Use of the SRM interface

- Use case
- What is the SRM?
  - Who develops it?
  - Is it a standard?
  - What should it do?
- Available tools
- Critical features
- Conclusions

Use case

- VO typically has heterogeneous storage resources
  - Various computer centers, administrative domains
  - Various hardware
  - Various software to manage the storage

- VO should be shielded from all those differences by a common Storage Resource Management interface
  - API used by VO applications
  - Define common/proper usage paradigms
    - Improve efficiency
    - Simplify client and server codes
    - Outlaw bad practices

What is the SRM?

- Client-server interface for Storage Resource Management
  - De facto standard (see further on), GGF working group
    - http://sdm.lbl.gov/srm-wg/
  - Secure web service
  - Defines functions that allow storage resources to be managed from both client and server perspectives
    - Different requirements, optimizations, concerns

- Big step up compared to "Classic SE" (simple GridFTP server)
  - Comes at a price:
    - No direct GridFTP access to data
      - Need to ask SRM for Transfer URL, generally cannot be predicted
    - NFS access to data unavailable in most implementations
      - Simple implementation would interfere with server-side space management
      - StoRM project uses POSIX file system (e.g. GPFS) with "just-in-time" ACLs
        - » http://www.egrid.it/sw/storm

Who develops “the” SRM?

- SRM collaboration institutes develop different implementations
  - CERN + RAL + INFN
    - CASTOR-2
  - CERN/LCG
    - DPM
  - FNAL (+ DESY)
    - dCache
  - JLAB
    - J-SRM
  - LBNL
    - DRM, HRM
  - EGRID/INFN/GridIt
    - StoRM
- Big computing facilities with different user communities
  - Different requirements, priorities, legacy interfaces
- The goal is to make them compatible from the grid perspective

Is the SRM a standard?

- "The nice thing about standards is that there are so many to choose from." - Andrew S. Tanenbaum

- Version 1.1 in widespread use
  - But implementations have subtle incompatibilities due to ambiguities in the "standard"
  - Various basic functionalities not defined

- Version 2.1 implemented to various extents by some projects
  - Try to get a critical subset implemented on WLCG by autumn 2006
    - Use cases defined by LHC experiments, see next pages
  - Still lacks some features
  - Incompatible with version 1
    - Clients and servers need to support both versions during transition period
      - May last a long time

- Version 3 definition many months away
  - Again incompatible

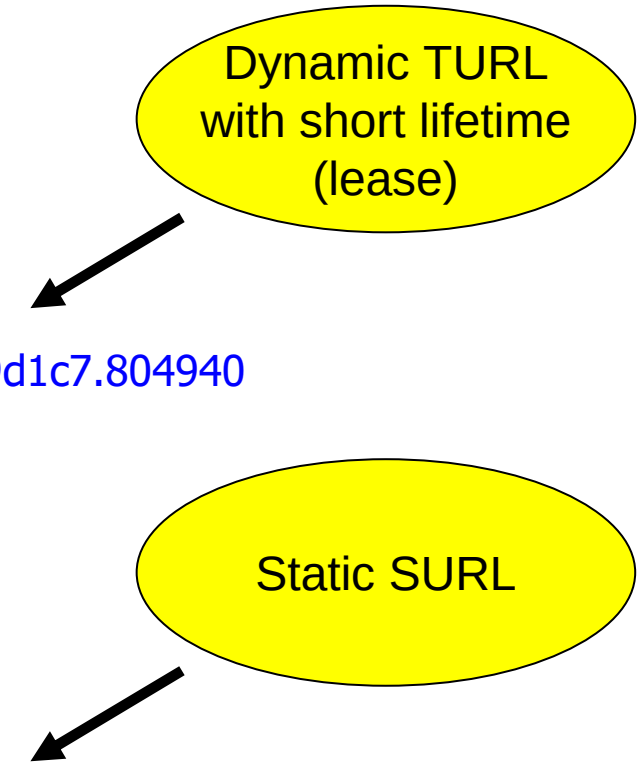
What should the SRM do?

(A. Shoshani, PPDG Review, 28 Apr 2003)

- Manage space dynamically
  - Any disk caches and Mass Storage Systems
  - Space reservation and negotiation
  - Manage "lifetime" of spaces
- Manage files dynamically
  - Pin files in storage till they are released
  - Manage "lifetime" of files, and action when lifetime expires
- Manage file sharing
  - Policies on what to evict when space is needed
    - Currently always decided by back-end
- Manage multi-file requests
  - A brokering function: queue file requests, pre-stage files
  - Invoke file transfer services
- Permit site-SRM over multiple storage systems
- Negotiate transfer protocols

Example user exposure to SRM

```
$ lcg-cr -v --vo dteam file:`pwd`/my_file -d castorsrm.cern.ch -l lfn:/grid/dteam/my_LFN
Using grid catalog type: lfc
Source URL: file:/afs/cern.ch/user/m/maart/my_file
File size: 2463
VO name: dteam
Destination specified: castorsrm.cern.ch
Destination URL for copy:
  gsiftp://castorgrid04.cern.ch:2811//shift/lxfsrk5104/
  data02/cg/stage/filecf87efc6-4e3d-4fb7-af19-762deda9d1c7.804940
# streams: 1
# set timeout to 0 seconds
Alias registered in Catalog: lfn:/grid/dteam/my_LFN
              0 bytes 0.00 KB/sec avg 0.00 KB/sec inst
Transfer took 630 ms
Destination URL registered in Catalog:
  srm://castorsrm.cern.ch/castor/cern.ch/grid/
  dteam/generated/2006-03-01/filecf87efc6-4e3d-4fb7-af19-762deda9d1c7
guid:f2c637ea-e699-44cc-adb9-9ec9445b59d8
```



SRM details usually “hidden”

- SRM methods usually considered low-level by VO applications
- Command line tools available
  - Simple use cases handled e.g. by “lcg-util” suite
    - lcg-cr, lcg-cp, lcg-rep, lcg-del, …
  - Bulk operations handled e.g. by File Transfer Service
  - VO tools
- API available through higher-level libraries
  - Grid File Access Library
  - lcg-util library
  - VO libraries
- SRM used in conjunction with information system and catalogs

Critical features for WLCG

- Result of WLCG Baseline Services Working Group
  - http://cern.ch/lcg/PEB/BS
- Originally planned to be implemented by WLCG Service Challenge 4
  - Delayed until autumn 2006
- Features from version 1.1 + critical subset of version 2.1

(Nick Brook, SC3 planning meeting – June '05)

  - File types
  - Space reservation
  - Permission functions
  - Directory functions
  - Data transfer control functions
  - Relative paths
  - Query supported protocols

File types

- Volatile
  - Temporary and sharable copy of an MSS resident file
  - If not pinned it can be removed by the garbage collector as space is needed (typically according to LRU policy)
- Durable
  - File can only be removed if the system has copied it to an archive
- Permanent
  - System cannot remove file
- Users can always explicitly delete files
- For SC4 the experiments only want durable and permanent

Space reservation

- v1.1
  - Space reservation done on file-by-file basis
  - User does not know in advance if SE will be able to store all files in multi-file request
- v2.1
  - Allows for a user to reserve space
    - But can 100 GB be used by a single 100 GB file or by 100 files of 1 GB each?
    - MSS space vs. disk cache space
  - Reservation has a lifetime
  - "PrepareToGet(Put)" requests fail if not enough space
- v3.0
  - Allows for "streaming"
    - When space is exhausted requests wait until space is released
  - Not needed for SC4
- What about quotas?
  - Strong interest from LHC VOs, but not yet accepted as task for SRM

Permission functions

- v2.1 allows for POSIX-like ACLs
  - Can be associated per directory and per file
  - Parent directory ACLs inherited by default
  - Can no longer let a simple UNIX file system deal with all the permissions
    - Need file system with ACLs or ACL-aware permission manager in SRM
      - May conflict with legacy applications
- LHC VOs desire storage system to respect permissions based on VOMS roles and groups
  - Currently only supported by DPM
- File ownership by individual users not needed in SC4
  - Systems shall distinguish production managers from unprivileged users
    - Write access to precious directories, dedicated stager pools
    - Supported by all implementations

Directory functions

- Create/remove directories
- Delete files
  - v1.1 only has an "advisory" delete
    - Interpreted differently by different implementations
      - Complicates applications like the File Transfer Service
- Rename directories or files (on the same SE)
- List files and directories
  - Output will be truncated to implementation-dependent maximum size
    - Full (recursive) listing could tie up or complicate server (and client)
      - May return huge result
      - Could return chunks with cookies → server would need to be stateful
    - It is advisable to avoid very large directories
- No need for "mv" between SEs

Data transfer control functions

- StageIn, stageOut type functionality
  - prepareToGet, prepareToPut
- Pinning and unpinning files
  - Avoid untimely cleanup by garbage collector
  - Pin has a lifetime, but can be renewed by client
    - Avoid dependence on client to clean up
- Monitor status of request
  - How many files ready
  - How many files in progress
  - How many files left to process
- Suspend/resume request
  - Not needed for SC4
- Abort request

Relative paths

- Everything should be defined with respect to the VO base directory
- Example:

  srm://castorsrm.cern.ch/castor/cern.ch/grid/lhcb/DC04/prod0705/0705_123.dst

- SE defined by protocol and hostname
- VO base directory is the storage root for the VO
  - Advertized in information system, but unnecessary detail
    - Clutters catalog entries
    - SRM could insert VO base path automatically
      - Available in dCache
- VO namespace below base directory

Query supported protocols

- List of transfer protocols per SE available from information system
  - Workaround, complicates client
  - SRM knows what it supports, can inform client
- Client always sends SRM a list of acceptable protocols
  - gsiftp, (gsi)dcap, rfio, xrootd, root, …
  - SRM returns TURL with protocol applicable to site
- Query not needed for SC4

Conclusions

- SRM is a cornerstone of grid data management
  - Can make heterogeneous storage facilities look similar
- Version 1.1 already in widespread use, but lacks important functionality
- Version 2.1 big step forward, but not widely available before autumn
  - Only critical functionalities considered
- Version 3 far away
  - May deal with quotas
- Tools and libraries available to shield users from SRM details