Achieving metric oriented load balancing

Paulo Canilho[1,*], *Ignacio* Reguero[1,**], and *Pablo* Saiz[1,***]

[1]CERN Esplanade des Particules 1, 1217 Meyrin Switzerland

Abstract. The Load Balance Service at CERN handles more than 500 aliases, distributed over more than 2000 nodes. After being in production for more than fifteen years, it has been going through a major redesign over the last two years. Last year, the server part was reimplemented in the Go Programming Language, taking advantage of its concurrency features to improve the scaling of the system. This year, the client side has been the main focus. This article describes the client side and its configuration, including the tools used at CERN. All the components used by the Load Balance Service are open source, and they can be deployed at other places.

1 Introduction

The aim of this paper is to describe the process of how a service administrator can achieve a metric-based load-balancing system. Section 2 will give an overview of the whole system and its different components. The server part was described in a previous article (see "Concurrent Adaptive Load Balancing at CERN" [1]). This article will focus more on the client side. To achieve this, section 3 will describe the component of the system that decides the healthiness of the nodes, the *lbclient*. It will also show how to configure the *lbclient* using local metrics. After that, section 4 will describe how a user defines an alias. Concluding, future-work will be discussed in section 5 and a final summary will be presented in section 6.

2 DNS load balancing at CERN

CERN runs an in-house developed DNS load balancing solution since the beginning of the 2000s. It is currently serving over 500 aliases running on thousands of nodes. The main feature of this solution is that the alias member nodes provide feedback on their load and health to an arbiter. This arbiter uses the information according to the policies associated with the alias to choose which IP addresses the alias should present and updates the DNS server accordingly. Even if the solution is limited by the slow responsiveness of DNS changes, it is very convenient for applications that can live with this restriction as it supports all protocols, the bulk of the network traffic continues accessing the nodes directly and it requires little maintenance. In the current operating mode, the arbiter, called *Load Balancing Daemon (lbd)*, gets the feedback from the nodes using the SNMP [2] protocol and updates the alias in the DNS server using the DDNS [3] protocol. This is depicted in Figure 1.

[*]e-mail: paulo.canhilo@cern.ch
[**]e-mail: ignacio.reguero@cern.ch
[***]e-mail: pablo.saiz@cern.ch

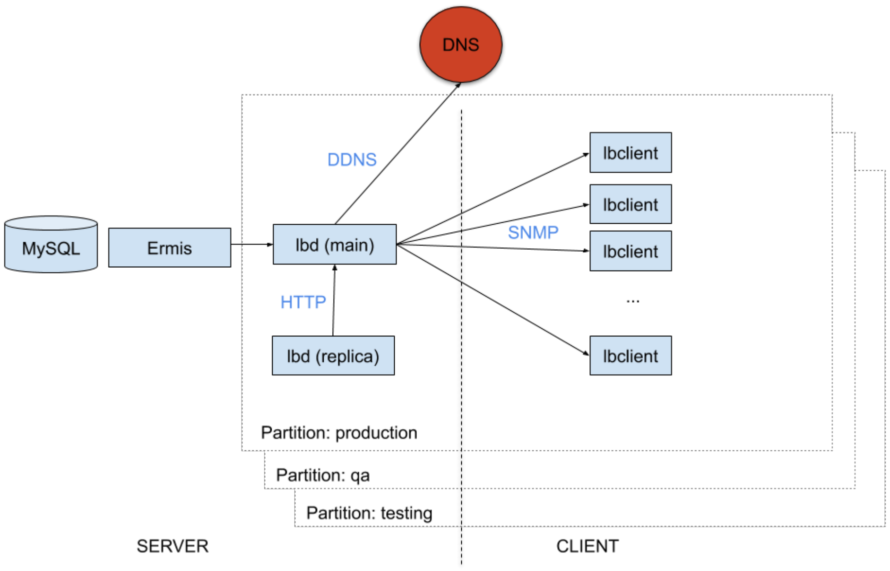


Figure 1. Components of the load balance solution

The *lbd* operates in a high availability mode, where a replica runs concurrently with the main server. The replica will check whether the main server is available, and it will only make alias updates when the main server is not available. On top of that, it is possible to separate the aliases in partitions, and use different *lbd* for each partition. This way, changes can be tested on a subset of the aliases before being applied to all of them.

The *lbd* is implemented as a concurrent Go program [4], which makes it very scalable, allowing to serve hundreds of aliases using a couple of small Openstack VMs. The feedback from the nodes is conveyed through a load metric that is generated locally in the alias members from monitoring and system information using the *lbclient* [5].

The *lbclient* can be configured to use different system/monitoring metrics as well as health monitoring checks to generate the load metric in a way that is consistent with the application running on the node that is to be accessed through the alias. The *lbclient* is a local program, currently implemented in Go, that outputs the health and the load of the node. In the current configuration, the alias member nodes get SNMP requests from the *lbd*, the snmpd calls the *lbclient* and sends the output of *lbclient* back to the *lbd*.

The final component depicted in figure 1 is the RESTful Web service containing the information of all the aliases and their parameters. The component is called *Ermis*. It offers command line and web interfaces with which to interact. Thanks to *Ermis*, users can specify the nodes that could appear behind an alias, how often to check them, how many of them should be returned and what to do in case there are no healthy nodes. The Puppet [6] system accesses *Ermis* to create the configuration of the lbd nodes.

The three components of this solution (*Ermis*, *lbd* and *lbclient*) can be split on a server side (*Ermis* and *lbd*) and a client side (*lbclient*). The server side was described on [1]. This article will be instead focused on the client side.

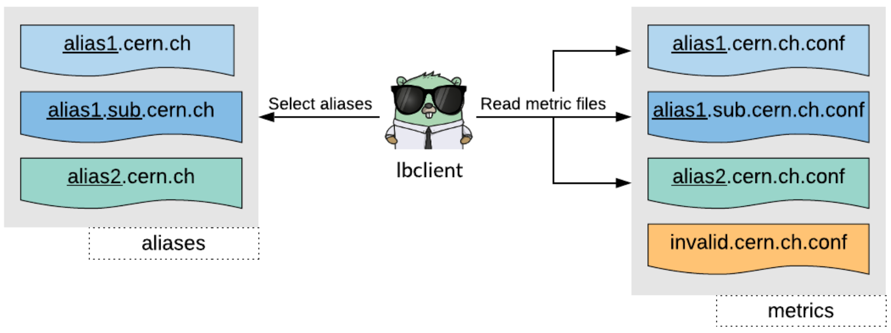


Figure 2. Configuration overview of the standalone client (lbclient) within a host machine

There are other common load balancing solutions that come out-of-the-box, as it is the case for Kubernetes-managed hosts. The solution presented in this article targets an audience looking for a configurable, standalone metric-based approach to load balancing. The current features give users the ability to load balance different aliases, within a single host, whilst restricting different or shared metrics for each of them. In the world of fullstack-development it is often the case that when a service is being developed or maintained, several environment types need to co-exist so that testing, quality-assurance and production releases can fully tested. The solution presented in this article is able to supply such functionality whilst restricting its usage complexity in the configuration side, making it easy to scale for a large number of aliases. The load balancing server [4] can also be used to achieve redundancy and high-availability of aliases by supplying the capabilities to configure the number of hosts that need to be offered by a given alias. Furthermore, the system contains a monitoring component that stores the result of the *lbclient* evaluation on all the nodes. This information can then be used for error/alarm classification. Using this approach, users are aware of host issues by looking at monitoring solutions instead of having to connect to any of the individual hosts to query their state.

3 Standalone load balancing client (lbclient)- overview

The LBD system supports any given number of DNS aliases. On top of that, each alias could be based on different metrics. To achieve this, the *lbclient* [5] reads an aliases configuration file to identify the aliases and their corresponding metrics, as seen in Figure 2. Afterwards, the client will then lookup for individual metric files within a configuration folder. A different metric configuration file can be used for each of the aliases. It is important to note that, the application will only attempt to read a file which its naming pattern matches with the previously found aliases. If the metric configuration file is not found for the required alias, the application will then calculate a default load metric.

3.1 Supported metric types

The metric-based load balancing system is provided as an out-of-the-box solution, this means that it can run in any UNIX-based operating system without being attached to a particular architecture or predefined environment. At the same time, to be able to use its full feature set, the installation of certain services might be required. Table 1 shows the current client release functionalities as well as their corresponding dependencies.

Metric Name	Usage	Required dependencies
afs	Checks if the CERN AFS mounting path is accessible	AFS: https://www.openafs.org/
collectd	Checks a c-style arithmetic expression for the given collectd metric(s)	collectd https://collectd.org
collectd_alarms	checks that the given collectd metric9s) are in the desired state	collectd
command	The command is evaluated on the node	bash
constant	Loads the supplied number into the metric output	
daemon	Checks if a service listens on the desired port, protocol and IP version	
nologin	Checks if users are allowed to log into the node	
roger	Checks that the roger state of the host is the desired one	CERN Roger
tmpfull	checks that the "/tmp" directory is not full	

Table 1. Metrics supported by the load balancing client

4 Alias definition: Ermis

The first step that a user has to follow to use the system is to define the name of the alias and its parameters. This is done with *Ermis*, either with the web interface or the equivalent command line interface. The necessary steps and components are as follows:

- Registration of a new alias:

  The request and approval of a new alias is done via a web portal named *aiermis.cern.ch*. The user may specify the desired alias name and indicate whether this alias is to be visible from outside CERN's intranet. It is important to note that duplicate aliases are not allowed. To ensure this, the system will disqualify any names that already exists in the network, as well as names that do not comply with the DNS rules.

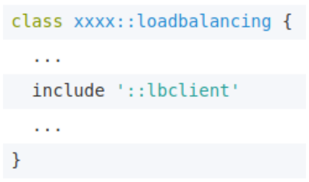
  CERN uses Puppet [6] to configure the nodes and Foreman [7] to group the hosts. Building on top of that, *Ermis* restricts that all the nodes behind an alias should be on the same Foreman hostgroup. This is to ensure that no rogue nodes can be injected behind an alias. Since multiple hosts can serve the same alias simultaneously, the user can also specify how many healthy hosts should be visible at any given moment. The user can supply as well alternative names for the alias, called canonical name records (cname records).

- Installation and configuration of the load balancing application:

  Puppet is the solution currently being used for the configuration management of all the hosts that need to serve aliases. The user specifies that the node needs the *lbclient*, and the configuration is done as described in the Figure 3. Then, Puppet is responsible for the installation of the *lbclient* and the management of both the aliases and metrics configuration files as shown in Figure 4.

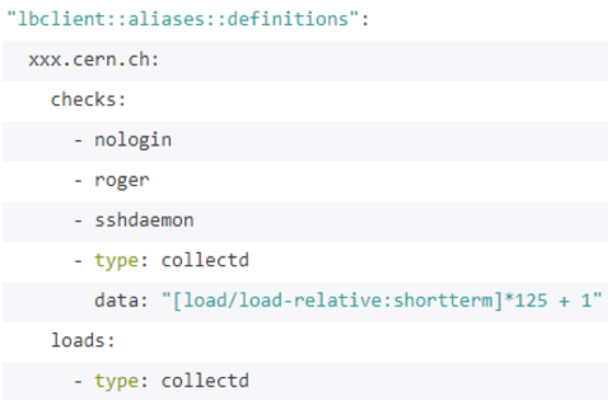
5 Future work

This chapter will give the reader an overview of the planned improvements to the existing standalone client and the overall setup used at CERN. It is important to note that the following



```
class xxxx::loadbalancing {
  ...
  include '::lbclient'
  ...
}
```

Figure 3. Puppet manifest to include lbclient



```
"lbclient::aliases::definitions":
  xxx.cern.ch:
    checks:
      - nologin
      - roger
      - sshdaemon
      - type: collectd
        data: "[load/load-relative:shortterm]*125 + 1"
    loads:
      - type: collectd
```

Figure 4. Puppet hiera declaration of aliases and metrics for lbclient

content does not describe a plan of activities. It is an insightful critic to how the load balancing solution could be further improved.

5.1 Pull vs. Push

The current deployment of the load balancing solution at CERN heavily relies on the usage of SNMP to periodically query (pull) the alias member nodes to obtain their state. This implies the complexity of installing and maintaining the configuration for the SNMP agents of all the nodes that are alias members. Using the Net-SNMP [8] implementation, the lbclient output can be provided as the result of an operation to an specific SNMP Object Identifier (OID). A different approach to address this issue is being evaluated. The idea is to switch to a push mechanism where the alias member nodes send the evaluation output of their metrics via HTTPS to a service implementing the alias arbiter logic. Using such method would allow to react to state changes of the nodes without having to wait for an evaluation cycle, thus increasing the responsiveness of the solution. It would also give the ability to create an API gateway, RESTful or gRPC based, that could allow the users to query information about the state of their aliases in real time.

5.2 Feature addition - Directory/File size conditional evaluation

It is often the case that the space utilization of different mounting points and volumes needs to be monitored as to avoid service degradation or even malfunctioning. With this in mind, the plan is to create a new metric type that will be capable of comparing the size of one or more directories or files against a given condition.

5.3 Kubernetes deployment of the server

The current deployment of the server side relies on a classical primary-secondary architecture, where the primary node is responsible for the update of DNS records and, in case of failure, its role is delegated automatically to a secondary node. Despite offering high-availability and failover, the scalability of the system is somewhat limited due to the write and read operations allowed on primary and secondary nodes since the state of each alias nodes needs to be queried by the server. As discussed in subsection 5.1, using a push instead of a pull approach would make our architecture highly-scalable horizontally. The goal is to adapt the different components of the server side (Ermis, the alias arbiter, and any other required microservices) as docker images that could be deployed in parallel and scale horizontally. Therefore, a combination of docker images deployed in Kubernetes cluster(s) could be used to deploy the new architecture.

6 Summary

This article presented the objectives, advantages and usage of a metric-based load balancing solution that is actively maintained at CERN. The topics that have been covered are:

- An overview of the full DNS load balancing solution implemented at CERN.
- The versatility and high scalability that this metric-based approach offers to load balance aliases that other out-of-the-box solutions cannot provide.
- Overview and walk the reader through the steps of how to use our standalone application and setup a configurable metric-based evaluation.
- Discuss possible improvements and new features to the current architecture at CERN and standalone application.

References

[1] P. Canilho, I. Reguero, & P. Saiz 2019. Concurrent Adaptive Load Balancing at CERN. EPJ Web of Conferences, 214, 8028. https://doi.org/10.1051/epjconf/201921408028

[2] SNMP https://en.wikipedia.org/wiki/Simple_Network_Management_Protocol

[3] 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

[4] GitHub. 2020. GitHub - cernops/golbd: Go implementation of CERN lbd DNS Load balancing daemon. [ONLINE] Available at: https://github.com/cernops/golbd. [Accessed 10 January 2020]

[5] GitHub. 2020. GitHub - cernops/golbclient. [ONLINE] Available at: https://github.com/cernops/golbclient. [Accessed 10 January 2020]

[6] Powerful infrastructure automation and delivery | Puppet | Puppet.com. 2020. Powerful infrastructure automation and delivery | Puppet | Puppet.com. [ONLINE] Available at: https://puppet.com/. [Accessed 02 February 2020].

[7] Foreman:Lifecycle management tool for physical and virtual servers [ONLINE] Available at: https://www.theforeman.org/ [Accessed 17 June 2020]

[8] Net-SNMP http://net-snmp.sourceforge.net/