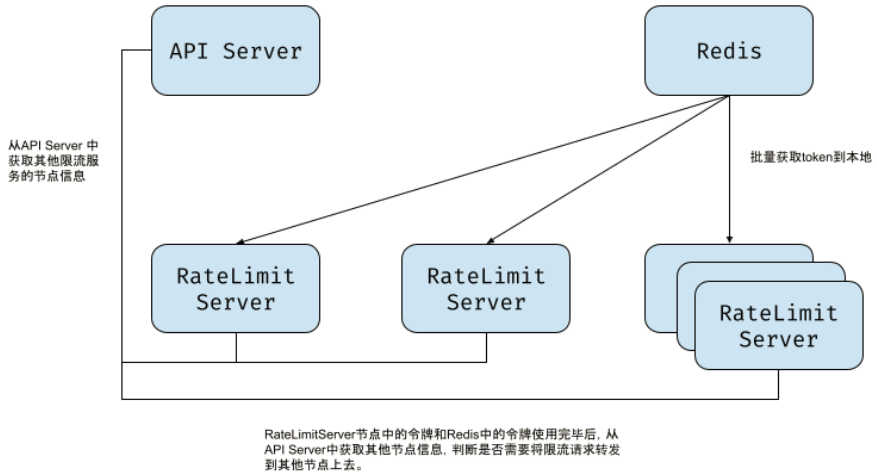
目录

高性能限流服务器



概述 基本原理如下: 组件说明：

- Redis：用于存放令牌的地方，RateLimit Server 会批量从 Redis 中获取一批令牌。
- TokenServer： 用于存储 RateLimit Server 上报的节点信息，然后将全量的节点信息下发给各个 RateLimit Server， 用于转发限流请求，提升限流精度。
- RateLimit Server：真正处理 Envoy 发过来的限流请求，内部启动一个协程，当令牌数不充足时， 从 Redis 中批量获取一批，如果本地无令牌可用，同时 Redis 中的令牌也取完了，可以根据配置选择 是否将限流请求转移到其他还有剩余令牌的节点上去。

原理如下：

- 用令牌桶算法来替换计数器固定窗口算法。
- 设计成分布式令牌桶算法，后端存储使用 Redis，采用“批量”的思想，大大减少对 Redis 的请求， 提升性能。
- Redis 中令牌不足时支持限流请求转移到还有剩余令牌的节点上等方式大大减少限流误差。

限流配置变化

相比较社区的限流配置，新增了 forward_enable , token_per_fetch 两个配置。

- forward_enable 默认值为 false，用于控制在 Redis 中的令牌取完了，同时内存中令牌数也消耗 没了，将限流请求转发给其他节点上去。
- token_per_fetch 默认值为10，用于配置每次批量从 Redis 中获取的令牌数。 > 注意：开启了转发限流请求时，会有一定的性能损耗，所以在限流精度和高性能上需要做一些取舍。

日志级别 默认日志级别为 info，可以通过以下方式动态调整某个限流服务器日志级别：

```
$ curl 127.0.0.1:8080
/debug/pprof/: root of various pprof endpoints. hit for help.
/debug/pprof/profile: CPU profiling endpoint
/debug/pprof/trace: trace endpoint
/healthcheck:
/logging: action=post, query/change logging levels, example curl -d '' 'http://127.0.0.1:8080/logging?level=debug'
/metrics:
/rlconfig: print out the currently loaded configuration for debugging

# 动态将日志级别调整成 debug
$ curl -d '' 'http://127.0.0.1:8080/logging?level=debug'
debug
```

后端存储Redis配置参数

Redis 的配置都是以环境变量的形式对外提供的。

```
// Redis settings
RedisType     string `envconfig:"REDIS_TYPE" default:"single"`
RedisUrl      string `envconfig:"REDIS_URL" default:"127.0.0.1:6379"`
RedisPoolSize int    `envconfig:"REDIS_POOL_SIZE" default:"10"`
RedisAuth     string `envconfig:"REDIS_AUTH" default:""`
RedisTls      bool   `envconfig:"REDIS_TLS" default:"false"`
```

异常情况对限流服务器的影响

序号	故障描述	影响	当前措施和原因
2	pod级别 -- pod触发OOM （due to container limit reached）	有影响，导致总令牌数减少，通过的请求变少，最大值 token_per_fetch	原因：pod中的内存达到了它的资源限制，因为资源限制是通过 Linux 的 cgroup 实现的，所以 cgroup 会将此容器强制杀掉，类似于 kill -9。
4	pod级别 -- 主动删除 pod	没有影响	程序监听了相应的信号，会触发令牌回收的操作，将程序内存中的令牌回收到后端存储（Redis）
5	pod级别 -- pod被驱逐	没有影响	程序监听了相应的信号，会触发令牌回收的操作，将程序内存中的令牌回收到后端存储（Redis）
1	pod级别 -- pod重启	没有影响	程序监听了相应的信号，会触发令牌回收的操作，将程序内存中的令牌回收到后端存储（Redis）
3	pod级别 – pod触发OOM （due to node limit overcommit）	有影响，导致总令牌数减少，通过的请求变少，最大值 token_per_fetch	原因：节点中的内存达到了上限，触发了 Linux 的 OOM Killer 程序开始强制杀掉 oom_score_adj 得分高的进程，类似于 kill -9。
6	pod级别 – 程序 hang 住了	没有影响	程序添加以下探针，hang住之后会导致探针不过，触发 pod 流量摘取和重启pod的操作。 • 就绪探针。 • 存活探针。

metrics

序号	名称	类型	说明
1	go_gc_duration_seconds	summary	A summary of the pause duration of garbage collection cycles.
2	go_goroutines	gauge	Number of goroutines that currently exist.
3	go_info	gauge	Information about the Go environment.
4	go_memstats_alloc_bytes	gauge	Number of bytes allocated and still in use.
5	go_memstats_alloc_bytes_total	counter	Total number of bytes allocated, even if freed.
6	go_threads	gauge	Number of OS threads created.
7	go_memstats_*		Stats about memory.
8	grpc_server_handled_total	counter	Total number of RPCs completed on the server, regardless of success or failure.
9	grpc_server_handling_seconds	histogram	Histogram of response latency (seconds) of gRPC that had been application-level handled by the server.
10	grpc_server_msg_received_total	counter	Total number of RPC stream messages received on the server.
11	grpc_server_msg_sent_total	counter	Total number of gRPC stream messages sent by the server.
12	grpc_server_started_total	counter	Total number of RPCs started on the server.

示例 前提： 安装了 istio。 安装步骤： 1.部署应用程序。

```
$ kubectl apply -f example.yaml
service/sample-http-server created
serviceaccount/sample-http-server created
deployment.apps/sample-http-server created
gateway.networking.istio.io/sample-http-server-gateway created
virtualservice.networking.istio.io/sample-http-server created
```

2.部署 redis 和 限流服务器的组件。

```
$ kubectl apply -f redis.yaml
service/redis created
deployment.apps/redis created
```

```
$ kubectl  apply -f rate-limit-service.yaml
configmap/ratelimit-config created
service/ratelimit-token-server created
deployment.apps/ratelimit-token-server created
service/ratelimit created
deployment.apps/ratelimit created
```

3.部署 envoyfilter

```
$ kubectl  apply -f envoyfilter.yaml
envoyfilter.networking.istio.io/filter-ratelimit-svc created
envoyfilter.networking.istio.io/filter-ratelimit created
```

4.跟社区版如何切换

```
$ kubectl delete -f rate-limit-service.yaml
configmap/ratelimit-config deleted
service/ratelimit-token-server deleted
deployment.apps/ratelimit-token-server deleted
service/ratelimit deleted
deployment.apps/ratelimit deleted

$ kubectl create -f shequ-rate-limit-service.yaml
configmap/ratelimit-config created
service/ratelimit created
deployment.apps/ratelimit created
```

验证

执行以下命令：

```
$ curl -I -X GET http://10.254.89.211:9080/api/v1/products
10.254.89.211为 sample-http-server 服务的 ClusterIP。
```

查询 Ratelimit Server 的日志，得到以下结果：

```
I0825 07:28:01.177011       1 ratelimit.go:149] applying limit: ratelimit={requests_per_unit=10000, unit=SECOND, unlimited=false, forward_enable=false, token_pre_fetch_ratio=
0.01}; descriptor: (PATH=/api/v1/products)
I0825 07:28:01.177082       1 cache_impl.go:222] "looking up cache key" key="productpage-ratelimit_PATH_/api/v1/products_1629876481"
I0825 07:28:01.180616       1 cache_impl.go:83] "acquire token from the storage" cacheKey="productpage-ratelimit_PATH_/api/v1/products_1629876481" tokenPerFetch=100 storageT
okenCounts=9900
I0825 07:28:01.180648       1 base_limiter.go:55] cache key: productpage-ratelimit_PATH_/api/v1/products_1629876481 currentToken: 99
I0825 07:28:01.180622       1 cache_impl.go:187] "set the expire of the cacheKey in storage" cacheKey="productpage-ratelimit_PATH_/api/v1/products_1629876481" ttlSecond=61
```

可以看到，当限流请求过来的时候，RateLimit Server 会一次批量从 Redis 从获取100个令牌，大大减少了跟 Redis 的交互，提升了性能。

示例yaml

- example.yaml

```
##################################################################################################
# apps
##################################################################################################
apiVersion: v1
kind: Service
metadata:
name: sample-http-server
labels:
app: sample-http-server
spec:
ports:
- port: 9080
  name: http
selector:
app: sample-http-server
---
apiVersion: v1
kind: ServiceAccount
metadata:
name: sample-http-server
labels:
account: sample-http-server
---
apiVersion: apps/v1
kind: Deployment
metadata:
name: sample-http-server
labels:
app: sample-http-server
spec:
replicas: 1
selector:
matchLabels:
  app: sample-http-server
template:
metadata:
  labels:
    app: sample-http-server
spec:
  serviceAccountName: sample-http-server
  containers:
    - name: sample-http-server
      image: hub.kce.ksyun.com/kasm-public/sample-http-server:v0.0.1
      imagePullPolicy: IfNotPresent
      ports:
      - containerPort: 9080
---
##################################################################################################
# networking
##################################################################################################
apiVersion: networking.istio.io/v1alpha3
kind: Gateway
metadata:
name: sample-http-server-gateway
spec:
selector:
istio: ingressgateway # use istio default controller
servers:
- port:
  number: 80
```

```
    name: http
    protocol: HTTP
  hosts:
  - "*"
  ---
  apiVersion: networking.istio.io/v1alpha3
  kind: VirtualService
  metadata:
  name: sample-http-server
  spec:
  hosts:
  - "*"
  gateways:
  - sample-http-server-gateway
  http:
  - match:
  - uri:
      prefix: /api/v1/products
  route:
  - destination:
      host: sample-http-server
      port:
        number: 9080
```

- redis.yaml

```
apiVersion: v1
kind: Service
metadata:
name: redis
labels:
app: redis
spec:
ports:
- name: redis
port: 6379
selector:
app: redis
---
apiVersion: apps/v1
kind: Deployment
metadata:
name: redis
spec:
replicas: 1
selector:
matchLabels:
  app: redis
template:
metadata:
  labels:
    sidecar.istio.io/inject: "false"
    app: redis
spec:
  containers:
  - image: hub.kce.ksyun.com/kasm-public/redis:alpine
    name: redis
    ports:
    - name: redis
      containerPort: 6379
  restartPolicy: Always
  serviceAccountName: ""
```

- rate-limit-service.yaml

```
##################################################################################################
# Redis service and deployment
# Ratelimit service and deployment
```

Note: a configmap is needed to make the rate limit deployment work properly, for example:

apiVersion: v1 kind: ConfigMap metadata: name: ratelimit-config data: config.yaml: | domain: productpage-ratelimit descriptors:

- key: PATH value: "/api/v1/products" rate_limit: unit: second requests_per_unit: 10000 forward_enable: false token_per_fetch: 20

---

apiVersion: v1 kind: Service metadata: name: ratelimit labels: app: ratelimit spec: ports:

- name: http-port port: 8080 protocol: TCP
- name: grpc-port port: 8081 protocol: TCP

- name: udp-port port: 8082 protocol: UDP selector: app: ratelimit

  apiVersion: apps/v1 kind: Deployment metadata: name: ratelimit spec: replicas: 1 selector: matchLabels: app: ratelimit template: metadata: labels: sidecar.istio.io/inject: "false" app: ratelimit spec: containers:

  - image: hub.kce.ksyun.com/kasm-public/kasm-ratelimit-server:v0.0.3 args:
    - server imagePullPolicy: Always name: ratelimit resources: requests: cpu: 500m memory: 100M limits: cpu: 500m memory: 100M readinessProbe: failureThreshold: 3 httpGet: path: /healthcheck port: 8080 scheme: HTTP initialDelaySeconds: 1 periodSeconds: 2 successThreshold: 1 timeoutSeconds: 1 livenessProbe: failureThreshold: 3 httpGet: path: /healthcheck port: 8080 scheme: HTTP initialDelaySeconds: 1 periodSeconds: 2 successThreshold: 1 timeoutSeconds: 1 env:
    - name: USE_STATSD value: "false"
    - name: HOST valueFrom: fieldRef: fieldPath: status.podIP
    - name: RUNTIME_ROOT value: /data
    - name: RUNTIME_SUBDIRECTORY value: ratelimit
    - name: REDIS_URL value: redis:6379 ports:
    - containerPort: 8080
    - containerPort: 8081
    - containerPort: 8082 volumeMounts:
    - name: config-volume mountPath: /data/ratelimit/config volumes:
  - name: config-volume configMap: name: ratelimit-config

- envoyfilter.yaml
```java
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
name: filter-ratelimit-svc
spec:
workloadSelector:
# select by label in the same namespace
labels:
app: sample-http-server
configPatches:
- applyTo: HTTP_FILTER
match:
context: SIDECAR_INBOUND
listener:
  filterChain:
    filter:
      name: "envoy.filters.network.http_connection_manager"
      subFilter:
        name: "envoy.filters.http.router"
patch:
operation: INSERT_BEFORE
# Adds the Envoy Rate Limit Filter in HTTP filter chain.
value:
  name: envoy.filters.http.ratelimit
  typed_config:
    "@type": type.googleapis.com/envoy.extensions.filters.http.ratelimit.v3.RateLimit
    # domain can be anything! Match it to the rate limit service config
    domain: productpage-ratelimit
    failure_mode_deny: true
    timeout: 10s
    rate_limit_service:
      grpc_service:
        envoy_grpc:
          cluster_name: outbound|8081||ratelimit.default.svc.cluster.local
      transport_api_version: V3
```

---

apiVersion: networking.istio.io/v1alpha3 kind: EnvoyFilter metadata: name: filter-ratelimit spec: workloadSelector: labels: app: sample-http-server configPatches:

- applyTo: VIRTUAL_HOST match: context: SIDECAR_INBOUND routeConfiguration: vhost: name: "inbound|http|9080" route: action: ANY patch: operation: MERGE

Applies the rate limit rules.

```
value:
  rate_limits:
    - actions: # any actions in here
        - request_headers:
            header_name: ":path"
            descriptor_key: "PATH"
```

- shequ-rate-limit-service.yaml
```java
# Note: a configmap is needed to make the rate limit deployment work properly, for example:
apiVersion: v1
kind: ConfigMap
metadata:
name: ratelimit-config
data:
config.yaml: |
domain: productpage-ratelimit
descriptors:
  - key: PATH
    value: "/api/v1/products"
    rate_limit:
      unit: second
      requests_per_unit: 10000
---
```

apiVersion: v1 kind: Service metadata: name: ratelimit labels: app: ratelimit spec: ports:

- name: http-port port: 8080 targetPort: 8080 protocol: TCP
- name: grpc-port port: 8081 targetPort: 8081 protocol: TCP

- ## name: http-debug port: 6070 targetPort: 6070 protocol: TCP selector: app: ratelimit

  apiVersion: apps/v1 kind: Deployment metadata: name: ratelimit spec: replicas: 1 selector: matchLabels: app: ratelimit template: metadata: labels: app: ratelimit spec: containers:

  - image: hub.kce.ksyun.com/kasm-public/ratelimit:6aaad7c3 imagePullPolicy: Always name: ratelimit command: ["/bin/ratelimit"] env:
    - name: LOG_LEVEL value: debug
    - name: REDIS_SOCKET_TYPE value: tcp
    - name: REDIS_URL value: redis:6379
    - name: USE_STATSD value: "false"
    - name: RUNTIME_ROOT value: /data
    - name: RUNTIME_SUBDIRECTORY value: ratelimit
    - name: RUNTIME_WATCH_ROOT value: "false"
    - name: RUNTIME_IGNOREDOTFILES value: "true" ports:
    - containerPort: 8080
    - containerPort: 8081
    - containerPort: 6070 volumeMounts:
    - name: config-volume mountPath: /data/ratelimit/config volumes:
  - name: config-volume configMap: name: ratelimit-config