Flomesh Software Load Balancer White Paper

Product Overview

The FLB (Flomesh Software Load Balancer) is a software-based application load balancing solution that operates at the infrastructure level. It is a reliable, efficient, and scalable solution that uses "BGP+ECMP+eBPF+Pipy" to offer both Layer 4 and Layer 7 load-balancing capabilities.

FLB has the flexibility to operate in different settings such as physical and virtual machines, cloud hosts, and k8s container platforms. It caters to a diverse range of scenarios, from the Internet edge (POP point) to the DMZ of the Internet access area, to the entrance of the container cluster (Ingress & Gateway API).

FLB offers the same level of throughput, low latency, and high stability as traditional hardware load balancing solutions. In addition, it provides the advantages of software-based scalability with its "small start and easy expansion" approach. This cloud-native, platform-based software load-balancing tool facilitates swift and effortless deployment across several environments, enhancing business access capabilities and application delivery.

FLB is a software load balancing tool that boasts high performance while consuming minimal resources. It is well-suited for use in container environments and cloud hosts, even those with limited processing power and memory, as low as 1C2G. FLB provides a seamless balance between low latency and high throughput, while also offering simple management capabilities.

FLB offers extensive customization options, enabling users to easily integrate with DevOps, cloud management, and other platforms through personalized management control interfaces (Admin REST API). Furthermore, it supports secondary development and extension capabilities that are based on PipyJS (PJS), allowing users to utilize JS syntax to swiftly customize management interfaces, routing policies, access control policies, load balancing policies, and more for Layer 4 and Layer 7 load balancing.

Chapter One: Evolution and History of Software Load Balancing

To appreciate the significance of Flomesh Software Load Balancer, it is vital to have a grasp of the progression and background of software load balancing. This section will provide a concise overview of the different phases of software load balancing, noteworthy technical frameworks, constituents, and traits.

First-generation software load balancing: hardware load balancing replacement

The rise of the internet after the turn of the millennium prompted the development of software load balancing. Hardware load balancers were costly and challenging to scale, presenting a problem for internet companies. To overcome this, internet engineers sought alternative solutions in open-source software. The outcome was the standard solution of using "Layer 4 load balancing based on LVS + Layer 7 load balancing based on proxy servers", which evolved rapidly. Commonly used Layer 7 proxy servers included Nginx, Haproxy, and Varnish. This gave rise to the first-generation software load balancing, known as "LVS + Nginx" or "LVS + Haproxy" scheme. During this

era, internet operations teams were responsible for building and using software load balancing, along with developing their own supporting operation and maintenance tools. This led to the formation of a toolchain centered around the core "LVS + Nginx/haproxy", which included automated operation and maintenance, monitoring, and metrics.

Second-generation software load balancing: Self-service

The rise of cloud computing paved the way for the second generation of software load balancing. Initially, cloud providers focused on enhancing the first-generation solutions by improving multi-tenancy, self-service, and elasticity. The fundamental technology stack remained "LVS + Nginx/haproxy," with the software load balancer being developed by cloud platform product teams and utilized by cloud tenants. The segregation between developers and users allowed for more precise management of the load balancer and a higher degree of automation.

The emergence of cloud computing providers has led to the development of commercial software load balancing software for private clouds. Examples include Nginx Plus from Nginx and AVI Networks, which was acquired by VMware. The second-generation software load balancer, also known as "self-service software load balancer," is characterized by its emphasis on self-service.

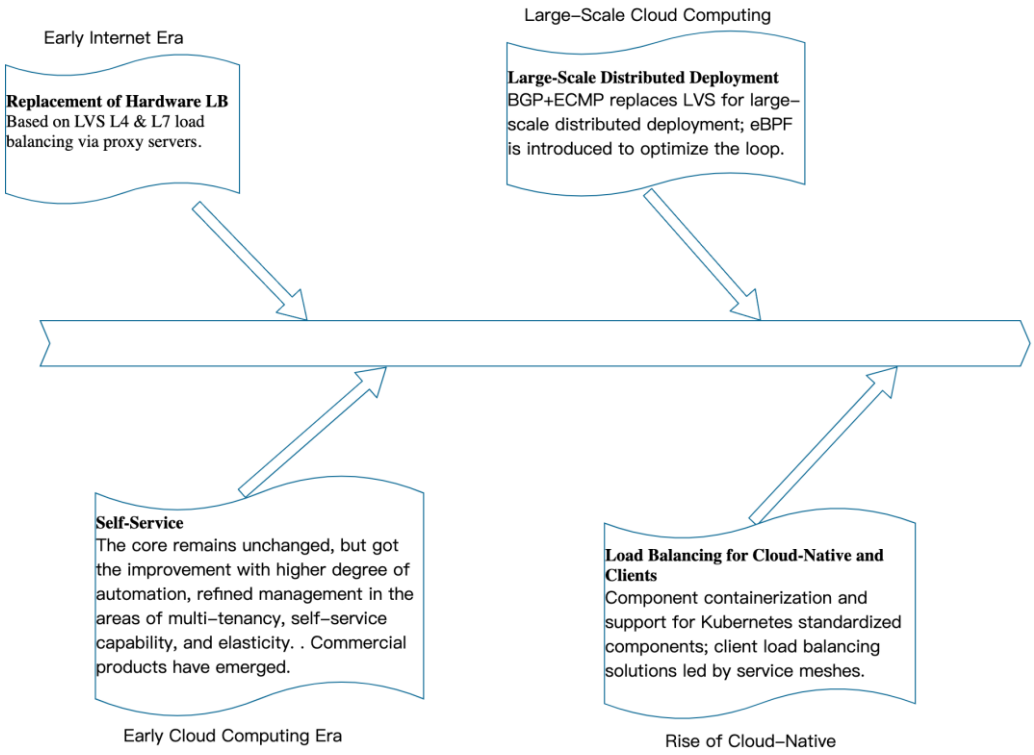
Third-generation software load balancing: Large-scale distributed deployment

The third generation of software load balancing has emerged to overcome the limitations of the second generation when dealing with large-scale expansion.

This version uses the "BGP + ECMP" scheme instead of LVS for layer 4 load balancing and incorporates eBPF technology to optimize loopback and other issues. A popular solution is Facebook's open-source Katran. For layer 7 load balancing, proxy-based solutions like Nginx/haproxy are still common. With the introduction of BGP, third-generation load balancing can handle larger scales in horizontal expansion and typically follows a large-scale distributed deployment model. As a result, it's often referred to as "distributed software load balancing".

Fourth-generation load balancing: Cloud-native load balancing

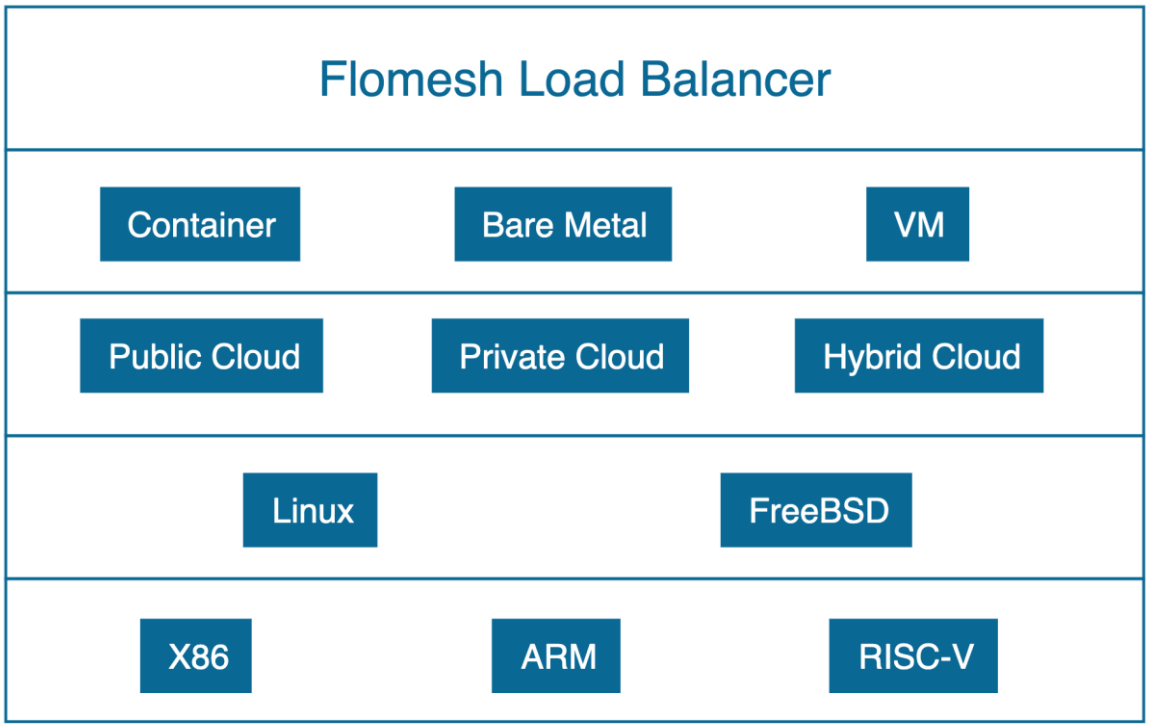
With the growing popularity of container platforms, fourth-generation load balancing (also called cloud-native load balancing) has emerged. Its main features include containerization and client-side load balancing. Containerization not only involves containerizing different components of load balancing, but also using standard frameworks like Kubernetes Ingress as load-balancing components. The adoption of client-side load balancing aims to address the load balancing of east-west traffic introduced by the surge in microservices. By front-loading load balancing to the service caller, client-side load balancing offers more flexibility. The sidecar proxy in the service mesh is the typical load balancing technique used in this generation.



Chapter 2: Product Attributes

2.1 Product Positioning

Flomesh is a next-level software load balancer that offers "load balancing as a service" for cloud platforms such as public, private, and hybrid clouds. By using Flomesh software load balancer, users can enjoy load-balancing capabilities across various platforms, clusters, regions, and vendors. This surpasses the performance of cloud platforms themselves and provides more adaptable scalability and a consistent user experience.



2.2 Solution Comparison: Private Cloud

Flomesh outperforms load-balancing solutions created by public or private cloud operation teams using open-source software in terms of universality and portability. It can be easily customized to suit the requirements of enterprise customers who use hybrid and multi-cloud solutions on various cloud platforms. Furthermore, Flomesh provides scalable capabilities that allow large enterprise users to build and personalize their own exclusive software load-balancing platforms using Flomesh as a foundation.

2.3 Solution Comparison: Public Cloud

Flomesh provides a more cost-effective and profitable solution compared to cloud platform-specific load balancing software like NLB and ALB from AWS. Flomesh's load balancing software costs as low as one-tenth or even less when processing the same throughput as equivalent cloud services. Additionally,

Flomesh uses a unified technology stack that can adapt to various scenarios such as physical machines, virtual machines, container platforms, microservices platforms, and others, resulting in a more consistent management experience and lower learning curve. This leads to the "one-time investment, use everywhere" effect.

	AWS LB	Flomesh LB
Billing Model	Charged based on request volume, with additional charges per million	Charged based on licenses or instances
Private Deployment	N	Y

Private Deployment	Nginx	Flomesh LB
Free	Y	Y
Open source	N	Y
Customization/Development	High cost	Low cost

Chapter 3 Product Architecture System

3.1 Control Console

The Strapi-based FLB Control Console employs a relational database for the storage of data pertaining to tenants, configurations, permissions, and various other details. PostgreSQL or MySQL are the default storage options for the console.

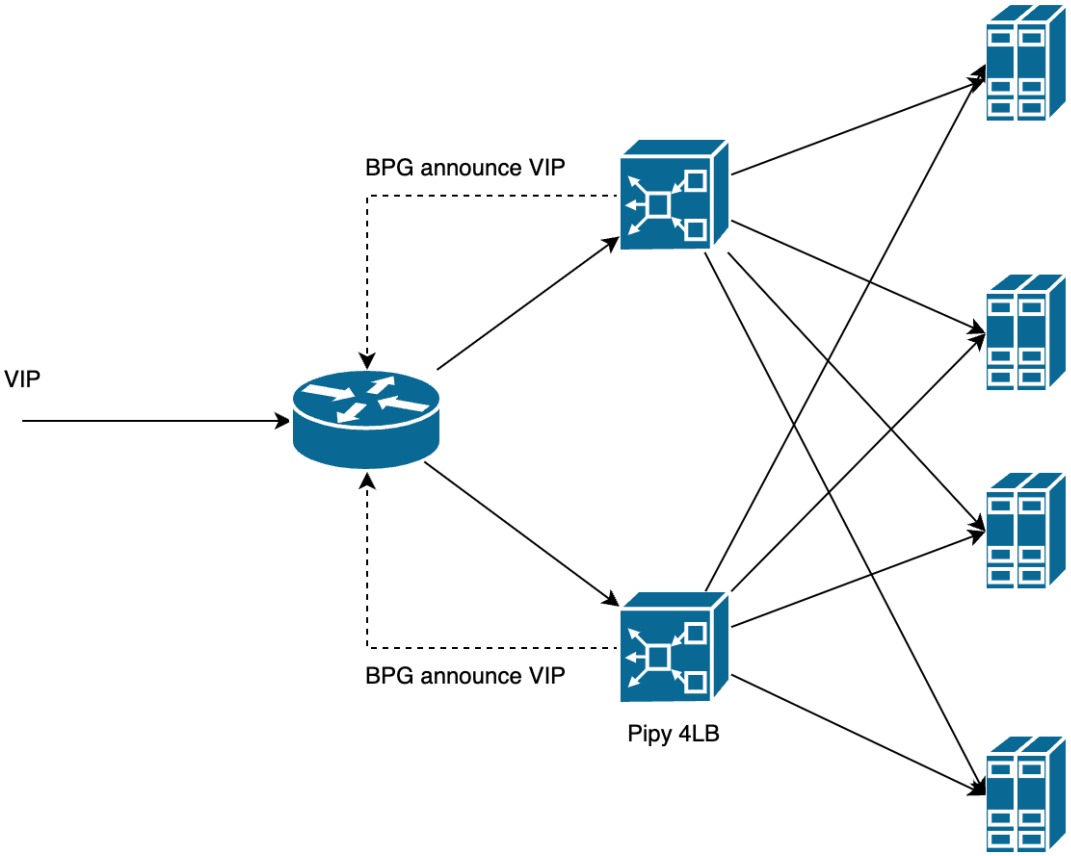
3.2 Pipy Repo (Configuration Center)

The Flomesh Software Load Balancer (FLB) utilizes Pipy Proxy, which is a Layer

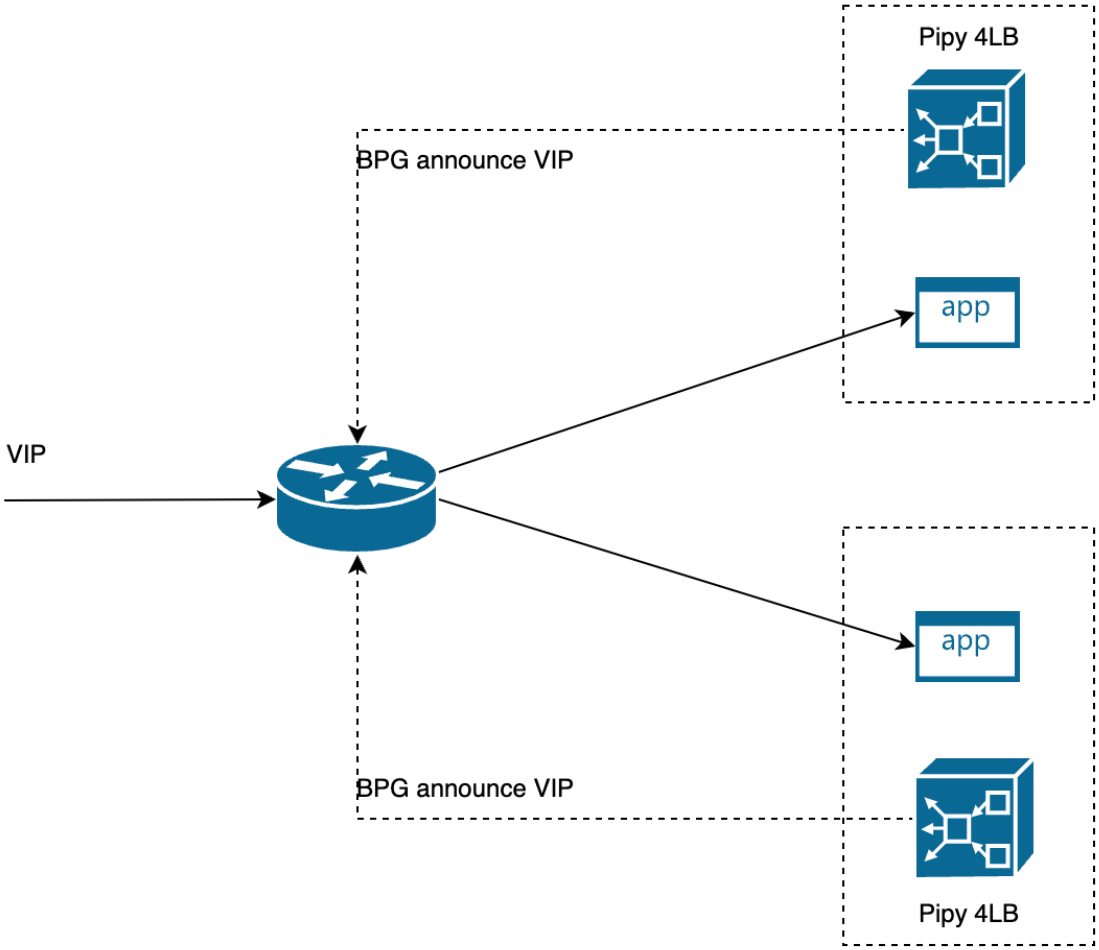
7 proxy that operates through various distributed Pipy instances. Pipy Repo works as a registration and configuration center. Whenever a new Pipy instance is initiated, it registers with Pipy Repo to access its configuration data. Therefore, Pipy Repo plays a crucial role in managing the registration and configuration of FLB.

3.3 Pipy 4LB

When operating in centralized deployment mode, a Pipy 4LB utilizes BGP protocol to announce a single IP externally. Data packets are then forwarded by the router to various nodes in the Pipy 4LB cluster based on the ECMP policy announcement information. Pipy then listens on the designated IP ports and distributes received connections to upstream servers based on the selected load balancing policy.

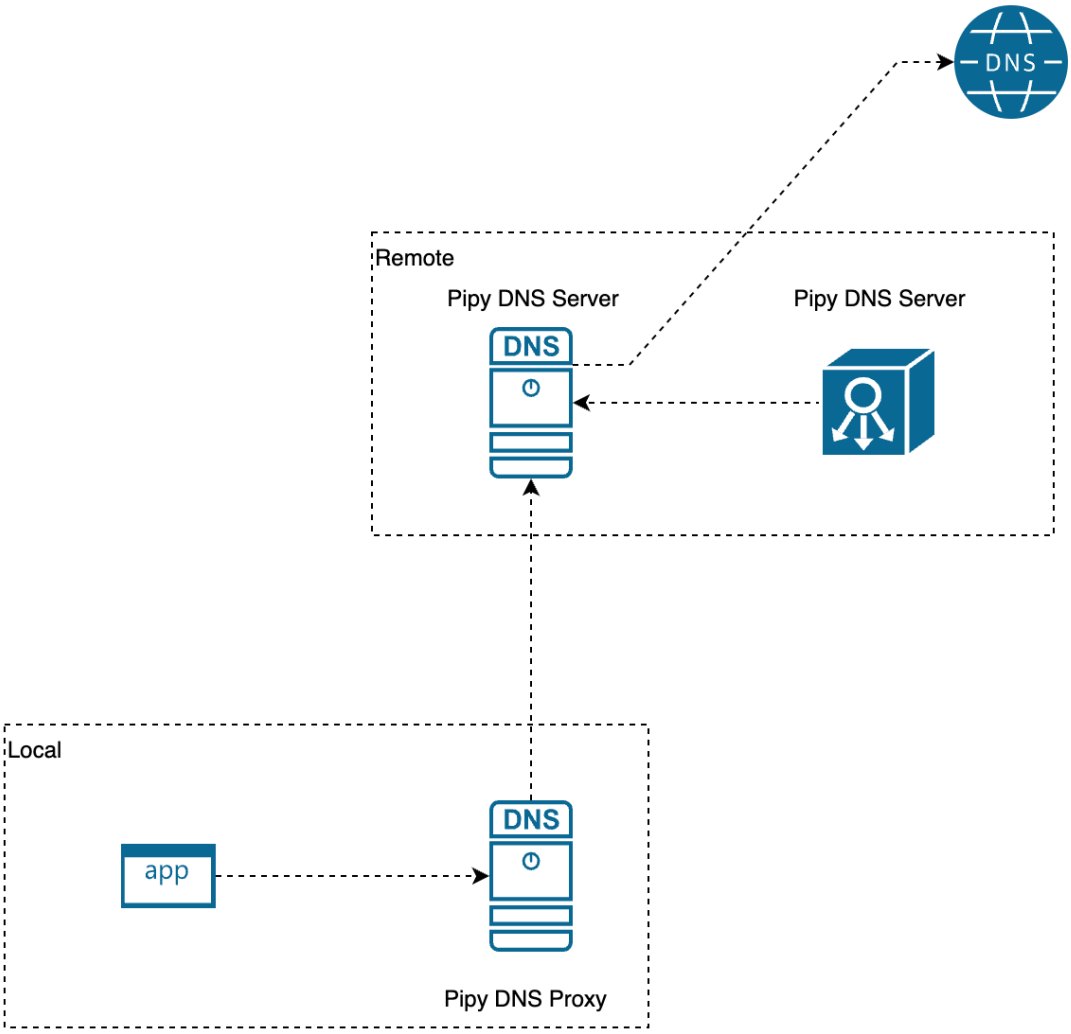


When operating in edge mode, the Pipy 4LB is utilized in conjunction with the business process. The responsibility of Pipy is to declare the IP address to the upstream router, while the business process is responsible for monitoring the specified IP address on the port. Unlike the centralized mode, traffic is not routed through Pipy in edge mode; instead, it is directed straight to the business process.



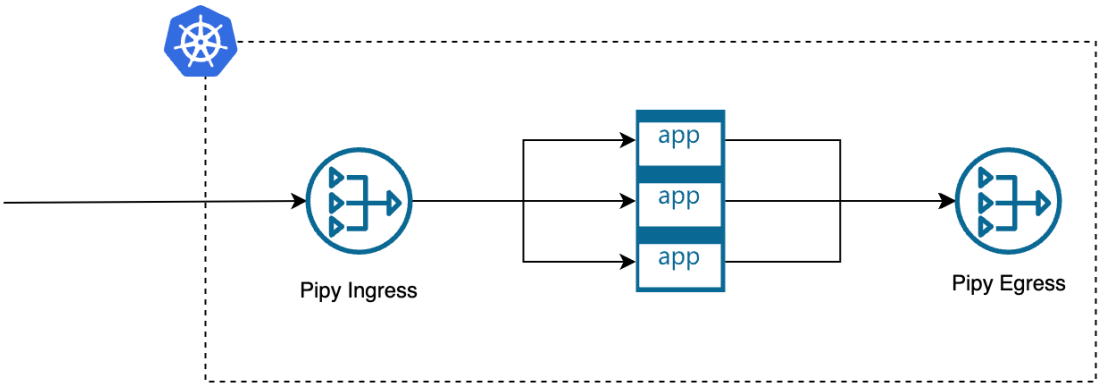
3.4 DNS Load Balancing (Optional)

Pipy has the capability to serve as a DNS server or proxy, performing on both the client and server ends. It can react to DNS requests from clients and retrieving and managing application IP addresses obtained from the control components. Pipy can provide diverse resolution records to various clients based on policies that take into consideration geographic location or health status. DNS load balancing is a feature of Flomesh that is optional and can be activated as per user requirements.



3.5 Ingress/Egress Controller (Optional)

FLB provides Ingress and Egress functionalities to a Kubernetes cluster with the aid of an optional Ingress/Egress Controller. The Controller can be set up in a specified namespace in Kubernetes and keeps track of changes in different resources in the cluster via the API Server. It then generates Pipy configurations and scripts based on these changes, which are then uploaded to Pipy Repo. Subsequently, these configurations are obtained, dynamically loaded, and executed automatically by Pipy Ingress instances. The Controller also supports the Gateway API.



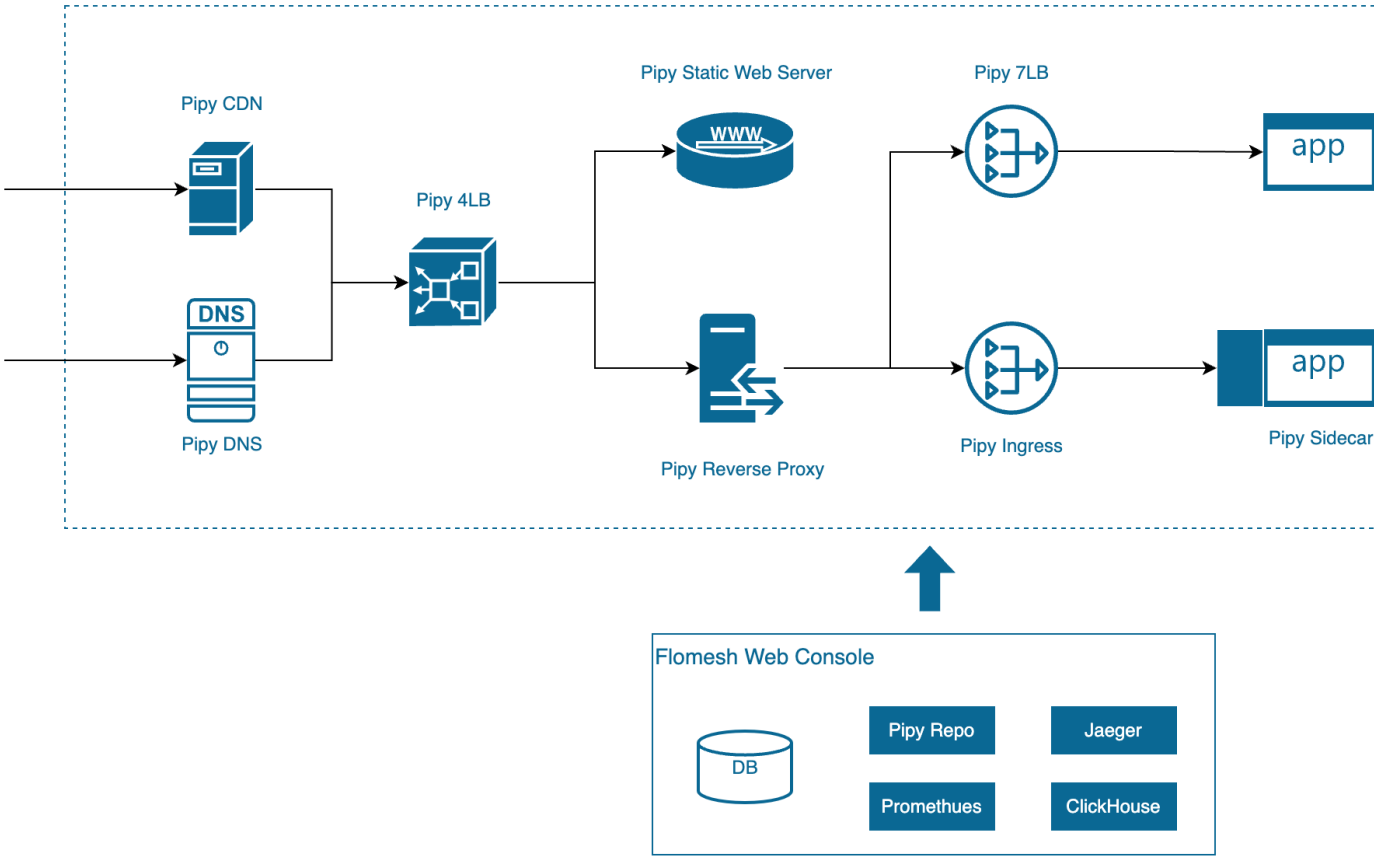
3.6 Sidecar Controller (Optional)

The FLB sidecar controller is an optional component that facilitates client load balancing. When deployed in this mode, the application process and Pipy process are activated side by side. Upon application start-up, the Pipy sidecar registers the service. Whenever an external service is accessed, FLB intercepts the request and redirects it to the Pipy sidecar through iptables or eBPF. The Pipy sidecar then completes service discovery and undertakes load balancing. The sidecar controller also offers sidecar injection, start/stop, and configuration hot reload functions, making sidecar deployment easy to manage and maintain.

3.7 ELB Controller (Optional)

Flomesh Software Load-balancer can be deployed to a Kubernetes cluster in order to to provide ELB services. Once deployed, the ELB controller will monitor services of type LoadBalancer and register their ExternalIP with BGP routers outside of the cluster as needed.

Chapter 4 Component Topology



Chapter 5: Product Feature List

Flomesh software load balancer can be divided into several major functional aspects:

1. Multi-tenant management
2. Technical component management
3. Layer 4 load balancing
4. Layer 7 load balancing
5. Client load balancing
6. DNS load balancing
7. Kubernetes Ingress and Egress
8. Kubernetes ELB

9. API load balancing
10. Graphical user console

5.1 Multi-Tenancy Management

Flomesh software load balancer is a multi-tenant load balancing as a service platform-oriented product designed for enterprise users. Users on the platform are organized and managed by "organizations," with each user belonging to a particular "organization." These "organizations" can form a hierarchical relationship with each other, and authorization, resource access control, and other operations are typically performed on an "organization" basis.

The platform also features project management capabilities, with all resources grouped by "projects." These resources include registry, services, APIs, load balancing, and more, each belonging to different projects. Within the same project, the access between resources can be in "loose" access mode, meaning that resources within the same project can access each other without constraints. Alternatively, the access between resources within the same project can be in "strict" mode, which means that each resource has its own independent authorization and access control rules. Resources between different projects usually require some kind of authorization to access.

Flomesh software load balancer platform also provides standard user management capabilities, including user registration and information editing. Administrators can authorize users, and authorized users can perform specified operations, all of which are recorded and auditable. The platform adopts the Role Based Access Control (RBAC) model to manage the access rights of all resources, with permissions assigned to specific "roles." Administrators further bind "roles" and users, so that users can access resources.

In addition, users can view application logs, application full-text logs (which need to be configured to enable), and user access and operation records within their authorization range on the console. The software load balancer platform itself can generate various events, such as alarms and prompts. The software load balancer platform can also connect to managed technical components, receive events, and centrally display and manage them on the console, such as receiving and processing events from k8s.

5.2 Component management

The Flomesh software load balancer platform is comprised of various components, including both self-developed features by the Flomesh team and third-party components that serve specific functional and management requirements. All self-developed components are accessible through the Flomesh team's open-source repository on flomesh-io · GitHub. Some of these components can be installed and deployed directly through the Flomesh software load balancer platform, while others can be utilized by configuring parameters. These components are categorized based on their respective functions.

- 4LB cluster. The 4LB cluster consists of multiple Pipy, which provide core functions such as proxy, routing, load balancing, and BGP announcement.
- Tracing components
  - Jaeger
- Metric components
  - Prometheus
- Database components
  - MySQL

    - PostgreSQL
- NoSQL component
    - Clickhouse
    - ElasticSearch

5.3 Layer 4 Load Balancing

Flomesh offers layer 4 load balancing services for multiple tenants, including IP high availability, TCP/UDP packet forwarding, load balancing, proxying, and tunneling. For traffic between subnets, BGP+ECMP is used for IP high availability and load balancing, while eBPF is used for traffic within the same subnet. Pipy Proxy is used for advanced control policies like access control and dynamic fault migration during the load balancing process.

IP high availability is achieved via BGP+ECMP when the source and destination addresses are in different subnets or switch LEAFs. Pipy makes BGP announcements to the BGP router, declaring that the target address process has a specified VIP, and the BGP router performs load balancing based on these announcements and policies. Pipy also monitors the target process and port and removes the failed destination address route from the BGP router. For requests within the same subnet, Flomesh uses eBPF technology to convert the requested VIP address to the real destination address based on the load balancing strategy.

The BGP announcement module has been added to Pipy, which can be deployed centrally or as a sidecar deployment with the target address process. Flomesh load balancer listens to a registration center like Eureka in central

mode and announces multiple providers' IP addresses as a single VIP to the BGP router. In Pipy sidecar mode, Pipy's address process starts and obtains the real IP address together with the target address process, announcing the VIP to the BGP router based on the pre-configured VIP.

During layer 4 load balancing, Flomesh software load balancer intercepts traffic and directs it through Pipy Proxy to execute required policies like identity recognition, access control, and link encryption. At the Layer 4 level, the software load balancer supports TCP proxy and socks proxy.

For cases where the source and destination addresses are in different networks or regions, Flomesh software load balancer establishes a tunnel between the addresses and forwards, routes, and load balances packets within the tunnel, supporting tunneling technology based on HTTP/1.1, HTTP/2, and gRPC protocols. The software load balancer supports multiple TCP connections in a single tunnel, using multiplexing technology to reduce resource consumption on underlying network devices like routers.

Policy management is essential during the load balancing process, and Flomesh software load balancer uses traffic interception technology to pass the traffic through Pipy Proxy, where specific policies are executed, like identity recognition, access control, bandwidth limitation, and connection establishment frequency. Pipy Proxy communicates with a centralized statistical and computing component through a long connection to perform rule detection. Multiple policies are applied in a chain, wrapped into a plugin and executed in the order of the chain.

5.4 Layer 7 Load Balancing

The Flomesh software load balancer system offers Layer-7 capabilities for load balancing across multiple tenants, clusters, and protocols. These capabilities are created using Pipy Proxy and are integrated with cloud platforms like Kubernetes and controllers, and presented in various forms including Ingress/Egress, Gateway API, API Gateway, Kubernetes ELB, and Sidecar Proxy.

The Layer-7 load balancing supports a range of protocols such as HTTP/1.x, HTTP/2, DNS, Redis, MQTT, Dubbo, gRPC, Thrift, and TCP short messages. It includes typical load-balancing functions such as routing (CBR), forwarding, proxying, message rewriting, load balancing, and Failover. Security extensions like content filtering (WAF filtering), identity recognition, and access control are implemented through extension plugins. Flomesh also supports service governance functions like service discovery, circuit breaking and degradation, and canary release.

To simplify configuration and management, Flomesh provides a mode that can run both Layer-4 and Layer-7 load balancing in the same Pipy. The Layer-7 load balancing also supports modular policy execution through plugins, which can be configured or dynamically injected into Pipy. The Pipy hot reload function enables dynamic effects without restarting Pipy, providing higher service level agreements (SLA) and quality of service (QoS) to avoid service interruption caused by reloading configuration.

Flomesh uses PipyJS technology to implement full plugin functionality using

home-built JavaScript engine, allowing for the deployment of new plugins without redeploying or upgrading Pipy. This improves the QoS and SLA of services while reducing the complexity of upgrading and managing the Pipy executable file.

5.5 Client-side Load Balancing

Client-side load balancing is a widely used technique in microservices that allows service consumers to access multiple service providers based on load-balancing policies. This can be achieved through an SDK or sidecar process, such as the Consul Agent in HashiCorp's Consul microservices framework. FLB provides similar client-side load balancing capabilities to that of the Consul Agent, regardless of whether the microservice consumer process runs on a physical machine, virtual machine, or container. The Pipy sidecar is manually or automatically injected by FLB, intercepting requests from service consumers and entering the Pipy sidecar proxy to complete load balancing. In addition to load balancing, Pipy Sidecar can provide various non-load balancing technical capabilities, such as probing, enforced policy checking, and service registration. FLB also provides process-level failover capabilities and two solutions for multi-cluster service discovery scenarios.

5.6 DNS Load Balancing

The technique of DNS Load Balancing involves associating multiple IP addresses with a singular domain name to distribute client requests to different IP addresses. The Flomesh software load balancer has a DNS module, Pipy, that utilizes PipyJS to develop flexible DNS resolution solutions. In Kubernetes environments, basic DNS service discovery capabilities are

provided through a Pipy sidecar or Pipy proxy. Flomesh software load balancer intercepts DNS queries and directs them to the Pipy DNS module, which uses specific logic to resolve queries or customize PipyJS scripts based on user requirements. Pipy sends DNS requests to upstream DNS servers, typically CoreDNS in a Kubernetes environment. The absence of NAT or proxy processes between ClusterIP and actual IP addresses speeds up the resolution process. The DNS load balancing feature is not limited to container environments and can be used for Global Traffic Management DNS services and Local Traffic Management DNS services. The Pipy DNS module and PipyJS script capability provide a basis for dynamic DNS resolution. Flomesh software load balancer's expert services provide strong customization requirements for this type of functionality. DNS service discovery for container environments is primarily used for DNS service discovery in Kubernetes environments.

5.7 Kubernetes Ingress & Egress

The two primary forms of load balancing in container platform environments are Ingress and Egress, which have interface specifications including Ingress, Gateway API, and Egress/EgressGateway CRD. For load balancing in Kubernetes environments, Flomesh software load balancer offers several controllers developed primarily in Go and available in the open-source repository of Flomesh at **GitHub - flomesh-io/ErieCanal: ErieCanal is a MCS(multi cluster service https://github.com/kubernetes-sigs/mcs-api) implementation, provides MCS, Ingress, Egress, GatewayAPI for Kubernetes clusters.**. These controllers are encapsulated and can be deployed easily using the Kubernetes toolchain, with a user experience that aligns with the habits of the Kubernetes community. The Flomesh software load balancer console contains these functions in the Ingress menu, where users can configure and manage them graphically.

5.8 Kubernetes ELB

The standard Kubernetes implementation includes a service type known as "LoadBalancer," but it does not automatically assign an External IP. Different public cloud platforms have their own variations, while the open-source community offers MetalLB as a typical implementation. Flomesh software loadbalancer also has its own ELB implementation, which has two modes. The first mode is in a flat network, where Flomesh software load directly announces a specific external IP to the BGP router through BGP announcement. The second mode exists outside the container platform, where a Flomesh Layer 4 load-balancing cluster is in place. In this scenario, Flomesh ELB will automatically register the forwarding rules of ExternalIP on the Layer 4

load balancing cluster outside the cluster, creating a "Layer 4 load balancing outside the cluster + Ingress Layer 7 load balancing" network solution. For traditional non-BGP networks, Flomesh software load offers a customized ARP-based ELB service, which is currently not part of the standard product and is optional and accessible only through Flomesh expert customization service.

5.9 Load Balancing for Application Interfaces (APIs)

Flomesh offers comprehensive load balancing capabilities at the API level for REST API users. By specifying the API using a combination of "domain name + path + HTTP method," users can manage and balance traffic for that API. Flomesh functions as an API gateway in this mode.

In addition to API load balancing, Flomesh supports several levels of load balancing:

- Load balancing for internet-facing entrances, including DNS load balancing, Layer 4 load balancing, reverse proxying, and load balancing for Layer 7.
- Load balancing for subnet entrances, used for building Layer 4 and Layer 7 network boundaries.
- Load balancing for hosts and instances, like the usage and functionality of traditional load-balancing hardware.
- Load balancing for container platforms, covering Ingress, Egress, and Multi-Cluster Service Management (MCS).
- Load balancing for microservices, including DNS service discovery and client load balancing.
- Load balancing for application interfaces (APIs).

5.10 Graphical Web Console

The Flomesh software load balancer has an open-source graphical user console available on GitHub at [GitHub - flomesh-io/traffic-guru: TrafficGuru is one stop GUI for cloud native traffic management of Service Mesh, Ingress, GatewayAPI, ELB, MCS, API management.]().

This console allows users to manage most functionalities of the load balancer visually, but some customizations may be necessary based on individual customer needs. Along with multi-tenant load balancing management capabilities, the console also offers customizable dashboards. Users can create their own dashboards or request customization from the Flomesh support team. The console operates in a container environment by default and can be deployed using the Helm chart. Basic Kubernetes management capabilities are included, along with a web terminal for convenient administration and management in a browser.