

Load Balancing is one of the technologies enabling deployment of large scale applications on cloud resources. A DNS Load Balancer (LBD) has been developed at CERN long ago as a cost-effective way to balance applications accepting DNS timing dynamics and not requiring affinity. We serve over 450 load balanced aliases with two small VMs acting as master and slave. These aliases are based on 'delegated' DNS subdomains that we manage using DDNS according to a load metric collected with SNMP from the alias member nodes. In the last years we have done several improvements to the software, for instance support for IPV6 AAAA records, parallelization of the SNMP status requests, as well as implementing the client in python to allow for multiple aliases with differentiated state on the same machine, support for Roger application state and other new features. The configuration of the Load Balancer is managed by a Puppet type that gets the alias member nodes dynamically from PuppetDB and consumes the alias definitions from a REST service. We have produced a self-service GUI for the management of the LB aliases based on the REST service above thus implementing a form of Load Balancing as a Service (LBaaS). Both the GUI and REST API have authorization based in Foreman hostgroups. All this is implemented with Open Software without much CERN specific code.

## LBD MAIN SCHEMA

### LBD Master and Slave

The LBD master periodically collects a load metric from the alias member nodes using SNMP transport to choose the ones whose state looks best to be presented by the LB alias. This is a great advantage relative to other load balancers using round-robin mechanism where there is no feedback from the member nodes so nodes not responding are not excluded. Each LB alias is mapped to a DNS subdomain corresponding to its name that can be updated independently by the LBD using DDNS.

The LBD slave collects the load metric from all nodes like the master, but it will only do the DDNS updates when the LBD master is not available. This is checked in a simple way by trying to access a heartbeat file served via HTTP by the LBD master. This implements a simple but effective form of high availability mechanism.

### LB Alias member nodes and LB Client

The similarly configured nodes that belong to the same cluster to implement a service are called LB alias member nodes. The IP addresses of these nodes can be presented by the LB alias so they can be accessed through the DNS name of the LB alias depending on their load and health monitoring state.

The lbclient runs in the LB alias member nodes to produce a load metric that measures the load and health state of the node. It is started by the SNMP daemon when getting and SNMP get request on a specific MIB OID. It gives back an integer number that is the load metric value. Negative values of the load metric indicate health monitoring conditions where the node is to be temporarily excluded, i.e. Its IP address not considered for the LB alias.

## APPLICATION EXAMPLES

Here follow some examples of the applications using one of the over 450 DNS LB aliases managed by the LBD

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
DASHB-MB	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 for LHC experiments	PANDA, RUCIO, FRONTIER, SQUID

## LOAD METRICS and HEALTH MONITORING

**LOAD METRICS** While the lbclient offers a built-in metric, in the spirit of the Linux Load Average, it also offers the possibility of building ad-hoc load metrics using a combination of Lemon monitoring metrics and constants.

**Health Monitoring** 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 can check whether the application state is production in the Roger server or check the return code of arbitrary scripts. Other arbitrary health monitoring checks can also be configured using conditions on Lemon monitoring metrics defined for the node.

**IPv6** Since the CERN network started supporting dual stack IPv4 - IPv6 machines, we needed to support the IPv6 AAAA DNS resource records, together with IPv4 A DNS resource records on the DNS zones used for LB aliases. The support for AAAA resource records implied replacing IPv4 only calls such as gethostname() with calls that work both for IPv4 and IPv6 such as getaddrinfo() and getnameinfo(). This support allows that LB aliases present a mixture of IPv4 and IPv6 addresses.

**DDNS** As each LB alias is mapped to a DNS subdomain corresponding to its name, the state of each alias is maintained in the DNS resource records of this domain, more specifically on the A and AAAA records that define the IP addresses that are presented when referencing the name of the LB alias. DDNS allows to update this state. The LBD gets access to the DNS server using DDNS for updates authenticating with a TSIG key.

**SNMP** The SNMP monitoring protocol is used to collect the status of the nodes that are members of the LB alias thus providing the LBD server with information on the load and availability of the nodes. The lbclient gives back an integer that is the load metric value. The recent Python versions of the lbclient allow multiple LB aliases on the same node, so instead of an integer, they can also give a string that encodes a key value pair list with the values of the metrics for all the LB aliases on the node. The LBD SNMP requests are authenticated using a SNMPv3 user and password.

## USER INTERFACES

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 web service that we called Ermis. This service consumes a SOAP interface provided by the CERN Network group that allows creation, deletion and modification of DNS subdomains.

Ermis has been implemented using Django Tastypie.

The Lbconf Puppet type and provider have been written to generate the LBD configuration consuming the information from Ermis.

The Aiermis user facing GUI has been implemented using Django views on top of the Ermis service with some dynamic content in Javascript using Ajax calls.

The Kermis CLI has been implemented as thin layer using the Python Requests module calling the Ermis service.

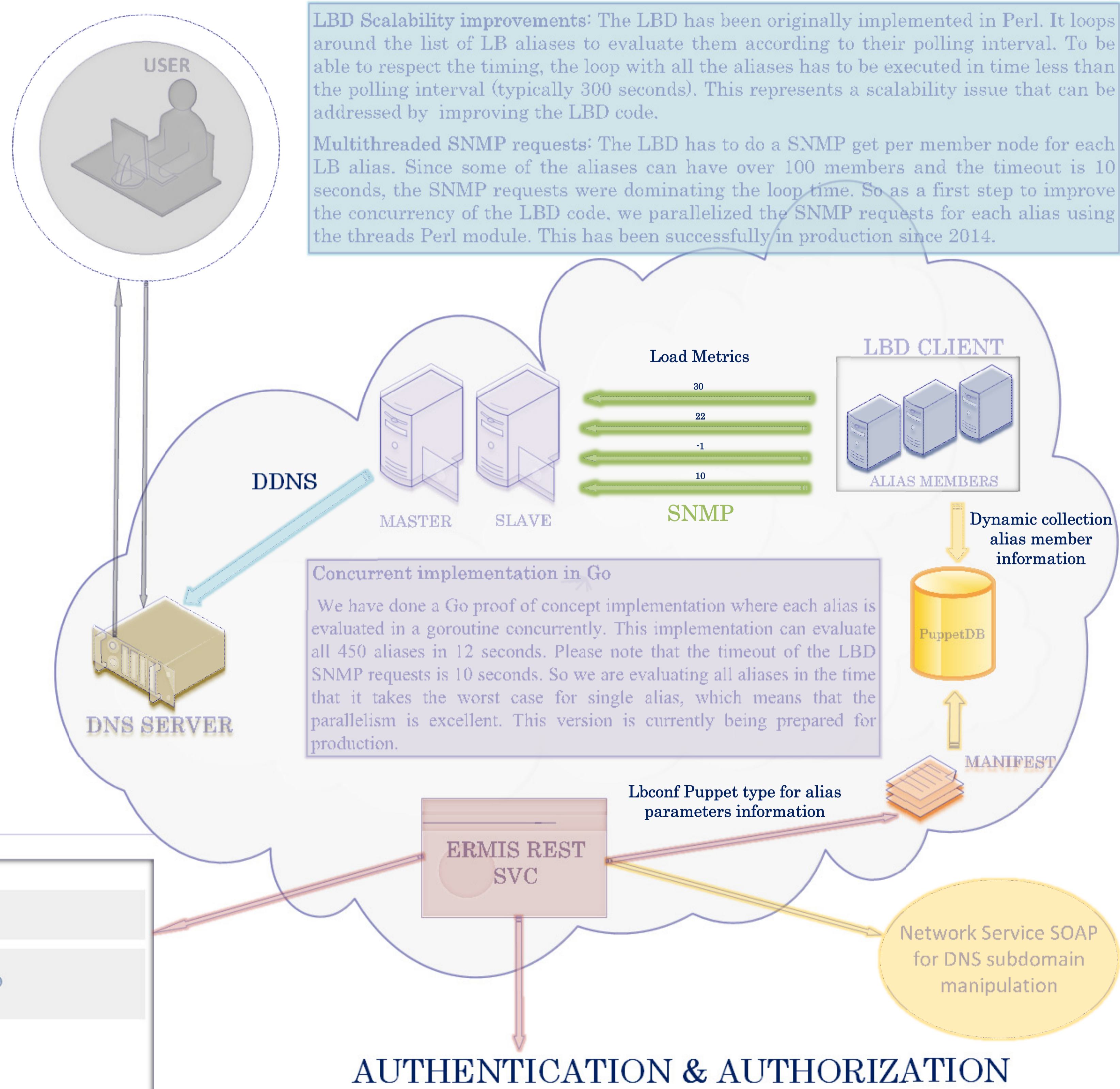
All this implements a form of Load Balancing as a Service (LBaaS).

## CONCLUSIONS

Using DNS allows to balance the load without modifying traffic patterns or adding network gateways. Another advantage is that it supports all protocols, without having to do special work.

On the other hand, the applications need to allow the slow switching due to DNS caching and the lack of affinity. The cases that cannot meet these restrictions can use industry standard layer 4 or 7 load balancers such as HAProxy. However, a large number of applications can use DNS load balancing. At CERN we run over 450 DNS LB aliases in a very cost effective way with the LBD running on a couple of nodes (master and slave) hosted on small Openstack VMs.

While the CERN LBD has been running in production for more than ten years, it has continuously evolved. New features have been added, such the support for IPV6 AAAA records. Scalability questions have been addressed, for instance with the parallelization of SNMP status queries. The migration to the cloud has prompted the integration with a service-oriented architecture where the management of the alias information is done through the Ermis RESTful web service that is consumed by Puppet and by user facing GUIs implementing a form of Load Balancing as a Service (LBaaS).



**LBD Scalability improvements:** The LBD has been originally implemented in Perl. It loops around the list of LB aliases to evaluate them according to their polling interval. 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 by improving the LBD code.

**Multithreaded SNMP requests:** The LBD has to do a SNMP get per member node for each LB alias. Since some of the aliases can have over 100 members and the timeout is 10 seconds, the SNMP requests were dominating the loop time. So as a first step to improve the concurrency of the LBD code, we parallelized the SNMP requests for each alias using the threads Perl module. This has been successfully in production since 2014.

**Concurrent implementation in Go**  
We have done a Go proof of concept implementation where each alias is evaluated in a goroutine concurrently. This implementation can evaluate all 450 aliases in 12 seconds. Please note that the timeout of the LBD SNMP requests is 10 seconds. So we are evaluating all aliases in the time that it takes the worst case for single alias, which means that the parallelism is excellent. This version is currently being prepared for production.

## AUTHENTICATION & AUTHORIZATION

**Authentication:** Every interaction with the Ermis service requires that the user is authenticated. Anonymous users cannot interact with the service. The Aiermis GUI uses the CERN Single Sign on that is based on Sibboleth. The kermis CLI uses Kerberos, ie. it requires a valid Kerberos ticket. The Ermis API can be accessed with Kerberos or SSL authentication.

**Authorization:** The Ermis service supports authorization based on Foreman hostgroups. All managed hosts belong to a hostgroup. When creating an LB alias, a hostgroup has to be assigned to it. All alias member nodes have to belong to the same hostgroup as the LB alias. Users are only allowed to perform create/delete/update operations on LB aliases belonging to hostgroups that they administer.