dubbo-go实操

方银城

- 快速开始
- 泛化
- 自定义filter
- 现状

快速开始

准备工作

github.com/apache/dubbo-go

github.com/apache/dubbo-samples/golang

zk注册中心

golang开发环境和ide

配置

server.yml

```
registries :
  "hangzhouzk":
    protocol: "zookeeper"
    timeout : "3s"
    address: "127.0.0.1:2181"
    username: ""
    password: ""

services:
  "UserProvider":
    # 可以指定多个registry，使用逗号隔开;不指定默认向所有注册中心注册
    registry: "hangzhouzk"
    protocol : "dubbo"
    # 相当于dubbo.xml中的interface
    interface : "com.ikurento.user.UserProvider"
    loadbalance: "random"
    warmup: "100"
    cluster: "failover"
    methods:
      - name: "GetUser"
        retries: 1
        loadbalance: "random"
```

client.yml

```
registries :
  "hangzhouzk":
    protocol: "zookeeper"
    timeout : "3s"
    address: "127.0.0.1:2181"
    username: ""
    password: ""

references:
  "UserProvider":
    # 可以指定多个registry，使用逗号隔开;不指定默认向所有注册中心注册
    registry: "hangzhouzk"
    protocol : "dubbo"
    interface : "com.ikurento.user.UserProvider"
    cluster: "failover"
    methods :
      - name: "GetUser"
        retries: 3
```

export CONF_PROVIDER_FILE_PATH="xxx"
export APP_LOG_CONF_FILE="xxx"

export CONF_CONSUMER_FILE_PATH="xxx"
export APP_LOG_CONF_FILE="xxx"

接口

```go
func init() {
    config.SetProviderService(new(UserProvider))
    // ------for hessian2------
    hessian.RegisterPOJO(&User{})
}

type User struct {
    Id   string
    Name string
    Age  int32
    Time time.Time
}

func (u User) JavaClassName() string {
    return "com.ikurento.user.User"
}
```

provider

```go
func init() {
    config.SetProviderService(new(UserProvider))
    // ------for hessian2------
    hessian.RegisterPOJO(&User{})
}

type UserProvider struct {
}

func (u_*UserProvider) GetUser(ctx context.Context, req []interface{}) (*User, error) {
    println( format: "req:%#v", req)
    rsp := User{ Id: "A001", Name: "Alex Stocks", Age: 18, Time: time.Now()}
    println( format: "rsp:%#v", rsp)
    return &rsp, nil
}

func (u *UserProvider) Reference() string {
    return "UserProvider"
}
```

consumer

```go
func init() {
    config.SetConsumerService(userProvider)
    hessian.RegisterPOJO(&User{})
}

type User struct {
    Id   string
    Name string
    Age  int32
    Time time.Time
}

type UserProvider struct {
    GetUser func(ctx context.Context, req []interface{}, rsp *User) error
}

func (u *UserProvider) Reference() string {
    return "UserProvider"
}

func (User) JavaClassName() string {
    return "com.ikurento.user.User"
}
```

启动

```
import (
    "github.com/apache/dubbo-go/common/logger"
    "github.com/apache/dubbo-go/config"
    _ "github.com/apache/dubbo-go/protocol/dubbo"
    _ "github.com/apache/dubbo-go/registry/protocol"

    _ "github.com/apache/dubbo-go/common/proxy/proxy_factory"
    _ "github.com/apache/dubbo-go/filter/filter_impl"

    _ "github.com/apache/dubbo-go/cluster/cluster_impl"
    _ "github.com/apache/dubbo-go/cluster/loadbalance"
    _ "github.com/apache/dubbo-go/registry/zookeeper"
)
```

```
type UserProvider struct {
    GetUser func(ctx context.Context, req [
}

func (u *UserProvider) Reference() string {
    return "UserProvider"
}
```

```
func main() {
    config.Load()
    time.Sleep( d: 3e9)

    user := &User{}
    err := config.GetConsumerService( name: "UserProvider").(*UserProvider).GetUser(context.TODO(), []interface{}{"A001"}, user)
    if err != nil {
        panic(err)
    }
    println( format: "response result: %v\n", user)
    initSignal()
}
```

1．临时没有zk注册中心可以运行例子，怎么办？

```
registries :
  "hangzhouzk":
    protocol: "zookeeper"
    timeout : "3s"
    address: "127.0.0.1:2181"
    username: ""
    password: ""

references:
  "UserProvider":
    # 可以指定多个registry，使用逗号隔开;不指定默认向所有注册中心注册
    registry: "hangzhouzk"
    protocol : "dubbo"
    interface : "com.ikurento.user.UserProvider"
    cluster: "failover"
    methods :
      - name: "GetUser"
        retries: 3
```

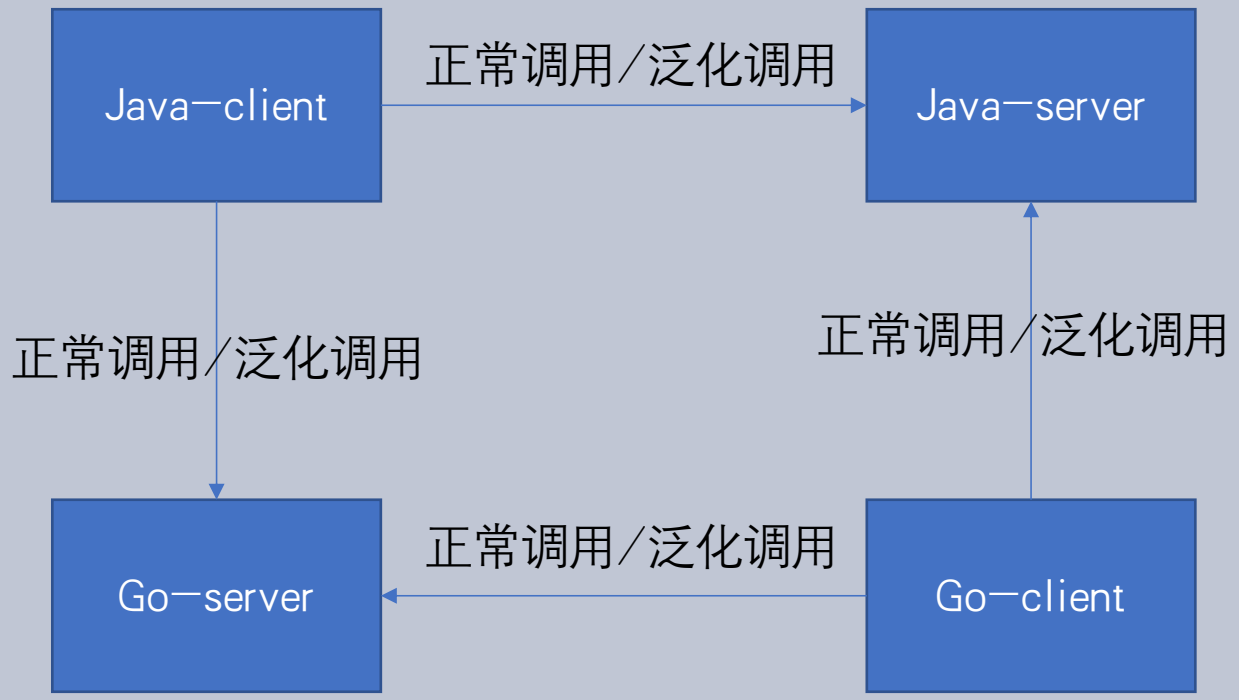
```
registries :
  "hangzhouzk":
    protocol: "zookeeper"
    timeout : "3s"
    address: "127.0.0.1:2181"
    username: ""
    password: ""

references:
  "UserProvider":
    # 可以指定多个registry，使用逗号隔开;不指定默认向所有注册中心注册
    #registry: "hangzhouzk"
    protocol : "dubbo"
    url:  "dubbo://127.0.0.1:20000"
    interface : "com.ikurento.user.UserProvider"
    cluster: "failover"
    methods :
      - name: "GetUser"
        retries: 3
```

2. Java方法都是小写开头的，dubbo-go如何适配？

```go
func (s *UserProvider) MethodMapper() map[string]string {
    return map[string]string{
        "GetUser": "getUser",
    }
}
```

```go
type UserProvider struct {
    GetUser func(ctx context.Context, req []interface{}, rsp *User) error `dubbo:"getUser"`
}
```

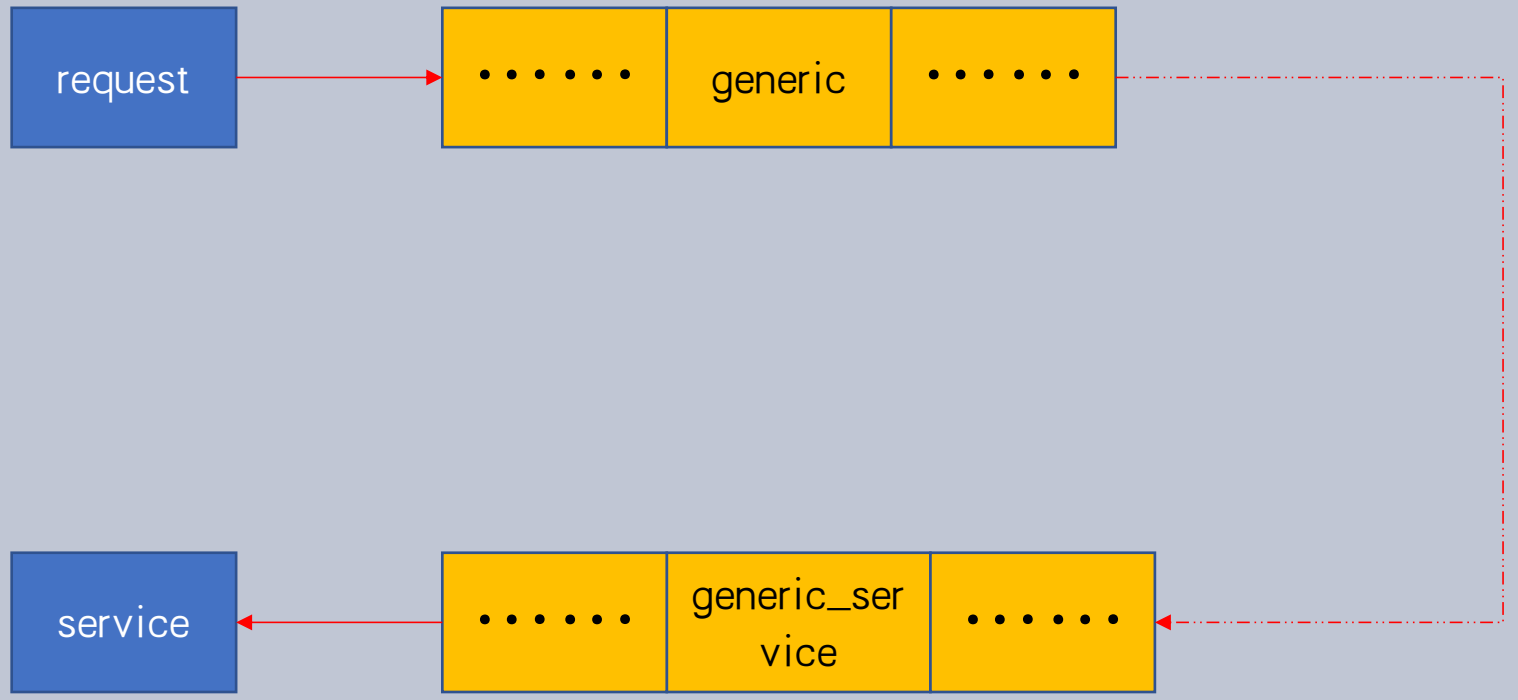
泛化


Java—client
正常调用/泛化调用
Java—server
正常调用/泛化调用
正常调用/泛化调用
Go—server
正常调用/泛化调用
Go—client

客户端

```
var appName = "UserProviderGer"
var referenceConfig = config.ReferenceConfig{
    InterfaceName: "com.ikurento.user.UserProvider",
    Cluster:       "failover",
    Registry:      "hangzhouzk",
    Protocol:      dubbo.DUBBO,
    Generic:       true,
}
referenceConfig.GenericLoad(appName) //appName is the unique identification of RPCService

time.Sleep(3 * time.Second)
println( format: "\n\n\nstart to generic invoke")
resp, err := referenceConfig.GetRPCService().(*config.GenericService).Invoke(context.TODO(),
    []interface{}{"GetUser", []string{"java.lang.String"}, []interface{}{"A003"}})
if err != nil {
    panic(err)
}
println( format: "res: %+v\n", resp)
println( format: "succ!")
```

自定义filter



accesslog

active

echo

execute

generic

generic_service

cshutdown

pshutdown

hystrix_consumer

hystrix_provider

metrics

token

tps

tracing

```go
// Filter interface defines the functions of a filter
// Extension - Filter
type Filter interface {
    // Invoke is the core function of a filter, it determins the process of the filter
    Invoke(context.Context, protocol.Invoker, protocol.Invocation)
    // OnResponse updates the results from Invoke and then returns
    OnResponse(context.Context, protocol.Result, protocol.Invoker,
}
```

```yaml
# dubbo server yaml configure file
filter: "MyCustomFilter"

# application config
application:
  organization : "ikurento.com"
  name : "BDTService"
  module : "dubbogo user-info server"
  version : "0.0.1"
  owner : "ZX"
  environment : "release"

registries :
  "hangzhouzk":
    protocol: "zookeeper"
    timeout : "3s"
    address: "127.0.0.1:2181"
    username: ""
    password: ""
```

```go
    extension.SetFilter( name: "MyCustomFilter", GetMyCustomFilter)

    // or using the singleton
    // filter.SetFilter("MyCustomFilter", GetMyCustomFilterSingleton)
}

type myCustomFilter struct{}

func (mf myCustomFilter) Invoke(ctx context.Context, invoker protocol.Invoker, invocation protocol.Invo
    // the logic put here...
    // you can get many params in url. And the invocation provides more information about
    url := invoker.GetUrl()
    serviceKey := url.ServiceKey()
    println( format: "Here is the my custom filter. The service is invoked: %s", serviceKey)
    return invoker.Invoke(ctx, invocation)
}

func (mf myCustomFilter) OnResponse(ctx context.Context, result protocol.Result, invoker protocol.Invok
    // you can do something here with result
    println( format: "Got result!")
    return result
}

func GetMyCustomFilter() filter.Filter {
    return &myCustomFilter{}
}
```

```go
// Protocol
// Extension - protocol
type Protocol interface {
    // Export service for remote invocation
    Export(invoker Invoker) Exporter
    // Refer a remote service
    Refer(url common.URL) Invoker
    // Destroy will destroy all invoker and exporter, so it only is called once.
    Destroy()
}
```

```go
// Registry Extension - Registry
type Registry interface {
    common.Node

    Register(url common.URL) error

    UnRegister(url common.URL) error

    Subscribe(*common.URL, NotifyListener) error

    UnSubscribe(*common.URL, NotifyListener) error
}
```

现状

registry

dubbo-go supported registries:

zookeeper | etcdv3 | consul | kubernetes | nacos

configuration center

dubbo-go supported configuration center:

zookeeper | nacos | apollo

protocol

dubbo-go supported protocol:

dubbo | grpc | jsonrpc | rest

metrics

dubbo-go supported metrics:

prometheus

- 目前v1.5.0正在发版投票中
- Submodule改造进行中
- 文档站点正在建设中，https://dubbogo.github.io/dubbo-go-website

谢谢观看