SERVICE MESH SUMMIT 2022
服务网格峰会

主办方 云原生社区 Cloud Native Community

轻舟服务网格的无侵入增强 Istio 经验

方志恒（网易数帆 云原生技术专家）

目录

关于侵入、无侵入

- **立目标，求上得中**
- **为什么强调“无侵入”？**
  - 业务适配、快速落地、定制需求 等，有太多“侵入”的理由
  - 长期维护、社区对齐、版本演进 等，一分“侵入”一分成本

定开、维护的经验

1. **原生的API，无侵入扩展**

- 直接使用
- 做上层的封装、转换

“计算机科学领域的任何问题都可以通过增加一个间接的中间层来解决”

2. **将扩展增强内容封装在库中做最小修改、替换**

3. **保持一致性的继续“前行”**

撸猫原则： 如果你非要撸猫，记得顺着毛撸

服务与配置扩展 - 配置

- 多configSource + MCP-over-xds ①

```
configSources:
- address: xds://mesh-registry.istio-system.svc:16010?type=serviceentry&type=sidecar
- address: k8s://
```

- istio-mcp“平替” 原生的`Adsc` ②
  - 增量推送
  - 增强的revision机制
  - 更灵活的分派： ”type“ selector

服务与配置扩展 - 服务

- `ServiceEntry` ①

```
configSources:
- address: xds://mesh-registry.istio-system.svc:16010?type=serviceentry&type=sidecar
- address: k8s://
```

- MCP-over-xds（istio-mcp） ①
- mesh-registry（MCP server） ①
  - dubbo/zk
  - eureka
  - nacos
  - ...

插件扩展 - EnvoyPlugin ①


```yaml
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: bookinfo-gateway-sampling
  namespace: istio-system
spec:
  configPatches:
  - applyTo: HTTP_ROUTE
    match:
      context: GATEWAY
      routeConfiguration:
        portNumber: 80
        vhost:
          name: "*:80"
    patch:
      operation: MERGE
      value:
```

```yaml
apiVersion: networking.istio.io/v1alpha3
kind: EnvoyFilter
metadata:
  name: reviews-lua
  namespace: bookinfo
spec:
  workloadSelector:
    labels:
      app: reviews
  configPatches:
    # The first patch adds the lua filter to the listener/http connection manager
  - applyTo: HTTP_FILTER
    match:
      context: SIDECAR_INBOUND
      listener:
        portNumber: 8080
        filterChain:
          filter:
            name: "envoy.filters.network.http_connection_manager"
            subFilter:
              name: "envoy.filters.http.router"
    patch:
      operation: INSERT_BEFORE
```

插件扩展 - EnvoyPlugin ①

SERVICE MESH SUMMIT 2022 服务网格峰会

- 问题
  - 因为操作的是xDS内容，所以使用者不得不去理解一些下层概念（vhost、routeconfiguration、特殊http filter如router 等）同时也耦合了istio的特性实现
  - 插件内容这个自然没办法，但内容以外的部分，可以考虑以istio语义来表达

详见 slime-plugin

```
apiVersion: microservice.slime.io/v1alpha1
kind: PluginManager
metadata:
  name: my-plugin
  namespace: default
spec:
  workload_labels:
    app: my-app
  plugin:
  - enable: true          # switch
    name: {{plugin-1}}      # plugin name
    inline:
      settings:
        {{plugin_settings}} # plugin settings
  # ...
  - enable: true
    name: {{plugin-N}}
```

```
apiVersion: microservice.slime.io/v1alpha1
kind: EnvoyPlugin
metadata:
  name: reviews-ep
  namespace: istio-samples
spec:
  workloadSelector:
    labels:
      app: reviews
  gateway:
  - gateway-system/prod-gateway
  host:
  - reviews.istio-samples.svc.cluster.local
  route:
    - ratings.istio-samples.svc.cluster.local:80/default
    - prefix-route1
  plugins:
  - name: envoy.filters.network.ratelimit
    enable: true
    inline:
      settings:
        {{plugin_settings}} # plugin settings
```

插件扩展 - EnvoyPlugin ①

类似的思路，我们还做了智能限流等模块，见 smartlimiter

```
apiVersion: microservice.slime.io/v1alpha2
kind: SmartLimiter
metadata:
  name: review
  namespace: default
spec:
  sets:
    v1:
      descriptor:
      - action:
          fill_interval:
            seconds: 1
          quota: "10"
          strategy: "single"
        condition: "{{.v1.cpu.sum}}>10"
        target:
          port: 9080
```

插件扩展 - Rider

- 支持 Lua 语言扩展
- 支持 Envoy 动态加载、更新、移除 Lua 插件
- 支持定义 Lua 插件配置
- 支持自定义 Lua 插件生效范围，网关级/项目级/路由级
- 性能优于 Envoy 社区 Lua 扩展和 WASM 扩展

插件扩展 - Rider

VS envoy Lua

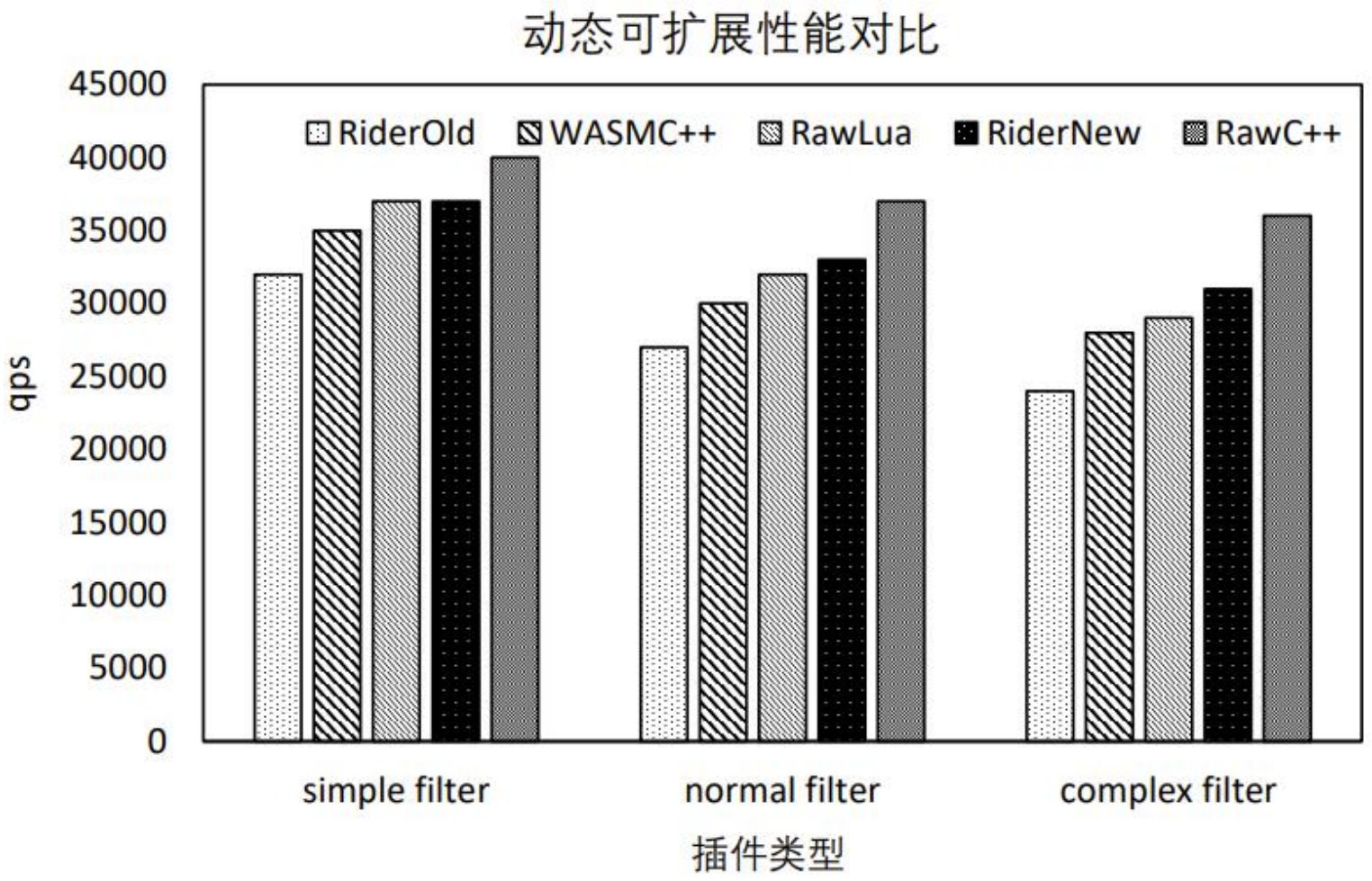
- 支持插件配置
- 支持更多API
- 更好的性能

VS envoy wasm

- 更好的易用性（脚本可见即可得）
- 便于分发（没有编译过程）
- 支持更多API
- 更好的性能

扩展方案	是否支持插件配置	是否支持动态扩展	性能	支持语言	开发复杂度
原生 C++	是	否	最优	C++	复杂
社区 Lua	否	是	较差	Lua	简单
社区 WASM	是	是	差	C++/Rust/ Go等	中等

插件扩展 - Rider


动态可扩展性能对比
45000
40000
35000
30000
25000
20000
15000
10000
5000
0
qps
RiderOld
WASMC++
RawLua
RiderNew
RawC++
simple filter
normal filter
complex filter
插件类型

插件扩展 - Rider


```
envoy.req.get_header(name)
envoy.req.get_header_size(name)
envoy.req.get_header_index(name, index)
envoy.req.get_headers()
envoy.req.get_body()
envoy.req.get_metadata(key, filter_name)
envoy.req.get_dynamic_metadata(key, filter_name)
envoy.req.get_query_parameters(max_args)
envoy.req.set_header(name, value)
envoy.req.set_headers(headers)
envoy.req.clear_header(name)
envoy.resp.get_header(name)
envoy.resp.get_header_size(name)
envoy.resp.get_header_index(name, index)
envoy.resp.get_headers()
envoy.resp.get_body()
envoy.resp.set_header(name, value)
envoy.resp.set_headers(headers)
envoy.resp.clear_header(name)
envoy.streaminfo.start_time()
envoy.streaminfo.current_time_milliseconds()
envoy.streaminfo.downstream_local_address()
envoy.streaminfo.downstream_remote_address()
envoy.streaminfo.upstream_cluster()
envoy.streaminfo.upstream_host()
envoy.logTrace(message)
envoy.logDebug(message)
envoy.logInfo(message)
envoy.logWarn(message)
envoy.logErr(message)
envoy.filelog(msg)
envoy.get_base_config()
envoy.get_route_config()
envoy.httpCall(cluster, headers, body, timeout)
envoy.respond(headers, body)
```

插件扩展 - WasmPlugin/RiderPlugin ③


```
# wasm_plugin.yaml

apiVersion: extensions.istio.io/v1alpha1
kind: WasmPlugin
metadata:
  name: test1.rider
spec:
  imagePullPolicy: IfNotPresent
  imagePullSecret: qingzhou-secret
  pluginConfig:
    destination: Body
    message: C++ is awesome!!
    source: Static
  selector:
    matchLabels:
      app: reviews
  sha256: nil
  url: oci://slimeio/rider_plugin:0.1.0
```

- **使用上**

  不能说差不多，简直是一模一样；

- **实现上**

  不能说没侵入，但只有一点点　　；

插件扩展 - WasmPlugin/RiderPlugin ③


```
route_config:
  name: local_route
  virtual_hosts:
  - name: local_service
    domains:
    - "*"
    # Plugin config here applies to the VirtualHost
    #
    # typed_per_filter_config:
    #   proxy.filters.http.rider:
    #     "@type": type.googleapis.com/proxy.filters.http.rider.v3alpha1.RouteFilterConfig
    #     plugins:
    routes:
    - match:
        prefix: "/static-to-header"
      route:
        cluster: web_service
      # Plugin config here applies to the Route
      #
      # plugins is a list of plugin route configs. Each entry has a name and its config.
      # The filter will look up the list by order given a plugin name, and use the first match entry.
      typed_per_filter_config:
        proxy.filters.http.rider:
          "@type": type.googleapis.com/proxy.filters.http.rider.v3alpha1.RouteFilterConfig
          plugins:
            - name: echo
              config:
                message: "Lua is awesome!"
                source: Static
                destination: Header
                header_name: x-echo-foo
```

```
http_filters:
- name: proxy.filters.http.rider
  typed_config:
    "@type": type.googleapis.com/proxy.filters.http.rider.v3alpha1.FilterConfig
    plugin:
      vm_config:
        package_path: "/usr/local/lib/rider/?/init.lua;/usr/local/lib/rider/?.lua;"
      code:
        local:
          filename: /usr/local/lib/rider/examples/echo/echo.lua
      name: echo
      config:
        message: "C++ is awesome!"
        source: Static
        destination: Body
- name: envoy.filters.http.router
  typed_config: {}
```

插件扩展 - WasmPlugin/RiderPlugin ③

基于以上，EnvoyPlugin对wasm/rider的支持接近完善，近期会release。

协议扩展 - dubbo ③

- 数据面
  - (L7) dubbo filters

```
"typed_per_filter_config": {
 "proxy.filters.dubbo.locallimit": {
  "@type": "type.googleapis.com/udpa.type.v1.TypedStruct",
  "type_url": "type.googleapis.com/proxy.filters.dubbo.local_limit.v2.\
    ProtoCommonConfig",
  "value": {
```

  - DRDS
  - dubbo协议嗅探

- 控制面
  - VS/DR

```
repeated HTTPRoute http = 3;
repeated TLSRoute tls = 5;

repeated DubboRoute dubbo = 102;
```

  - authn/authz（`AuthorizationPolicy`）
  - `Sidecar`/lazyload
  - `EnvoyFilter`

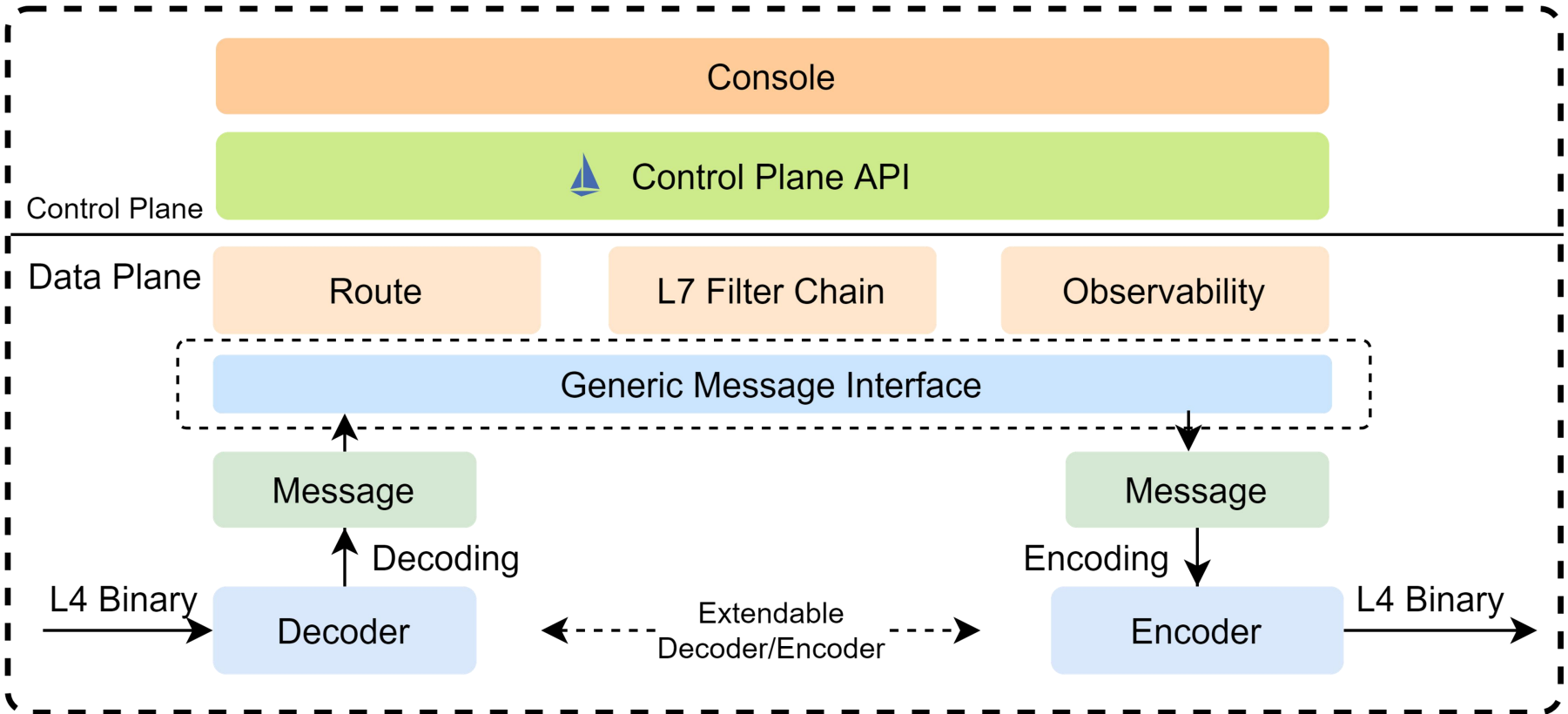
```
- applyTo: DUBBO_FILTER
match:
  context: SIDECAR_OUTBOUND
  listener:
    filterChain:
      filter:
        name: envoy.filters.network.dubbo_proxy
        subFilter:
          name: envoy.filters.dubbo.router
patch:
  operation: INSERT_BEFORE
  value:
    config:
      '@type': type.googleapis.com/udpa.type.v1.TypedStruct
      type_url: type.googleapis.com/proxy.filters.dubbo.traffic_mark.v2.\
        ProtoCommonConfig
```

协议扩展 - 通用七层扩展框架

多协议支持是服务网格落地过程中不可回避的问题，但是在数据面中添加新的七层协议支持是一件成本和复杂度非常高的事情。

除了必要的协议编解码之外，开发者这还需要实现请求路由，上下游连接管理以及事件管理，请求流抽象和生命周期管理，七层插件链实现和管理等等。如果需要

开发者只需要实现其所需要支持协议的编解码器即可直接在数据面中添加新的协议代理支持，并且所有七层能力都可以在多种不同协议之间复用。



协议扩展 - 通用七层扩展框架

- 目前： 用该通用框架实现dubbo
- 后续：
  - 替换原dubbo proxy
  - 与相关方一起推动控制面接入

关于slime - 演进

- **架构/工程**
  - 模块化设计，统一的框架层提供基础能力
  - 更多的部署形态： 单独部署、slime-boot（helm-operator）、bundle聚合
  - 代码工程： 单一项目 -> 模块拆分 -> 模块化大仓
- **更丰富的基础能力**
  - metric框架，作为上层的metric data source
  - 多集群支持
  - 统一的服务模型数据（k8s service + ServiceEntry）支持
  - 模块聚合、管理能力

```
func main() {
    module.Main("bundle", []module.Module{
            &limitermod.Module{},
            &pluginmod.Module{},
            &pilotadminmod.Module{},
            &sidecarmgrmod.Module{},
            &tracetiomod.Module{},
            &meshregistrymod.Module{},
    })
}
```

关于slime - 演进

- 更多的模块（releasing...）

  meshregistry, pilotadmin, sidecarmgr, tracetio ...

- 生态

  i9s ...

- 社区

  istio ecosystem, CNCF landscope —-> CNCF sandbox

云原生社区
Cloud Native Community

活动由云原生社区主办

SERVICE MESH
SUMMIT 2022
服务网格峰会

感谢观看