Possible transition to an asynchronous srmLs in dCache: summary from a mail thread.

12/03/09: Flavia triggers a discussion about the possible consequences of a a change to an asynchronous srmLs in dCache. Especially, what should be changed in the SRM clients?

Alex Sim: our Bestman server uses synchronous, immediate returns on srmLs most of the time, especially when there is only disk backend. But it also supports asynchronous srmLs when MSS/HPSS backend is deployed and target directory has a very large number of files that make the listing slow. Our srm-ls client already supports asynchronous srmLs, when an SRM server returns proper status code and request token.

Andrea asks to Patrick: will the asynchronous srmLs be mandatory, or optional? If it is optional, sites could have the freedom to decide not to enable it until all relevant clients fully support it.

13/03/09: Patrick answers: It would indeed be possible to make it optional. However, that would make the overall setup even more complicated. The client tools would then have to know to which storage element to talk which version of the srmLs.
(Why? In principle that would be easy, as only an srmLs request that is asynchronous would return SRM_REQUEST_QUEUED, so the code would know what to do next..)
a) If only a single server would answer asynchronously, the client would already have to understand it anyway, so would have to be modified.
(mmhh not clear either:because in the context of WLCG, one could issue a recommendation to sites not to enable the functionality until clients are compliant. It would be better than to forcefully couple it to other (possibly critical) bug fixes in 1.9.3)
b) We are trying to solve an urgent stability problem. And this a piece (beside other methods in discussion) from which we would benefit a lot to our current understanding.
The first version which will have async SrmLs will be two releases away. (The current recommended release is 1.9.1 and the aysnc srmLs one will be 1.9.3) So we have plenty of time. (...) As everybody understands that commands which require a long processing time have to be asynchronous we have to understand that srmls is becoming a time-consuming command. It therefor blocks processing subsequent commands. BTW : The bestman client (OSG) and the ARC clients are already prepared.

Andrea: In practice, how would things change? I guess that an srmLs would be queued and eventually processed or timed out, but contrarily to other requests, there is no SRM_REQUEST_TIMED_OUT status, not even SRM_ABORTED. Similarly, there is no concept of desiredTotalRequestTime. If the goal is to limit the rate of "ls" on PNFS and the rate of incoming srmLs is too high, the requests will accumulate, but either they can stay in the queue forever with increasingly longer processing times, or abort. In both cases the client should poll for the status until it gives up or the request is aborted. Another way of solving the PNFS overload would be to keep the request **synchronous** and return SRM_INTERNAL_ERROR whenever dCache decides that PNFS is overloaded: the client could then retry with an exponential backoff as discussed in the past weeks. The advantage would be that srmLs calls would stay simple and the clients should not have to adapt. I guess that if the result of srmLs is necessary for what comes next, the code would still be blocked even with async srmLs; if not being blocked is important, one could fork a dedicated process. Patrick:Exactly, we can hold this particular sequence of commands a bit while having other processed. I'm concerned about blocking our input channels not blocking the client. As storage providers we have to optimize the overall throughput and responsiveness of a system (including FTS and the workernode clients) and not at all in the time for the individual client.

Patrick: The whole point is that the server has a fixed number of pipes to communicate with the client. This is enforced by tomcat. Unless we rewrite parts of tomcat, this is just how it is. A synchronous operation blocks one of those pipes for the time it takes to perform the entire operation. If all pipes are filled, no communication is possible any more although the server could e.g. easily reply on 'get status' requests or could initiate new bring onlines. Gerd did some detailed investigation on the karlsruhe system and found this happening very often. If we could queue srmLs's (on directories), we would be able to continue serving less

resource intensive requests while internally queuing the 'ls'. We don't want to have them 'timing out' we just want the system to continue while we collect directory entries before we reply.
a) This helps to overcome peaks which would other block the entire system and
b) it allows to partition the various requests (type wise) which is not possible on the tomcat level. A spike in an sync operation let all other operations suffer while with asyn, we can make only srmls slower if there is a spike and let others proceed.

The point is to overcome the deficiencies in the way tomcat is interfacing between the clients and the web-service and to allow smart scheduling and partitioning of request.

Andrea: ok, now it's clear. I assume that the reason why srmLs may take so long is because of the PNFS "bottleneck". Would the other storage systems have the same problem of srmLs blocking other kinds of requests, if srmLs was too slow for them?

srmLs in Bestman server by Alex Sim

From our experience with asynch srmLs with MSS backend, status code INPROGRESS can be returned along with QUEUED. 26000 or 40000 files in a flat directory can take a long time to be returned even if the query is already started. HEP community may not have those large flat directories, but it exists in other domains. We implemented asynch srmLs in our bestman server in a way that request token is always assigned for srmLs, but when the results can be returned in a "short" period of time and not blocking the communication channel, we return SUCCESS along with the listing results. All or most of the disk based queries for us return right away, and all existing SRM ls clients work along this line as we have tested with. When bestman server cannot get the results "soon" enough, we return QUEUED or INPROGRESS, and it happens many of our MSS backend cases because we limit communication channels to a few to the underlying MSS. We have not yet seen QUEUED/INPROGRESS returned with disk based bestman server from any of our deployed sites, but as we implemented in such a way, it is not at all impossible.

It seems that dCache may return listing results right away synchronously when result can be returned immediately (From Patrick mentioning queuing srmLs(directories))... or is it always asynchronous...? And most of the client use case for srmLs is one file at a time.... currently? If a clients expect a large listing or directory listings, they can implement handling asynchronous srmLs first while others can take their time... only if dCache may return reasonably immediately on a file. None of us (servers) wants users hanging on the connection for sure... connection is a resource too...

We can add/return those a couple (TIMED_OUT and ABORTED) if we agree it helps.

-- ElisaLanciotti - 19 Mar 2009

This topic: LCG > PreStagingTestsSummarysrmLs
Topic revision: r1 - 2009-06-05 - AndreaSciaba