KubeCon | CloudNativeCon

Europe 2018

gRPC Loadbalancing on Kubernetes

Jan Tattermusch (GitHub:@jtattermusch), Google

Why Load Balancing?

- Build scalable services
- Improve throughput, decrease latency
- Avoid overloading of a single backend
- Improved Tolerance for backend failures
- Allows updating service on the fly

LB is of key importance in microservice architecture

Concepts: L4 vs L7

Connection based (L4) vs **Stream-based (L7)** balancing:

- What is the granularity of picking a backend?
- L4 works fine for HTTP1.1/REST APIs
- gRPC uses HTTP/2: every RPC is a separate stream in the *same* TCP/IP connection
- L7 LB needed for gRPC traffic
- Potential Problem: Kubernetes LB is only L4 (= in service types `ClusterIP` and `LoadBalancer)`

Concepts: Client LB vs Proxy LB

KubeCon | CloudNativeCon
Europe 2018

Proxy LB

+ simple client, untrusted clients are fine
- higher overhead & latency

"Sidecar" deployment possible on Kubernetes

Client LB

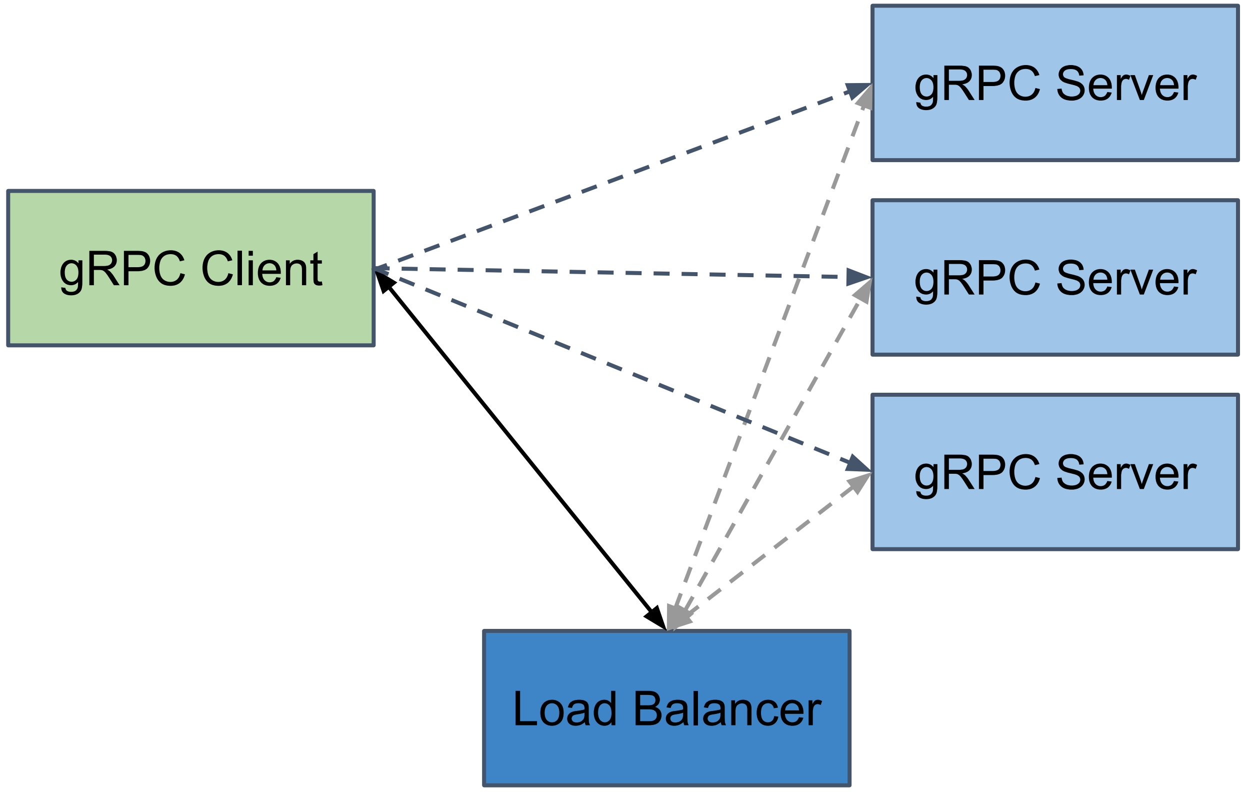
+ low latency, low overhead, no proxy management
- only good for simple LB logic

gRPC implements RoundRobin and "grpclb" lookaside

KubeCon CloudNativeCon Europe 2018

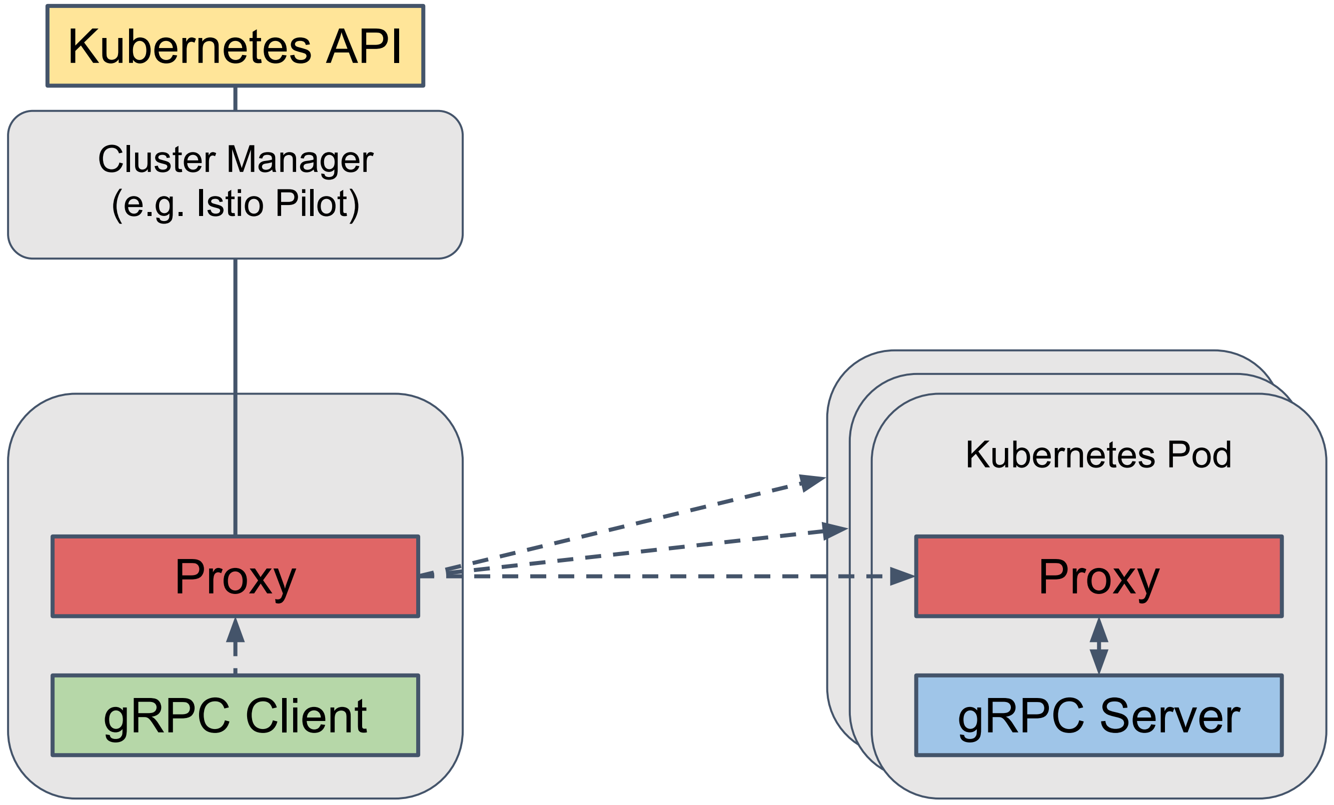
Client Lookaside LB

- Complex logic implemented by Balancer
- Extensible
- Can accommodate server load info



Service Mesh LB

- proxy deployed as a service side-car
- LB performed by the proxy
- many additional features available



gRPC Loadbalancing Options

Proxy LB

- Envoy
- nginx (full gRPC support from Mar 2018)
- proxies that support both HTTP/2 and LB should work

Proxy LB in a Service Mesh

- Envoy / Istio
- Linkerd

gRPC Loadbalancing Options

Client LB

- simple built-in RoundRobin loadbalancer (comes with gRPC)

Lookaside Client LB

- client talks to a balancer that implements simple **grpclb** protocol and instructs how to balance the load
- **grpclb** client is built into gRPC library
- Problem: **grpclb** server implementation not available publicly

Future of gRPC Lookaside LB

- Envoy uses Universal data plane API to discover endpoints
- gRPC will implement **Universal data plane API** support
  - API adjustments might be needed
- 2 possible deployment models
  - Envoy proxy does the lookaside load balancing (AVAILABLE NOW)
  - gRPC client consumes data plane API directly (as **grpclb** alternative) - NOT AVAILABLE YET
- **grpclb** will continue to be supported

Example: Round Robin LB

https://github.com/jtattermusch/grpc-loadbalancing-kubernetes-examples

How to do this in Kubernetes

- Use "headless" service (`clusterIP: None`) to expose all replicas as DNS entry
- Set loadbalancing policy in gRPC clients
  `new ChannelOption("grpc.lb_policy_name", "round_robin")`
- Connect to the service as usual
  `new Channel("greeter-server.default.svc.cluster.local:8000", …)`

+ Simple setup, works out of the box
- does not take server load into account
- handling "scale up" correctly requires a workaround

Example: LB with Envoy sidecar

https://github.com/jtattermusch/grpc-loadbalancing-kubernetes-examples

Statically configured Envoy proxy

- Use "headless" service (`clusterIP: None`) to expose all replicas as DNS entry
- Setup Envoy proxy as a sidecar container
- Direct all client traffic to the envoy proxy
- Use Envoy's STRICT_DNS cluster type

Dynamically configured Envoy proxy

- Install istio (or other cluster manager)
- Deploy client with a sidecar using "istioctl kube-inject"
- Connect to the service as usual
  `new Channel("greeter-server.default.svc.cluster.local:8000", ...)`
- Envoy will obtain configuration from Cluster Manager (istio pilot)

Example: LB in Service Mesh

https://github.com/jtattermusch/grpc-loadbalancing-kubernetes-examples

How to do this in Kubernetes

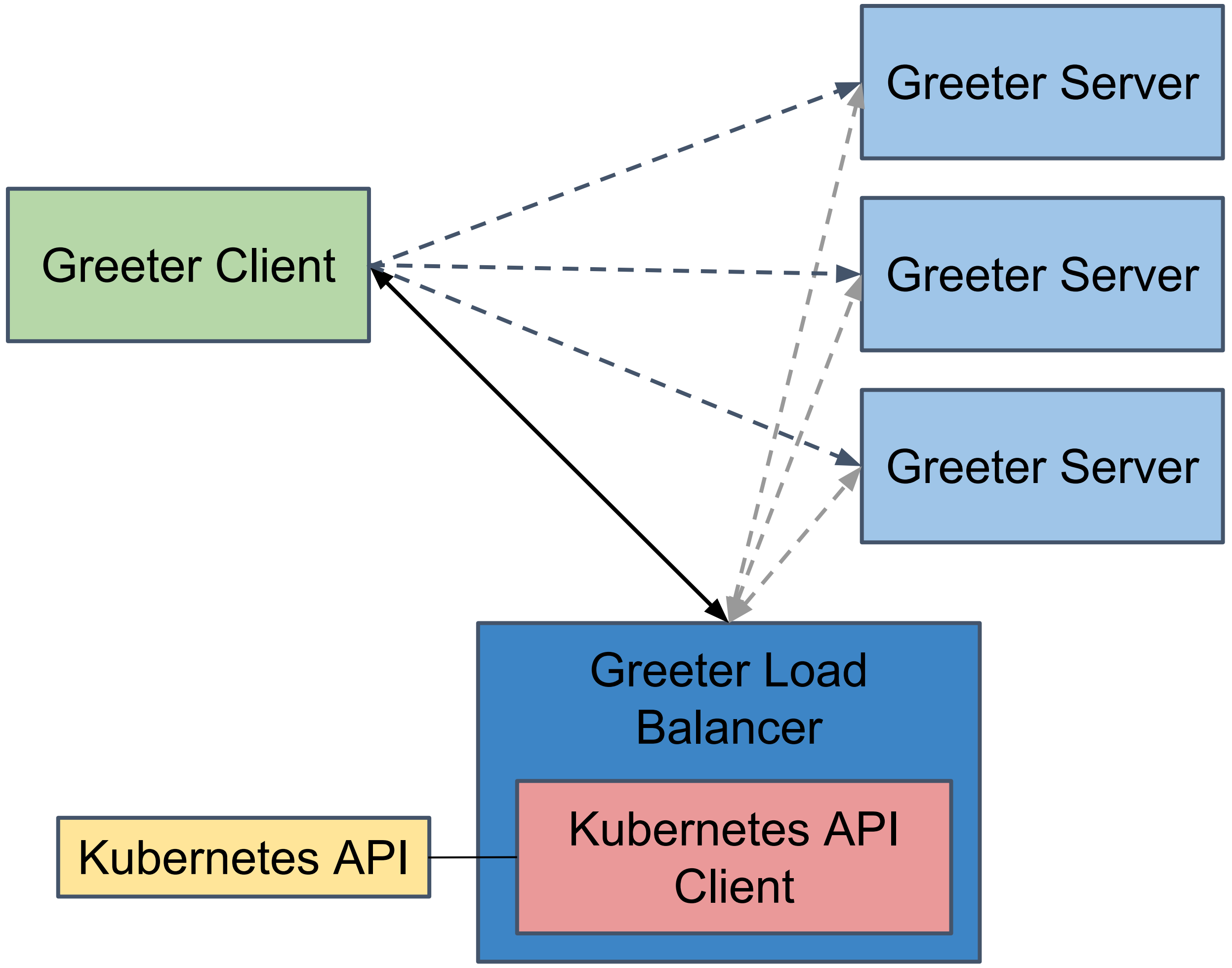
- Install istio
- Deploy server and client using "istioctl kube-inject"
- Use port names "grpc" or "grpc-mysuffix" for your service otherwise route rules (and load balancing) won't work
- Connect to the service as usual
  `new Channel("greeter-server.default.svc.cluster.local:8000", ...)`

Example: Lookaside LB

KubeCon | CloudNativeCon Europe 2018

Simplified scenario with external load balancer

- client discovers balancer via SRV DNS records
- balancer watches backend list via Kubernetes Endpoint API.



Example: Lookaside LB

How to do this in Kubernetes:

- Expose a **named port** called "grpclb" for the balancer service (=> publishes the necessary _grpclb._tcp.yourservice.default.internal SRV records)
- Use a headless balancer service and headless backend service

Implement the balancer service

- Dummy **grpclb** server in our example - only watches the available backends using kubernetes API and published the server list.

Balancing streaming RPCs

- Traditional RPCs are short-lived
- gRPC load balancing is done per-call
  - Fine for single request - single response
  - Potentially no balancing is happening for long-lived streaming calls.
  - It is difficult to assign weights to streaming calls - we don't know how long they are going to take

What to do

- restart streaming calls periodically
- can set MAX_CONNECTION_AGE to limit lifetime of connections
- keep this in mind when designing APIs

What if I can only do L4 LB?

Poor man's approach to making L4 load balancing less bad for gRPC

- Set `grpc.max_connection_age_ms` on your servers
- Established connections will reconnect periodically -> connection based LB can kick in.
- Can be use with Kubernetes 'LoadBalancer' and 'ClusterIP' services.

Reference:

https://github.com/grpc/proposal/blob/master/A9-server-side-conn-mgt.md

Conclusion

gRPC office hours in CNCF booth

- Thu 11:30 - 12:30

Other gRPC talks

- "gRPC Deep Dive" - Thu 16:35

Please fill out feedback survey

https://bit.ly/2HsEMcS

Resources

KubeCon | CloudNativeCon

Europe 2018

Examples repository

https://github.com/jtattermusch/grpc-loadbalancing-kubernetes-examples

Overview

https://github.com/grpc/grpc/blob/master/doc/load-balancing.md

https://grpc.io/blog/loadbalancing

Other useful links

https://blog.envoyproxy.io/the-universal-data-plane-api-d15cec7a

https://www.nginx.com/blog/nginx-1-13-10-grpc/