Journal of Physics: Conference Series **219** (2010) 042014 doi:10.1088/1742-6596/219/4/042014

A lightweight high availability strategy for Atlas LCG File Catalogs

Barbara Martelli, Alessandro de Salvo, Daniela Anzellotti, Lorenzo Rinaldi, Alessandro Cavalli, Stefano dal Pra, Luca dell'Agnello, Daniele Gregori, Andrea Prosperini, Pier Paolo Ricci, Vladimir Sapunenko
Viale Berti Pichat
6/2, Bologna I-40127, IT

E-mail: barbara.martelli@cnaf.infn.it

Abstract. The LCG File Catalog is a key component of the LHC Computing Grid middleware [1], as it contains the mapping between Logical File Names and Physical File Names on the Grid. The Atlas computing model foresees multiple local LFC housed in each Tier-1 and Tier-0, containing all information about files stored in the regional cloud. As the local LFC contents are presently not replicated anywhere, this turns out in a dangerous single point of failure for all of the Atlas regional clouds. In order to solve this problem we propose a novel solution for high availability (HA) of Oracle based Grid services, obtained by composing an Oracle Data Guard deployment and a series of application level scripts. This approach has the advantage of being very easy to deploy and maintain, and represents a good candidate solution for all Tier-2s which are usually little centres with little manpower dedicated to service operations. We also present the results of a wide range of functionality and performance tests run on a test-bed having characteristics similar to the ones required for production. The test-bed consists of a failover deployment between the Italian LHC Tier-1 (INFN - CNAF) and an Atlas Tier-2 located at INFN - Roma1. Moreover, we explain how the proposed strategy can be deployed on the present Grid infrastructure, without requiring any change to the middleware and in a way that is totally transparent to end users and applications.

1. Introduction

One of the main missions of the Worldwide LHC Computing Grid (WLCG) software infrastructure is to ensure optimal load balancing and failover between the centres taking part in the collaboration. In spite of this theoretical target, several Grid services like the File Transfer Service (FTS) and the LHC File Catalog (LFC) have been deployed in a high availability setup internally to a particular Tier-1, but are still prone to downtime if all the facilities of the hosting Tier-1 become unavailable. In the case of LFC, some Virtual Organizations (VOs) like LHCb have coped with the problem of the complete Tier-1 downtime studying an high availability (HA) deployment which provides read-only fault tolerance with respect to the whole Tier-1 failures [2]. Anyway, for the ATLAS VO the problem is even more complicated, because the regional LFC needs to be able to failover to the stand-by LFC keeping allowed both the read and the write capabilities. This requirement is motivated by the particular usage and deployment of the ATLAS LFC, which will be described in more detail in the remainder of this document. In this

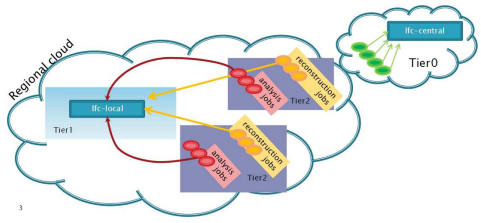

Figure 1. Present ATLAS LFC deployment: geographical view

paper we describe a lightweight HA strategy developed, tested and put in production at the Italian cloud, which consists in replicating the LFC database contents and the LFC front-end setup from INFN-CNAF (the Italian Tier-1 located in Bologna) to INFN-Roma1 (an Italian Atlas Tier-2 located in Rome). This solution is based on the Oracle Data Guard technology, exploited to replicate the database data, and on a set of scripts we developed in order to implement an automatic failover of the LFC front-end, in a way which is completely transparent to all users and applications. In section 2 we explain deeply the problems of the present LFC deployment which motivated this work and we give a bird view of the solution proposed. In section 3 we give an overview of the possible solutions to add HA capabilities to the LFC front-end and to the Oracle Database back-end, motivating our choice. In section 4 we describe our solution: the failover setup at database level and LFC front-end level. In section 5 we present the results of a wide range of functionality and performance tests run in order to validate our solution. In section 6 we draw the conclusions and suggest some possible future developments.

2. Present ATLAS LFC deployment and issues

Presently, ATLAS is using the LFC as local catalog to obtain the Site URLs (SURL) of the files stored in each cloud [3]. Each Tier1 centre houses an LFC for the cloud it represents, and both analysis and production jobs query the Distributed Data Management system DQ2 [4] to determine the location of the files in the Storage Elements. In this scenario a single LFC instance in a cloud represents a single point of failure, i.e. a show-stopper for all the cloud activities when the Tier-1 centre or the LFC machine are in downtime. Figure 1 shows schematically the situation of the Italian cloud, which is similar to the one of all other ATLAS regional clouds. Figure 2 shows a zoom of the present LFC setup inside CNAF where the deployment is in full HA: the back-end database is an Oracle Real Application Cluster (RAC) [5] database composed by 2 nodes.

An Oracle service is configured with both instances in *active* status, and Transparent Application Failover (TAF) policy enabled in *basic* mode. The application front-end consists in 2 LFC servers completely independent one another and connected to the previously mentioned Oracle RAC service. The *round-robin* DNS capability of BIND 9 [6] is exploited to give a unique entry point to users who want to connect to the LFC. In fact, a unique name *lfc.cr.cnaf.infn.it* is published in the BDII. Such name is resolved to two different IPs, which are the ones identifying the front-end LFC servers. A Nagios [7] script is used to monitor the status of both front-ends: should a machine become unavailable, the scripts updates the DNS via *nsupdate* in order to remove the failed IP [8].

Each machine involved in this setup (both front-end and back-end ones) is located in a different rack inside the data centre room, in a way that no hardware device is shared between any of the servers; thus no single point of failure is present inside the computing room. The high availability inside the Tier-1 computing room is therefore full, but the setup is no resilient

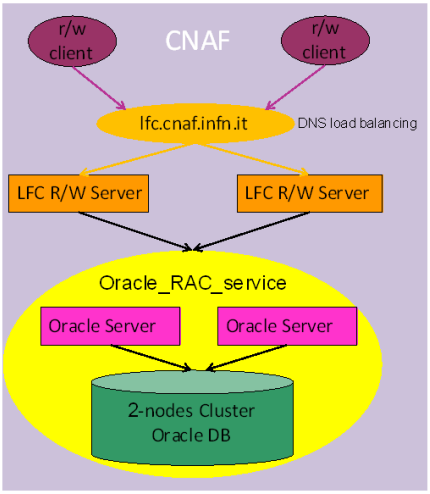

Figure 2. Present ATLAS LFC setup at CNAF

to fault which involve the whole data centre. This is due to three unresolved issues:

- the LFC database contents are not replicated anyware
- the local LFC service housed at INFN-CNAF is not replicated any ware
- the primary DNS is housed at INFN-CNAF, thus is subject to failures if CNAF is in trouble, making impossible any update

In the following section we'll explore the possible solutions to this issues.

3. Possible HA options

Today the state of the art in HA technologies is quite advanced and is possible to find many solutions on both free software and legacy software sides. We have investigated several technologies and in the next paragraphs we describe some solutions, which could be the bricks for a complete LFC HA deployment.

3.1. HA for Database

By now almost all main Relational Data Base Management Systems (RDBMS) offer a multiplicity of solutions to address the database HA problem. The LFC software supports both Oracle and MySQL RDBMSs, but we concentrated on the Oracle one, as nearly all Tier-1 LFCs sit on Oracle and the Tier-1s which are still running on MySQL are planning to migrate to Oracle as well. In the next paragraphs we describe the main solution for Oracle database replication/HA and give a critical discussion on their pros and cons, explaining the reasons which brought us to the choice of Data Guard.

3.1.1. Oracle replication/HA options Oracle replication/HA solutions can be classified into 4 composite groups:

- Real Application Cluster (RAC) [5]: RAC exploits the redundancy provided by clustering to deliver availability with n-1 node failures in an n-node cluster. All users have access to all data as long as there is one available node in the cluster.

- Data replication via Materialized Views [9]: A materialized view is a database object which contains a complete or partial copy of a target master object from a single point in time, typically the target master is a table at a master site. A materialized view can be *read-only, updatable* or *writeable*.
- Data replication via Streams [10]: is the process of sharing database objects and data from a *source* database to one or more *destination* databases. A change at the source DB is propagated to the destination ones through a three steps process: textitcapture, propagation, apply.
- Data Guard [11]: An Oracle Data Guard configuration is a collection of loosely connected systems, consisting of a single primary database and up to nine standby databases. The databases in a Data Guard configuration can be connected by a LAN in the same data center or can be geographically dispersed over a WAN and connected by Oracle Net Services. A Data Guard configuration is transparent to applications.

3.1.2. Discussion The most interesting solutions in our environment are Streams replication and Data Guard, because Oracle RAC is not feasible in our environment due to the fact that the nodes are connected via WAN and materialized views don't provide automatic failover. The Streams peculiarity is the sophisticated buffered-queue infrastructure, which turns into a very reliable replication system: if any of the components is overloaded, and the replication is not able to keep abreast of the updates, the buffered queues can mitigate the replication rate until the peak is ended. Another important feature of Streams is the possibility to add filtering rules along the replication path: you can selectively replicate data to different sites. On the other side, Data Guard is particularly suited to disaster recovery scenarios. The focus in this technology is to be able to failover automatically in case of problems at the primary site. In fact Data Guard can detect automatically a fault at the primary database and automatically failover to the standby database without any human intervention. This can be achieved with a particular configuration which requires the installation of a Data Guard Broker and an Observer process in charge of the failure detection. Unlike Streams, Data Guard doesn't provide filter rules or buffered queues. In the final analysis Streams is particularly targeted to replication systems in which all components (masters and replicas) are thought to be up and available to users at the same time, and it's very advantageous if you need to filter data that you want to replicate, or you need some flow control over the replication process. On the contrary Data Guard is focused on automatic failover: it has a less sophisticated replication mechanism, but it provides a way to autonomously detect a failure and to failover towards the standby site. Thank to its features, Streams replication has been successfully deployed by the LCG 3D group [12] in an analogous situation: the replication of the LFC for the LHCb experiment [2]. In that case, the catalog was a central catalog housed at CERN, and the aim was to replicate the whole catalog to read-only LFCs at Tier-1s. If the central, read-write catalog at Tier-0 fails, as it is completely replicated at each Tier-1, most LHCb activities can continue with a read-only Tier-1 LFC. If a Tier-1 LFC fails, LHCb can simply read from another catalog, as all the LFCs contain the same data; the failover for read operations is implemented at application level by the DIRAC software, which tries all LFCs in a round-robin algorithm. Anyway the ATLAS use case is completely different, because the Tier-1 catalogs contain only the data partition relative to files stored at that regional cloud, therefore all Tier-1 and Tier-0 LFCs are different and constitute single point of failure for both read and write operations. In this case, we need to be sure that every Tier-0/Tier-1 LFC is always available in read/write mode, that is the focus in this case is to automatically detect a fault and to failover to another catalog (housed at a remote site)in read-write mode. All these actions must be triggered and undertaken by a software, without any human intervention. Unlike Streams, Data Guard provides such functionalities, still ensuring reliable data replication. We therefore chose to try-out the Data Guard technology. The other advantage of Data Guard

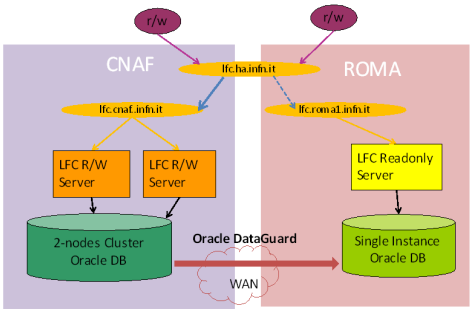

Figure 3. HA deployment for ATLAS LFC

is that it is a lightweight replication strategy, in a sense that it requires very low administration effort compared to Streams, being a good choice for Tier-2s where usually there isn't any Oracle specialized personnel.

3.2. HA for the LFC front-end

The LFC software is provided with HA capabilities developed thinking at the LHCb use case, which can be useful also for the ATLAS one. The software foresees the possibility to configure multiple front-ends against the same DB back-end; moreover the LFC front-end contains the configuration variable

`RUN_READONLY`

which can force the catalog to be a read-only instance. User attempts to write entries will return an error and no data will be written into the database. It's possible to change a read-only catalog into a read/write one or vice versa, by simply changing the value of

`RUN_READONLY`

and restarting the LFC service (`service lfcdaemon restart`). This capability can be exploited to create a hierarchy of catalogs with a read/write catalog as root and one or more read-only replicas. In order to provide load balancing and failover functionalities to GRID jobs accessing the LFC, some features of BIND 9 [6] can be exploited like *dynamic update* and *load balancing* Dynamic Update is a method for adding, replacing or deleting records in a master server by sending to it a special form of DNS messages. The format and meaning of these messages is specified in RFC 2136 [13]. Dynamic update can be enabled on a zone-by-zone basis, therefore inside a domain can be created a dynamic subdomain, restricting dynamic updates only to that one. Load balancing can be achieved in the DNS by using multiple A records for one name. When a resolver queries for these records, BIND will rotate them and answer to the query with the records in a different order. Clients will use the first record returned and will discard the rest.

4. Chosen strategy

In this section we describe the complete HA system for the ATLAS LFC (HA LFC), built using the previously described technologies. The system is structured in 2 layers: database, and LFC front-end Figure 3. The HA LFC consists of a master LFC located at CNAF and a slave LFC located at Roma. The master catalog runs in read/write mode and it's deployment involves 4 servers: 2 of them are part of a 2-nodes Oracle RAC and the other 2 are LFC front-ends where the LFC daemon runs. The slave catalog runs at Roma on 2 servers: one of them houses the

Oracle back-end while the other runs the LFC daemon in read-only mode. Oracle Data Guard is installed in order to replicate the database contents from CNAF to Roma. In the reminder of this section we describe in deeper detail the deployment at both layers.

4.1. Database HA

At database level, at CNAF there is an Oracle 2-nodes RAC, publishing a RAC service with *basic* TAF policy, one node *active* and one node *available*. This means that normally connections coming form the LFC front-end nodes will be routed only to one node (the *active* one) and only in case of a failure on the *active* node, the *available* node will be turn into *active* state and connections will be diverted to it. Because the TAF policy is *basic*, LFC front-end connections to a failed node are re-created to the available node only after the failure. SELECT statements are automatically recovered, while transactions are automatically rolled back and restarted from the beginning. The CNAF Oracle RAC is connected via Data Guard to a single instance database in Roma, in order to be able to tolerate failures of the whole cluster by means of an automatic failover to Roma. The Data Guard configuration chosen is *logical standby* in *maximum availability* protection mode. We also decided to exploit Oracle Data Guard Broker which automates complex creation and maintenance tasks and provides dramatically enhanced monitoring, alert, and control mechanisms. It uses background agent processes that are integrated with the Oracle database server and associated with each Data Guard site to provide a unified monitoring and management infrastructure for an entire Data Guard configuration. Two user interfaces are provided to interact with the Data Guard configuration, a command-line interface (DGMGRL) and a graphical user interface called Data Guard Manager. Oracle Data Guard Manager is integrated with Oracle Enterprise Manager, which is already in use at CNAF as monitoring and managing tool and it is the official Oracle monitoring tool for all WLCG sites. Data Guard Manager provides wizards to help you easily create, manage, and monitor the configuration. This integration lets you take advantage of other Enterprise Manager features, such as the event service for alerts, the discovery service for easier setup, and the job service to ease maintenance. We found it very useful in setting up the software and essential in order to proactively monitor the system. Finally, in order to automate the failure detection, we enabled *fast-start failover* and installed the *observer*. *Fast-start failover* allows the broker to automatically fail over to a previously chosen, synchronized standby database in the event of loss of connectivity with the primary database. Fast-start failover quickly and reliably fails over the target standby database to the primary database role, without requiring the database administrator to perform any manual step to invoke the action. The observer is a separate Oracle Call Interface (OCI) client-side component which runs on a different computer from the primary database and monitors its availability. Once the observer is enabled, no further user interaction is required. If both the observer and the standby database lose connectivity to the primary database, the observer waits for the amount of time specified by the FastStartFailoverThreshold property before initiating a fast-start failover. Moreover, after the failover completes, the former primary database is automatically reinstated as a standby database in the new broker configuration when a connection to it is re-established.

4.2. LFC front-end HA

Unlike database faults, the failover operation at LFC front-end level is not triggered automatically by any given software. In fact, even if multiple LFC servers can be installed in front of the same database, they are not aware one of the other. We need to allow both intra-site and inter-site failover: we talk about intra-site failover if only one piece of software fails in a local redundant deployment: for example if one database node fails in a multi-node RAC, or an LFC server fails in a multiple front-end deployment. We talk about inter-site failover if all redundant components of a certain kind fail at a site: all nodes which are part of a multiple

node RAC or all LFC servers which sit in front of the same database and there is the need to fail over towards a different site. We therefore have developed an automatic LFC failover script as a nagios plug-in, which monitors the status of the front-ends. In order to manage both intra-site and inter-site failovers, we have deployed 2 nagios servers and 2 DNS servers. Two nagios servers accomplish the task of monitoring both CNAF and Roma LFCs, one of them is installed at CNAF and the other one in Roma. They monitor both sites concurrently. This is possible because actions taken by the script are idempotent, that is, if both nagios detect the same fault and both take some action, the result doesn't change. In this way, should the entire CNAF site fail (nagios server included), Roma nagios server can detect the event and take the necessary actions. Two DNS servers are needed in order to allow the service failover from CNAF to Roma. One of them, housed at GARR [14], is *primary master* of the domain `ha.infn.it`, while the other housed at CNAF, is the secondary. The `ha.infn.it` domain has been created expressly for this work, and in the future could be used to implement inter-site failover between any couple of INFN sites; it contains global CNAMES for the services at different sites, which also have local names. For example: the published LFC service name belongs to the `ha.infn.it` domain (i.e `lfc.ha.infn.it`) and is a CNAME for a local site LFC service name. Normally, it will point to a CNAF service name (i.e `lfc.cr.cnaf.infn.it`), but should it fail, it will be updated to point to the standby LFC in Roma (`lfc.roma1.infn.it`). Having this global DNS housed at GARR protects against possible CNAF failures involving the local DNS. In case of GARR DNS failure, only the write operations on the DNS will be impossible, while reads will be still available through the secondary DNS housed at CNAF. Now we describe the details of the nagios script we developed. It monitors all local and remote LFC front-ends for the following events:

- E1 Certificate Validity: the test fails if the X.509 server certificate is expired;
- E2 Read Operations: the test fails if the command `lfc-ls /grid/atlas` returns any error;
- E3 Write Operations: the test fails if the command `lfc-mkdir /grid/atlas/testdir` returns any error.
- E4 Database Role Reversal: the test launches an SQL command against the standby database. It fails if the original *standby* database is found in state *primary*, advising that a role reversal has happened. As the Data Guard has been proven to be a reliable tool, we can assume that a role reversal happens if and only if the entire RAC suffers of a major problem (data corruption or whole cluster failure).

The nagios script returns a "CRITICAL" status if any of the above test fails for a given LFC front-end. In that case different actions are taken, depending on the number and type of failure.

- A1 Failure of one CNAF LFC front-end out of two: one or more of nagios checks number E1, E2 or E3 is CRITICAL on one LFC front-end, while the other is OK. In this case a local DNS failover is triggered. The nagios sensor executes an `nsupdate` operation and deletes the failed IP from the LFC service name.
- A2 Failure of the entire CNAF LFC Oracle RAC (regardless of the status of the LFC front-ends): check number E4 is CRITICAL. In this case the failover operation at database level has already been triggered automatically by the Data Guard software and the status of the standby database in Roma has already been switched to *primary*. The nagios script triggers a GARR DNS update via the `nsupdate` command, which deletes the record `lfc.ha.infn.it CNAME lfc.cr.cnaf.infn.it` and adds the record `lfc.ha.infn.it CNAME lfc.roma1.infn.it`.
- A3 Failure of both CNAF LFC front-ends with the Oracle RAC running correctly. Nagios check number E4 is OK, while both LFC front-ends fail at least one of E1, E2 or E3 tests. In this case, in order to failover to Roma the nagios script triggers a Data Guard operation with

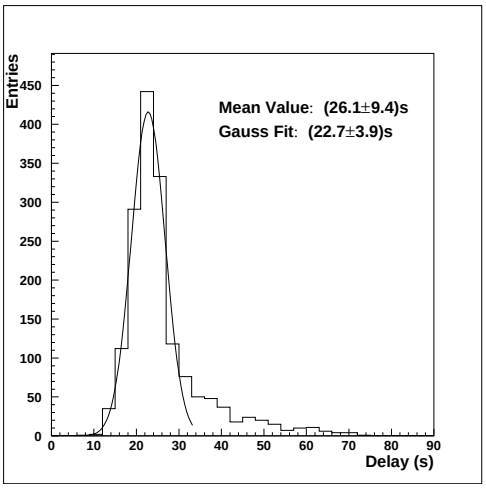

Figure 4. Delay vs time

the `SWITCHOVER` DGMGR command [15]. As soon as the switchover is completed, nagios check number E4 results CRITICAL, triggering the events described at point A2.

Like the database, the LFC front-end is set up enforcing high availability inside CNAF for faults involving only one front-end [8], and high availability outside CNAF for whole site downtimes.

5. Functionality and performance tests

We now provide a detailed description of the functionality and performance tests that we run in order to validate our proposal.

5.1. Test-bed

The testbed is composed by a primary LFC at CNAF and a standby LFC in Roma, in both cases the front-end server sits on a server independent from the hardware housing the back-end database Figure 3. At CNAF the database back-end is installed on a 2-nodes Oracle RAC made of 2 DELL PowerEdge 1950 (2 Intel(R) Xeon(R) 5130 @ 2.00GHz with hyperthreading enabled, 4GB RAM, 2 120GB SATA disks) and 500 GB of shared storage installed on an EMC CX3-80 device and accessed via 4Gbit Fibre Channel redundant connections. Two front-end daemons are installed on two separate machines: Super Micro bi-processors, Intel(R) Xeon(TM) 2.40GHz with hyperthreading enabled, 2 GB RAM, 2 120 GB SATA disks and one Fast Ethernet network interface. In Rome the back-end database is a single instance installed on a Dual quad-core Intel E5410, 16 GB RAM, two 500 GB disks in RAID1, in this case the database is stored on the local disks. The front-end server is installed on a XEN HVM virtual machine with 2 virtual CPUs, a 20 GB HD and 1 GB RAM.

5.2. Functionality tests

This Data Guard setup has two functional purposes: allow database administrators to make interventions on the CNAF LFC which imply a complete shutdown of the whole Oracle RAC, without any service downtime for ATLAS; allow to tolerate incidents having a critical impact at CNAF with a negligible downtime (tens of seconds). In our functionality tests we simulated those events with the intent of trying out the *switchover* and *failover* procedures at both database and application level. In order to exercise the switchover, we started ten jobs which executed

read and write operations such as

```
lfc_getreplica , lfc_addreplica, lfc_delreplica
```

[16] in an infinite loop. In case an error is detected during any of the catalog operations, the job logs the error and continue the loop. During the job execution, we performed the actions defined in the "to-do list" for scheduled operations, which in brief are summarized in the following points:

1 from the Oracle Grid Console push the *switchover* button to start the database switchover process;

2 check that the execution of the application switchover script has been triggered (or run it manually if the monitoring has been previously stopped).

After the switchover, we checked the job log files: errors were logged during the time interval of the switchover operation. As soon as the switchover was completed, the log come back to a clean state. We performed this test ten times, measuring a mean application switchover time of 3 minutes and 4 seconds and a standard deviation of 17 seconds. Afterwards we tested the failover, that is a role reversal triggered by an unscheduled down of the primary RAC. In order to simulate the worst case situation, we physically unplugged all the primary RAC servers simultaneously. We observed the system behaviour: the Oracle Data Guard Observer detected the critical state on the primary database and triggered the failover to the database in Rome. As soon as the failover was completed, the state of the Rome database was changed to **primary**, this fact triggered the application failover script which completed the failover operation. This test has been performed 10 times with a mean failover time of 2 minutes and 56 seconds and a standard deviation of 21 seconds. As in the switchover test, the test jobs logged LFC access errors during the time interval of the failover, and came back to a clean state just after the failover completion.

5.3. Performance tests

The purposes of the performance tests were:

P1 to understand if the installation of the Data Guard infrastructure introduced any relevant delay to applications;

P2 to measure the replication delay accumulated during a massive insertion of entries in the catalog [17].

In order to test P1, we have developed a script which inserted in the LFC 100 k files plus one replica for each file. Inserts were interleaved by a delay of one second. We wrote down the elapsed execution time with the Data Guard replication inactive and active and measured the difference. After 10 runs, the mean delay resulted to be negligible (less then one second). In order to test P2, we have developed a stress test script which launched a number of simultaneous jobs (10, 20, 40) accessing the LFC. The access to the catalogue has been performed simulating a typical ATLAS access pattern:

- open a session
- create a new entry
- add its replica
- close the session

At each run, a total of 1 M of requests were processed, splitted in 40 parallel threads. The whole run took few hours to be completed. We measured the delay accumulated by Data Guard in writing the information in the backup catalogue. The results are shown in Figure 4 The delay time shows a normal-tailed distribution.

The average measured time is 26.1±9.4 s. A fit of the gaussian peak, in which the events of the tail were excluded, gives a mean value of 22.7 ± 3.9s. Since the test was performed concurrently with other processes, the fraction of entries written with a higher delay is not negligible.

This delay can be splitted in two part: the *transport lag* and the *apply lag*. The first one which consists of the time needed to transmit the data on WAN and depends mainly on the round trip time; the second one is function of the standby database speed (CPU, disks, etc...) and gives an hint on the failover time.

6. Conclusions

We have proposed, tested and put in production an high available LFC for ATLAS. The peculiarity of this work is the fact that it is able to manage both *intra-site* and *inter-site* failovers. The solution is based on Oracle Data Guard and a set of monitoring scripts deployed as nagios plug-in. Extensive functionality and performance tests have demonstrated that this approach is robust, completely transparent to clients and doesn't suffer of any relevant performance penalty. As this solutions is built on technologies widely used inside the WLCG 3D project, it can be easily adopted by the others Tier-1s, creating a distributed, homogeneous and highly available ATLAS LFC service world wide.

7. References

[1] gLite Users Guide.*https://edms.cern.ch/file/722398//gLite-3-UserGuide.html*
[2] Bonifazi et al.LHCb experience with LFC replicationJ. Phys.: Conf. Ser. Proceedings of International Conference on Computing in High Energy and Nuclear Physics, Victoria, Canada, pp.042005
[3] ATLAS computing: Technical Design Report
[4] Managing ATLAS data on a petabyte-scale with DQ2Mario Lassnig et al.Journal of Physics: Conference SeriesSeptemberInstitute of Physics PublishingBristol, England
[5] Oracle Database Oracle Clusterware and Oracle Real Application Clusters Installation Guide 10g Release 2 for Linux
`http://download.oracle.com/docs/cd/B19306_01/install.102/b14203/toc.htm`
[6] Bind 9 Administrators Reference Manual.*http://www.bind9.net/manuals*Copyright 2004 Internet Systems Consortium, Inc. ("ISC")Copyright 2000-2003 Internet Software Consortium
[7] http://www.nagios.org/
[8] A Cavalli et al.Geographical failover for the EGEE-WLCG grid collaboration toolsProceedings of International Conference on Computing in High Energy and Nuclear Physics, Victoria, CanadaJournal of Physics: Conference Series 119 0620222008
[9] Oracle Database Advanced Replication 10g Release 2
`http://download.oracle.com/docs/cd/B19306_01/server.102/b14226/toc.htm`
[10] Streams Replication Administrator's Guide 10g Release 2
`http://download.oracle.com/docs/cd/B19306_01/server.102/b14228/toc.htm`
[11] Oracle Data Guard Concepts and Administration 10g Release 2
`http://download.oracle.com/docs/cd/B19306_01/server.102/b14239/toc.htm`
[12] LCG 3D project status and production plans.*http://twiki.cern.ch/twiki/bin/viewfile/PSSGroup/TalksAndDocuments?rev = 1;filename = 3d-chep06.pdf*
[13] IETF RFC 2136*http://rfc-ref.org/RFC-TEXTS/2136/index.html*
[14] *http://www.garr.it/garr-b-home-engl.shtml*
[15] Oracle Data Guard Broker 10g Release 2
`http://download.oracle.com/docs/cd/B19306_01/server.102/b14230/toc.htm`
[16] LFC Administration Guide.
`http://edms.cern.ch/file/579088/1/LFC-Administrator-Guide-1.3.4.pdf`
[17] Performance analysis of a file catalog for the LHC computing gridProceedings of the High Performance Distributed Computing, 2005. HPDC-1414th IEEE International SymposiumPages 91-992005