Universidade Federal de Pernambuco

Departamento de Ciência da Computação

Curso de Ciência da Computação

Guia para Balanceamento de carga do gRPC em ambientes Kubernetes

Proposta de Trabalho de Conclusão de Curso de Graduação

por

João Filipe da Matta Ribeiro Moura

Orientador: Prof. Nelson Souto Rosa

Recife, Junho / 2021

RESUMO

Kubernetes se tornou um padrão da indústria em orquestração de containers, indispensável no uso de microserviços, traz integração com balanceadores de carga das principais soluções do mercado, que facilitam escalabilidade e monitoramento.

O protocolo gRPC (g Remote Procedure Call) vem ganhando relevância na indústria devido sua capacidade de transportar dados de maneira eficiente, estruturada e retrocompatível. gRPC é mais eficiente que outras soluções pelo fato de ele manter conexões persistentes que idealmente duram todo o tempo de vida da aplicação. Essa característica implica em alguns desafios ao balancear carga, fazendo com que alguns dos balanceadores do tipo proxy, mais utilizados tenham que implementar módulos de balanceamento de carga especial. Além disso, nem sempre um balanceador de carga da camada de aplicação é a melhor escolha para um dado contexto.

Neste trabalho será descrito como algumas das principais soluções de balanceamento de carga interagem com o gRPC em Kubernetes, e serão definidos parâmetros para a escolha mais eficiênte do balanceador de carga, bem como testes comparativos para identificar qual o melhor balanceador em cada cenário definido.

1 INTRODUÇÃO

TODO

1.1 Métodos Disponíveis

Existem dois grandes métodos para fazer balanceamento de carga do gRPC. Balanceamento no lado do cliente e balanceador do tipo Proxy. Dentro desses dois há várias ramificações, as principais serão abordadas a seguir, juntamente com seus pontos positivos e negativos.

1.1.1 Balanceamento no lado do cliente

Nesse método cada uma das réplicas do cliente irá se responsabilizar por balancear a carga entre as réplicas do servidor. Para isso, o cliente abre um conjunto de conexões, uma com cada réplica do servidor. A cada operação o algorítmo de balanceamento irá escolher uma conexão para utilizar.

Existe ainda um outro componente nessa solução, que é um observador, que em um intervalo determinado, ou na ocasião de um erro de conexão, irá atualizar o conjunto de conexões para garantir que este esteja sempre atualizado, visto que uma réplica do servidor por ser encerrada, ou mais réplicas serem criadas. Sem esse componente poderiam haver sucessivos erros de conexão, por conta de servidores que foram desligados, ou haver um desbalanceamento de carga, já que as novas réplicas do servidor nunca receberiam requisições.

O principal ponto positivo em relação a balanceadores do tipo proxy é a performance. O balanceamento no lado do cliente não tem a necessidade de fazer uma parada extra exigida no caso do proxy, o que adiciona latência, principalmente por conta do processamento extra necessário.

O principal ponto negativo é o custo de implementação e manutenção. Cada cliente precisará implementar o seu método de balanceamento, o que aumenta o número de pontos de falha, podendo causar sobrecargas nos servidores de produção e dificultar diagnóstico em caso de interrupções de serviço em produção.

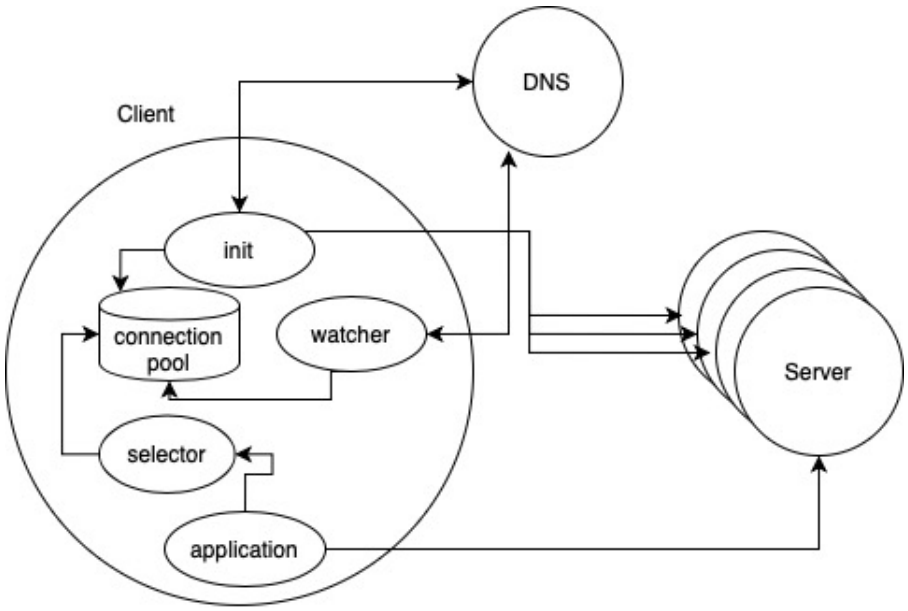

Figura 1: Blanceamento no lado do cliente.
Fonte: https://www.linkedin.com/feed/.

1.1.2 Balanceador Proxy

Balanceadores do tipo proxy funcionam como um ponto central que tem um conjunto de conexões para cada réplica do servidor, e assim como o balanceador no lado do cliente também observa por mudanças no conjunto de servidores.

A grande diferença é que no caso do proxy toda essa complexidade fica concentrada nesse agente externo e abstrai a responsabilidade de balanceamento do cliente. Tudo que um cliente precisa fazer nesse caso é abrir uma conexão (ou um conjunto de conexões) com o proxy, e ele lida com o restante.

As principais vantagem desse método são a maior simplicidade de implementação e manutenção, e a facilita também a observabilidade, o que ajuda na investigação durante interrupções de serviço.

As principais desvantagens são adição de latência devido a processamento extra e por se tornar um único ponto de falha, podendo comprometer a disponibilidade do servidor se não for bem mantido.

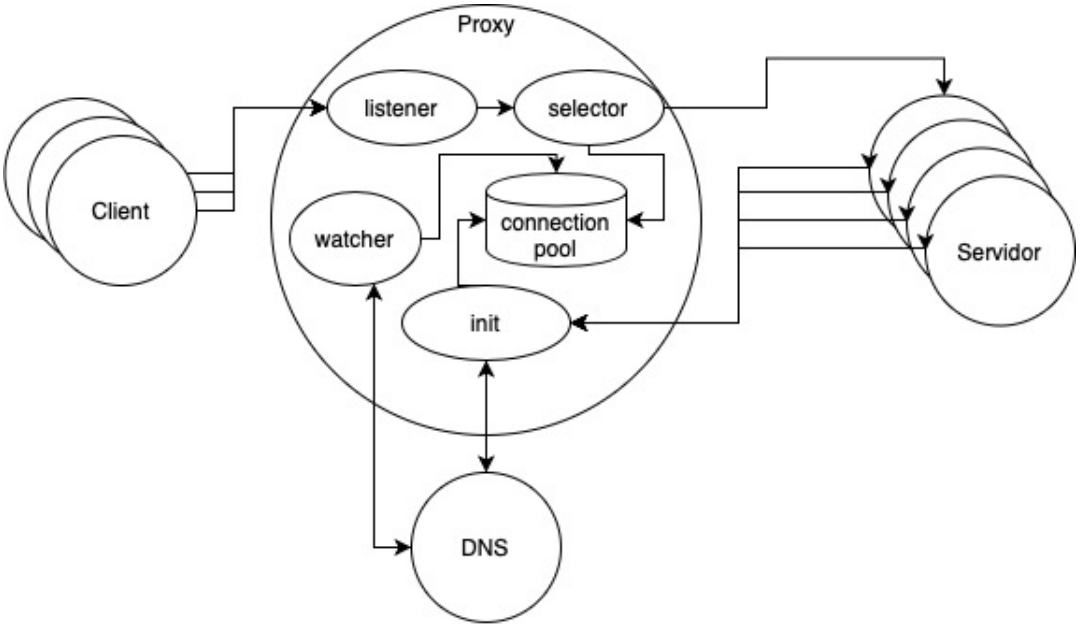

Figura 2: Balanceador Proxy.
Fonte: https://www.spirometry.guru/.

2 OBJETIVOS

Este trabalho tem como objetivo elaborar um guia para escolha da melhor solução de balanceamento de carga para cada cenário de um conjunto pré-definido em Kubernetes.

- Definir parâmetros para medir as seguintes características do balanceador de carga: nível de balanceamento de carga, a resiliência em casos de inserção ou remoção de uma réplica do serviço, escalabilidade, latência adicionada na comunicação entre os sistemas e monitoramento.
- Definir cenários de carga a serem analisados.
- Implementar um gerador de carga que simule todos os tipos de carga definidos.
- Configurar um cluster Kubernetes como código para facilitar a reprodução dos experimentos.
- Configurar um sistema de coleta de métricas/monitoramento.
- Criar os manifestos de lençamento dos recursos para o cluster Kubernetes.
- Implementar scripts para possibilitar diversas execuções dos experimentos.
- Compilar e analisar os dados extraídos dos experimentos.

3 METODOLOGIA

Será definido quais parâmetros serão utilizados para avaliar as características listadas no primeiro tópico dos objetivos, e quais categorias de balanceamento de carga deverão ser incluídas na avaliação, para então encontrar as soluções de balanceamento para cada categoria. Em seguida deverá também ser definido os tipos de carga que serão inclusos na avaliação e será determinado quais aplicações servirão como testes, além dos tamanhos/tipos de dados que serão trafegados, visto que todos esses pontos podem influenciar nas métricas de avaliação. Serão ainda designadas as ferramentas de monitoramento utilizadas para extração das métricas.

Com todos esses pontos definidos será iniciada a implementação. Serão desenvolvidas aplicações simples com a instrumentação necessária, clientes que gerarão diferentes tipos de carga e um servidor que irá responder às requisições, fazer todos os manifestos para o lançamento no cluster de Kubernetes, incluindo os manifestos de instalação dos recursos de balanceamento de carga e a instalação da ferramenta de monitoramento, lançar o cluster Kubernetes via código e executar os experimentos.

Com os experimentos executados serão compilados os dados e serão feitas todas as análises, visando a recomendação de pelo menos uma tecnologia de balanceamento de carga para cada um dos tipos de carga definidos.

4 CRONOGRAMA

	Periodo															
Junho	**Julho**				**Agosto**				**Setembro**				**Outubro**			
Revisão Bibliográfica	X	X	X	X												
Implementação					X	X	X	X								
Experimentos								X	X	X	X					
Avaliação dos Resultados											X	X				
Escrita do TG												X	X	X	X	
Apresentação														X	X	X

REFERÊNCIAS