Ron's SRM survey (Tier I's only)

With replies from

- CNAF (Luca)
- BNL (Pedro)
- Fermi (Jon)
- Sara (Ron)
- fzk (Jos)
- RAL (Shaun)

Ron's Questions

Problems with current srmv2.2 spec and implementation

- *What do you think is exactly wrong with the SRM v2.2 spec?*
- *What issues do you see at the moment with the current SRM implementations? Please specify what implementation you are running.*
  - *How could these issues be solved, do you think?*
- *What features do you think are necessary or useful in the current SRM implementations and should be retained?*
- *What features do you think are not necessary?*

SRM replacement

- *If there would be a chance to come up with a something new, replacing the SRM. What functionality should it possess apart from put and get and namespace operations?*
- *Are there certain implementation features that it should possess?*
- *If, in order to have an efficient implementation, it would be necessary to have a SRM replacement interface that would differ from implementation to implementation, would this be acceptable?*

Pedro Salgado (BNL, dCache)

Performance :

Multiple SRM servers for load balancing

Maybe partitioning by activity

SRM should protect back-end (name-service)

for me, GSI is a burden specially for read operations and it seems to be one of the bottlenecks on dCache's SRM server.

Future :

there should be a main API that each implementation should respect but if a particular implementation wanted to provide more methods, I think they should be allowed to.

Jon Bakken (Fermi, dCache)

What do you think is exactly wrong with the SRM v2.2 spec?

- *The spec is new enough that some items are under-defined and open to interpretation between the various providers.*
- *Choice of authentication protocol may not be the best choice today. Standard TLS may be a better choice today.*
- *The spec is complex, only a small subset is really needed. The*
- *extra requirements have caused many problems and distracted from stabilizing the core requirements.*

Jon Bakken (Fermi, dCache)

What issues do you see at the moment with the current SRM implementations? Please specify what implementation you are running. How could these issues be solved, do you think?

dCache implementation problems only

- *Current version is not scalable. This is being solved by introducing a horizontal load balancing SRM server.*
- *If there are problems, diagnosing the issue is difficult. Reporting of errors is not very good.*
- *Recovery from errors does not work well.*
- *The configuration is complex and documentation is lacking in some cases.*
- *Underlying dCache problems manifest themselves as SRM problems. (pnfs overloading for example)*

Jon Bakken (Fermi, dCache)

What features do you think are necessary or useful in the current SRM implementations and should be retained?

Anything already written that does not hinder performance should be retained.

What features do you think are not necessary?

- *We don't use Space Management. Access latency, retention policy, etc are all concepts that do not belong to a srm system.*
- *Recursive ls is a dangerous idea.*
- *Auto directory creation demonstrates a lack of planning and has created a mess.*

Jon Bakken (Fermi, dCache)

SRM replacement:

If there would be a chance to come up with a something new, replacing the SRM. What functionality should it possess apart from put and get and namespace operations?

- *We need the copy function for performance of wan transfers.*
- *We need access control for some operations (srm-bring-online should be limited for example)*

If, in order to have an efficient implementation, it would be necessary to have a SRM replacement interface that would differ from implementation to implementation, would this be acceptable?

No, we need a common interface or we should drop the whole idea.

Jos van Wesel (gridKa,Scc, dCache)

SRM and possible improvements

- What do you think is exactly wrong with the SRM v2.2 spec?
  - implements to many features
  - forms a non scalable bottleneck

- What issues do you see at the moment with the current SRM implementations?

Please specify what implementation you are running.

FZK uses dCache

  - its slowing down the dCache system responses
  - it takes a huge toll on cycles on the SRM machine
  - it introduces DB inconsistencies between subcomponents of the system

- How could these issues be solved, do you think?

  reduce the number of SRM features

- What features do you think are necessary or useful in the current SRM implementations and should be retained?

  remote authenticated access to the SE

Rons Tier I SRM survey

Jos van Wesel (gridKa,Scc, dCache)

What features do you think are not necessary?

- *spacetokens*

SRM replacement:
If there would be a chance to come up with a something new, replacing the SRM. What functionality should it possess apart from put and get and namespace operations?

- *The new interface would be MRS: Mass Reversed Storage :-)*

Are there certain implementation features that it should possess?

- *The MRS should be able to exchange its capabilities and features with clients so clients stay upward and downward compatible.*
- *MRS implementations should be stateless*

If, in order to have an efficient implementation, it would be necessary to have a SRM replacement interface that would differ from implementation to implementation, would this be acceptable

- *No*

Rons Tier I SRM survey

Luca dell'Agnello (CNAF,SToRM, CASTOR)

Problems with current srmv2.2 spec and implementation:

- What do you think is exactly wrong with the SRM v2.2 spec?
It's hard to say from our (i.e. storage provider) point of view what is exactly wrong. From our experience we can say that some of the requested functionalities are never used in practice such as the additional degree of freedom of space-tokens respect to SURLs.
- What issues do you see at the moment with the current SRM implementations? Please specify what implementation you are running. How could these issues be solved, do you think?
At CNAF we have currently two different storage systems: StoRM (for D1T0 and D1T1) and Castor (for D0T1).
For StoRM we would like to have the support for multiple back-ends (currently only multiple front-ends are supported) in order to enable a full fail-over mechanism. Additional points are: an administration tool (mainly a configuration utility) and a monitoring tool for the service (presently we use the standard lemon/nagios tools and we parse the log files). In any case there are no significant issues related to StoRM.

Luca dell'Agnello (CNAF,SToRM, CASTOR)

- What issues do you see at the moment with the current SRM implementations? Please specify what implementation you are running. How could these issues be solved, do you think?

The use of Castor srm end-points is much lower than those of StoRM; Anyway in this case the issues generally raise from the Castor system rather than from the SRM layer itself.

- What features do you think are necessary or useful in the current SRM implementations and should be retained?

- *put/get/namespace operations.*
- *the neutrality respect to the transfer protocols offered by the underlying storage system and we should avoid the temptation of the unique protocol (we do not believe xrootd is the protocol....).*

- What features do you think are not necessary?

As mentioned above, we think that the degree of freedom given by the space

nother point to retain from the current srm specs is the neutrality resp
not believe xrootd is the protocol....).

Luca dell'Agnello (CNAF,STorRM, CASTOR)

SRM replacement:

- If there would be a chance to come up with a something new, replacing the SRM. What functionality should it possess apart from put and get and namespace operations?

In any case, any future replacement (or more likely a new version of the srm specs) should not interfere with the functionalities of the underlying storage system. A typical case is given by a cluster file-systems (such as GPFS or Lustre) which natively aggregate logical volumes in a unique namespace. Moreover some functionalities (such as the permission enforcement) should be based on the functionalities offered by the underlying storage system.

In a few words: keep it simple as possible.

Luca dell'Agnello (CNAF,STorRM, CASTOR)

- Are there certain implementation features that it should possess?

Besides put/get/namespace operations, a quota management system per VO would be desidered (i.e. the VOmaneger can manage, via srm, the storage space assigned to a group)

- If, in order to have an efficient implementation, it would be necessary to have a SRM replacement interface that would differ from implementation to implementation, would this be acceptable?

ABSOLUTELY NOT !!

Shaun De Witt (RAL, CASTOR)

Problems with current srmv2.2 spec and implementation:

- What do you think is exactly wrong with the SRM v2.2 spec?

- *Too many interfaces - space management should be outside scope*
- *Too many 'concepts', and the useful ones are used in the wrong place. Specifically I would not use accessLatency/transferMode, retentionPolicy is useful only on PUT requests, and FileLocality only useful on Ls requests*
- *Far too many optional parameter can be passed in to most operations, making maintenance harder and allowing more scope for users to make mistakes.*

- What issues do you see at the moment with the current SRM implementations? Please specify what implementation you are running. How could these issues be solved, do you think?

- *(With developers hat on) - no problems at all ;)*
- *Can be difficult to configure w/o expert knowledge*
- *Restricting access to space tokens difficult/impossible to set up*
- *GSI certificate decoding very CPU intensive; meaning good h/w required*
- *Difficult to trace requests from the SRM into CASTOR*

Rons Tier I SRM survey

Shaun De Witt (RAL, CASTOR)

- What features do you think are necessary or useful in the current SRM implementations and should be retained?

- Transfer and namespace operations. Space interrogation (GetSpaceMetadata, GetSpaceTokens)

- What features do you think are not necessary?

- Pinning (can be achieved using Disk1Tape1 + ReleaseFiles)
- Space Management
- BringOnline (IMO could just as easily be done using Get)

Shaun De Witt (RAL, CASTOR)

SRM replacement:

- If there would be a chance to come up with a something new, replacing the SRM. What functionality should it possess apart from put and get and namespace operations?

- Could try and take charge of the transfer, so the whole process could be within the scope of the new method.
- Reduced security overhead to allow use of less performant hardware

- Are there certain implementation features that it should possess?

- Aside from the current transfer and namespace operations
- fairshare for shared SRMs

Rons Tier I SRM survey

Shaun De Witt (RAL, CASTOR)

- If, in order to have an efficient implementation, it would be necessary to have a SRM replacement interface that would differ from implementation to implementation, would this be acceptable?

Yes and No;

- *for non-knowledgeable user anything dCache, castor, gpfs or whatever in the interface should be removed even if it means the storage system may not behave optimally for knowledgeable users hints to the back end storage system are useful.*
- *Much as I hate to say it, a simple basic interface and the use of keyword/value pairs seems like the optimal solution*

Ron Trompert (Sara, dCache)

- *What do you think is exactly wrong with the SRM v2.2 spec?*

- *It is far too complex for the relatively simple use cases it has to cater for. It is far too bloated. Complexity is the enemy of reliability. This is what we have seen since we run srm v2.2.*
- *Things are not defined well enough leading to different interpretations among developers. The ONLINE, NEARLINE issue is a good example of this*

Ron Trompert (Sara, dCache)

- What issues do you see at the moment with the current SRM implementations? Please specify what implementation you are running.

- The problems we see is that we are not able to protect our srm server from abuse by naive users. Only this week our srm got get turl request for rfio at a rate of 170 Hz causing the load on our srm server to go though the roof. The srm managed to survive though due to the fact that we have set all retries to zero. We banned the culprit users from gPlazma but that did not help much because rejecting the requests and producing all these tracebacks in the catalina.out file also caused a high load on the machine. Things are not defined well enough leading to different interpretations among developers. The ONLINE, NEARLINE issue is a good example of this.
- Lack of scalability.
- The SRM spec is only partly implemented which renders, for example, a storage class as T1D1 useless. By the way, the pin lifetimes supplied to srmbringOnline and srmPrepare ToGet are integers. This would mean that it would be possible to pin a file for 68 years which is a pretty good aproximation of T1D1. So why bother to have t1d1 at all.
- This is not so much a problem for just SRM itself but for datamanagement in general. The experiment frameworks do brokering with their data transports, the FTS does brokering and the srm does brokering. This may lead to mysterious behaviour which is hard to understand for site admins trying to track down problems. Since we are dealing with data transport here so another complicating factor is that their are more than one site involved, making debugging even more difficult.

Ron Trompert (Sara, dCache)

- What features do you think are necessary or useful in the current SRM implementations and should be retained?

- put/get and namespace operations should be retained
- srm is useful as a load balancer over gridftp, (gsi)dcap or xrootd servers
- The ability to issue bulk stage requests. This is something that the other protocols are not able to provide.
- The ability for a client to specify the access-latency in a uniform way for the different storage systems

Ron Trompert (Sara, dCache)

- *What features do you think are not necessary?*

- *Space reservation is useless*
- *In the dCache setting a space token is useless because you can achieve the same thing by copying files to different directories. But I can imagine that it is useful to be able to flag files so that they will be copied to a certain tape set. This way you would have a uniform way to do that with the different storage systems.*
- *T1D1 is useless*
- *Recursive srmLs is a time bomb. This has no place in peta scale storage systems. srmLs should only work on a single file or single directory (non-recursively). Considering the fact that we will have directories with more than a million files in it, I would argue to place an upper limit to the number of results returned doing an srmLs on a directory.*

Ron Trompert (Sara, dCache)

- If there would be a chance to come up with a something new, replacing the SRM. What functionality should it possess apart from put and get and namespace operations?

- *A light weight interface to storage systems containing necessary functionality the other protocols do not provide. I think that one of the reasons that the road for srm v2.2 implementation has been so rough is that it needed to be implemented in already existing storage systems. It did not really fit well (probably an understatement) with the existing storage systems. A far less rich interface would probably be a lot easier to implement into existing systems.*

Ron Trompert (Sara, dCache)

- *Are there certain implementation features that it should possess?*

* *put, get with load balancing over the servers providing the transfer protocols and namespace operations*
* *bulk staging*
* *file pinning and unpinning*
* *the ability to specify accesslatency for a client. Hereby I deliberately do not mention retention policy since this has no added value. Having only one tape copy of a file like we have for our HEP VOs hardly qualifies as custodial storage. Users are interested in how fast the data access is. The quality of the storage should be part of an agreement the VOs have with the sites but does not need to be included in an SRM spec.*

Ron Trompert (Sara, dCache)

- If, in order to have an efficient implementation, it would be necessary to have a SRM replacement interface that would differ from implementation to implementation, would this be acceptable?

- If there is no other way, there is no other way but preferably not. If it is not possible to have a uniform, functional, stable, scalable and performing srm implementation we should throw it over board altogether and just be happy with xrootd :-)

Summary

Problems with current srmv2.2 spec and implementation:

- What do you think is exactly wrong with the SRM v2.2 spec?

- Protocol too complicated, too many features.
- Protocol open to different interpretation.
- GSI protocol instead of standard SSL.
- Some functions are not used in practice.

- What issues do you see at the moment with the current SRM implementations? Please specify what implementation you are running. How could these issues be solved, do you think?

- dCache : SRM is not scalable yet.
- dCache and Castor : SRM problems mostly are back-end problems
- dCache and Castor : difficult to configure
- dCache and Castor : problem diagnositic is difficult.

Summary

Problems with current srmv2.2 spec and implementation:

- What features do you think are necessary or useful in the current SRM implementations and should be retained?

- ***Anything already written that does not hinder performance should be retained.***
- ***remote authenticated access to the SE***
- ***put/get/namespace operations.***
- ***Space interrogation (GetSpaceMetadata, GetSpaceTokens)***
- ***the neutrality respect to the transfer protocols offered by the underlying storage system and we should avoid the temptation of the unique protocol (we do not believe xrootd is the protocol....).***

Summary

Problems with current srmv2.2 spec and implementation:

- What features do you think are not necessary?

- Space Management.
- Access latency, retention policy, etc are all concepts that do not belong to a srm system.
- Recursive ls is a dangerous idea.
- Auto directory creation demonstrates a lack of planning and has created a mess.
- Bring Online
- T1D1 is useless

Summary

SRM replacement

- Protection for space
- exchange its capabilities and features with clients so clients stay upward and downward compatible
- coscheduling would be required (FTS), We need the copy function for performance of wan transfers.
- The ability for a client to specify the access-latency in a uniform way for the different storage systems
- Any new functionality should be based on the features of the underlying storage system and should be orthogonal.

- If, in order to have an efficient implementation, it would be necessary to have a SRM replacement interface that would differ from implementation to implementation, would this be acceptable?
  - NO