FERMILAB UNIX FILE SYSTEM SRM

Fermilab provides a reference implementation of the SRM-Storage interface to a Unix File System. You can read more about SRM interface at [1,2,3,4]. Fermilab's srm implementation conforms to the SRM version 1.1 specification. Fermilab srm implementation is developed as a part of DESY – Fermilab dCache project [5]. Its source is available for download from [4]. It provides the following features: fully blown implementation of srm v1.1 specification, support for srm "get", "put" and "copy" functions, gsiftp as a transfer protocol for "get " and "put" operations, built-in gsiftp client for SRM to SRM or MSS transfers. This reference implementation provides highly configurable transfer scheduling mechanism, which utilizes queues and limits the number of simultaneous transfers performed by either client or server itself. It retries each operation upon failure and uses database to store its state to provide a high level of robustness and reliability.

CVS access to File System SRM and SRM client.

The latest version of this product can be downloaded from the publically accessible cvs repository.
To access srm public DCACHE CVS repository, execute the following cvs commands:

```
$ cvs -d :pserver:cvs@cvs.dcache.org:/cvs login
Logging in to :pserver:cvs@cvs.dcache.org:2401/cvs
CVS password: (password is 'cvs')
$ cvs -d :pserver:cvs@cvs.dcache.org:/cvs co srm srmclient
```

The prerequisite to the srm installation if the presence of the host certificates and the functional gsiftp server installation from globus toolkit on the node. SRM product uses of the postgress database, which should be installed and running on one of the network reachable server, with username and password available. SRM server is a java program, which will need the java compiler for building and java virtual machine for execution. Please refer to [6], [7] and [8] for installation and configuration instructions of gridftp server, postgres database server and java development kit (jdk).

Building.

To build SRM , make the root directory of srm product your current than execute the following command:

```
$ ./makeUnixfsSRM.sh
```

Note: The script `makeUnixfsSRM.sh` relies on the presence of Sun Microsystem's java 1.4 development kit. The system path should include the kit's bin directory containing javac compiler.

Configuration.

Once the srm is built, you would need to edit .srmconfig/config.xml, and put in the correct values for sql database user, password, url and jdbc driver (Postress jdbc driver is included in pgjdbc2.jar), and the locations of the host x509 private key and certificate.

You will also need to edit the script bin/run-unix-fs and put in the correct values for gridftp server host and port (variables GRIDFTP_HOST and GRIDFTP_PORT), name of the SRM server (variable NAME), which needs to be unique for each SRM server using the same database, in case of multiple SRMs, and location of the log file (variable LOG).

In order to add users to the system, you will have to modify two files:
1. /etc/grid-security/grid-mapfile to add the mapping from the user certificate to the local account
2. .srmconfig/dcache.kpwd

to add the mapping and the login entries for the user. There is a description of the structure of this file in the prefix of dcache.kpwd file itself.

Running the server.

Once it is done, you can start the server by running the following commands as a root from the root directory of the srm product:

```
$ bin/run-unix-fs
```

Once this is done, you will see an srm admin shell prompt on the screen, type help to see the list of available commands. This will allow you to monitor the state of the system and to cancel the requests.
Note: SRM will create all the database tables automatically.

Building the srm client.

Once the system is running, you can use srmcp client to access the system.
To build the client do the following:
Assign the path to the srm server installation directory to an exported enviroment variable SRM_SERVER:

```
$ export SRM_SERVER= <path to srm>
```

execute the following commands:

```
$ cd srmclient
$ cp -r $SRM_SERVER/lib/globus lib/globus
$ cp -r $SRM_SERVER/lib/axis lib/axis
$ cp -r $SRM_SERVER/lib/glue lib/glue
$ cp $SRM_SERVER/lib/*.jar  lib/
$ make
```

Read README text file to learn about srmcp usage

REFERENCES

[1] Arie Shoshany at al, SRM Joint Design v.1.0, http://sdm.lbl.gov/srm-wg/doc/srm.v1.0.pdf
[2] Arie Shoshany at al, SRM Interface Specification v.2.1, http://sdm.lbl.gov/srm-wg/doc/SRM.spec.v2.1.final.pdf
[3] The Storage Resource Manager Collaboration Pages, http://sdm.lbl.gov/srm-wg/
[4] Fermilab SRM Project , http://www-isd.fnal.gov/srm
[5] DCache, Disk Cache Mass Storage System, http://www.dcache.org/
[6] http://www.globus.org
[7] http://www.postgresql.org/
[8] http://java.sun.com