解构 Apache/Dubbo-go

dubbogo社区　于雨

CSDN

Content

个人介绍

CSDN

于雨（@AlexStocks）

- dubbogo 社区负责人
  https://github.com/dubbogo
- 基础系统从业者
  IM/NoSQL/RPC/ServiceMesh/k8s
- 开源项目贡献者
  kv: Redis/Pika/Pika-Port/etcd
  rpc：muduo/Dubbo/Dubbo-go/Sentinel-golang

1. 总体框架

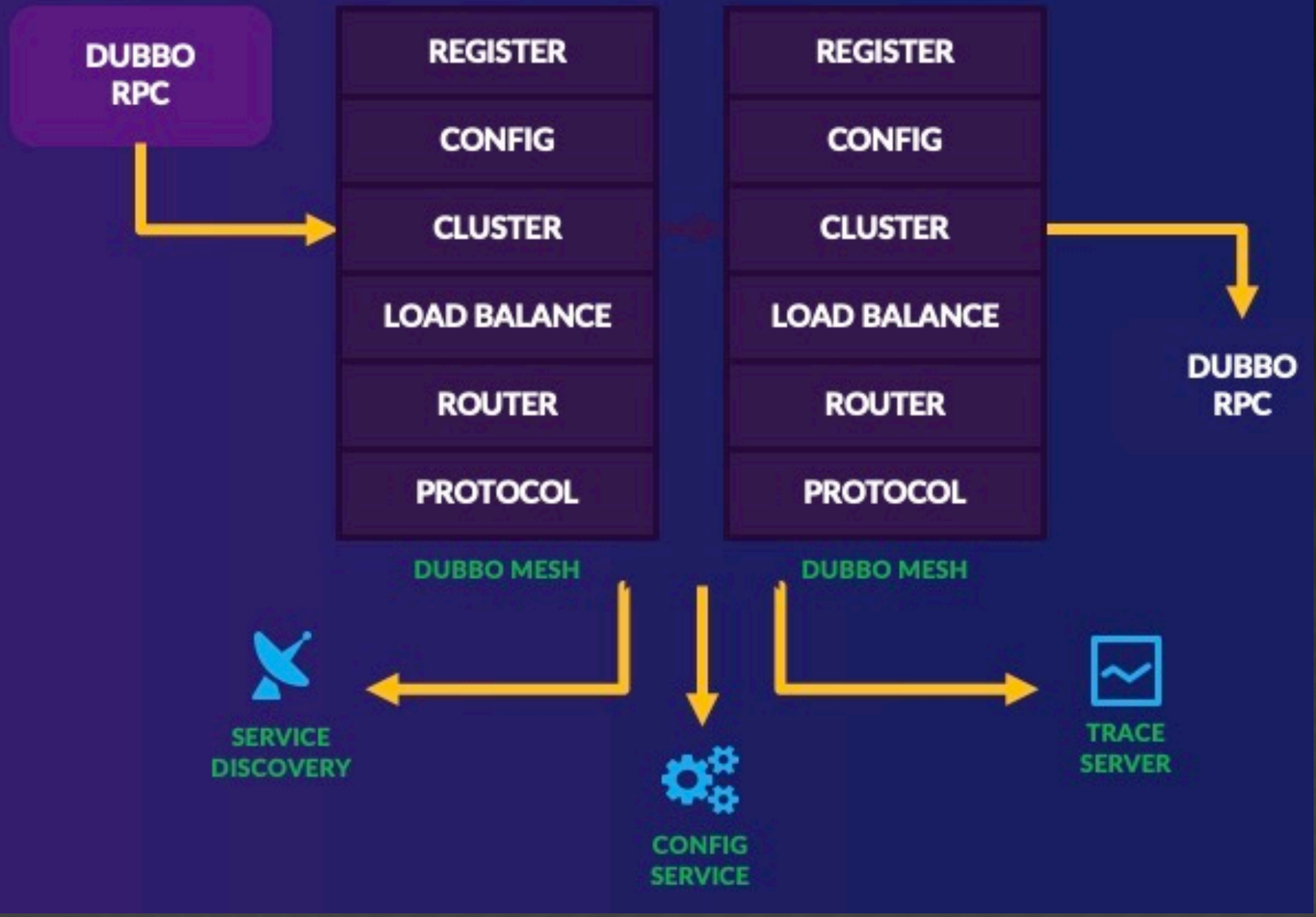
Dubbo vs Spring Cloud

服务治理能力

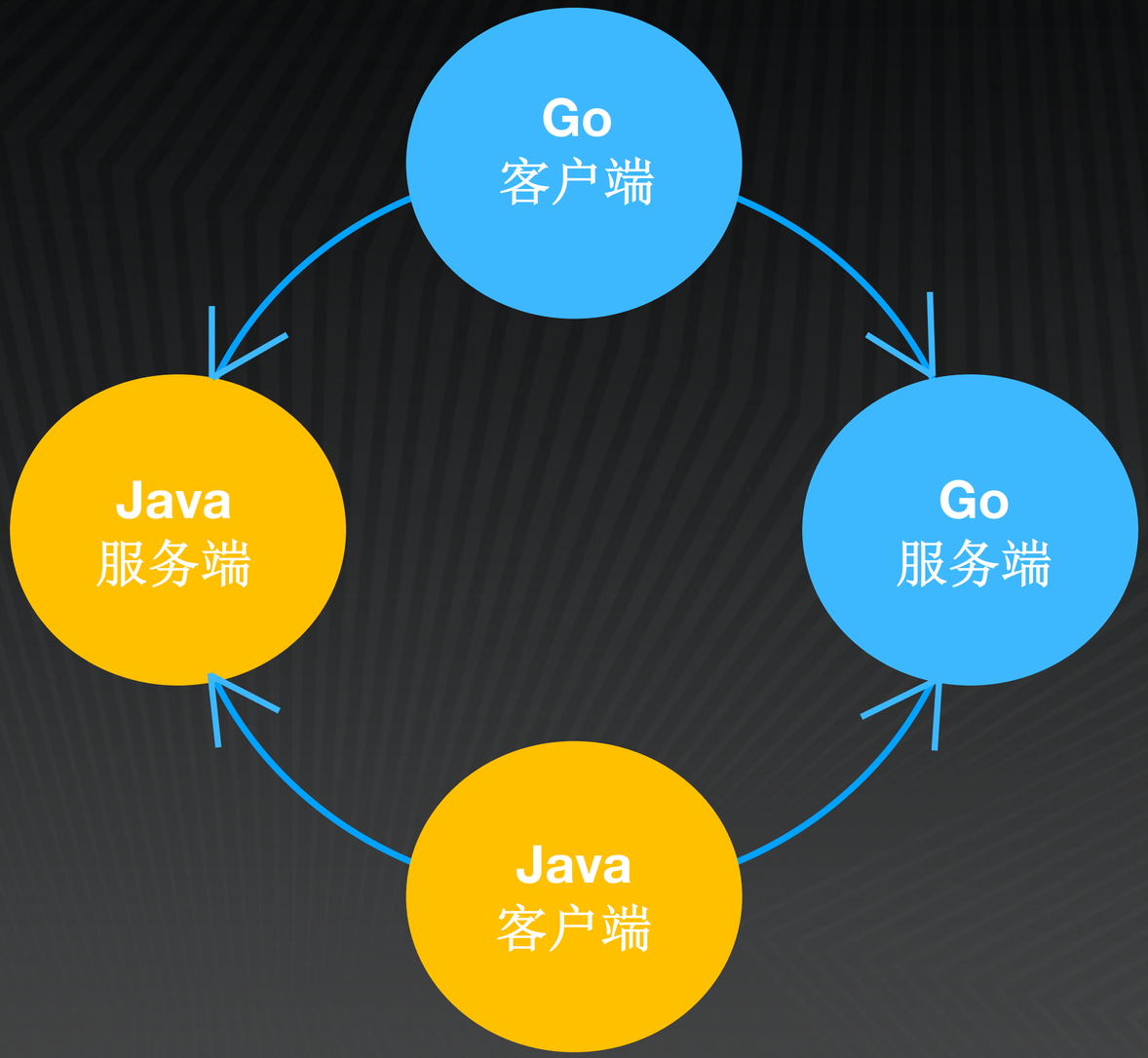
未来的趋势是通过将服务治理能力 sidecar 化，应用无需与特定的技术栈绑定

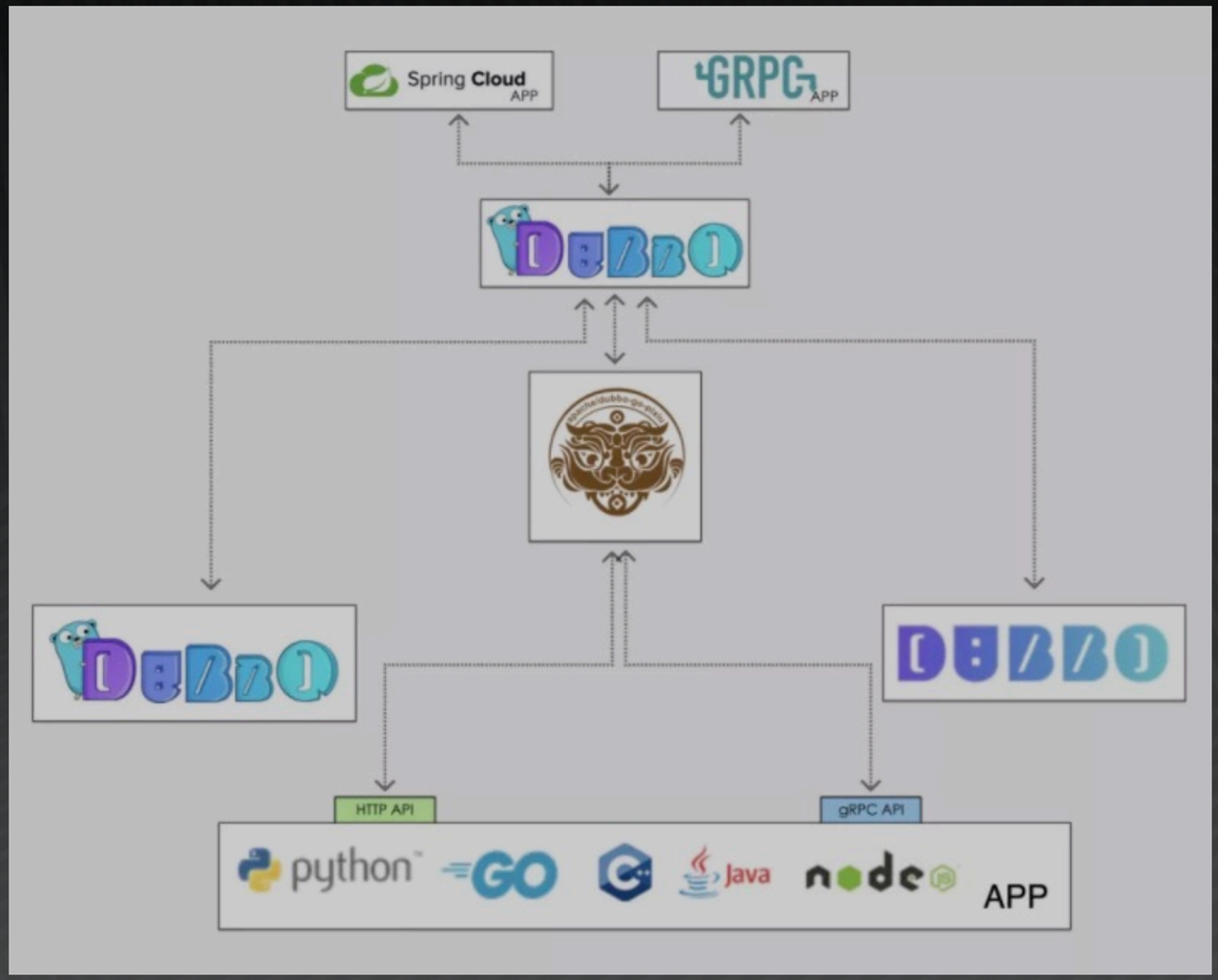
多语言支持

Dubbo 目前已经支持 Go/Java/Js/Python/Erlang 等多种语言，这也是 Spring Cloud 方案最大的短板



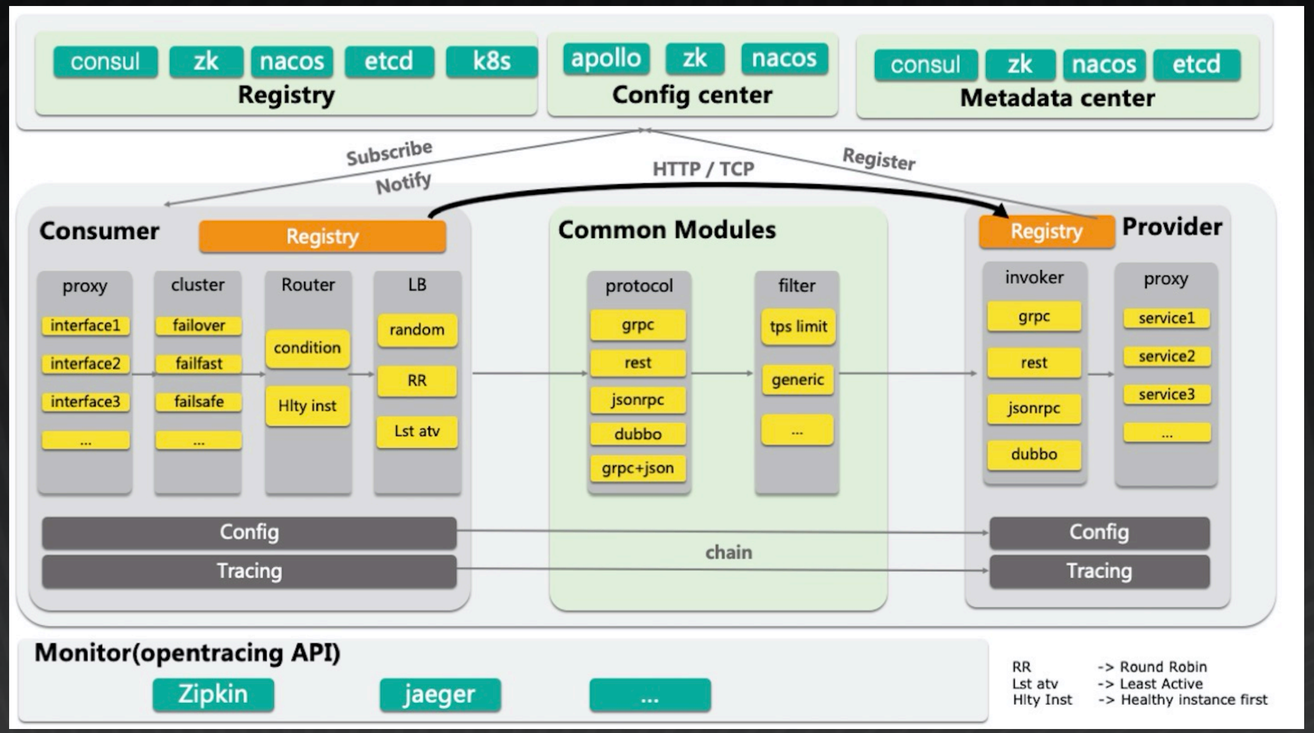
bridging the gap between Java/Dubbo and Go/X

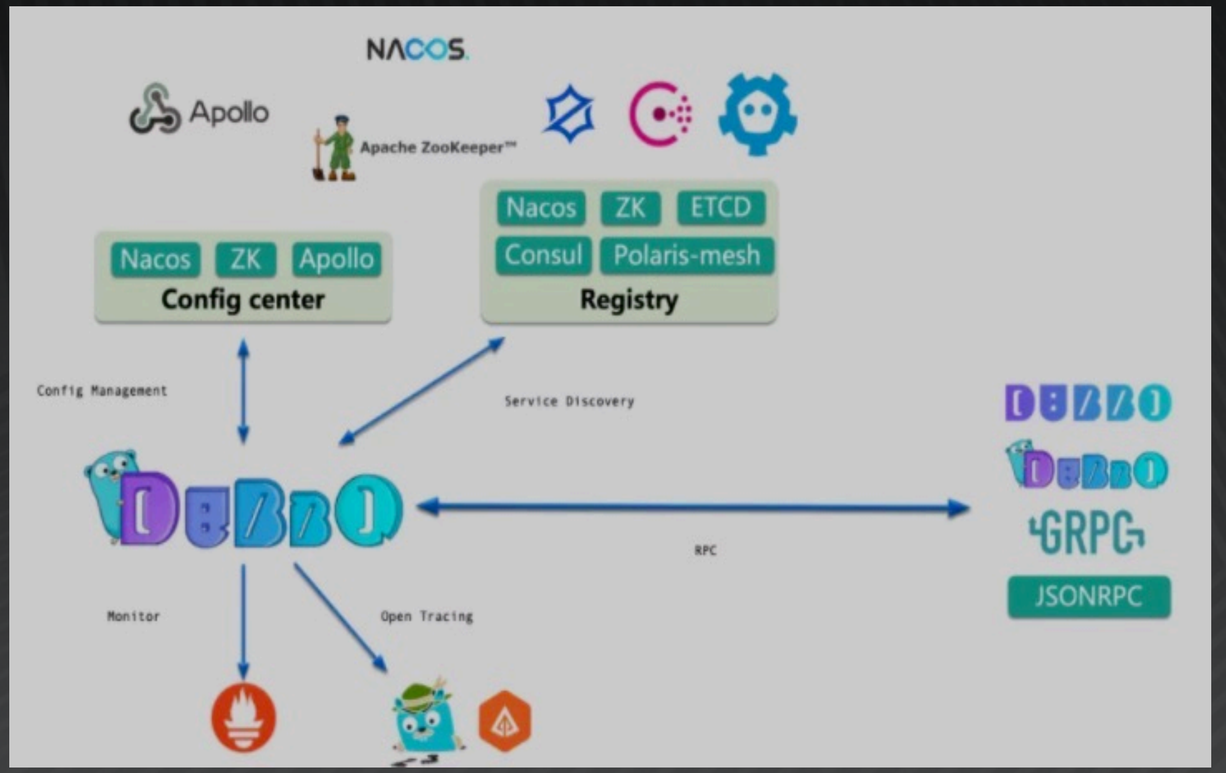




- 同一份Go客户端代码，可调用Go服务端和Java服务端。
- 同一份Go服务端代码，可以被Go客户端和Java客户端调用。

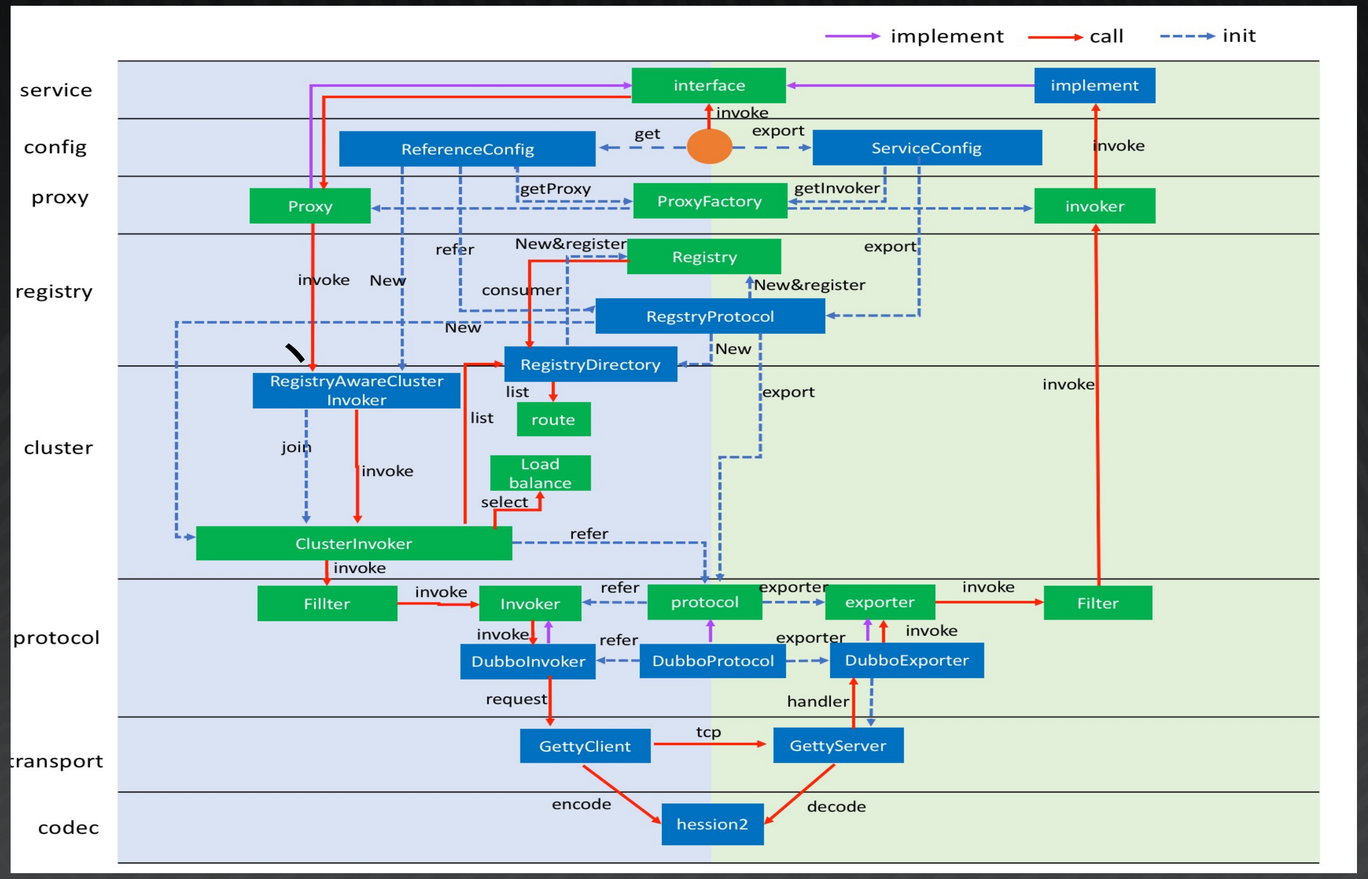
整体框架


consul
zk
nacos
etcd
k8s
Registry
apollo
zk
nacos
Config center
consul
zk
nacos
etcd
Metadata center
Subscribe
Notify
HTTP / TCP
Register
Consumer
Registry
proxy
interface1
interface2
interface3
...
cluster
failover
failfast
failsafe
...
Router
condition
Hlty inst
LB
random
RR
Lst atv
Common Modules
protocol
grpc
rest
jsonrpc
dubbo
grpc+json
filter
tps limit
generic
...
Registry
Provider
invoker
grpc
rest
jsonrpc
dubbo
proxy
service1
service2
service3
...
Config
Tracing
chain
Config
Tracing
Monitor(opentracing API)
Zipkin
jaeger
...
RR -> Round Robin
Lst atv -> Least Active
Hlty Inst -> Healthy instance first


NACOS
Apollo
Apache ZooKeeper™
Nacos
ZK
Apollo
Config center
Nacos
ZK
ETCD
Consul
Polaris-mesh
Registry
Config Management
Service Discovery
RPC
Monitor
Open Tracing
DUBBO
DUBBO
GRPC
JSONRPC

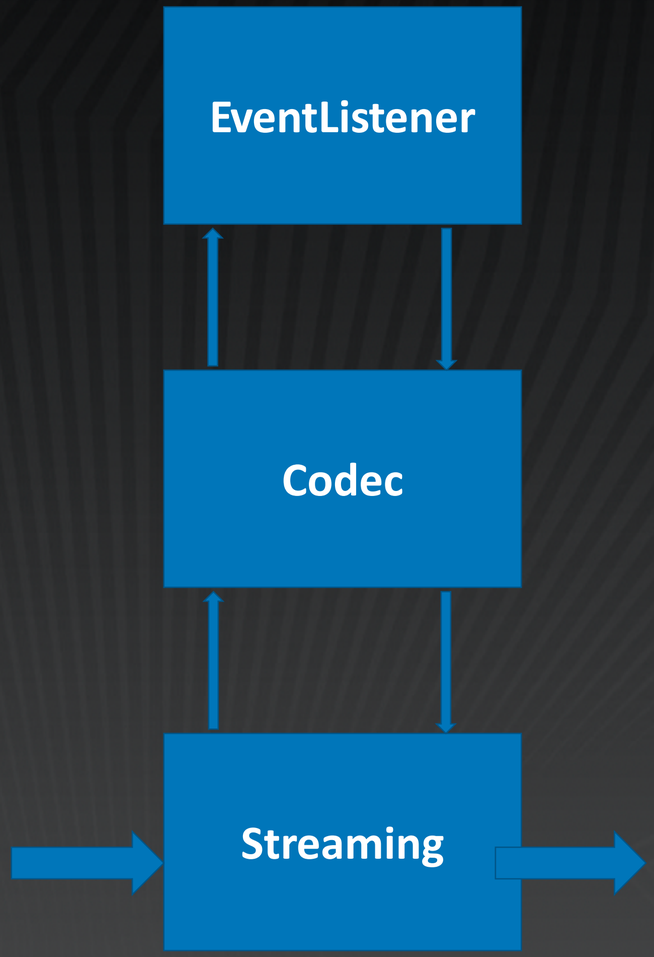
整体设计

- 借鉴 dubbo 的分层设计，易拓展性
- 语言不同，部分模块实现思路不一致



2. 异构的基础： Protocol

Getty 三层模型



应用层处理

处理网络包以及各种事件

协议

针对不同协议分别进行序列化/反序列化

网络IO

网络读写的封装

OnOpen

连接过多判断、应用层连接池添加

OnClose/OnError

连接关闭或发生错误、应用层连接池清理

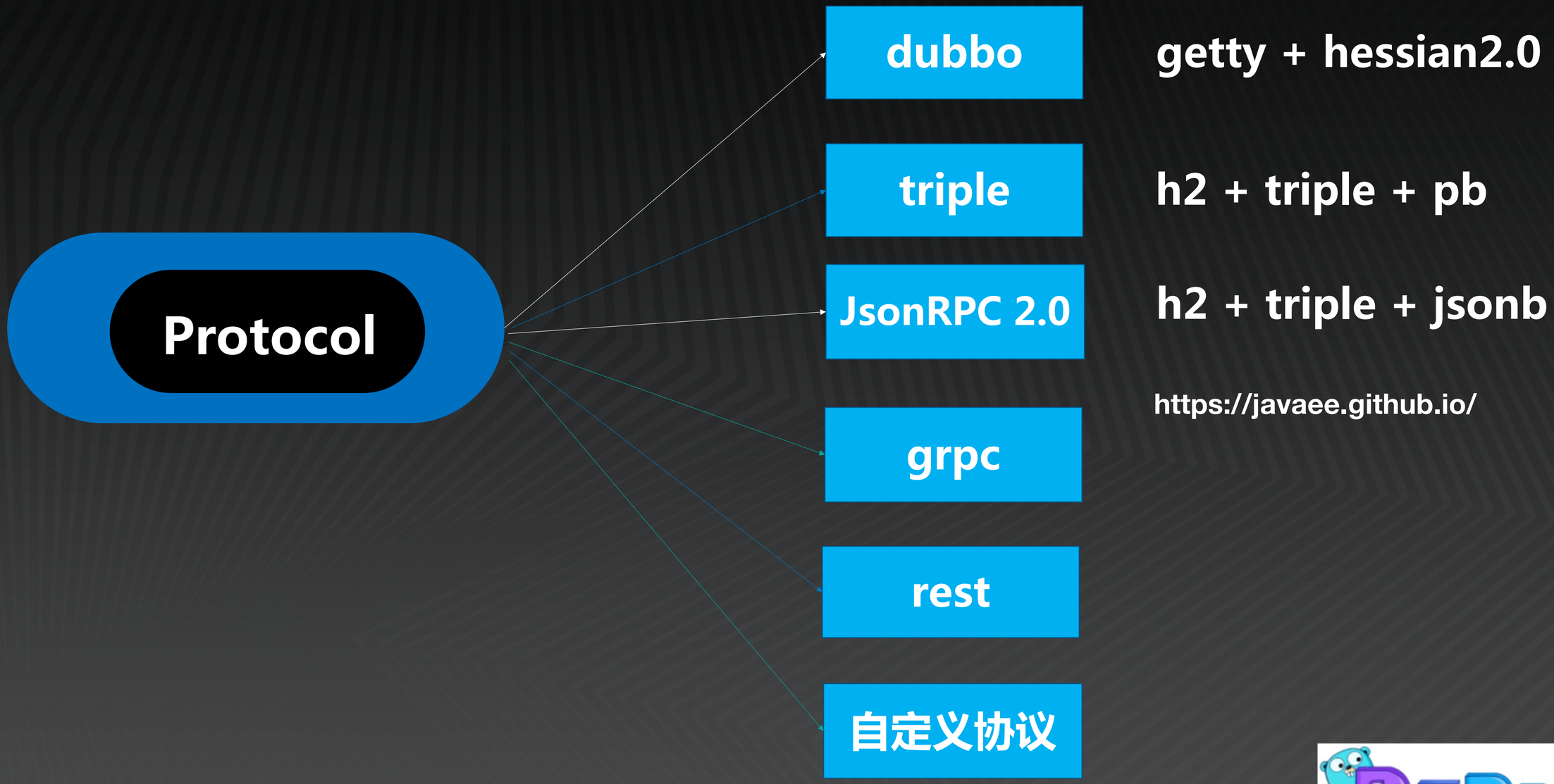
OnCron

定时任务、心跳(websocket除外)

OnMessage

逻辑包处理、阻塞Streaming/handleLoop

Protocol抽象与扩展



Dubbo-go-hessian2

- All JDK Exceptions
- Field Alias
- Java BigDecimal
- Java Date & Time
- Java Generic Invocation
- Dubbo Attachments
- Skipping unregistered POJO
- Emoji

Dubbo-go-hessian2 类型映射表

hessian type	java type	golang type
null	null	nil
binary	byte[]	[]byte
boolean	boolean	bool
date	java.util.Date	time.Time
double	double	float64
int	int	int32
long	long	int64
string	java.lang.String	string
list	java.util.List	slice
map	java.util.Map	map
object	custom define object	custom define struct
OTHER COMMON USING TYPE		
big decimal	java.math.BigDecimal	github.com/dubbogo/gost/math/big/Decimal
big integer	java.math.BigInteger	github.com/dubbogo/gost/math/big/Integer
date	java.sql.Date	github.com/apache/dubbo-go-hessian2/java_sql_time/Date
date	java.sql.Time	github.com/apache/dubbo-go-hessian2/java_sql_time/Time
date	all java8 sdk time	github.com/apache/dubbo-go-hessian2/java8_time

Go struct 与 Java class 映射： class+ member

```go
type POJO interface {
    JavaClassName() string
}

type POJOEnum interface {
    POJO
    String() string
    EnumValue(string) JavaEnum
}
```

```go
type User struct {
    Id   string
    Name string
    Age  int32
    Time time.Time
}
func (User) JavaClassName() string {
    return "com.ikurento.user.User"
}
```

Go struct 与 Java class 映射

java-server

```java
public abstract class User {
}

public class MyUser extends User implements Serializable {

    private String userFullName;

    private String familyPhoneNumber;
}

public interface UserProvider {
    String GetUser(User user);
}

public class UserProviderImpl implements UserProvider {
    public UserProviderImpl() {
    }

    public String GetUser(User user) {
        MyUser myUser=(MyUser)user;
        return myUser.getUserFullName();
    }
}
```

go-client

```go
type MyUser struct {
    UserFullName      string   `hessian:"userFullName"`
    FamilyPhoneNumber string   // default convert to => familyPhoneNumber
}

func (m *MyUser) JavaClassName() string {
    return "com.company.MyUser"
}

func (m *MyUser) JavaParamName() string {
    return "com.company.User"
}

type UserProvider struct {
    GetUser func(ctx context.Context, user *MyUser) (string, error) `dubbo:"GetUser"`
}
```

Go struct 与 Java class 映射: 陷阱

`go-hessian2` supports inheritance struct, but the following situations should be avoided.

- **Avoid fields with the same name in multiple parent struct**

The following struct `C` have inherited field `Name` (default from the first parent), but it's confused in logic.

```
type A struct { Name string }
type B struct { Name string }
type C struct {
        A
        B
}
```

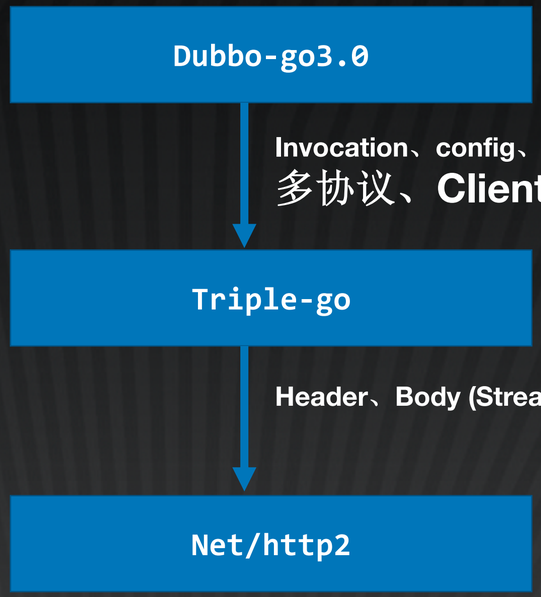
- **Avoid inheritance for a pointer of struct**

The following definition is valid for golang syntax, but the parent will be nil when create a new Dog, like `dog := Dog{}`, which will not happen in java inheritance, and is also not supported by `go-hessian2`.

```
type Dog struct {
        *Animal
}
```

dubbogo 3.0 Triple

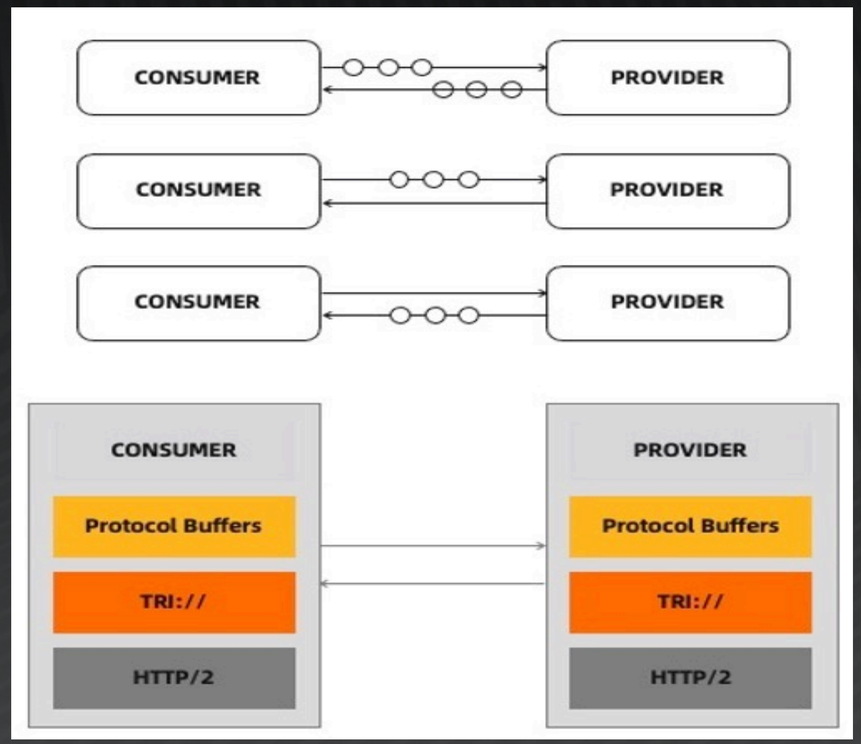
https://github.com/dubbogo/triple



Triple-go 与 gRPC 互通（普通RPC、流式RPC）

Triple-go 与 Dubbo3互通

序列化协议支持：Protobuf\Hessian2\Msgpack(DingDing)



https://github.com/dubbogo/net

TPS	Consumer CPU (%)	Provider CPU(%)	RT(ms)
1000	8.9	4.6	0.6
2000	16.5	8.5	0.6
5000	18.8	10	1
10000	37	18.3	1
20000	72	37	1
25000	77	40	1.2

3. 服务治理

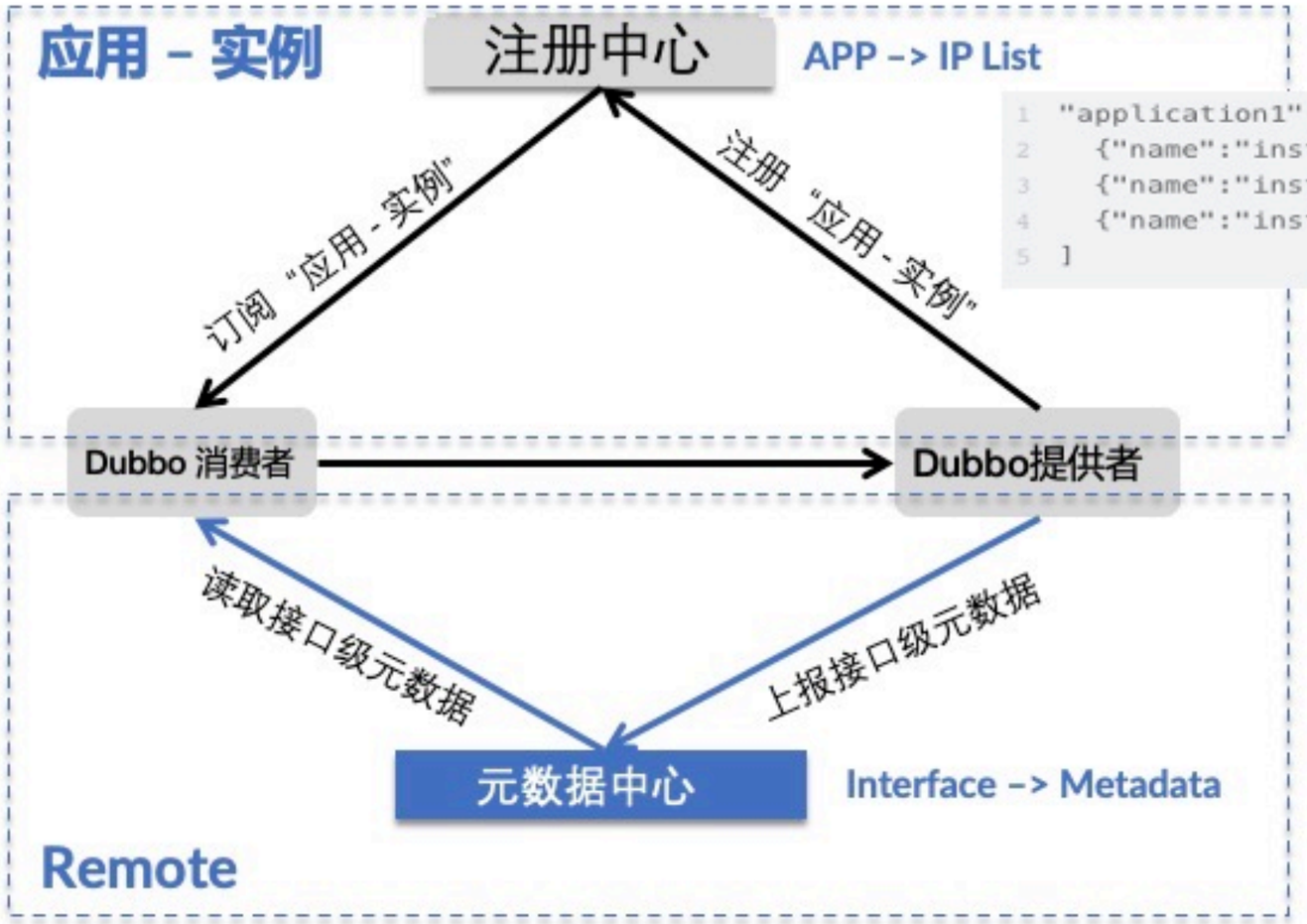
应用注册模型 接口粒度

```
<dubbo:service interface="org.apache.dubbo.samples.basic.api.DemoService" ref="demoService"/>
<dubbo:service interface="org.apache.dubbo.samples.basic.api.GreetingService" ref="greetingService"/>
```

dubbo://192.168.0.104:20880/org.apache.dubbo.samples.basic.api.GreetingService?anyhost=true&application=demo-provider&default=true&deprecated=false&dubbo=2.0.2&dynamic=true&generic=false&interface=org.apache.dubbo.samples.basic.api.GreetingService&methods=greeting&pid=995&release=2.7.7&side=provider&timestamp=1596988170816


```
{
    "name": "demo-provider",
    "id": "192.168.0.104:20880",
    "address": "192.168.0.104",
    "port": 20880,
  "metadata": {
    "dubbo.endpoints": "[{\"port\":20880,\"protocol\":\"dubbo\"}]",
    "dubbo.metadata.storage-type": "local",
    "dubbo.revision": "78296358123700993 87"
  },
    "time": 1583461240877
}
```

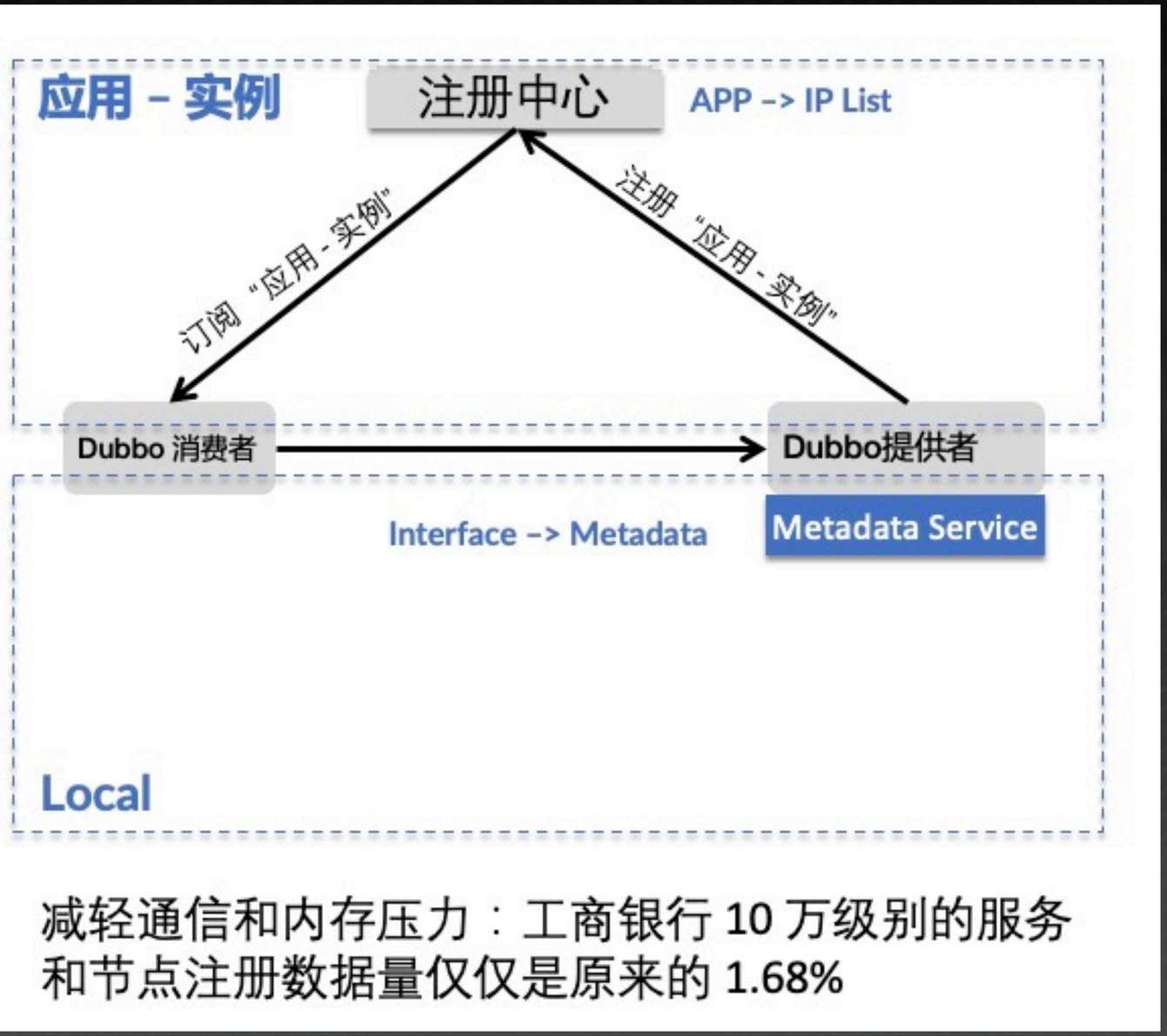
应用注册模型remote metadata center




```
"application1": [
  {"name":"instance1", "ip":"127.0.0.1", "metadata":{}},
  {"name":"instance2", "ip":"127.0.0.2", "metadata":{}},
  {"name":"instanceN", "ip":"127.0.0.3", "metadata":{}}
]
```

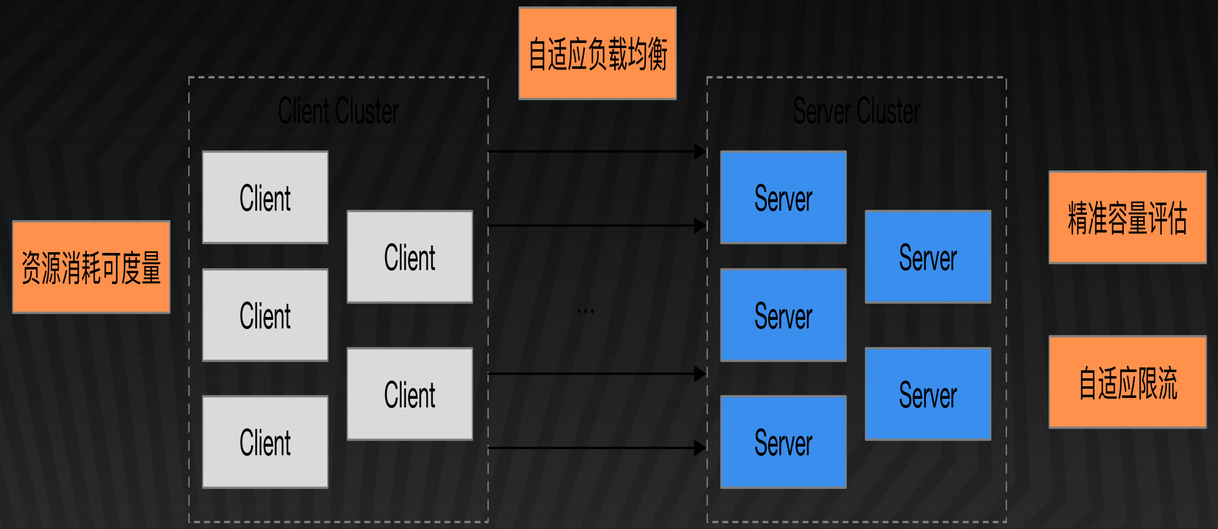
```
"com.xxx.User":[
  {"name":"instance1", "ip":"127.0.0.1", "metadata":{"timeout":1000}},
  {"name":"instance2", "ip":"127.0.0.2", "metadata":{"timeout":2000}},
  {"name":"instance3", "ip":"127.0.0.3", "metadata":{"timeout":3000}},
]
```

应用注册模型 local metadata center



减轻通信和内存压力：工商银行 10 万级别的服务和节点注册数据量仅仅是原来的 1.68%

柔性服务



云原生时代微服务特点：

1 节点异常是常态
2 云原生时代的微服务：服务不垮掉是最低要求，然后尽可能挖掘系统能力提供更优质的服务
3 长时间内服务容量评估测不准

机器规格：大规模服务下机器规格难免异构，同规格机器老化
服务拓扑复杂：分布式服务拓扑结构在不断进化
服务流量不均衡：有洪峰有波谷
依赖的上游服务能力不确定性：缓存/db 能力

4. 云原生形态

Kubernetes VS dubbo-go

CSDN

k8s service：一套完全独立的服务管理体系

dubbo interface：

key: interface/version/group
value: ip:port/revision/ service/method/role

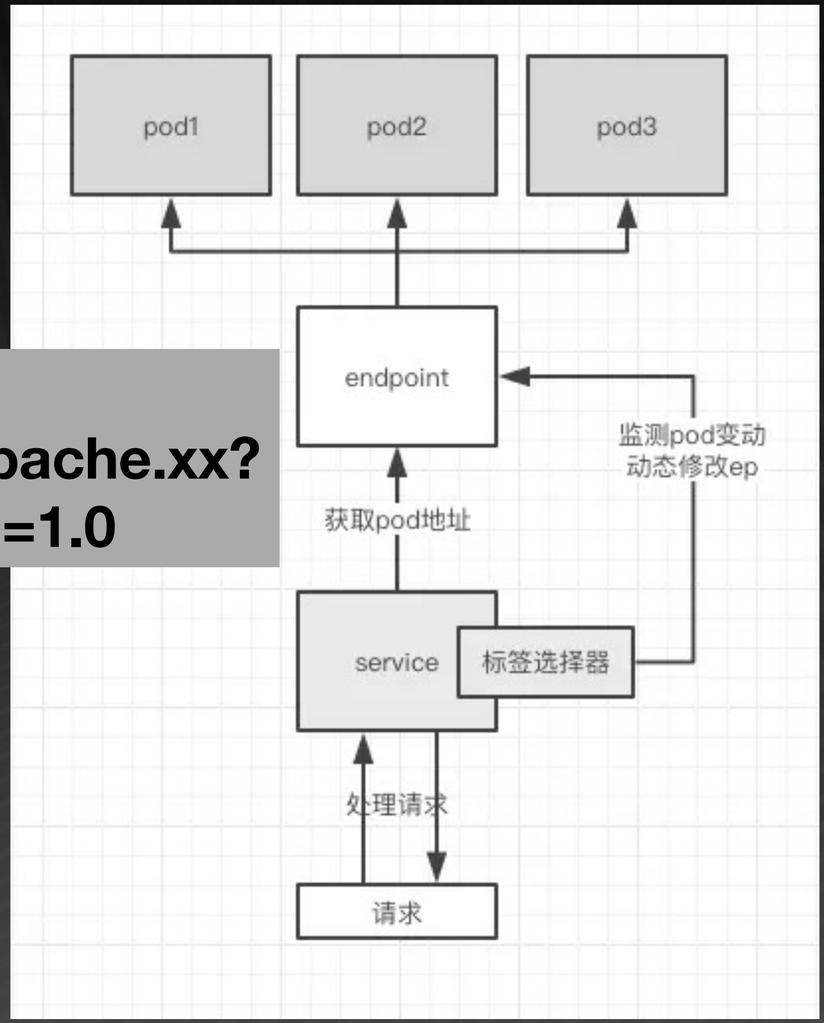
k8s service ≈ dubbo Directory + Router + Load Balance + Filter

Kubernetes注册中心——endpoint维度

- 服务注册：将每个pod提供的interface信息放到endpoint的metadata的annotation中注册
- 监听k8s 的api server中的对应的endpoint的变化，利用k8s健康检查，有效剔除无用的节点

缺点：一个endpoint代表一个服务发现的监控单元，需要根据每个service的group和version组合，生成多个endpoint

Metadata：
annotation:com.apache.xx?group=xx&version=1.0



Kubernetes注册中心——Operator维度

目前k8s 提供operator的方式来发布应用，operator可以定制化 。

dubbo-go operator:

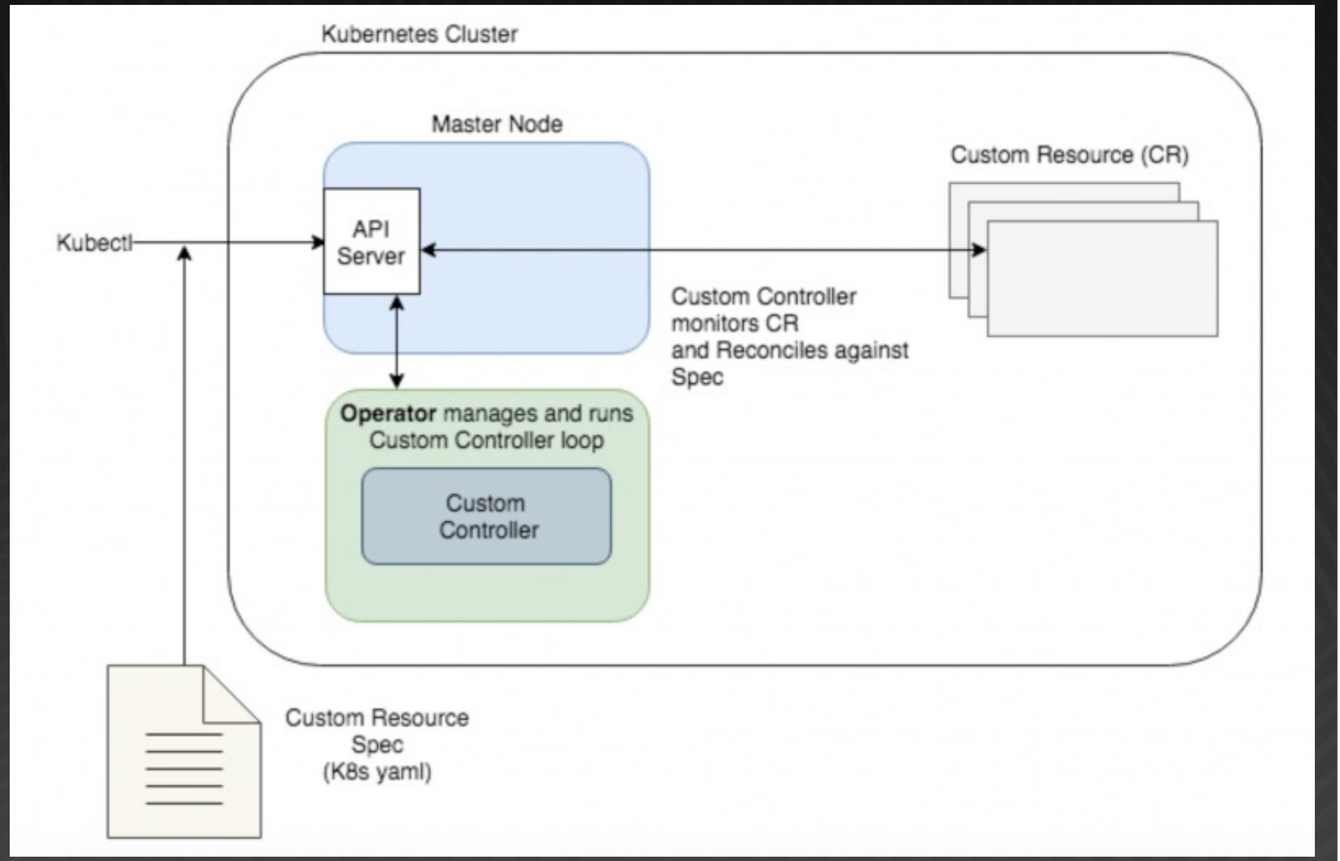
- 支持服务注册，通过**k8s api server** 以**crd**的形式写入**k8s etcd**
- 支持服务的健康检查，及时的剔除不健康的服务

dubbo-go:

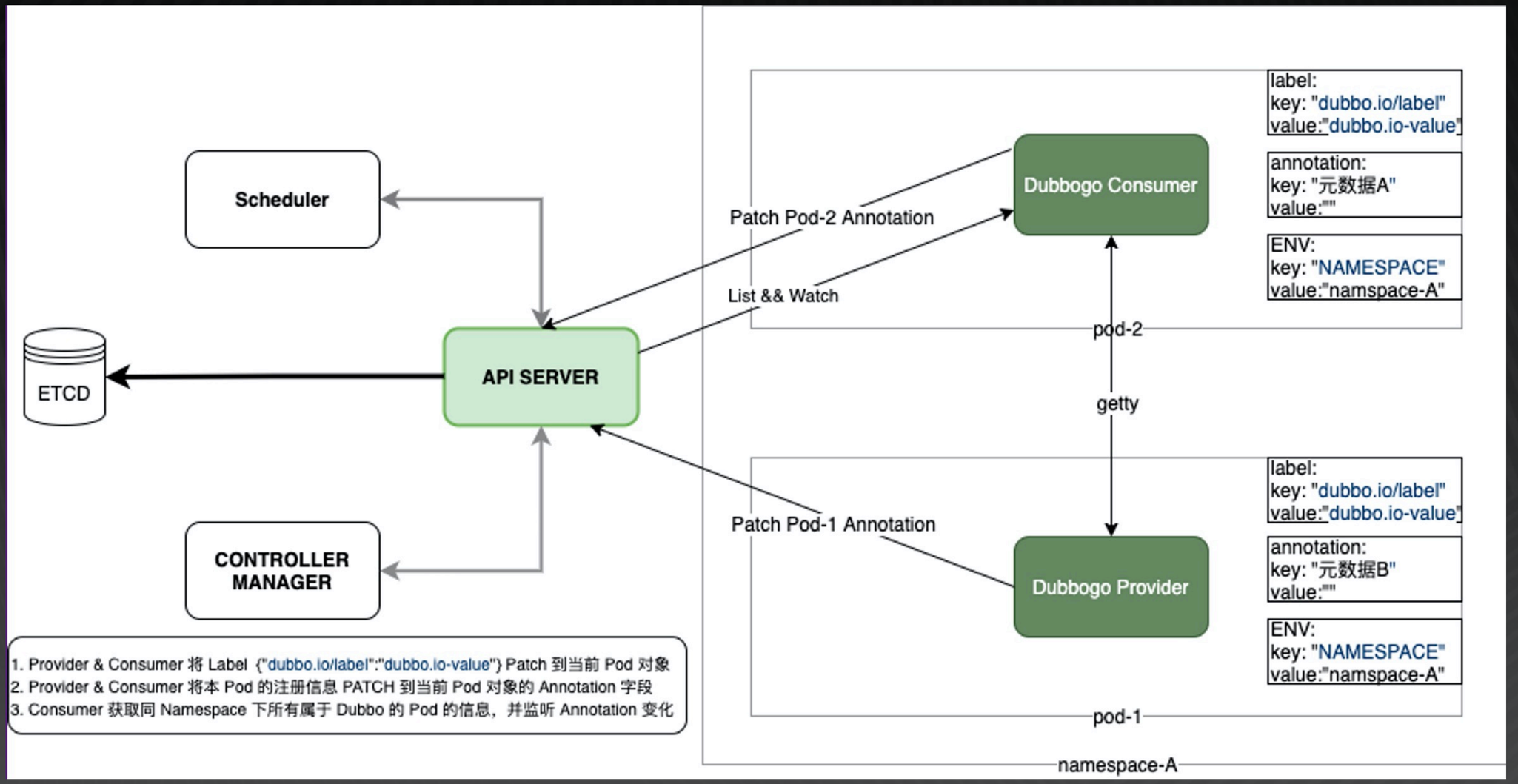
- 通过**k8s api server** 对**crd**资源进行查询
- **crd** 资源的监听，及时变更**crd** 资源的变化

优点：主流方案，高度定制化

挑战：单独维护**operator**的运维成本



Kubernetes注册中心——Dubbogo 方案

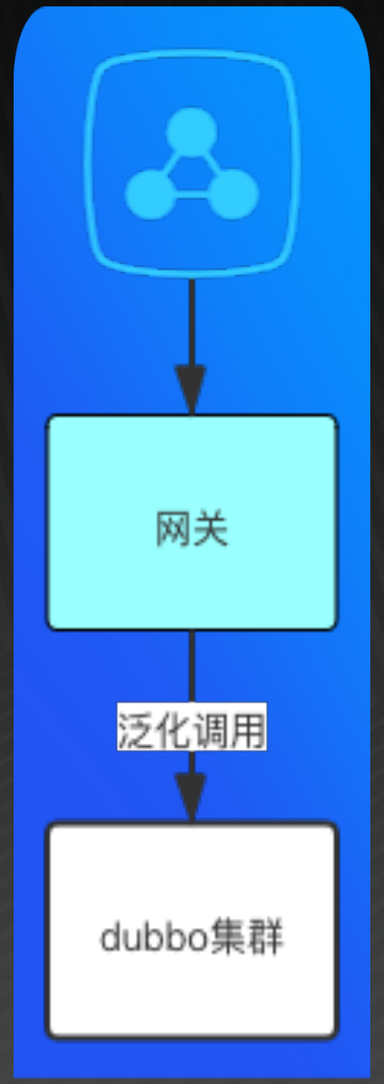


5. Not Only RPC

泛化调用

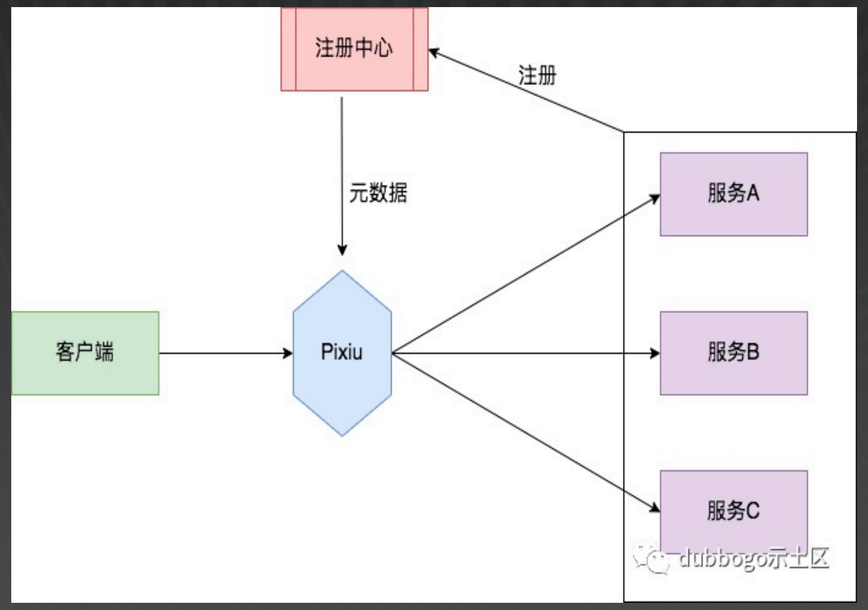
```go
type GenericService struct {
    Invoke       func(req []interface{}) (interface{}, error) `dubbo:"$invoke"`
    referenceStr string
}
```

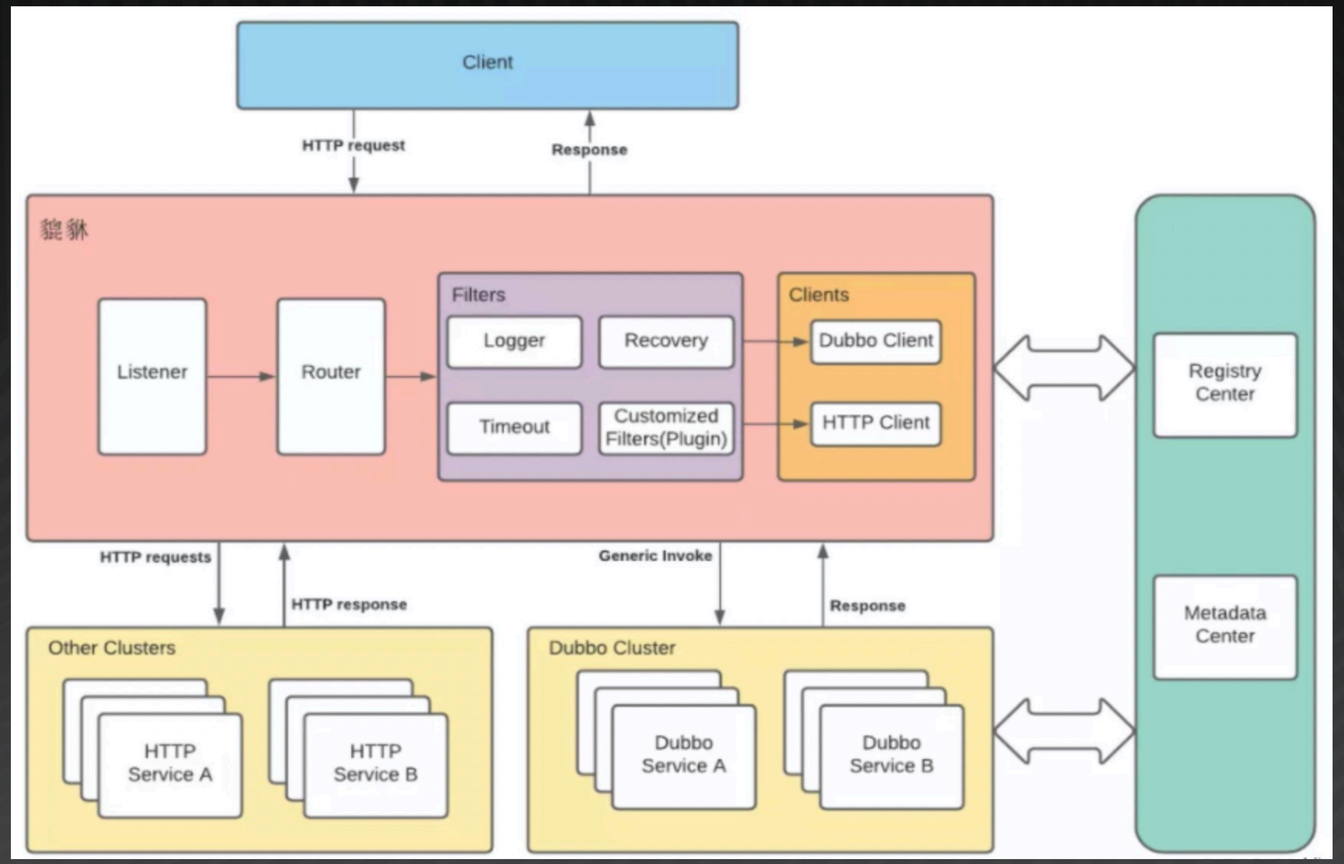
```go
// UserProvider 客户端 stub
type UserProvider struct {
  // dubbo标签，用于适配go侧客户端大写方法名 -> java侧小写方法名
  // 只有 dubbo 协议客户端才需要使用
  GetUser  func(ctx context.Context, user *GoUser) (*User, error) `dubbo:"getUser"`
}


// 一个客户端泛化调用
genericService.Invoke(context.TODO(),
  "getUser",  // 通过传入方法名，和参数值
  []string{(&GoUser{}).JavaClassName(), (&GoUser{}).JavaClassName()}, // 方法签名
  []interface{}{&GoUser{Name: "alex"}} // 参数值
)

flow:
1 框架客户端层接收到如上这三个参数后，会构造出泛化请求，发送至服务端
2 框架服务端在 filter 中过滤 $invoke 请求，构造请求向业务层发起调用

usage:
1 mock 压测
2 流量回放
3 API 网关
4 命令行服务测试
5 云原生场景下的服务探活
```



Pixiu gateway/sidecar

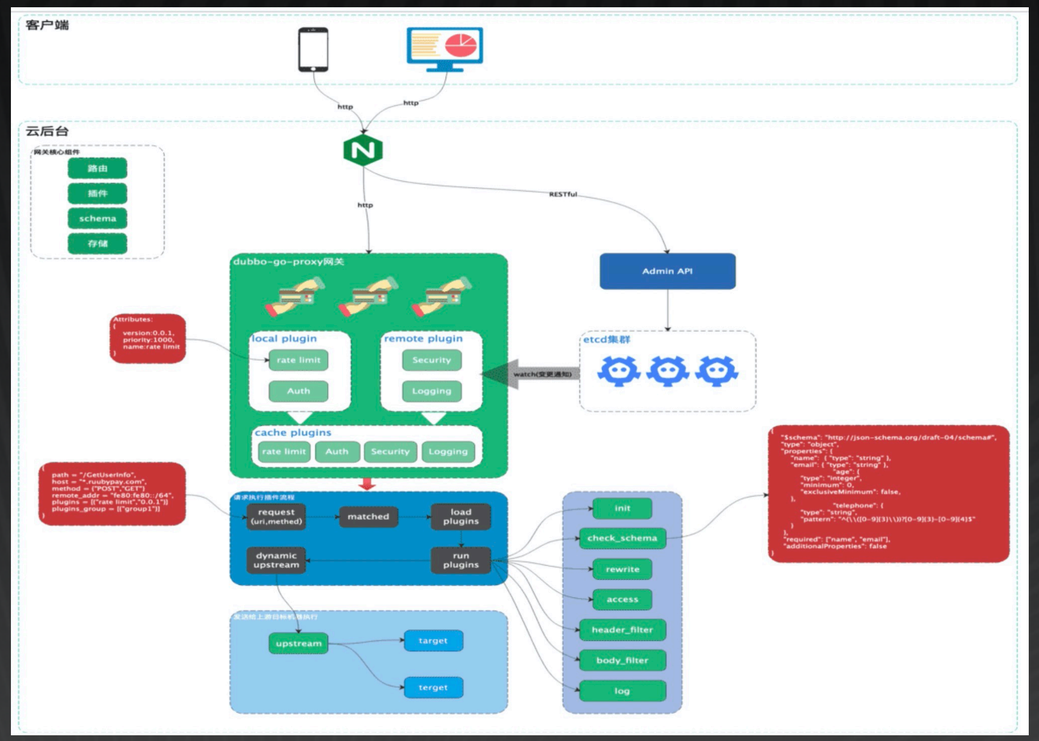
github.com/apache/dubbo-go-pixiu

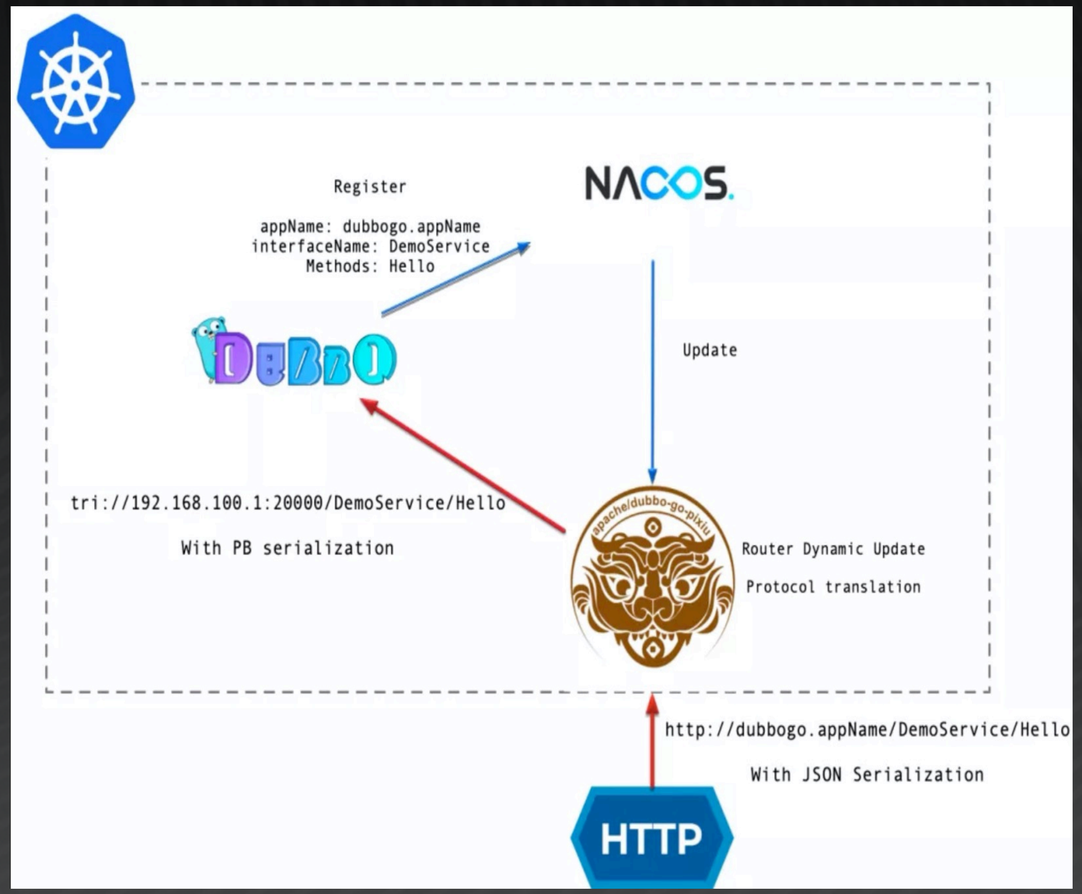




Pixiu → Gateway

github.com/apache/dubbo-go-pixiu





Pixiu 不足：

1 扩展性不足
2 稳定性需要更多场景验证
3 性能需要向 envoy 看齐

What's service mesh

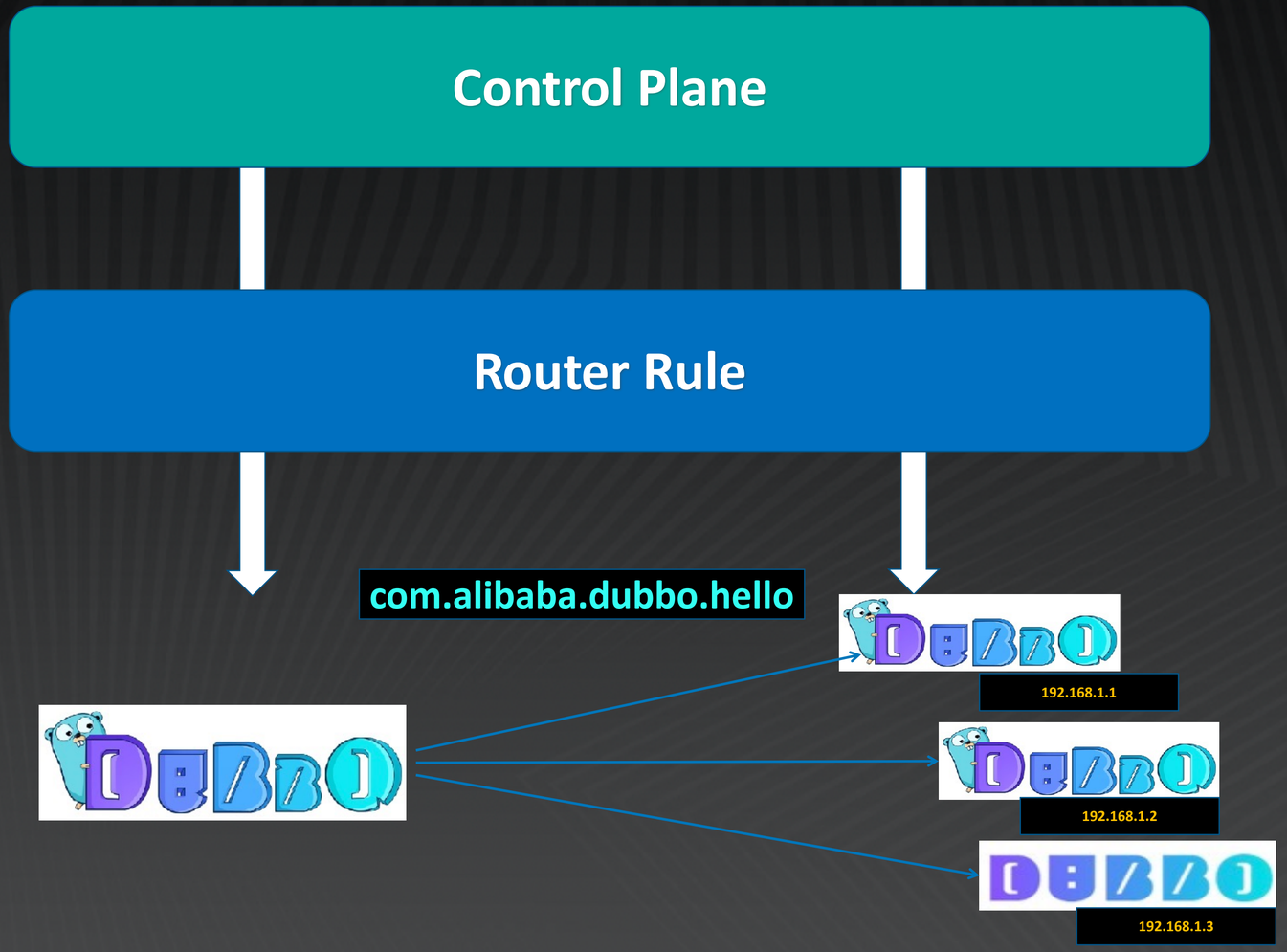
The term service mesh is used to describe the network of microservices that make up such applications and the interactions between them.

https://istio.io/latest/docs/concepts/what-is-istio/

For all the hype, the service mesh is architecturally pretty straightforward. It's nothing more than a bunch of *userspace proxies*, stuck "next" to your services (we'll talk about what "next" means in a bit), plus a set of management processes.

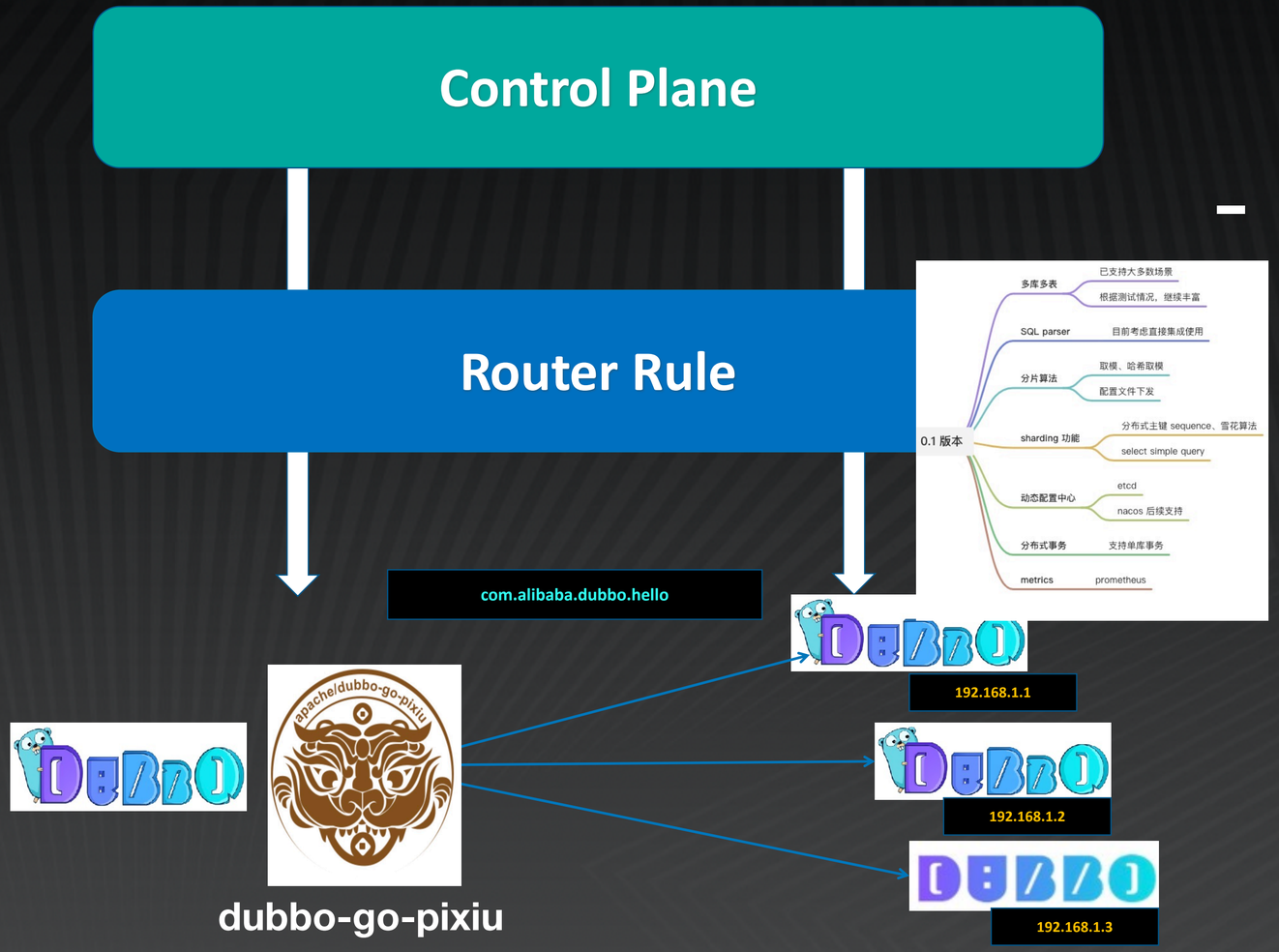
https://buoyant.io/service-mesh-manifesto/

Dubbogo Proxyless Service Mesh



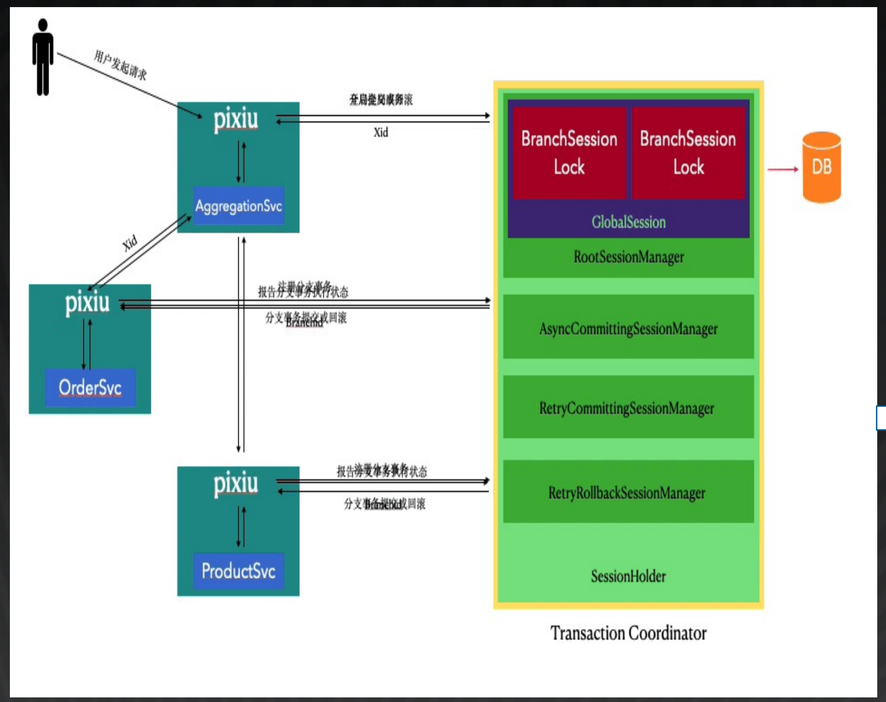
- 业务改造成本低
- 迁移成本低
- 无 **proxy** 性能损失

Dubbogo Proxy Service Mesh



- 解决 dubbo多语言问题

DBMesh

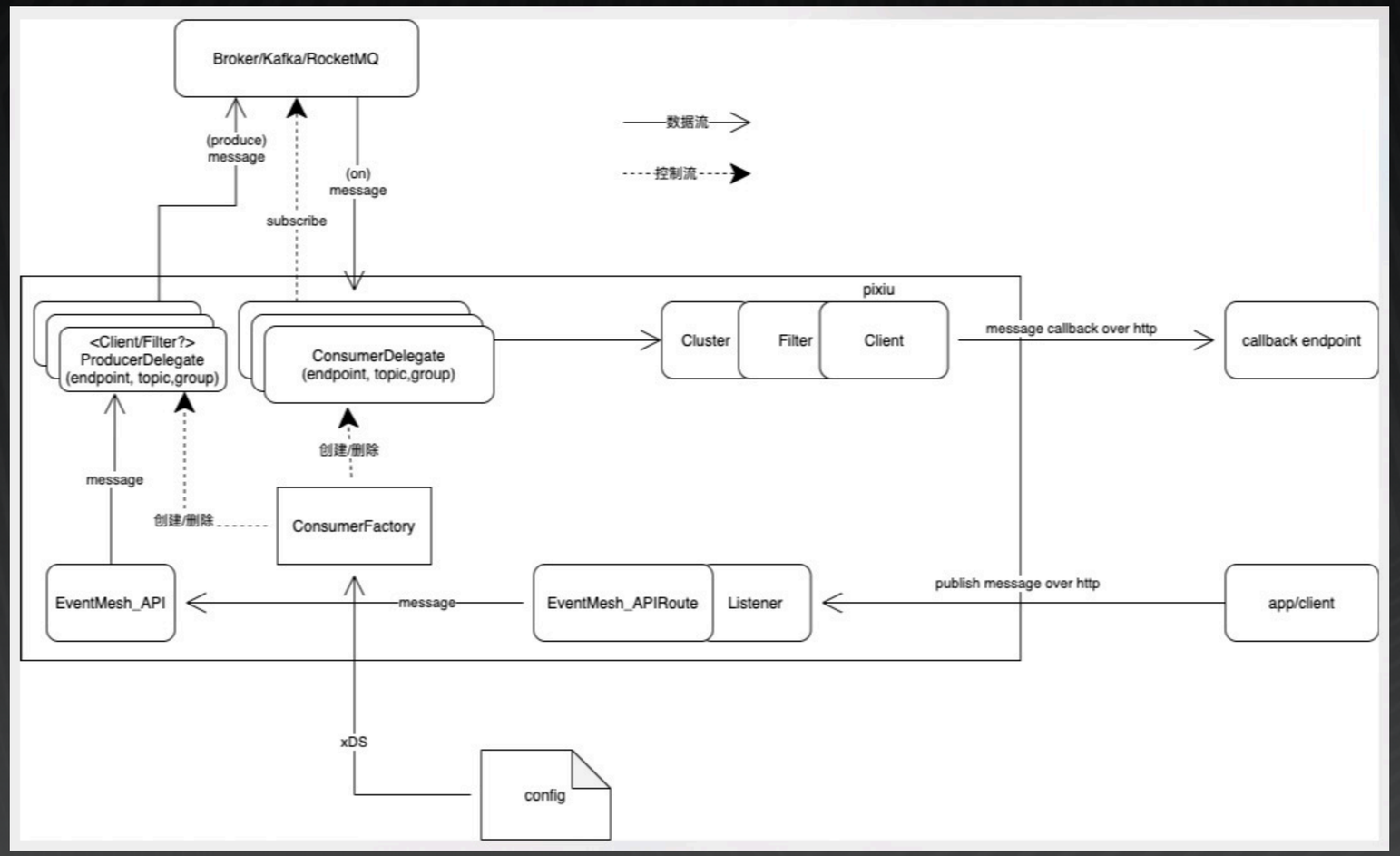


github.com/dubbogo/arana

DBMesh 特点：

1 开发者只关心通用语言 SQL
2 介于 Proxyless 和 Proxy 形式之间，性能和多语言优势兼顾
3 数据面 + 控制面 + 工具箱

EventMesh


Broker/Kafka/RocketMQ
(produce)
message
(on)
message
subscribe
数据流
控制流
pixiu
<Client/Filter?>
ProducerDelegate
(endpoint, topic,group)
ConsumerDelegate
(endpoint, topic,group)
Cluster
Filter
Client
message callback over http
callback endpoint
创建/删除
message
创建/删除
ConsumerFactory
EventMesh_API
message
EventMesh_APIRoute
Listener
publish message over http
app/client
xDS
config

6. 社区

2021 dubbogo

3801 **STARS** 8339 **COMMITS** 655 **FORKS**

截止 20220108

User List

If you are using apache/dubbo-go and think that it helps you or want to contribute code for Dubbo-go, please add your company to the user list to let us know your needs.

See more user cases

program	lines
dubbogo/gost	17521
apache/dubbo-go-hessian2	20007
apache/dubbo-getty	26352
dubbogo/triple	5658
dubbogo/triple-benchmark	2229
dubbogo/v3router	2412
apache/dubbo-go	96152
dubbogo/tools	1483
apache/dubbo-go-samples	master 34887/1.5分支 45456
apache/dubbo-go-pixiu	30369
dubbogo/dubbo-go-pixiu-filter	524
dubbogo/pixiu-admin	44001
dubbogo/handsonlabs	2812
dubbogo/arana	25894
total	320870/355757

2021 Dubbogo Meetup

Thank you !