with 8.0 yet.

Tested platforms

Tested platforms for RFT:

- Linux

- Fedora Core 1 i686
- Fedora Core 3 i686
- RedHat 7.3 i686
- RedHat 9 x86

- Debian Sarge x86
- Debian 3.1 i686

Associated standards

Associated standards for RFT:

- WS-RF
- WS-Addressing
- WS-Security

For More Information

Click here[1] for more information about this component.

Notes

1. index.html

Chapter 6. GT 4.0 RFT Public Interface Guide

Semantics and syntax of APIs

Programming Model Overview

The Reliable Transfer Service (RFT) is a WSRF based service that provides interfaces for controlling and monitoring third party file transfers using GridFTP servers. The client controlling the transfers (in this case RFT) is hosted inside of a Grid service so it can be managed using the soft state model. It is essentially a reliable and recoverable version of the GT2 `globus-url-copy` tool and more. In 3.9.5 RFT can also perform file deletion, recursive directory deletion operations. It is also used by GRAM to perform all the staging operations and cleanup operations.

Component API

Some relevant API :

- Service API[1]
- Common API[2]
- Client API[3]

Semantics and syntax of the WSDL

Protocol overview

RFT Service implementation in 3.9.5 uses standard SOAP messages over HTTP to submit and manage a set of 3rd party GridFTP transfers and to delete files using GridFTP. The user creates a RFT resource by submitting a list of URL pairs of files that need to be transferred/deleted to RFT Factory service. The user also specifies the time to live for the resource the user is creating to a GT 3.9.5 Container in which RFT is deployed and configured. The resource is created after the user is properly authorized and authenticated. RFT service implementation exposes operations to control and manage the transfers (the resource). The operations exposed by both RFT factory and RFT service are briefly described below. The resource the user created also exposes the state of the transfer as a resource property to which the user can either subscribe for changes or poll for the changes in state periodically using standard command line clients.

Operations

Please find below operations of both RFT Factory and RFT Service Implementation.

RFT Factory Service

Used to create a Reliable File Transfer resource. The operations exposed by the factory are as follows:

- `createReliableFileTransfer`: Creates a Reliable File Transfer Resource.
  - Input Parameters: Initial Termination time , Transfer Request or Delete Request

- Output parameters: Termination time, Current time, Endpoint reference of the Resource created. (This should be stored by the user as it is needed to query the status of the resource and to perform any further operations on the resource.
- Fault: createReliableFileTransferFault:

RFT Service

Used to manage the Resource created using the RFT Factory Service. The operations exposed by the service are as follows:

- `start`: Starts executing the transfers/deletes
  - Input Parameters: None
  - Output Parameters: None
  - Fault: RepeatedlyStartedFault:
- `getStatus`: To get the status of a particular file.
  - Input Parameters: A source URL of the file that is part of the request.
  - Output Parameters: `Transfer Status Type`
  - Fault: RFTDatabaseFault
- `getStatusSet` : To get the status of a set of files in a request
  - Input Parameters: int from (the relative position of the transfer in the request) and int offset (Number of files queried)
  - Output Parameters: An array of `TransferStatusType`
  - Fault: RFTDatabaseFault
- cancel: To cancel a transfer that is part of a resource.
  - Input Parameters : int from (the relative position of the transfer in the request) int to
  - Output Parameters: None
  - Fault: RFTDatabaseFault

Resource Properties:

Resource Properties of RFT Factory (which acts both as a resource and a service at the same time) and RFT Resource are found below:

RFT Factory Service

- `ActiveResourceInstances`: A dynamic resource property of total number of active rft resources in the container at a given point of time.
- `TotalNumberOfTransfers`: A dynamic resource property of total number of transfers/deletes performed since the RFT service is deployed in this container
- `TotalNumberOfActiveTransfers`: A dynamic resource property of number of active transfers across all rft resources in a container at a given point of time.
- `TotalNumberOfBytesTransferred`: A dynamic resource property of total number of bytes transferred by all RFT resources created since the deployment of the service.

- `RFTFactoryStartTime`: Time when the service was deployed in the container. Used to calculate uptime.
- `DelegationServiceEPR`: The end point reference of the Delegation resource that holds the delegated credential used in executing the resource.

RFT Service

- `OverallStatus`: This is a complex type providing overall status of a RFT resource by providing number of transfers pending, active, finished, retrying, failed, cancelled. Each of those values can be obtained by invoking getTransfers(Finished/Active/Failed/Restarted/Pending/Cancelled) on OverallStatus Resource Property.Note that this Resource Property gets updated every time one of the transfers changes state, so there can be and will be more than one update, in the life time of a RFT resource, if you subscribe to this RP. This Resource Property also includes the last fault (if thrown) from a transfer and can be accessed by invoking getFault on OverallStatus. This will indicate why a transfer has failed.
- `RequestStatus`: This is complex type resource property providing status of a RFT resource in form of Pending/Active/Done/Failed. The status can be obtained from RequestStatusType by invoking getRequestStatus(). This will result in one of four status strings(Pending/Active/Done/Failed/Cancelled). This RP also contains a fault that denotes the last fault in a RFT resource and can be accessed by invoking getFault(). If a client is subscribed to this RP there will be only be 2 updates in the life time of a rft resource (Pending->Active->Done, Pending->Active->Failed, Pending->Active->Cancelled, Pending->Cancelled).
- `TotalBytes`: provides the total number of bytes transferred by the resource
- `TotalTime`: provides the total time taken to transfer the above mentioned total bytes.

Faults

Faults from RFT Factory Service and RFT service can be found below:

RFT Factory Service

- `createReliableFileTransferFault`: All the errors encountered during the creation of the RFT resource are mapped to this fault. Any security related errors are caught even before the factory and are thrown to the user/client.

RFT Service

- `RepeatedlyStartedFault`: This is raised if a client calls start more than once on a resource.
- `RFTDatabaseFault`: Thrown when the service is unable to find the resource the user/client is querying for.

WSDL and Schema Definition

- Reliable Transfer Factory Port Type[4]
- Reliable Transfer Port Type[5]

You can find links to all the RFT schemas here.[6]

Command-line tools

Reference I, *Commandline Reference*

Overview of Graphical User Interface

There is no GUI for RFT service in this release.

Semantics and syntax of domain-specific interface

Please look here[7] for information on how RFT schemas look like.

Configuration interface

Configuration overview

RFT has the following prerequisites:

- Java WS Core[8] - this is built and installed in a default GT 4.0 installation[9].
- a host certificate (see Required Configuration[10].)
- GridFTP[11] Server - GridFTP perfoms the actual file transfer and is built and installed in a default GT 4.0 installation[12].
- PostgreSQL - PostgreSQL is used to store the state of the transfer to allow for restart after failures. The interface to PostgreSQL is JDBC so any DBMS that supports JDBC can be used, although no other has been tested. For instructions on configuring the PostgreSQL database for RFT, click here[13].

RFT can be registered to an MDS index service to facilitiate monitoring and discovery. The MDS documentation contains a note on registering RFT to an Index Service[14].

Syntax of the interface

The security of the service can be configured by modifying the security descriptor[15]. It allows for configuring the credentials that will be used by the service, type of authentication and authorization that needs to be enforced. By default, the following security configuration is installed:

- Credentials set for use by container is used. If that is not specified, default credentials are used.
- GSI Secure conversation authentication is enforced for all methods.

Note: Changing required authentication and authorization method will require suitable changes to the clients that contact this service.

To alter security descriptor configuration, refer to Security Descriptors[16]. The file to be altered is `$GLOBUS_LOCATION/etc/globus_wsrf_rft/security-config.xml`

Environment variable interface

The only Env variable that needs to be set for RFT is GLOBUS_LOCATION in order to run the command line clients, which should be set to globus installation.

Notes

1. http://www-unix.globus.org/api/javadoc-3.9.5/globus_wsrf_rft_service_java/
2. http://www-unix.globus.org/api/javadoc-3.9.5/globus_wsrf_rft_common_java/
3. http://www-unix.globus.org/api/javadoc-3.9.5/globus_wsrf_rft_client_java/
4. http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/reliable_transfer_factory_port_type.wsdl?re type=text/vnd.viewcvs-markup
5. http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/reliable_transfer_port_type.wsdl?rev=1.14& type=text/vnd.viewcvs-markup
6. http://viewcvs.globus.org/viewcvs.cgi/ws-transfer/reliable/common/schema/transfer/reliable/
7. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/rft_job_description.html
8. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/common/javawscore/
9. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/admin/docbook/
10. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/admin/#requiredconfig
11. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/gridftp/
12. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/admin/docbook/
13. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/admin/#postgresql
14. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/info/index/admin/registering-rft.html
15. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/security/authzframe/security_descriptor.html
16. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/security/authzframe/security_descriptor.html

Chapter 7. GT 4.0 RFT: Quality Profile

Test coverage reports

Not available right now.

Code analysis reports

Not available right now.

You can find list of outstanding bugs in RFT by following this: link[1]

Bug Fixes

- Bug 2749[2]
- Bug 2724[3]
- Bug 2683[4]
- Bug 2662[5]
- Bug 2703[6]
- Bug 2847[7]
- Bug 2826[8]
- Bug 2312[9]
- Bug 2879[10]
- Bug 2930[11]
- Bug 2935[12]
- Bug 2852[13]
- Bug 2986[14]
- Bug 3017[15]
- Bug 2984[16]
- Bug 2965[17]
- Bug 2666[18]
- Bug 2927[19]
- Bug 3072[20]
- Bug 2916[21]
- Bug 2721[22]
- Bug 2999[23]
- Bug 3110[24]
- Bug 3091[25]
- Bug 3130[26]
- Bug 2914[27]
- Bug 3115[28]
- Bug 2956[29]

Performance reports

A recent performance report can be found here[30].

Notes

1. http://bugzilla.globus.org/globus/buglist.cgi?bug_status=NEW&bug_status=ASSIGNED&bug_s
2. http://bugzilla.globus.org/globus/show_bug.cgi?id=2749
3. http://bugzilla.globus.org/globus/show_bug.cgi?id=2724
4. http://bugzilla.globus.org/globus/show_bug.cgi?id=2683
5. http://bugzilla.globus.org/globus/show_bug.cgi?id=2662
6. http://bugzilla.globus.org/globus/show_bug.cgi?id=2703
7. http://bugzilla.globus.org/globus/show_bug.cgi?id=2847
8. http://bugzilla.globus.org/globus/show_bug.cgi?id=2826
9. http://bugzilla.globus.org/globus/show_bug.cgi?id=2312
10. http://bugzilla.globus.org/globus/show_bug.cgi?id=2879
11. http://bugzilla.globus.org/globus/show_bug.cgi?id=2930
12. http://bugzilla.globus.org/globus/show_bug.cgi?id=2935
13. http://bugzilla.globus.org/globus/show_bug.cgi?id=2852
14. http://bugzilla.globus.org/globus/show_bug.cgi?id=2986
15. http://bugzilla.globus.org/globus/show_bug.cgi?id=3017
16. http://bugzilla.globus.org/globus/show_bug.cgi?id=2984
17. http://bugzilla.globus.org/globus/show_bug.cgi?id=2965
18. http://bugzilla.globus.org/globus/show_bug.cgi?id=2666
19. http://bugzilla.globus.org/globus/show_bug.cgi?id=2927
20. http://bugzilla.globus.org/globus/show_bug.cgi?id=3072
21. http://bugzilla.globus.org/globus/show_bug.cgi?id=2916
22. http://bugzilla.globus.org/globus/show_bug.cgi?id=2721
23. http://bugzilla.globus.org/globus/show_bug.cgi?id=2999
24. http://bugzilla.globus.org/globus/show_bug.cgi?id=3110
25. http://bugzilla.globus.org/globus/show_bug.cgi?id=3091
26. http://bugzilla.globus.org/globus/show_bug.cgi?id=3130
27. http://bugzilla.globus.org/globus/show_bug.cgi?id=2914
28. http://bugzilla.globus.org/globus/show_bug.cgi?id=3115
29. http://bugzilla.globus.org/globus/show_bug.cgi?id=2956
30. rft_scalability_3_9_4.doc

Chapter 8. GT 4.0 Migrating Guide for RFT

The following provides available information about migrating from previous versions of the Globus Toolkit.

Migrating from GT2

This does not apply to RFT.

Migrating from GT3

The RFT implementation in GT4 and GT3 are not interoperable as they are built on different GT core implementations. In order to migrate to GT4 RFT, you should follow the installation instructions of GT4 RFT which can be found here[1].

Notes

1. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/admin/docbook/

I. Commandline Reference

rft

Name

`rft` — Submit and monitor a 3rd party GridFTP transfer.

Synopsis

rft

Tool description

Submits a transfer to the Reliable File Transfer Service and prints out the status of the transfer on the console.

Command syntax and options

```
rft [-h <host-ip of the container defaults to localhost>
-r <port, defaults to 8080>
-l <lifetime for the resource default 60mins>
-m <security mechanism. 'msg' for secure message or 'conv' for
 secure conversation and 'trans' for transport. Defaults to
   secure transport.>
-p <protection type, 'sig' signature and 'enc' encryption,
 defaults to signature >
-z <authorization mechanism can be self or host. default self>
-file <file to write EPR of created Reliable  File Transfer Resource]>
-f <path to the file that contains list of transfers>
```

This is a sample transfer file that the command-line client will be able to parse. It can also be found in **$GLOBUS_LOCATION/share/globus_wsrf_rft_client/** along with other samples for directory transfers and deletes (lines starting with # are comments):

```
This option when it is set to true means to perform transferr in bi-
nary
form, if it is set to false transfer is done in ASCII. Default is binary.
true
#Block size in bytes that is transferred. Default is 16000 bytes.
16000
#TCP Buffer size in bytes
#Specifies the size (in bytes) of the TCP buffer to be used by the un-
derlying
ftp data channels. This is critical to good performance over the WAN.  Use the
bandwidth-delay  product as your buffer size.

16000

#Notpt (No thirdPartyTransfer): turns third-party transfers off is this op-
tion
is set to false (on if set to true).
Site firewall and/or software configuration may prevent a connection
between the two servers (a third party transfer).  If this is  the case,
RFT will "relay" the data.  It will do a  GET from the source and a PUT to
the destination. This obviously causes a performance penalty, but will al-
low
you to  complete a transfer you otherwise could not do.

false

#Number of parallel streams: Specifies the number of parallel data con-
nections
that should be used.
```

```
1

#Data Channel Authentication (DCAU): Turns off data channel authen-
tication for
FTP transfers  is set to false.(the default is true to authenticate the data
channel).
true
# Concurrency of the request: Number of files that you want to trans-
fer at any
given point. Default is set to one.
1
#Grid Subject name of the source gridftp server. This is used for Authorization
purposes. If the source gridftp server is running with host creden-
tials you can specify "null" here. By default host authorization is performed
/DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710
#Grid Subject name of the destination gridftp server. This is used for Au-
thorization purposes. If the destination gridftp server is running with host
credentials you can specify "null" here. By default Host authoriza-
tion is done.
/DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710
#Transfer all or none of the transfers: This option if set to true will make RFT
to clean up ( delete ) all the transfers that have been done already if one of the t
fers fails. Used in GRAM when staging.

false
#Maximum number of retries: This is number of times RFT retries a trans-
fer failed with a non-fatal error.

10

#Source/Dest URL Pairs: gsiftp urls of source followed by destina-
tion.
If directory is to be recursively transferred the source gsiftp url and
destination gsiftp url should end with "/". Currently RFT supports Di-
rectory -
Directory, File - Directory, File - File transfers. There can be more URL pairs
and all of them use the same options as above for performing the trans-
fer.

gsiftp://localhost:5678/tmp/rftTest.tmp
gsiftp://localhost:5678/tmp/rftTest_Done.tmp
```

Limitations

This command line client is very. simple and does not do any intelligent parsing of various command line options and the options in the sample transfer file. It works fine if used in the way documented here. For more information on all these options please refer to documentation of globus-url-copy here[1]. Also please note that maximum number of transfers command-line client can process before running out of memory is ~21K with default JVM heap size which is 64M in our tests. Please look at Performance reports[2] for more details

Notes

1. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/gridftp/user/#commandline
2. http://www-unix.globus.org/toolkit/docs/development/4.0-drafts/data/rft/rft_scalability_3_9_4.doc

rft-delete

Name

`rft-delete` — Command-line client to delete files using RFT.

Synopsis

rft-delete

Tool description

This command-line tool is used to submit a list of files to be deleted.

Command and options

```
rft-delete [-h <host-ip of the container default localhost>
-r <port, defaults to 8080>
-l <lifetime for the resource default 60mins>
-m <security mechanism. 'msg' for secure message or 'conv' for
 secure conversation and 'trans' for transport. Defaults to
   secure transport.>
-p <protection type, 'sig' signature and 'enc' encryption,
 defaults to signature >
-z <authorization mechanism can be self or host. default self>
-file <file to write EPR of created Reliable  File Transfer Resource]>
-f <path to the file that contains list of transfers>
```

This is a sample file that the command line client will be able to parse, it can also be found in **$GLOBUS_LOCATION/share/globus_wsrf_rft_client/** along with other samples for directory transfers and deletes (lines starting with # are comments):

```
# Subject name (defaults to host subject)
  /DC=org/DC=doegrids/OU=People/CN=Ravi Madduri 134710
  gsiftp://localhost:5678/tmp/rftTest_Done.tmp
  gsiftp://localhost:5678/tmp/rftTest_Done1.tmp
```

Limitations

No limitations with this commandline tool.

GT 4.0 Data Management Glossary

Bloom filter

Compression scheme used by the Replica Location Service (RLS) that is intended to reduce the size of soft state updates between Local Replica Catalogs (LRCs) and Replica Location Index (RLI) servers. A Bloom filter is a bit map that summarizes the contents of a Local Replica Catalog (LRC). An LRC constructs the bit map by applying a series of hash functions to each logical name registered in the LRC and setting the corresponding bits. (RLS)

client

As noted above, FTP is a command/response protocol. The defining characteristic of a client is that it is the process sending the commands and receiving the responses. It may or may not take part in the actual movement of data (see client/server and third party transfers below). (GridFTP)

client/server transfer

In a client/server transfer, there are only two entities involved in the transfer, the client entity and the server entity. We use the term entity here rather than process because in the implementation provide in GT4, the server entity may actually run as two or more separate processes. The client will either move data from or to, his local host. The client will decide whether or not he wishes to connect to the server to establish the data channel or the server should connect to him (MODE E described below, dictates who must connect). If he wishes to connect to the server, he will send the PASV (passive) command. The server will start listening on an ephemeral (random, non-privileged) port and will return the IP and port as a response to the command. The client will then connect to that IP/Port. If the client wishes to have the server connect to him, the client would start listening on an ephemeral port, and would then send the PORT command which includes the IP/Port as part of the command to the server and the server would initiate the TCP connect. Note that this decision has an impact on traversing firewalls. For instance, the clients host may be behind a firewall and the server may not be able to connect. Finally, now that the data channel is established, the client will send either the RETR "filename" command to transfer a file from the server to the client (GET), or the STOR "filename" command to transfer a file from the client to the server (PUT). (GridFTP)

command/response

Both FTP and GridFTP are command/response protocols. What this means is that once a client sends a command to the server, it can only accept responses from the server until it receives a response indicating that the server is finished with that command. For most commands this is not a big deal. For instance, setting the type of the file transfer to binary (called I for image in the protocol), simply consists of the client sending TYPE I and the server responding with 220 OK. Type set to I. However, the SEND and RETR commands (which actually initiate the movement of data) can run for a long time. Once the command is sent, the client's only options are to wait until it receives the completion reply, or kill the transfer. (GridFTP)

concurrency

When speaking of GridFTP transfers, concurrency refers to having multiple files in transit at the same time. They may all be on the same host or across multiple hosts. This is the equivalent to starting up different clients for different files, and having them all running at the same time. This can be effective

if you have many small files to move. The Reliable File Transfer (RFT) service utilizes concurrency to improve its performance. (GridFTP)

dual channel protocol

GridFTP uses two channels. One of the channels, called the control channel, is used for sending commands and responses. It is low bandwidth, and is encrypted for security reasons. The second channel is known as the data channel. Its sole purpose is to transfer the data. It is high bandwidth and uses an efficient protocol. By default, the data channel is authenticated at connection time, but no integrity checking or encryption is performed due to performance reasons. Integrity checking and encryption are both available via the client and libraries. Note that in GridFTP (not FTP) the data channel may actually consist of several TCP streams from multiple hosts. See the description of parallelism and striping below for more details. (GridFTP)

extended block mode (MODE E)

MODE E is a critical GridFTP components because it allows for out of order reception of data. This in turn, means we can send the data down multiple paths, and don't have to worry if one of the paths is slower than the others and the data arrives out of order. This enables parallelism and striping within GridFTP. In MODE E, a series of "blocks" are sent over the data channel. Each block consists of an 8 bit flag field, a 64 bit field indicating the offset in the transfer, and a 64 bit field indicating the length of the payload, followed by length bytes of payload. Note that since the offset and length are included in the block, out of order reception is possible, as long as the receiving side can handle it, either via something like a seek on a file, or via some application level buffering and ordering logic that will wait for the out of order blocks. [INSERT PICTURE HERE] (GridFTP)

improved extended block mode (MODE X)

This protocol is still under development. It is intended to address a number of the deficiencies found in MODE E. For instance, it will have explicit negotiation for use of a data channel, thus removing the race condition and the requirement for the sender to be the connector. This will help with firewall traversal. A method will be added to allow the filename to be provided prior to the data channel connection being established to help large data farms better allocate resources. Other additions under consideration include block checksumming, resends of blocks that fail checksums, and inclusion of a transfer ID to allow pipelining and de-multiplexing of commands. (GridFTP)

Local Replica Catalog (LRC)

Stores mappings between logical names for data items and the target names (often the physical locations) of replicas of those items. Clients query the LRC to discover replicas associated with a logical name. Also may associate attributes with logical or target names. Each LRC periodically sends information about its logical name mappings to one or more RLIs. (RLS)

logical file name

A unique identifier for the contents of a file. (RLS)

logical name

A unique identifier for the contents of a data item. (RLS)

mode command

In reality, GridFTP is not a "protocol", but a collection of several protocols. There is a protocol used on the control channel, but there is a range of protocols available for use on the data channel. Which protocol is used is selected by the MODE command. Four modes are defined: STREAM (S), BLOCK (B), and COMPRESSED (C) in RFC 959 for FTP, and EXTENDED BLOCK (E) in GFD.020 for GridFTP. There is also a new data channel protocol, or mode, being defined in the GGF GridFTP Working group, which for lack of a better name at this point, is called MODE X. (GridFTP)

network end points

A network endpoint is generally something that has an IP address (a network interface card). It is a point of access to the network for transmission or reception of data. Note that a single host could have multiple network end points if it had multiple NICs installed (multi-homed). This definition is necessary to differentiate between parallelism and striping, described below. (GridFTP)

parallelism

When speaking about GridFTP transfers parallelism refers to having multiple TCP connections between a single pair of network endpoints. This is used to improve performance of transfers on connections with light to moderate packet loss. (GridFTP)

physical file name

The address or the location of a copy of a file on a storage system. (RLS)

Replica Location Index (RLI)

Collects information about the logical name mappings stored in one or more Local Replica Catalogs (LRCs) and answers queries about those mappings. Each RLI periodically receives updates from one or more LRCs that summarize their contents. (RLS)

Replica Location Service (RLS)

A distributed registry that keeps track of where replicas exist on physical storage systems. The job of the RLS is to maintain associations, or mappings, between logical names for data objects and one or more target or physical names for replicas. Users or services register data items in the RLS and query RLS servers to find replicas. (RLS)

RLS attribute

Descriptive information that may be associated with a logical or target name mapping registered in a Local Replica Catalog (LRC). Clients can query the LRC to discover logical names or target names that have specified RLS attributes. (RLS)

server

The compliment to the client is the server. Its defining characteristic is that it receives commands and sends responses to those commands. Since it is a server or service, and it receives commands, it must be listening on a port somewhere to receive the commands. Both FTP and GridFTP have IANA registered ports. For FTP it is port 21, for GridFTP it is port 2811. This is normally handled via inetd or xinetd on Unix variants. However, it is also possible to implement a daemon that listens on the specified port. This is described more fully in the operation overview section below. (GridFTP)

stream mode (MODE S)

The only mode normally implemented for FTP is MODE S. This is simply sending each byte, one after another over the socket in order, with no application level framing of any kind. This is the default and is what a standard FTP server will use. This is also the default for GridFTP. (GridFTP)

striping

When speaking about GridFTP transfers striping refers to having multiple network endpoints at the source, destination, or both, participating in the transfer of the same file. This is normally accomplished by having a cluster with a parallel shared file system. Each node in the cluster reads a section of the file and sends it over the network. This mode of transfer is necessary if you wish to transfer a single file faster than a single host is capable of. This also tends to only be effective for large files, though how large depends on how many hosts and how fast the end to end transfer is. Note that while it is theoretically possible to use NFS for the shared file system, your performance will be poor, and would make using striping pointless.

target name

The address or location of a copy of a data item on a storage system. (RLS)

third party transfers

In a third party transfer, there are three entities involved. The client, who will only orchestrate, but not actually take place in the data transfer, and two servers one of which will be sending data to the other. This scenario is common in Grid applications where you may wish to stage data from a data store somewhere to a supercomputer you have reserved. The commands are quite similar to the client/server transfer described above. However, now the client must establish two control channels, one to each server. He will then choose one to listen, and send it the PASV command. When it responds with the IP/port it is listening on, the client will send that IP/port as part of the PORT command to the other server. This will cause the second server to connect to the first server, rather than the client. To initiate the actual movement of the data, the client then sends the RETR filename command to the server that will read from disk and write to the network (the "sending" server) and will send the STOR "filename" command to the other server which will read from the network and write to the disk (the "receiving" server). (GridFTP)