DNS load balancing in the CERN cloud

Ignacio Reguero Naredo and Lorena Lobato Pardavila
CERN IT Department, 1211 Geneva 23, Switzerland
E-mail: ignacio.reguero@cern.ch, lorena.lobato@cern.ch

Abstract. Load Balancing is one of the technologies enabling deployment of large-scale applications on cloud resources. A DNS Load Balancer Daemon (LBD) has been developed at CERN as a cost-effective way to balance applications accepting DNS timing dynamics and not requiring persistence. It currently serves over 450 load-balanced aliases with two small VMs acting as master and slave. The aliases are mapped to DNS subdomains. These subdomains are managed with DDNS according to a load metric, which is collected from the alias member nodes with SNMP. During the last years, several improvements were brought to the software, for instance: support for IPv6, parallelization of the status requests, implementing the client in Python to allow for multiple aliases with differentiated states on the same machine or support for application state. The configuration of the Load Balancer is currently managed by a Puppet type. It discovers the alias member nodes and gets the alias definitions from the Ermis REST service. The Aiermis self-service GUI for the management of the LB aliases has been produced and is based on the Ermis service above that implements a form of Load Balancing as a Service (LBaaS). The Ermis REST API has authorisation based in Foreman hostgroups. The CERN DNS LBD is Open Software with Apache 2 license.

1. Introduction
A simple model to achieve reliable scalability of computing services in a cloud environment is to provide the service with a (sometimes large) number of automatically configured cattle [1] VMs, that allows users access through an alias mechanism providing load balancing and failover. This model has been extensively used at CERN to migrate most of the computing services to an OpenStack-based private cloud.

Although the need for load balancing increases with the use of cloud computing, the load balancing techniques predate the deployment of clouds. A DNS [2] [3] Load Balancer Daemon (LBD) [4] has been developed at CERN as a cost-effective way to allow service deployments based on large numbers of commodity computer nodes.

The aliases managed by the LBD are DNS subdomains whose names provide access to the resources.

A remarkable feature of the LBD is that it allows application-level feedback and health monitoring by means of a load metric used to decide which IP addresses to present when referencing the alias name.

The LBD uses DNS for load balancing which has two definite advantages:

- It allows balancing the load without modifying the traffic patterns or adding network gateways that may become availability and performance bottlenecks.
- It supports transparently all protocols.
On the other hand, DNS presents two drawbacks:

- The time to switch nodes is limited by the caching time of DNS entries, which is defined by the Time to Live (TTL) parameter. Typically, we have a TTL of 1 minute and a polling interval of 5 minutes, which makes the alias change dynamics of the order of several minutes.
- There is no way to guarantee that all requests from a client go to the same member node so there is no persistence or sticky sessions.

Layer 4 or 7 load balancers such as HAProxy [5] can be used for the cases that need persistence or faster switching time. On the other hand, a large number of applications can use a DNS load balancer. Over 450 aliases are currently running at CERN with the LBD on a couple of nodes (master and slave) hosted on small VMs which is very cost effective. Below, the basics and the evolution of the LBD are described.

1.1. Brief summary of the LBD history
The journey started in the beginning of the 2000s. CERN was running load balancing based on the ISS DNS server implementation where LB client feedback was provided by using RSH to be changed to SSH in 2002. A couple of years later, a new DNS LB system was written using SNMP for feedback, which is considered the base of the current LBD system. Afterwards in 2006, the paper DNS Load Balancing and Failover mechanism at CERN was presented in the CHEP conference.

In 2011, the LBWeb GUI to request alias configuration changes as well as lbclient support for load metric built with Lemon [6] monitoring metrics were added.

In 2013, we had the migration to OpenStack private cloud and the migration to Puppet fabric management.

In 2014, we added IPv6 support, multithreaded SNMP requests and a new lbclient in Python. Since 2015, we have produced the Ermis REST service to manage LB aliases, the Kermis CLI for LBD admins, the Aiermis Self-service GUI to manage LB aliases, as well as the new concurrent LBD written in Go.

The timeline of the milestones above is shown in Figure 1.

![Figure 1. LBD milestones](image-url)
2. LBD basics
The LBD master is an arbiter that periodically collects a load metric from the alias member nodes using SNMP [7] transport to choose the ones whose metric looks best for the LB alias.

The lowest positive values of the load metric are considered the best, on the other hand, negative values of the load metric indicate health check conditions where the node is temporarily excluded.

The use of a load metric is a great advantage relative to other load balancers using round-robin [8] mechanisms where there is no feedback from the member nodes. Without feedback, unresponsive nodes cannot be automatically excluded.

Each LB alias is mapped to a DNS subdomain corresponding to its name that can be updated independently.

The LBD selects a number of nodes among the ones available in the cluster with the best valid metric values, so their IP addresses are presented by the LB alias. This is done by updating the DNS A and AAAA resource records of the DNS subdomain corresponding to the alias using DDNS [9]. The LBD authenticates with a TSIG [10] transaction signature key to the DNS server to do the DDNS updates.

The LBD slave collects the load metric from all nodes like the master, but it will only carry out the DDNS updates when the LBD master is not available. This is verified by trying to access via HTTP a heartbeat file periodically updated by the LBD master, which produces a simple but effective form of high availability mechanism.

![LBD communication mechanisms](image)

**Figure 2.** LBD communication mechanisms

2.1. Alias member nodes and lbclient
2.1.1. Alias member nodes The equally-configured nodes that belong to the same cluster to implement a service are called **alias member nodes**. The IP addresses of these alias member nodes can be presented by the LB alias so that the nodes can be accessed through the LB alias depending on their load and health check state.
2.1.2. The lbclient  The lbclient is a small program that runs in the alias member nodes to obtain a load metric that measures the load and health state of the node. We use a configuration of the NET-SNMP [11] daemon that calls the lbclient when getting an SNMP GET request on a specific MIB OID [12] that uniquely identifies our objects in the SNMP MIB hierarchy. The request is done with an SNMPv3 [13] user and password for security. Therefore, when the LBD server sends an SNMP request to the node on the load balancing MIB OID (.1.3.6.1.4.1.96.255.1), the lbclient is started and it gives back in stdout an integer number that is the load metric value. The NET-SNMP daemon subsequently transmits this value with the SNMP reply to the LBD server.

2.1.3. lbclient evolution  The lbclient was originally written in the C language. We have re-implemented it in Python to facilitate its evolution. We have subsequently extended it to support multiple LB aliases in the same node with differentiated states. In this case the load metric gives back a string with a comma-separated key-value pair list containing the alias names and the values of their respective metrics. The LBD was also extended to handle this polymorphism transparently.

2.2. Load metrics and health checks

2.2.1. Load metrics  While the lbclient offers a built-in load metric (in the spirit of the Linux Load Average) it also offers the possibility of building ad-hoc load metrics by doing a combination of Lemon monitoring metrics and constants. The support for constant metrics allows using the LB alias as a cheap high availability system, where the node with the lowest metric is the master. Its IP address is always displayed unless it is not available, switching the other (slave) node in this case.

2.2.2. Health checks  The lbclient offers a series of built-in health monitoring checks that are configured optionally, such as checking whether /tmp is full or whether /etc/nologin is set. It also allows configuring arbitrary health monitoring checks using conditions on Lemon monitoring metrics defined for the node. We have written a health check for application state so that the node is taken out of the LB alias when the state is not production. This works by querying the REST service of the CERN in-house developed state manager, while using cached values if the REST service is not available to avoid undesired state changes in this case. Finally, we have added support to use health checks based on the return code of arbitrary programs. Of course, these programs have to run within the time window of the SNMP timeout.

2.3. LBD scalability improvements

The LBD has been originally implemented in Perl. It loops around the list of LB aliases so that it evaluates each alias when its polling interval has run out. To be able to respect the timing, the loop with all the aliases has to be executed in time less than the polling interval (typically 300 seconds). This represents a scalability issue that can be addressed either by splitting (sharding) the LB alias list among several servers, or increasing the concurrency of the LBD code. Sharding the LB alias list would increase considerably the complexity of the LBD operations. Therefore, we rather chose to improve the concurrency of the LBD code.

2.3.1. Multithreaded SNMP requests  The LBD has to do an SNMP GET per alias member node for each LB alias. Since some of the aliases can have over 100 alias members and timeouts of 10 seconds are likely, the SNMP requests were dominating the loop time. So as a first step to improve the concurrency of the LBD code, while we still evaluate the aliases serially, we parallelized the SNMP requests for each alias so that the load metric is collected concurrently.
from all alias members using the threads Perl module. This has successfully been in production since 2014.

2.3.2. Concurrent implementation  As the number of aliases continued to grow, other scalability improvements were considered. However, the use of interpreter-based threads is officially discouraged in Perl and most libraries do not support them so we decided to try implementing the LBD in another language. Python was first considered but the limitations of its Global Interpreter Lock discouraged us although we later realized that these limitations would only apply to CPU intensive threads. In any case, the Go language was finally taken because it provides concurrency tools such as goroutines and channels natively. Therefore, a Go implementation has been produced where each alias is evaluated concurrently and the SNMP requests to the member nodes of each alias are also executed concurrently. This has been done using two layers of concurrent goroutines. This Go implementation can evaluate over 450 aliases in 12 seconds. Bearing in mind that the timeout of the LBD SNMP requests is 10 seconds, we are evaluating all aliases in the time that it takes the worst case for a single alias, which means that the parallelism is excellent. This version is currently being prepared for production.

| Application                   | Description                                      |
|-------------------------------|--------------------------------------------------|
| LXPLUS                        | Interactive Linux service                        |
| AIADM                         | Linux administration gateway                     |
| ZENODO                        | Research data repository                         |
| EOS                           | Physics data servers                             |
| CASTOR                        | Physics data servers                             |
| Grid Infrastructure           | VOMS, SRM, CE, FTS servers, etc.                 |
| INDICO                        | Document server                                  |
| TWIKI                         | Wiki web server                                  |
| DASHBMB                       | Message Brokers                                  |
| LICFLEX                       | License servers                                  |
| ITMON                         | Monitoring servers                               |
| METER, TIMBER                 | Kibana servers                                   |
| CDS                           | Document server                                  |
| OPENSTACK                     | Cloud servers                                    |
| GITLAB                        | Git server                                       |
| CMSWEB, CMSDOC                | Experiment web interfaces                        |
| Production infrastructure     | PANDA, RUCIO, FRONTIER, SQUID                    |

2.4. LB alias logs  A Kibana Dashboard to view the LB alias logs has been produced. It takes the LB alias name as an argument and provides the log of the LB alias in question. This allows system managers to view them without requiring access to the LBD server. This dashboard can be accessed through the CERN Service Availability Overview page with a point-and-click interface.
2.5. IPv6
As the CERN network started allowing dual stack IPv4-IPv6 machines, we needed to support the IPv6 AAAA DNS records, together with the traditional IPv4 A DNS records on the DNS zones used for LB aliases. This way, it allows LB aliases to present a mixture of IPv4 and IPv6 addresses.

The support for AAAA records has implied replacing IPv4-only calls such as `gethostbyname()` with calls that work both for IPv4 and IPv6 such as `getaddrinfo()` and `getnameinfo()`. As for transport, the Perl LBD keeps doing the SNMP queries on IPv4 UDP while the Go LBD has been improved to use IPv6 UDP6 for SNMP if the node is displaying any IPv6 address.

3. Ermis and Puppet
With the migration to the Cloud, we started to use the Puppet [14] configuration management system for the LBD. The LB alias member nodes were collected dynamically by means of a PuppetDB query. However, the LB alias parameter information was still stored in a Puppet manifest backed by Git. So manual intervention by privileged administrators was required to define and configure LB aliases. The users were creating support tickets for the operations requested through the Lbweb GUI. As the use of the LBD kept increasing, these operations were becoming a burden for LBD administrators as well as providing an uneven turnaround to the users. Hence, the study of a self-service GUI that could perform these tasks directly was started.

As we were aiming to interface with systems based on service-oriented architectures, such as Puppet, Foreman and OpenStack, we decided to manage the LB alias parameter information with a RESTful [15] web service that was called Ermis (Greek god of transitions and boundaries). Ermis has been implemented using the Django Tastypie library that facilitates building RESTful services. Ermis consumes a SOAP interface provided by the CERN Network Group that allows creation, deletion and modification of DNS subdomains.

The idea is that Ermis interfaces to all services needed to create or update the LB alias configuration such as the SOAP for DNS subdomains and Puppet for the LBD configuration. At the same time, user GUIs or other services can use the API provided by Ermis to manage the LB aliases in an automated way. In this context, the Lbconf Puppet type and its provider have been written to generate the LBD configuration consuming the information from Ermis. On the user side, the following interfaces have been coded:

- The Aiermis user-facing GUI. It uses Django views that access the Ermis API.
- The Kermis CLI. It is a thin layer using the Python Requests module calling the Ermis service.

Therefore, Ermis with the interfaces described above implements a form of load balancing as a Service (LBaaS).

3.1. Ermis authentication and authorization
Ermis supports authentication using Kerberos, SSL and the Shibboleth CERN Single Sign-on. They are used for the following cases:

- Kerberos is used for the Kermis CLI.
- SSL is used by the Lbconf Puppet type with an ACL of host certificates that is filled automatically by Puppet.
- Shibboleth is used by the Aiermis GUI.

To be able to provide a self-service GUI, it is not enough to authenticate. Therefore, the privileges of each user have to be delimited depending of the machines involved and their affiliation. As mentioned before, Foreman hostgroups are used for authorization:
As Foreman is acting as a Puppet external node classifier (ENC) [16], all managed hosts belong to a hostgroup.

When creating an LB alias, a hostgroup must be provided. All alias member nodes have to belong to the same hostgroup as the LB alias. When doing any operation on an LB alias, the user has to be administrator of the corresponding hostgroup. This is checked by the UserAuth class for the Ermis tastypie resource. To do so it queries the CERN in-house developed Teigi REST service per hostgroup and user. Teigi keeps track of which users are managers for each hostgroup.

4. Conclusions
While the LBD has been running successfully in production for more than ten years, it has been continuously growing. More than 450 LB aliases are currently running. New features have been added, such as the support for IPv6. Scalability has been improved with the parallelization of SNMP status queries and ultimately with the Go implementation that evaluates the LB aliases concurrently. The migration to the Cloud has prompted the integration with the Puppet configuration management system. It has also elicited the move to a service-oriented architecture where the management of the alias information is done through a RESTful web service. This service is consumed by Puppet and by the user-facing GUI and CLI, implementing a form of load balancing as a Service (LBaaS).

![Figure 3. LBD Integration with Puppet and the Ermis REST Service](image_url)
References

[1] The history of pets vs cattle http://cloudscaling.com/blog/cloud-computing/the-history-of-pets-vs-cattle/
[2] Domain Name System https://en.wikipedia.org/wiki/Domain_Name_System
[3] Mockapetris P 1987 Domain Names - concepts and facilities RFC https://tools.ietf.org/html/rfc1034
[4] Bahyl V and Garfield N 2006 DNS Load Balancing and Failover Mechanism at CERN 15th International Conference on Computing In High Energy and Nuclear Physics - CHEP 2006, Mumbai, India, 13 - 17 Feb 2006 519-523 https://cds.cern.ch/record/1081742
[5] Haproxy load balancer http://www.haproxy.org/
[6] Babik M, Fedorko I, Hook N, Lansdale T H, Lenkes D, Siket M and Waldron D 2011 LEMON - LHC Era Monitoring for Large-Scale Infrastructures J. Phys.: Conf. Series 331 052025 http://iopscience.iop.org/article/10.1088/1742-6596/331/5/052025/pdf
[7] SNMP https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol
[8] Round-robin scheduling https://en.wikipedia.org/wiki/Round-robin_scheduling
[9] Vixie P, Thomson S, Rekhter Y and Bound J 1997 Dynamic Updates in the Domain Name System (DNS UPDATE) RFC https://tools.ietf.org/html/rfc2136
[10] Vixie P, Gudmundsson O, Eastlake 3rd D and Wellington B 2000 Secret Key Transaction Authentication for DNS (TSIG) RFC https://tools.ietf.org/html/rfc2845
[11] Net-SNMP http://net-snmp.sourceforge.net/
[12] OIDs and MIBs http://www.networkmanagementsoftware.com/snmp-tutorial-part-2-rounding-out-the-basics/
[13] SNMPv3 https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol#Version_3
[14] Puppet https://puppet.com/
[15] Representational state transfer (REST) https://en.wikipedia.org/wiki/Representational_state_transfer
[16] Writing external node classifiers https://docs.puppet.com/guides/external_nodes.html