Replica Usage in the LDG

1. Motivation

Having replications (copies belonging to the same “logical” filename) of data is common practice within grid-technology. Though identical regarding the content, different replications of a file might differ in certain aspects of “quality”. E.g., a certain replicum might be known to be eventually deleted in the near future or it might be known to be stored without an expiration time (“permanent”). Also, the expected latency by accessing a certain replicum is a valuable information for planning and scheduling and mechanisms for optimisation in advance would be desirable. Although the exact access time of course cannot be foreseen, the fact whether a certain replicum in a certain Storage Element (SE) is accessible only on tape, or (additionally) on disk usually is the most essential influence to the total access time. All Storage Elements within LDG provide a “staging mechanism” which allows clients to request files to be put online to be performantly accessible in the near future. The grid-infrastructure in LDG supports replica-management, but at the moment it is not yet introduced in a solid specified way. The following text summarises known concepts and definitions and examines how LDG could benefit from these ideas and proposals regarding replica-management.

2. Relevant Concepts

2.1. Type Definitions in SRM 2.2

Within the SRM 2.2-definitions “Type Definitions” are proposed that can be used to manage the usage of replicas in a certain grid[1].

2.1.1. File Storage Types

[SRM 07] distinguishes three file storage types (a file equates with a SURL): “volatile”, “durable” and “permanent”. The first two of them have an expiration time. A file marked as “volatile” might be deleted by a system autonomously though “durable” means, the file should not be deleted even if the expiration time has expired.

2.1.2. Retention Policies

Regarding the retention of a file, three qualities are given: “Replica quality”, ”Output quality” and “Custodial quality”. Files with “Replica quality” might easily be replaced in case of loss, because a replicum exists somewhere. “Output quality”-files might also be restored, but this might take some effort, e.g. relaunching a computational process. Data that is “custodial” cannot be restored in case of data-loss.

2.1.3. Access Latency Modes

Within that definitions three types of access latency exist: “online”, “nearline” and “offline”. If a file is “online”, it is accessible with expected lowest latency, that is, directly from any kind of disk over the net to the client. “Nearline” means, a staging operation has to be done before the actual transfer begins, e.g. copying from a tape to a disk-cache. “Offline” requires human intervention.

2.2. Proposal of “Storage Classes” in LDG

In [WLCG 06] another concept called “storage classes” is presented. Each storage class represents the minimal number of copies of a file on tape and on disk that are requested in the grid as shown in the following table:

[1] Some further types like “size in bytes” are not described here, as usage is very intuitive and straight-forward on a low technical level.

Storage class	*Minimum Required Copies*	
	Tape	Disk
Tape0Disk1	0	1
Tape0DiskN	0	>1
Tape1Disk0	1	0
Tape1Disk1	1	1
Tape1DiskN	1	>1
TapeNDisk0	>1	0
TapeNDisk1	>1	1
TapeNDiskN	>1	>1

Figure 1 : Storage Classes

This must be seen as the “usual” access possibility. A temporarily reduced availability (e.g. a crashed disk that is being repaired from backup-data) does not mean that a change of storage class occurs.

3. Aspired Usage in LDG

3.1. Basic Assumptions

The following basic assumptions are made for the usage of replicas within LDG:

- User-Initiated creating and deleting of replicas should be possible only at the level of ensembles (and of course only with sufficient read/write access-rights)
- Replicas might be created for permanent or temporary usage
- Creating and deleting replicas should be done by using the lrep/ldelrep or eventually further LTOOLS commands exclusively (needed to guarantee consistency)
- Informations about replicas that need to be stored centrally will be hold by the FC as a comment, therefore
- "lfc-setcomment" must not be used by any user directly (otherwise information needed by the system might be overwritten)
- Replicating an ensemble is NOT seen as an atomic transaction. A replicum of an ensemble may therefore be incomplete in case the copying of any of the files fails. If lrep command may be invoked several times, files that have already been replicated will not be copied again. If remote copy of a file is already registered in FC, no copy operation will be performed.

3.2. Replica Types

Within LDG, by uploading a new ensemble to the grid this data initially is to be seen as "unique" with a need to be saved permanently somewhere and somehow. It is not intended to upload data which will be deleted completely from the grid at some time (beside error-correction). So, the default SRM-type for uploaded date always will be "permanent". This situation changes, if multiple replicas of that ensemble exist: after an ensemble is replicated to another SE', the configurations belonging to that data-set on SE' *might* be deleted at some time (partially or total).
It might also happen, that it is decided to keep the replicated data but to delete the prior uploaded data or it might be the case that both (or even more) replicas should be kept forever. Therefore, the default file storage type for replicated data also is "permanent", so the SE is not allowed to delete it autonomously.
Only if a user replicates an ensemble knowing in advance that this replication will be needed only for a certain time, after replicating he might mark this replicum as "volatile" so that the SE can delete that data after a time-stamp has expired.

3.3. Ltool-Commands for Replica Usage

3.3.1. Creating Replicas : lrep

syntax:

lrep <*ensembleURI*> [<*SE*> | <*SURL*>]

If the storage-element <*SE*> is provided, this has to be done in the format that is also used in $LROOT/etc/ltools.rsc, that is, at the moment:
dcache.zib.de, grid-se3.desy.de, globe-door.ifh.de or dcache.fz-juelich.de .

If "lrep <*ensembleURI*> <*SE*>" is executed, for each configuration $conf_i$ in <*ensembleURI*> the following lcg-command is executed:

lcg-rep --vo $vo -d srm://<*SE*>/<srm_middle_part>/<collaboration>/
<project>/<ensemble>/<Filename> lfn:<LFN>'

where <LFN>' is the logical filename of $conf_i$ converted in FC syntax;
<srm_middle_part> is taken from $LROOT/etc/ltools.rsc for this SE;
<collaboration>, <project>, <ensemble> and <Filename> are taken from the <LFN> of the configuration (<Filename> is the last part of the LFN).

In case of "lrep <*ensembleURI*> <*SURL*>" for each configuration simply the following command is executed:

lcg-rep --vo $vo -d <*SURL*> lfn:<*LFN*>'

Here <SURL> addresses a directory on the destination SE which may automatically be created by the SE. Using this syntax, it is completely left to the user to assure that the destination is correct.

If just "lrep <ensembleURI>" is executed, all files will be replicated to all SE which are supposed to provide a replicum.

If the replication tasks for the whole ensemble were successful, an appropriate message is given to the user and the execution stops. If the replication of a particular configurations fails, it is retried for three times; after three errors the whole operation stops with an error-message to the user (no roll-back concerning the replicated configurations).

3.3.2. Manual Deletion of Replicas : ldelrep

The command to delete a replicum (again possible only at ensemble-level) is

ldelrep <*ensembleURI*> <*SE*>

This will delete configurations of the ensemble <*ensembleURI*> that are stored in the

SE <*SE*> if (and only if):

- at least one other replicum of the configuration exists and
- this replicum is not defined 'master' on ensemble-level and
- there is no lock set in the FC for this ensemble (see below)

Again, this is not an atomic transaction, that is, it might happen that a number of replicated configurations still exist while others were deleted after ldelrep ended with an error message to the user. As lrep, ldelrep will retry deleting a particular replicated configuration three times before giving up. To be able to delete partial ensemble-replicas ldelrep will silently ignore if there is no replicum for a particular configuration but will deliver a success-message only if no replicated configuration is left on the given SE.

3.3.3. Protecting Replicas : ladm --set-mastercopy

To assure no valuable data is deleted, ldelrep will never delete the only replicum of an file. Above this, a replicum can be marked as “master” and then never will be deleted by ldelrep. This is done by an additional option introduced in the ladm – command. It is allowed to mark multiple replicas as “masters”, so none of these could be deleted. The command to mark a replicum as master is

ladm --set-mastercopy <*EnsembleURI*> <*SE*>

This information is centrally stored in the FC (see chapter below). Up to four replicas of an ensemble can be marked as “master”. To revoke this action,

ladm --unset-mastercopy <*EnsembleURI*> <*SE*>

might be launched. Also, if a SE is defined to provide a master copy, this effects the lrep command (see above). When lrep is invoked without further arguments, data is replicated to this SE.

3.3.4. Marking Replicas for Automatic Deletion: ladm --set-volatile

If a replicated ensemble is known to be needed temporary only, it can be marked as “volatile” by executing

ladm --set-volatile <Ensemble> <SE> <ExpirationTime>

Just as ldelrep, this can only be executed if at least one other replicum exists, the ensemble is not marked as “master” and the access is not locked in the FC.
Technically, this will launch an *srmChangeFileStorageType* to change the type to “volatile”for each configuration to inform the SE about the volatile status, which means the SE might delete that data after the time stamp expired autonomously (usage of the “durable”-type is not envisaged at the moment). As a kind of “undo” operation there is also a

ladm --unset-volatile command.

3.4. “Life cycle” of a Replicum

As a summary of the above, the potential “life cycle” of an ensemble stored in a certain SE is shown in figure 1: “life” might start by being uploaded or replicated, status might change between “replicum” and “master” and deletion might only occur in a “replicum-state” (that is, not “master”) and if at least one other replicum exists.

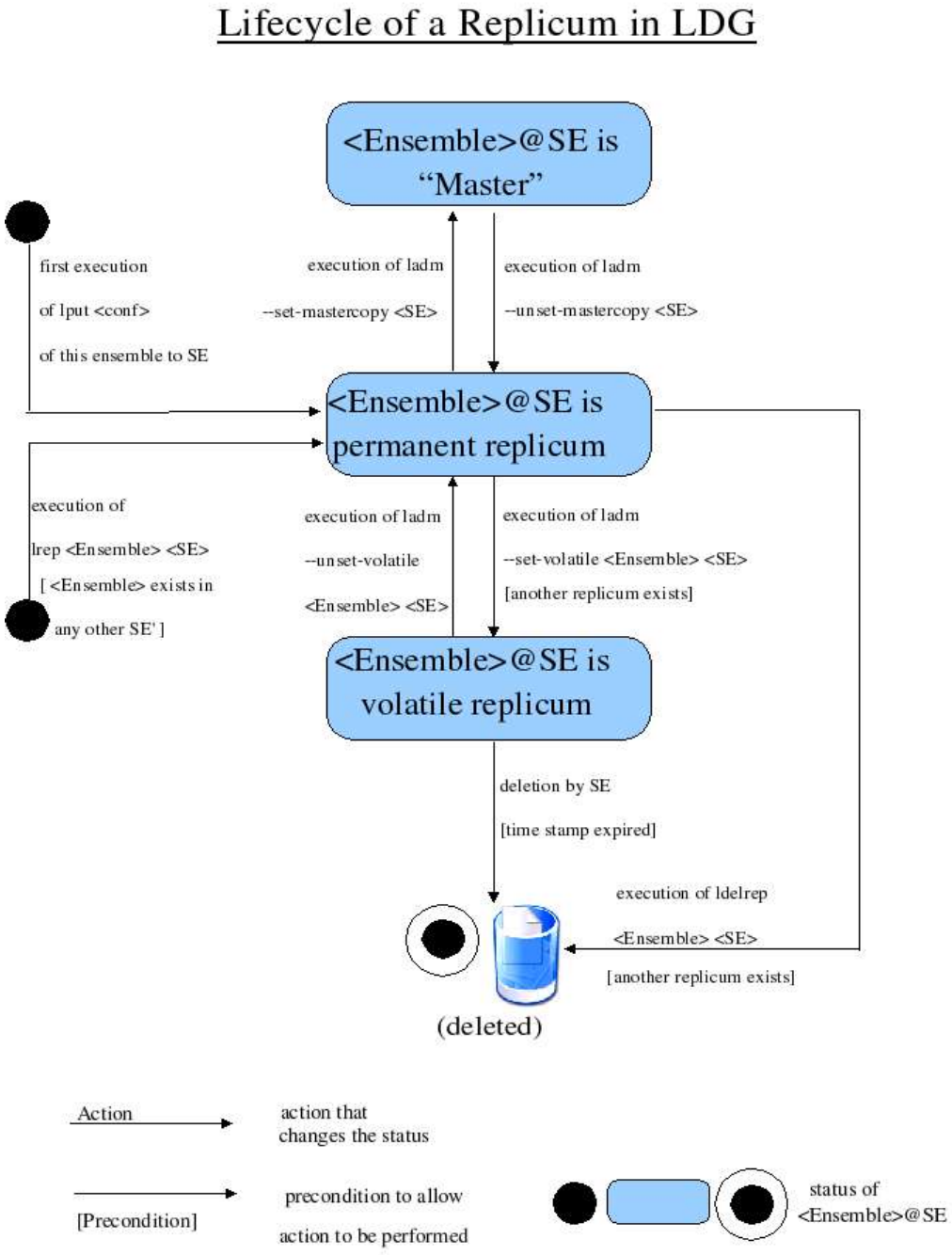

Figure 2 : Lifecycle of a Replicum

3.5. Retention Policies in LDG

All data in LDG except volatile replicas (see above) is seen as "custodial". That is, there is no assumption that any data could be restored from outside the grid if it was lost in the SE.

3.6. Access Latency

Within the LDG, all files are "online" or "nearline", that is, it is accessible directly from disk or a staging operation has to be done before the actual transfer begins, e.g. copying from a tape to a disk-cache. There are no "offline"-files where human intervention is required. If different replications exist, for a download-request of course an online "version" should be preferred to a nearline copy. Two possibilities to make this information available come to mind: this information could be published in the central FC or all SEs holding a replicum could be asked prior to the actual transfer.
The first alternative increases load to the FC and requires mechanisms to achieve consistency of the FC and the SEs regarding the file status.
The second one obviously increases communication-overhead, maybe even leading to the need of waiting for timeouts if a SE is down, but naturally delivers the true status of the file which is known only by the SE definitely. As ensembles often exceed hundreds of GB in total, at the moment it seems not too likely that complete ensembles are online in one SE[2], so the on- or nearline status cannot be stored at ensemble-level. Though replication is intended to be done at ensemble-level, downloading is done configuration per configuration, so there is no reason why different configurations should not be downloaded from different SEs, if this improves performance. Storing the status-information for each configuration in the FC and keeping this information consistent with the real status within the SEs seems to be quite a hard task, so at the moment it is proposed not to store this information centrally, but to extend the download-operation lget to ask the SEs which hold replicas about their status. Launching this questions could be done in parallel and actual download could begin as soon as the information about an online-replicum arrives (or the information that there is none in the grid). This would reduce the latency resulting from the communication-overhead.

3.7. Planning of Up-Downloads

Within the SRM-specification, two commands are foreseen to inform an SE in advance before the actual upload/download is executed: *srmPrepareToPut* and *srmPrepareToGet*. In LDG there is no need for a *srmPrepareToPut* at the moment, but *srmPrepareToGet* might be used, because this will initiate the staging mechanism to put a file online. Although this might not be fulfilled in total, requesting a SE to bring as much configurations online as possible would be useful if downloading is planned in the near future. But, as the size of an ensemble may be large compared to the available read pool at the moment "srmPrepareToGet" is used only inside the ltools, e.g. when executing lrep. That is, there is no user-initiated command foreseen

[2] This might change in the future.

to trigger the staging operation[3].

3.8. Storage Classes in LDG

Taking a look at these classes from the LDG point of view, most of the data belongs to "Tape1Disk0"-class. There might be T0-* -class data as well. At the moment, the information to which storage-class an ensemble belongs is not stored explicitly.

3.9. Extended Usage of the FC

3.9.1. Storing Informations About Replicas

Resulting from the concepts described in the previous sections, some information about replicas of ensembles within the LDG needs to be stored centrally.
It was decided to use the FC for this task, as this already serves as a central information repository for the actual location of the data. The comment-field of the ensemble-directory in the FC will be used to hold information about different replicas of that ensemble. Each "mastercopy" will have an entry with the SE where the replicum is stored, "temporary"/"volatile" replica will not have an entry here.
Usage of the comment-field in the FC implies the following restrictions:

- As the length of this field is restricted, only four mastercopies can be defined.
- Also, in case one SE should hold multiple copies of an ensemble, either all or none will be marked as master, as the hostname can be entered only once.
- Finally, the lfc-setcomment command must not be used manually by LDG-users, as this might override stored information about replicas.

3.9.2. Central Locking-Mechanism

Though not too likely in this context, ugly accidents resulting from parallel access on this central information medium might occur. Therefore an exclusive lock on ensemble-level is introduced: any ltool-command which deals with the replica-information will set a lock for this ensemble first, which is removed only after successful completion of the command. Any other ltool command will refuse to handle this ensemble in this context if such a lock exists. This is realised by creating a subdirectory ./lock in the FC in the ensemble directory, as directory-creation is an atomic operation which excludes any further problems resulting from parallelism.
If a replication task ends unsuccessfully, a lock might still exist and has to be removed somehow.
A stale lock can be deleted by any user that has write rights with respect to the ensemble.

[3] This also might change in the future.

The command to unlock an ensemble is

ladm --remove-stale-lock <Ensemble> .

Any ltool command that refuses to handle a replication task because of a lock will print information about who set the lock, from which machine and the date when it was done to allow error-tracing (technically this information is saved as a lfc-comment to the “lock”-directory).

4. Survey of concepts

As a kind of summary, the following table presents SRM-specifications or LCG proposals with the corresponding aspired usage in the LDG.

SRM-technology	***usage/concretion in LDG***
File Storage Types:	
volatile	Only possible with additional replicas, that is, configurations of an ensemble that are in an SE that is not labeled as “master” and to whom at least one other replicum exists might be volatile. A configuration can become “volatile” only if the ensemble is manually set to that type by launching ladm --set-volatile.
durable	--
permanent	Default type for all data.
Retention Policies:	
replica	--
output	--
custodial	All data in LDG is seen as “custodial” (except volatile replicas).
Access Latency Mode:	
online	Data that is in the SEs's disk cache.
nearline	All data is at least nearline (on tape in the SEs).
offline	--
srmPrepareToPut and srmPrepareToGet:	
srmPrepareToPut	--
srmPrepareToGet	Used from within ltool-commands to increase performance; no explicit end-user command foreseen.
Storage Classes:	
Tape0Disk1 .. TapeNDiskN	Not used explicitly as a conceptual qualifier. Tape1Disk0, TapeNDisk0 and Tape0DiskN occurs.

Figure 3 : SRM/LCG-specifications and LDG usage

5. References:

[SRM 07] http://sdm.lbl.gov/srm-wg/doc/SRM.v2.2.pdf

[WLCG 06] WLCG Data Management Coordination Group, "SRM Storage and File Types" (version 4), May 2006 (https://srm.fnal.gov/twiki/pub/WorkshopsAndConferences/SRMV2Changes/SRMStorageandFileTypes-v4.pdf)