LHCb experience with LFC replication

Federico Bonifazi[1], Angelo Carbone[1], Eva Dafonte Perez[3], Antimo D'Apice[1], Luca dell'Agnello[1], Dirk Duellmann[3], Maria Girone[3], Giuseppe Lo Re[1], Barbara Martelli[1], Gianluca Peco[2], Pier Paolo Ricci[1], Vladimir Sapunenko[1], Vincenzo Vagnoni[2], Dejan Vitlacil[1]

[1] INFN-CNAF v. Berti Pichat 6/2, 40127 Bologna, Italy.
[2] INFN Sezione di Bologna, v. Irnerio 46, 40126 Bologna, Italy.
[3] CERN, Geneva Switzerland

Abstract. Database replication is a key topic in the framework of the LHC Computing Grid to allow processing of data in a distributed environment. In particular, the LHCb computing model relies on the LHC File Catalog, i.e. a database which stores information about files spread across the GRID, their logical names and the physical locations of all the replicas. The LHCb computing model requires the LFC to be replicated at Tier-1s. The LCG 3D project deals with the database replication issue and provides a replication service based on Oracle Streams technology. This paper describes the deployment of the LHC File Catalog replication to the INFN National Center for Telematics and Informatics (CNAF) and to other LHCb Tier-1 sites. We performed stress tests designed to evaluate any delay in the propagation of the streams and the scalability of the system. The tests show the robustness of the replica implementation with performance going much beyond the LHCb requirements.

1. Introduction

In the framework of the High Energy Physics computing, databases are commonly used for a variety of purposes, notably including the storage and access of detector configuration information and data taking conditions, and the book-keeping of the content of data files. The usage of resources spread over the Grid requires discovering the location of physical data files, i.e. starting from a so-called Logical File Name (LFN) one needs to convert it to a Physical File Name (PFN), hence allowing the applications to address one or more storage resources hosting the single file or a dataset of files. This task is accomplished by means of file catalog services, e.g. the LHC File Catalog (LFC) [3]. The file catalog service is basically realized by a front-end interface authenticating the user and processing the specific request, and a backend database hosting the catalog data.

For high performance as well as for fault tolerance purposes, the replication of the file catalog service at different sites, hence in particular the replication of the backend database, is a key topic in the LHC Computing Grid environment. In particular, the computing model of the LHCb experiment foresees a LFC service to be replicated at each LHCb Tier-1 computing centre. The LCG 3D project is responsible for the database replication, and implements it by means of the Oracle Streams technology.

In the following sections we will first introduce some basic concepts concerning the LFC service, how the replication is actually realized and what is the usage of the LFC by LHCb. Then we will discuss the deployment of the LFC replication to the various computing sites involved, and finally

we will present the results of a series of stress tests performed to understand the efficiency of the database replication and the effective scalability of the system.

2. LCG File Catalog

LHC users and applications which need to locate files on various sites exploit the LFC service. The LFC maintains mappings between Logical File Names (LFNs), Grid Unique IDentifiers (GUIDs) and Storage URLs (SURLs). At present it is the only officially supported catalog in WLCG/EGEE.

The LFC architecture consists of an application frontend and a database backend. The frontend is a multi-threaded daemon written in C. It accepts user connections, performs user authentication, stores/retrieves/deletes entries in the catalog according to user requests and sends the query results back to the client, logging any operation. The database backend stores the catalog entries in a relational structure. Supported backends are MySQL and Oracle, but in the case catalog replication is needed only Oracle is supported.

Clients can access the LFC through a Command Line Interface (CLI) which provides a POSIX-like semantics. An Application Program Interface (API) is also available for allowing the development of specific applications.

The catalog contains a GUID as an identifier for logical files, and stores both logical and physical mappings for the file in the same database. There is a global hierarchical namespace of LFNs mapped to the GUIDs. GUIDs are mapped to the physical locations of file replicas in the physical storage (Storage File Names or SFNs). System attributes of the files (such as creation time, last access time, file size and checksum) are stored as attributes with the LFN. Multiple LFNs per GUID are allowed as symbolic links to the primary LFN. A sketch of the LFC database entries and their relationships is shown in Fig. 1.

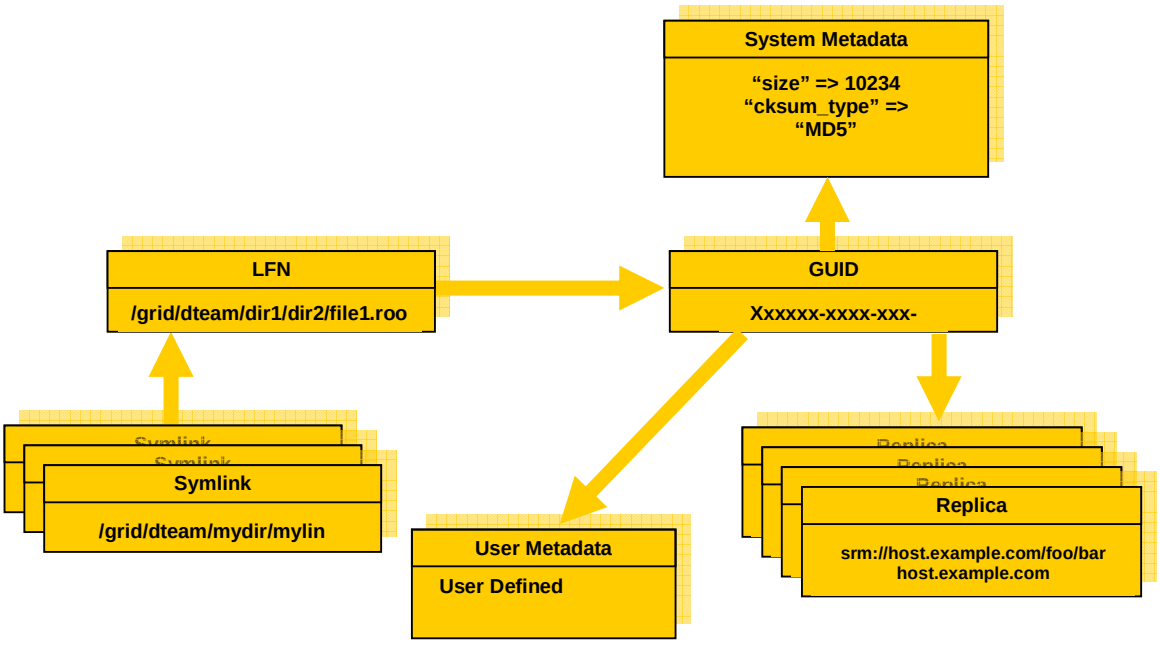


Figure 1: Sketch of the LFC database entries and their relationships.

3. LFC replication

The replication model implemented is a Master-Slave replication based on Oracle Streams. Namely only one LFC is the master catalog and allows read-write access. Other LFC servers can be deployed as replicas of the master one. LFC replicas are read-only catalogs containing all the entries present in

the LFC master. Users who need to write an entry must access the master catalog, the entries are then propagated to the read-only replicas. Users who need to access the catalog in read only mode can contact either the master or a replica LFC. This configuration enables the LFC to address geographical redundancy, high availability and scalability for read-only operations.

The Oracle Streams technology consists in a set of queues and background processes accessing them.

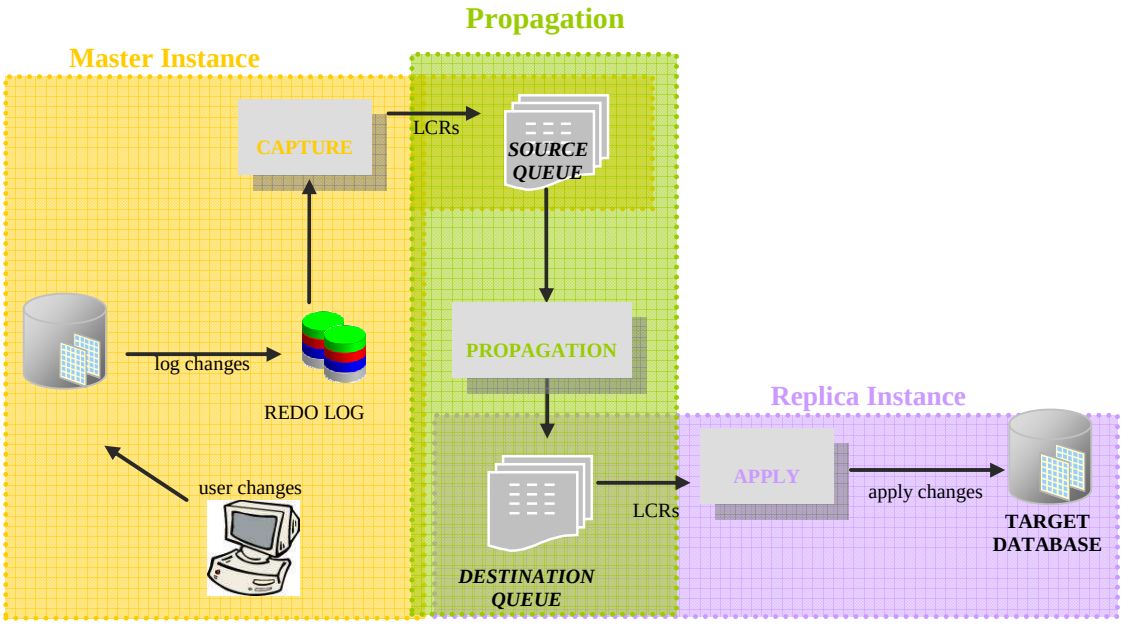


Figure 2: Graphical description of the Oracle Streams replication: processes and queues.

Change events originated by Data Definition Language (DDL) or Data Manipulation Language (DML) operations are propagated from the source database to the destination one in the form of Logical Change Records (LCRs). A LCR is a message with a specific format that describes a database change. Source and destination databases are kept synchronized through Oracle Streams in a three-phase process.

- *Capture* phase: at the source database a capture process reads changes from the *redo logs*, reformats them into LCRs and queues them. If the change was originated by a DML operation, then each LCR encapsulates a row change resulting from the DML operation to a shared table in the source database. If the change was a DDL operation, then the LCR encapsulates the DDL change that was made to a shared database object in the source database.
- *Staging* phase: Streams publishes captured LCRs into a staging area implemented as a queue. A propagation process propagates the staged LCR to another queue, which resides in the destination database.
- *Apply* phase: at the destination database, an apply process dequeues the LCRs and applies changes to the appropriate object.

A sketch of the Streams replication mechanism is depicted in Fig. 2.

Each queue is composed by a buffer in a shared memory and a persistent part, stored in an Oracle Data Dictionary table. LCRs are queued in the buffered part of the queues. Hence they are usually kept in memory and written to disk only when the total memory consumption of buffered messages approaches the available shared memory limit. The event of writing LCRs into disk is referred to as *spilling*.

Since spilling causes a degradation of performances, one of the main goals for Streams administrators is to adequately size the queue buffers in order to prevent spilling. In particular the Oracle instance parameter to be tuned is called STREAMS_POOL_SIZE.

4. LFC usage in the LHCb computing model

CERN is the central production centre and will be responsible for distributing the RAW data in quasi-real time to the Tier-1 centers. CERN will also act as a Tier-1 centre. Other 6 Tier-1 centres are involved for LHCb: INFN-CNAF (Italy), FZK (Germany), IN2P3 (France), NIKHEF (The Netherlands), PIC (Spain) and RAL (United Kingdom). LHCb will also exploit about 14 Tier-2 centres. CERN and the national Tier-1 centres will be responsible for all the production processing phases associated with the real data. The RAW data will be stored at CERN, with another copy distributed across the 6 Tier-1s. A schematic view of the hierarchical tier structure with the tasks of the various centres according to the LHCb computing model is given in Fig. 3.

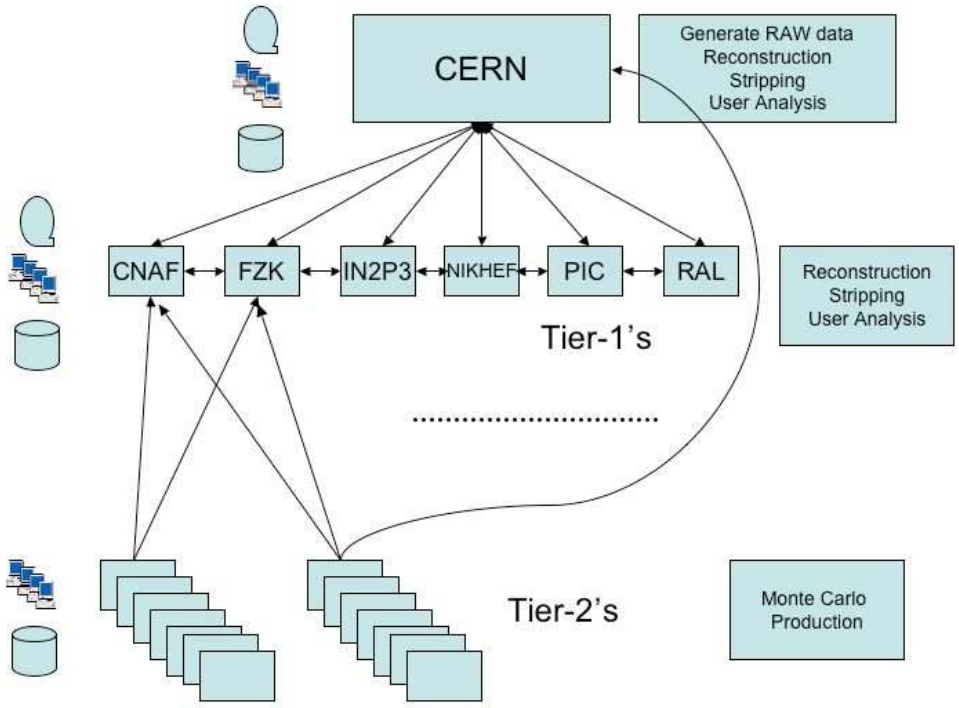


Figure 3: Schematic view of the tier structure of the LHCb Computing Model.

The LHCb computing model foresees a single LFC central catalog hosted at CERN and various read-only replicas located at each LHCb Tier-1. The LFC is used as a central catalog and stores informations about *all* files and file replicas stored at the Tier-0 and Tier-1s.

Each LFC entry is inserted at CERN and automatically replicated at the database backend level at each Tier-1.

The LFC comes into play in different contexts:

- Data processing: send the job to the Tier-1 site where the data are available and produce an output to be registered (**read/write**).
- Data transfer: find the replica to transfer, perform the transfer and register the new destination (**read/write)**.

- Monte Carlo simulation: transfer output from a Monte Carlo job to one or more Storage Elements and register the file in the catalog **(write).**
- Conditions data: the Condition Database is also replicated to Tier-1s. The various Condition Database replicas are inserted in the LFC in the same way as the file replicas. In order to locate the right Condition Database replica, the CORAL middleware needs to ask the LFC for the list of the replicas.

In order to efficiently use the replicated LFC it is mandatory that the master and replica databases are synchronized with low latency. LHCb requirements are not dramatically strict: *less than 30 minutes.*

5. LFC Replica Deployment

The first LHCb LFC replica has been setup in November 2006 at the CNAF Tier-1 and it has been running without relevant problems since then. Its deployment required the joint collaboration of the CERN and CNAF database teams in order to install the Oracle Streams and start the replication.

The process of replicating the LFC schema required a careful study of the implications on the LFC behavior in order to guarantee consistency and functionalities at the application layer. This process can be subdivided in three main phases: database schema replication, LFC front-end configuration and high availability setup.

5.1. Database schema replication

Streams one-way replication requires that no write is performed on the replica because it bases its consistency mechanism on SCN ordering. Each LCR propagated via Streams contains a SCN indicating the timestamp of the action. The LCRs are applied to the destination database in SCN ascending order. Should a write operation be done on a replica database, the SCN is incremented locally, interfering with Streams LCR ordering.

The LFC schema contains 2 tables which need to be updated by the LFC front-end even if the LFC is the replicated one. These tables are:

- CNS_USERINFO which stores information about users accessing the catalog (user id, X.509 certificate distinguish name).
- CNS_GROUPINFO which stores information about user groups (group id, group name).

The first time a user tries to read the catalog (master or replica), it is registered in these tables. The CNS_USERINFO and CNS_GROUPINFO tables are then excluded from the replication. This is achieved through the definition of a negative rule set for the propagation level on the source database with a statement like the following [5]:

```
DBMS_STREAMS_ADM.ADD_TABLE_PROPAGATION_RULES(
table_name              => 'CNS_USERINFO',
streams_name            => 'CAPTURE',
source_queue_name       => 'lfcmaster_queue',
destination_queue_name  => 'lfcreplica_queue',
include_dml             => true,
include_ddl             => true,
include_tagged_lcr      => false,
source_database         => 'lfc_master_service_name',
dml_rule_name           => 'my_dml_rulename',
ddl_rule_name           => 'my_ddl_rulename',
inclusion_rule          => false,
and_condition           => NULL);
```

5.2. LFC read-only front-end configuration

In order to guarantee that no write operation is performed on the replica site, the LFC front-end daemon has to be configured as read-only catalog (the default is read-write) [3]. This configuration is quite simple: the LFC administrator needs only to modify `/etc/sysconfig/lfcdaemon` file setting

the parameter `RUN_READONLY="yes".` Any further attempt to write to the catalog will return an error:

```
$ lfc-mkdir /grid/dteam/hello
cannot create /grid/dteam/hello: Read-only file system.
```

5.3. High availability setup

The LFC setup was carefully studied to implement a highly available service both at CERN and Tier-1 sites. In Fig. 4 a graphical description of the setup is shown.

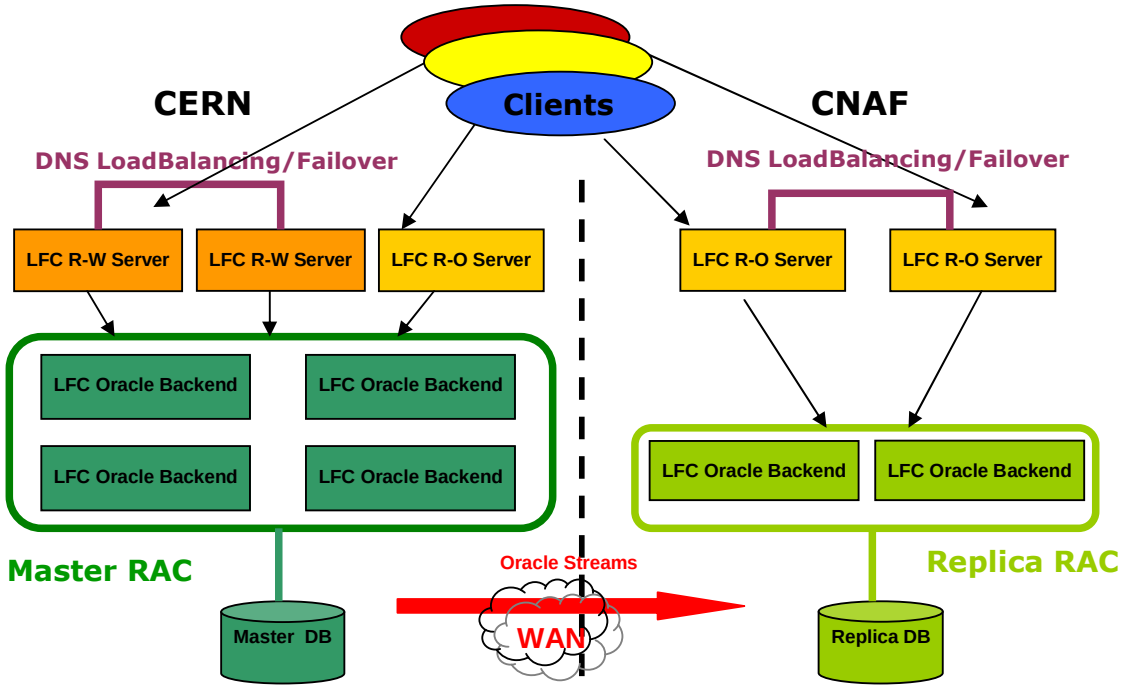


Figure 4: Schematic view of the LFC service realized for the LHCb experiment.

At each site front-end and back-end LFC services are load-balanced and highly-available. The Oracle backend is an Oracle Real Application Cluster (RAC). A virtual service name is associated to a RAC and published to the clients. Client connections to the virtual service are redirected to one of the RAC nodes. The Oracle Listener plays the role of choosing the least loaded node in the cluster, thus implementing a load-balancing policy.

Should a node in the RAC fail, the node is automatically removed from the cluster, and client connections previously established are dropped and reopened on a different node. This operation is performed by the Oracle Call Interface on the client side, an Oracle driver which is distributed as an RPM package and is a prerequisite of the LFC installation.

As far as the LFC front-end is concerned, we exploit DNS load balancing in order to assure a redundant, scalable and reliable service. Again, a virtual service name is configured in the DNS which is resolved to all IPs assigned to the LFC front-end machines. The DNS server is in charge of implementing a round-robin load-balancing policy through all IPs assigned to the virtual service name. If a LFC server becomes unreachable, a monitoring script updates the DNS in order to delete its IP from the virtual service name resolution.

6. LFC replication tests

Tests have been performed in two different scenarios: single replica and multi-replica. We measured replication throughput and latency with the single replica setup and repeated the same tests on the multi-replica setup in order to find out if enhancing the number of replicas would have had an impact performance.

6.1. Monitoring tool

Most of the measurements and plots shown are taken from Strmmon, the official Streams monitoring tool of the LCG 3D project [7]. This tool is designed to access the Oracle data dictionary views and collect all information about latency and throughput in all the phases of the Oracle Streams replication process. It stores the values in a dedicated repository database.

The most interesting metrics taken into account are:

- Total LCR latency: time elapsed between the creation of the LCR in the master DB and the LCR apply operation in the destination database.
- LCR replication rate: number of captured LCRs, queued LCRs, dequeued LCRs and applied LCRs per second.

6.2. Single replica setup: functionality, scalability and stability tests

Two different tests have been realized in order to evaluate the time latency between the master and replicated database and the performance of the LFC front-end with write/delete operations as a function of increasing number of clients.

Some python scripts were developed using LFC API functions `lfc_creatg`, `lfc_unlink`, `lfc_addreplica`, `lfc_delreplica` in order to insert files and file replicas into the master database. In order to test scalability and to simulate different access patterns, the tests were repeated increasing the number of simultaneously writing or deleting clients and changing the number of files and file replicas inserted. The two tests specifically consisted in:

- Test I: insert 8k files and 10 replicas per each file. This access pattern is similar to the LHCb usage. The test was run with 10, 20, 40 and 76 clients.
- Test II: insert 16k files and 25 replicas per each file. This is beyond the LHCb requirements and is run to prove LFC stability and scalability. This test was also run with 10, 20, 40 and 76 clients.

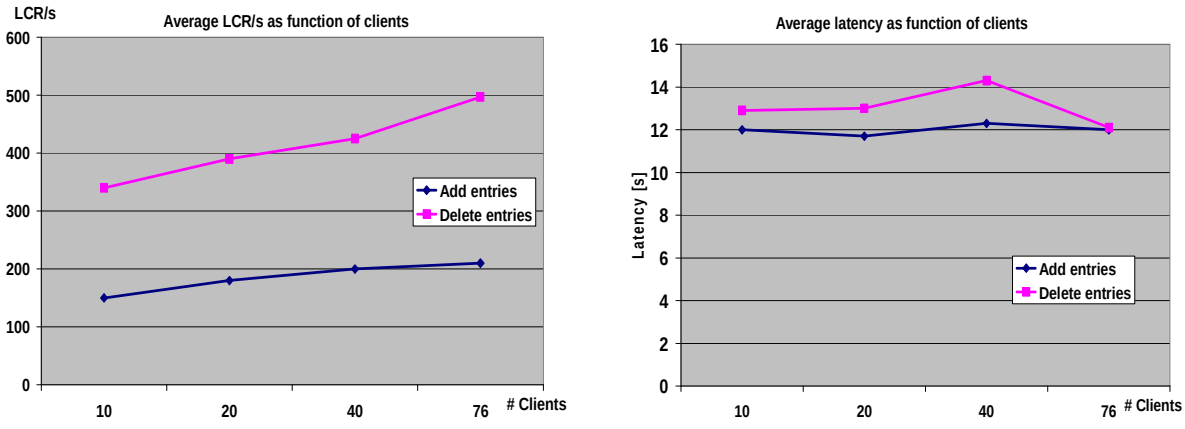


Figure 5: Test I: average LCR/s and latency as function of the number of clients.

As it is shown in Fig. 5, the increase of the number of clients slightly enhances the LCR replication speed for both *add* and *delete* operations. Add operations are quite slower than delete ones because of some overhead inside the LFC code. Due to this reason, *delete* operations can take more advantage from the addition of LFC clients than the *add* ones. On the other hand, latency is pretty much constant for both add and delete operations.

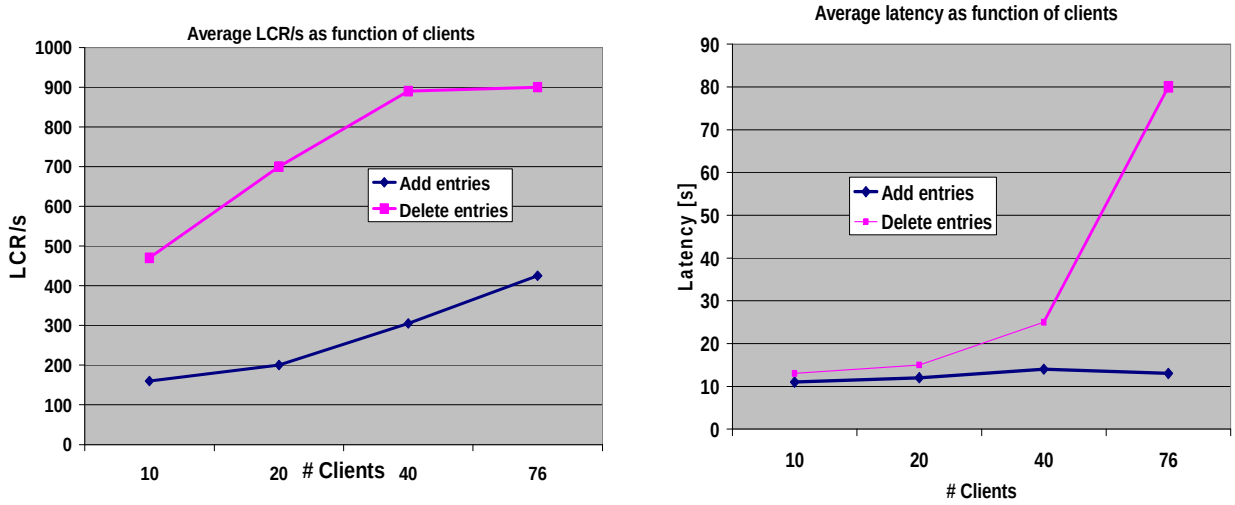


Figure 6: Test II: average LCR/s and latency as function of the number of clients.

In Fig. 6 the results of Test II are shown. Replication speed is greatly improved by increasing the number of files and file replicas added and deleted in each run. There is a linear increase of the LCR rate for both *add* and *delete* operations, but with about 40 clients the replication speed saturates at 900 LCR/s. Since from Strmmon we did not detect spilling nor lengthening of the Streams queues, we concluded that the LFC front-end became a bottleneck and prevented clients to queue new requests. On the latency side, the LFC front-end limit impacts the replication latency too, but if we examine the results for less than 40 clients, latency is quite stable. Anyway, even with 76 clients, a latency of 80 seconds is still much better than LHCb requirements (1800 seconds).

6.3. Multi-replica setup: functionality, scalability and stability tests

At present, 4 Tier-1s in addition to CNAF have added their LFC backend to the 3D replicated environment. They are RAL, IN2P3, GridKa and PIC. Fig. 7 shows the present LHCb LFC replication status as it appears from the Strmmon [7] monitoring tool.

The test suite previously used with the single replica setup has been run with the multi-replica too in order to understand if the setup scales with more than one LFC back-end replica. We have run the previous tests on the new environment inserting and deleting entries in the LFC front-end at CERN and subsequently monitoring the replication speed, latency and synchronization at each Tier-1.

While tests with the single replica were performed reading the entries from the LFC front-end at CNAF Tier-1, in this case we needed to *read directly from the database back-ends* because the LFC front-ends were not yet deployed to all the sites.

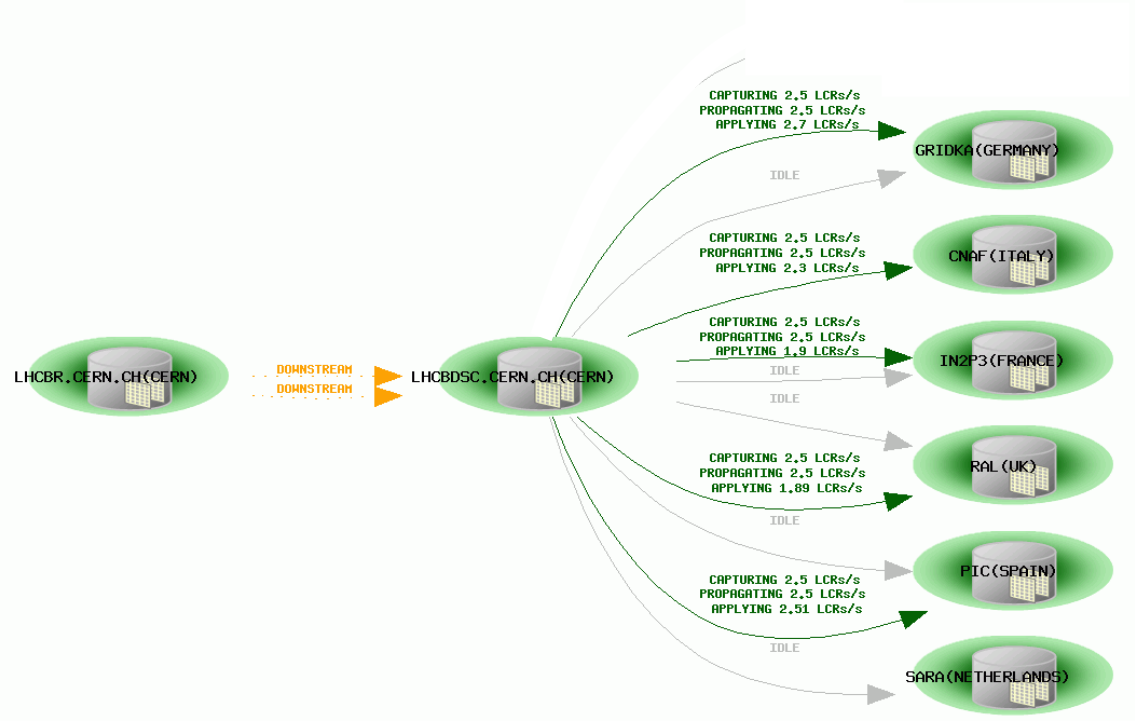
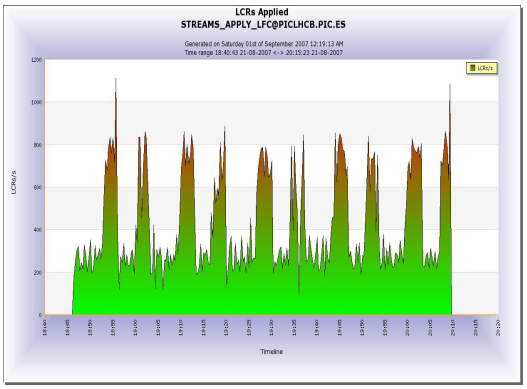


Figure 7: LHCb LFC database Tier-0 → Tier-1s replication.

Test I and Test II were ran again in the new environment. Replication speed is the same as in the single replica setup. All Tier-1s are monitored and all plots are similar in shape. An example is shown in Fig. 8a, in which the replication speed at PIC during Test II is depicted. Peaks and valleys are due to Streams queues filling up and emptying. LCR speed varies between 300 and 1000 LCR/s.



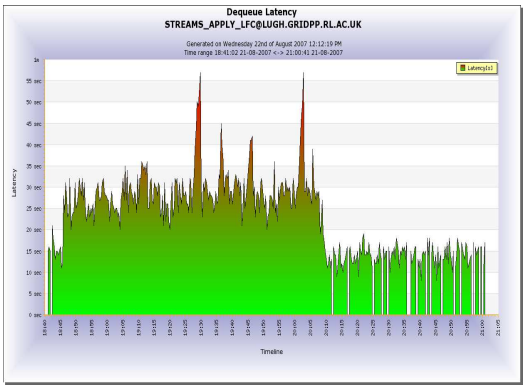


Figure 8a: Replication speed versus time at PIC. **Figure 8b:** Replication latency versus time at RAL.

During the longest test, the replication rate was sustained for 1 hour and half and no spilling was detected at any Tier-1. This means that Streams replication is not a bottleneck for the LFC. LCR latency is still pretty stable at about 20 seconds with peaks of 55 seconds as shown in Fig. 8b.

7. Conclusions and future work

High Availability is a key issue for database services and is well addressed by present Oracle technologies. The LCG 3D project has successfully been deployed with such technologies achieving good stability and reliability of the replications, first at CNAF as a pilot site, now at all the other Tier-1 centres. Adding replicas to the setup did not impact Streams replication performances, i.e. latency did not grow and replication speed did not decrease. Moreover, all the Tier-1s behaved in the same way, all the results about replication speed and latency were pretty much the same. Streams replication is not a bottleneck on LFC performances and the LHCb requirements about latency and performances are largerly met.

References

[1] gLite Users Guide. https://edms.cern.ch/file/722398//gLite-3-UserGuide.html

[2] LHCb Computing Model: LHCb TDR11. http://documents.cern.ch/cgi-bin/setlink?base=lhcc&categ=public&id=lhcc-2005-019

[3] LFC Administration Guide. https://edms.cern.ch/file/579088/1/LFC-Administrator-Guide-1.3.4.pdf

[4] LCG 3D project status and production plans. https://twiki.cern.ch/twiki/bin/viewfile/PSSGroup/TalksAndDocuments?rev=1;filename=3d-chep06.pdf

[5] Oracle Streams Administration Guide. http://download-uk.oracle.com/docs/cd/B14117_01/server.101/b10785/toc.htm

[6] Bind 9 Administrators Reference Manual. http://www.bind9.net/manuals

[7] Strmmon: Streams monitoring tool. https://twiki.cern.ch/twiki/bin/view/PSSGroup/StreamsMonitorHelp