云原生高性能服务框架 Kitex

陆少杰 / 字节跳动 基础架构

CSDN

个人简介

- 大连理工大学 软件工程
- 2014～2018 腾讯后台研发
- 2018年底加入字节跳动基础架构，专注 RPC 框架研发

团队简介

- 字节跳动 基础架构 服务框架
- 致力于 RPC/HTTP 框架、Service Mesh 等领域的技术探索
- 技术栈包括 C++/Go/Java/Rust/Python/……

CSDN

CloudWeGo

大纲

1. 字节微服务框架的挑战和演进
2. Golang RPC 框架 Kitex
3. Kitex 在字节内部的落地
4. Kitex 的开源实践

CSDN

CloudWeGo

一、字节微服务框架的挑战和演进

字节 Golang RPC 框架的演进历史

字节 Golang RPC 框架的演进历史

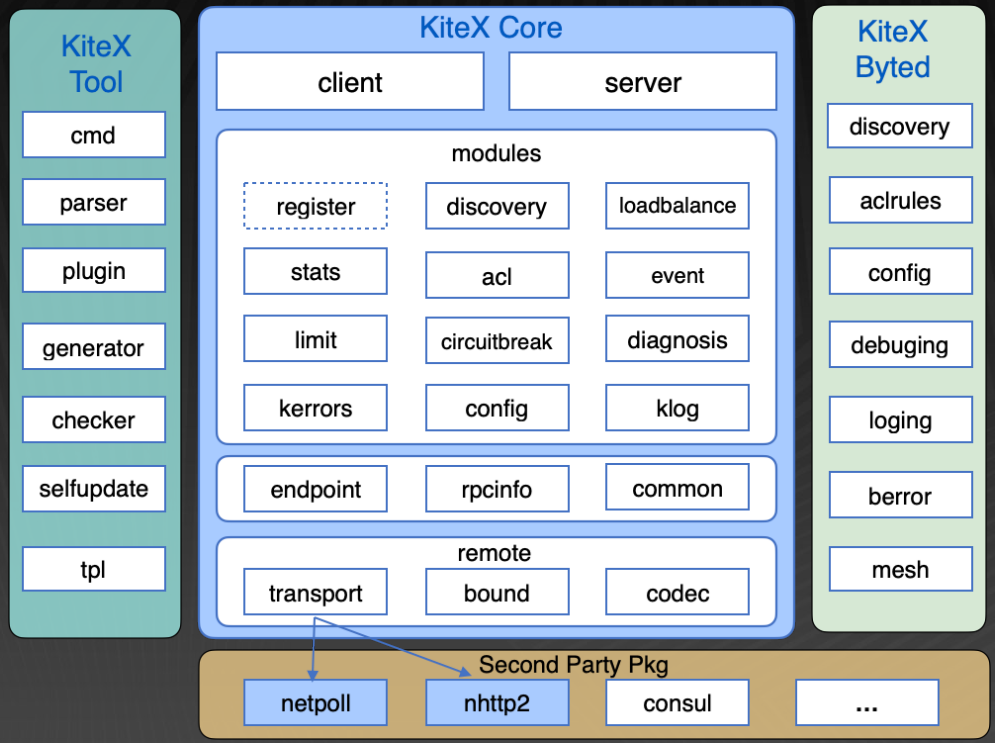
作为字节第一代 golang RPC 框架，kite 有一些历史原因的缺点：

1. 为了快速支持业务需求，耦合了部分中台业务的功能
2. 对 Go modules 支持不友好（Go modules 在 2019 年才进入语言核心）
3. 代码拆分成多仓库，版本更新推动困难
4. 强耦合了早期版本的 apache thrift，协议和功能拓展困难
5. 生成代码逻辑与框架接口强耦合，成为了性能优化的天花板
6. ……

业务的快速发展和需求场景的多样化，催生了新一代 golang RPC 框架的诞生

二、Golang RPC 框架 Kitex

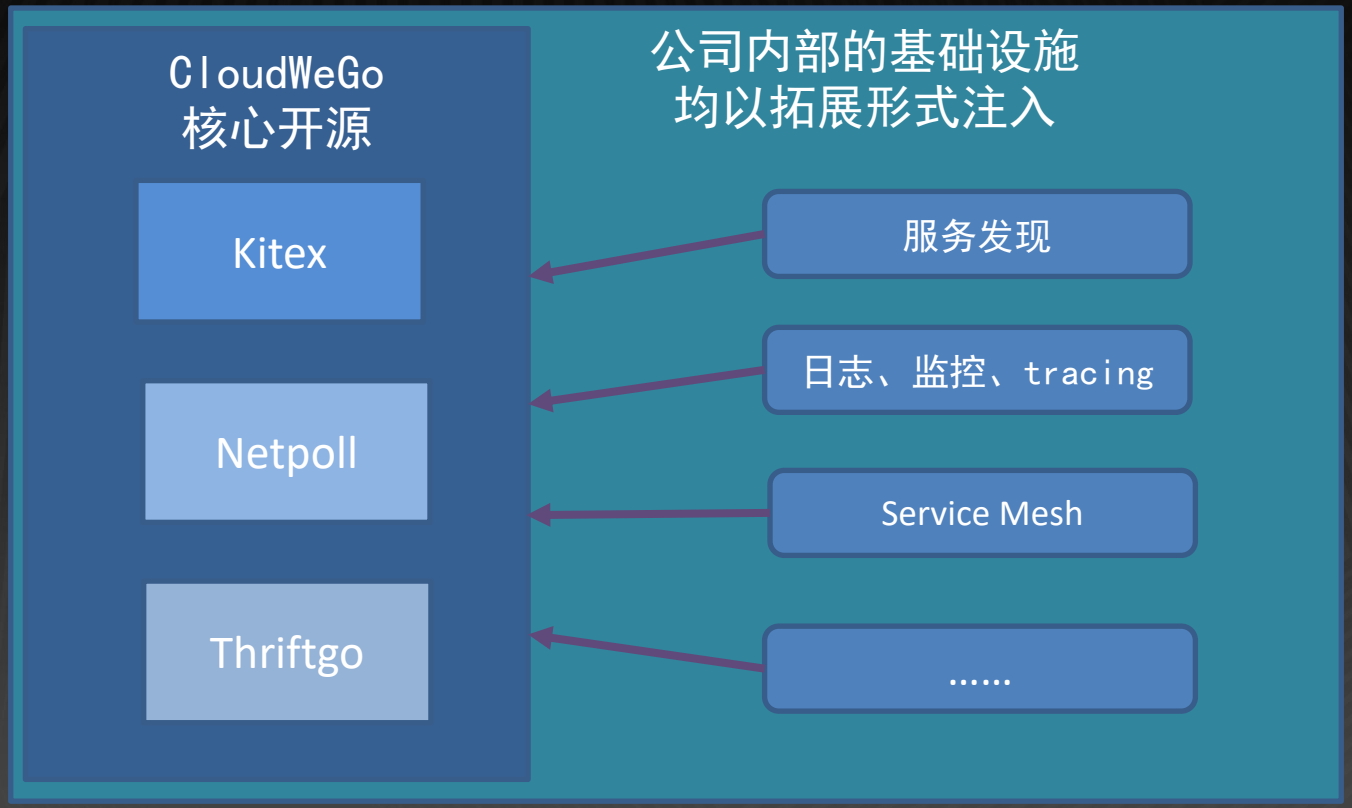
Kitex 的架构设计



Kitex 的主要特点：

1. 面向开源
2. 功能丰富
3. 灵活可拓展
4. 多协议
5. 高性能

面向开源的设计



CloudWeGo

面向开源的设计

```
// NewClient creates a kitex.Client with the given ServiceInfo, it is from generated code.
func NewClient(svcInfo *serviceinfo.ServiceInfo, opts ...Option) (Client, error) {
+--- 15 lines: ...----------------------------------------------------------------------
}

// NewServer creates a server with the given Options.
func NewServer(ops ...Option) Server {
+--  5 lines: ...-----------------------------------------------------------------------
}
```

Client 和 Server 的创建接口均采用 Option 模式，提供了极大的灵活性

CloudWeGo

Kitex 的功能特性：治理能力

CSDN

CloudWeGo

Kitex 的功能特性：治理能力

以服务发现为例

```
// Resolver resolves the target endpoint into a list of Instance.
type Resolver interface {
    // Target should return a description for the given target that is suitable for being a key for cache.
    Target(ctx context.Context, target rpcinfo.EndpointInfo) (description string)

    // Resolve returns a list of instances for the given description of a target.
    Resolve(ctx context.Context, desc string) (Result, error)

    // Diff computes the difference between two results.
    // When `next` is cacheable, the Change should be cacheable, too. And the `Result` field's CacheKey in
    // the return value should be set with the given cacheKey.
    Diff(cacheKey string, prev, next Result) (Change, bool)

    // Name returns the name of the resolver.
    Name() string
}
```

```
opt := client.WithResolver(new(MyResolver))
cli, err := myservice.NewClient("ServerName", opt)
if err != nil {
    panic(err)
}
```

Kitex 的功能特性：治理能力

使用 Suite 来打包自定义的功能
提供「一键配置基础依赖」的体验

```go
// A Suite is a collection of Options. It is useful to assemble multiple associated
// options as a single one to keep the order or presence in a desired manner.
type Suite interface {
    Options() []Option
}
```

```go
package mysuite

import "github.com/cloudwego/kitex/client"

type MySuite struct { ...  }

func (p *MySuite) Options() []client.Option {
    ... // do some calculation and decisions
    return []client.Option{
        client.WithResolver(myResolver),
        client.WithMiddleware(myMiddleware),
        client.WithLoadBalancer(myLoadbalancer),
        client.WithTracer(myOpenTracing),
        ...
    }
}
```

```go
cli, err := myservice.NewClient(
    "ServerName",
    client.WithSuite(new(mysuite)))
```

Kitex 的功能特性：多协议

基于自研的高性能网络库 Netpoll，支持了多种传输协议和调用方式

Kitex 的功能特性：代码生成工具

```
❯ kitex -module mydemo –service myservice demo.thrift

❯ tree
.
├── build.sh
├── demo.thrift
├── go.mod
├── handler.go
├── kitex_gen
│   └── demo
│       ├── demo
│       │   ├── client.go
│       │   ├── demo.go
│       │   ├── invoker.go
│       │   └── server.go
│       ├── demo.go
│       ├── k-consts.go
│       └── k-demo.go
├── main.go
└── script
    └── bootstrap.sh

4 directories, 13 files
```

简单易用的命令行工具

- 生成服务代码脚手架
- 支持 protobuf 和 thrift
- 内置功能丰富的选项
- 支持自定义的生成代码插件

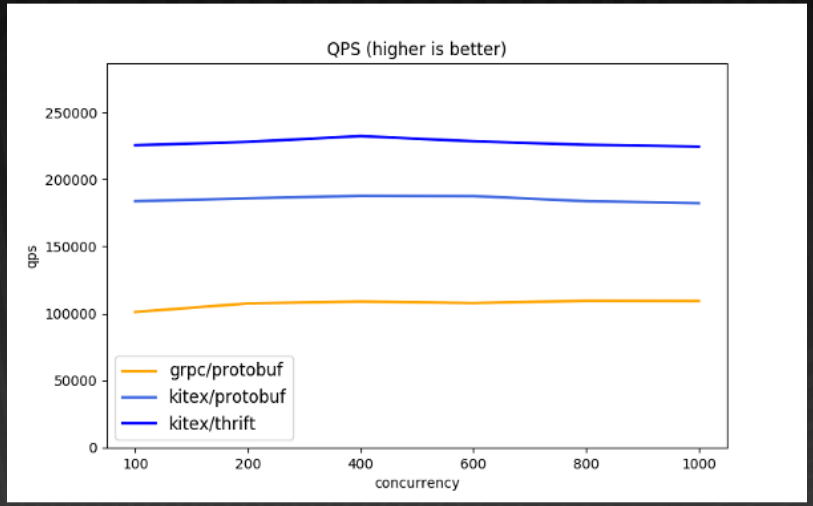
Kitex 的性能表现

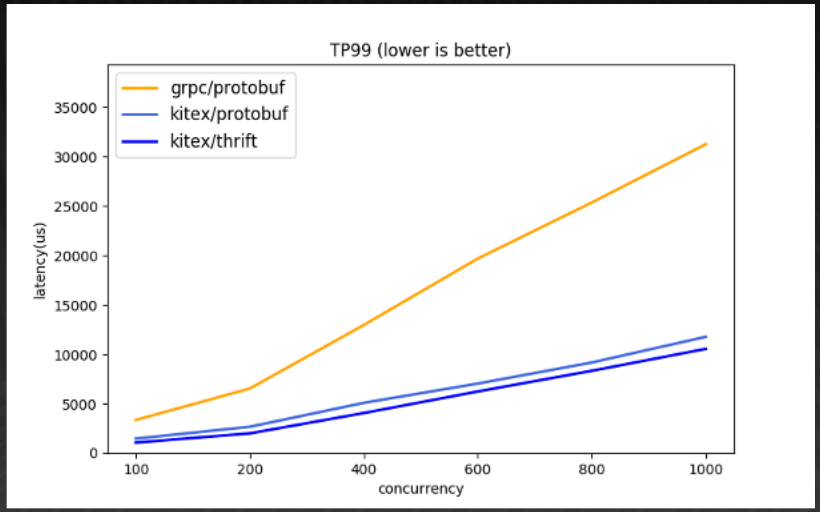
```
26 func (p *DemoTestArgs) FastRead(buf []byte) (int, error) {
27     var err error
28     var offset int
29     var l int
30     var fieldTypeId thrift.TType
31     var fieldId int16
32     _, l, err = bthrift.Binary.ReadStructBegin(buf)
33     offset += l
34     if err != nil {
35         goto ReadStructBeginError
36     }
37
38     for {
39         _, fieldTypeId, fieldId, l, err = bthrift.Binary.ReadFieldBegin(bu
   f[offset:])
40         offset += l
41         if err != nil {
42             goto ReadFieldBeginError
43         }
44         if fieldTypeId == thrift.STOP {
45             break
46         }
47         switch fieldId {
48         case 1:
49             if fieldTypeId == thrift.STRING {
50                 l, err = p.FastReadField1(buf[offset:])
51                 offset += l
52                 if err != nil {
53                     goto ReadFieldError
54                 }
55             } else {
56                 l, err = bthrift.Binary.Skip(buf[offset:], fieldTypeId)
57                 offset += l
58                 if err != nil {
```

结合 Netpoll 能力而设计的 FastRead/FastWrite 编解码实现

具有远超 apache thrift 生成代码的性能

Kitex 的性能表现





Kitex/gRPC性能对比（2022年1月数据）

注：kitex/protobuf 性能低于 kitex/thrift 是因为没有针对 pb 做 nocopy buffer优化
更多内容见：https://github.com/cloudwego/kitex-benchmark

Kitex：一个 demo

定义 IDL

demo.thrift

```
1  namespace * demo
2
3  service Demo {
4      string Test(1: string req);
5  }
6
```

生成代码

```
~/demo
❯ kitex -module mydemo -service cloudwego.kitex.demo demo.thrift

~/demo
❯ ls
build.sh    demo.thrift go.mod      handler.go  kitex_gen   main.go     script
```

Kitex：一个 demo

填充 handler 方法

handler.go

```go
package main

import (
    "context"
)

// DemoImpl implements the last service interface defined in the IDL.
type DemoImpl struct{}

// Test implements the DemoImpl interface.
func (s *DemoImpl) Test(ctx context.Context, req string) (resp string, err error) {
    resp = "你好，世界！"
    return
}
```

Kitex：一个 demo

编译、运行

```
~/demo
❯ go mod tidy

~/demo
❯ sh build.sh

~/demo
❯ ./output/bootstrap.sh
2022/01/08 11:09:20.707329 server.go:77: [Info] KITEX: server listen at addr=[::]:8888
```

Kitex：一个 demo

编写 client

client.go

```go
package main

import (
    "context"
    "log"
    "mydemo/kitex_gen/demo/demo"

    "github.com/cloudwego/kitex/client"
)

func main() {
    cli := demo.MustNewClient("
    ctx := context.Background()
    resp, err := cli.Test(ctx,
    if err != nil {
        panic(err)
    }
    log.Println("server:", resp)
}
```

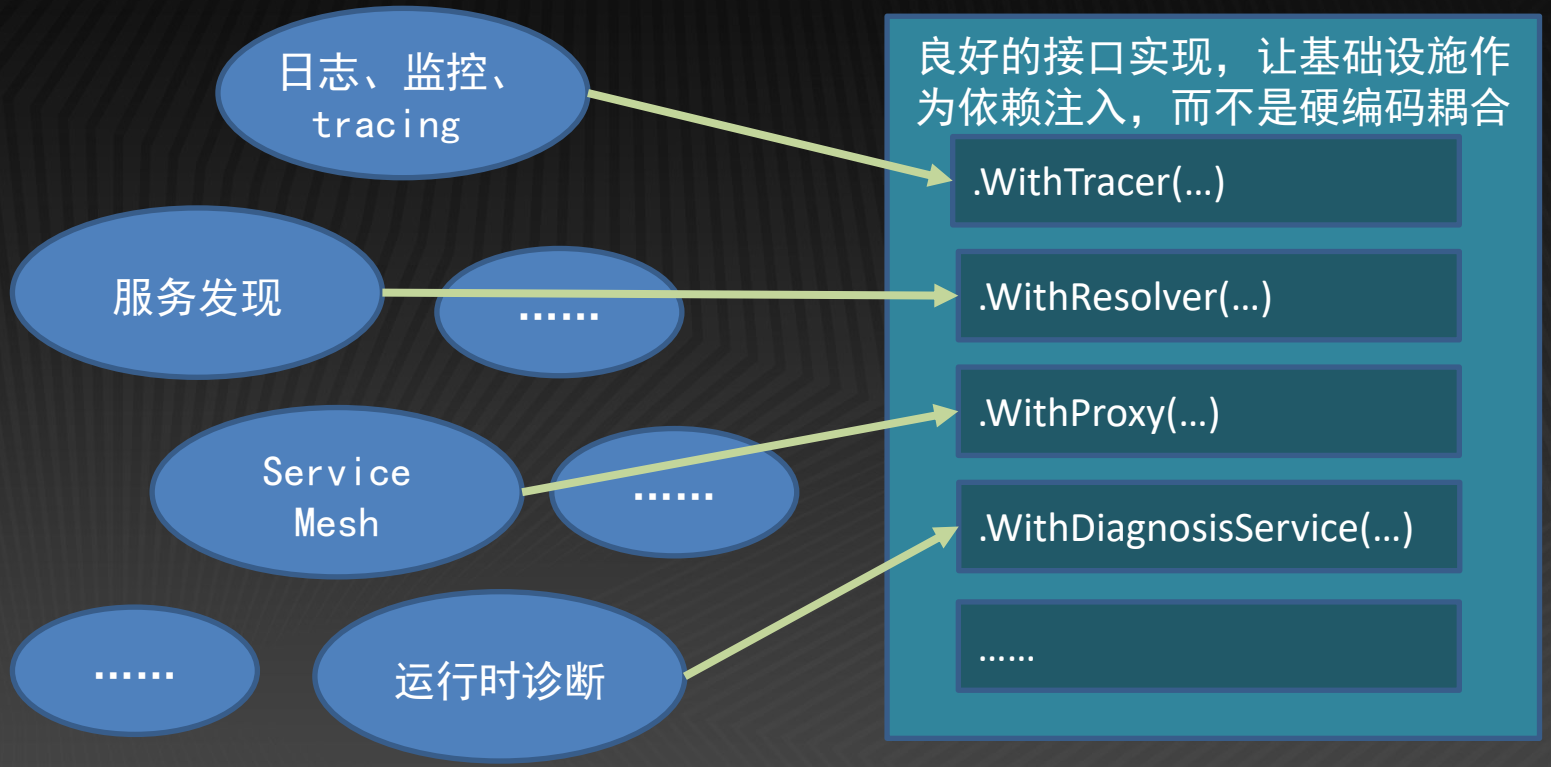
```
~/demo/client
❯ go run client.go
2022/01/08 11:11:33 server: 你好，世界！
~/demo/client
❯
```

CSDN

CloudWeGo

三、Kitex 在字节内部的落地

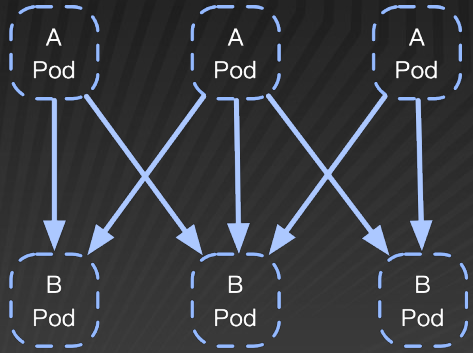
与内部基础设施的集成


日志、监控、tracing
服务发现
.....
Service Mesh
.....
.....
运行时诊断
良好的接口实现，让基础设施作为依赖注入，而不是硬编码耦合
.WithTracer(...)
.WithResolver(...)
.WithProxy(...)
.WithDiagnosisService(...)
.....

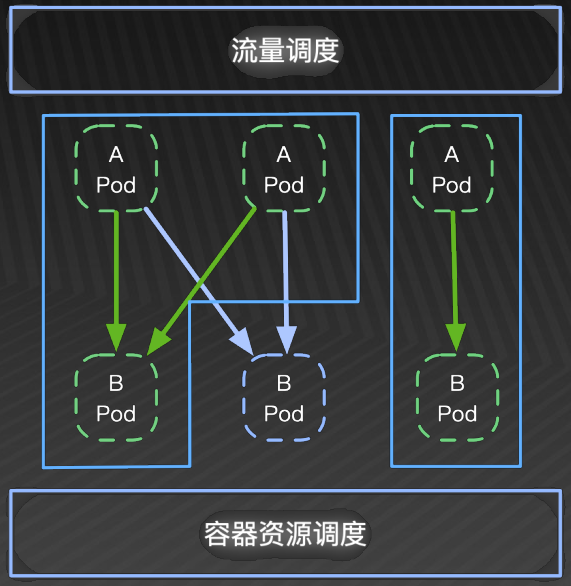
内部落地的典型案例：合并部署

- 微服务过微，网络传输和序列化开销越来越大
- 将强依赖的服务同机部署，有效减少资源消耗

非合并微服务调用



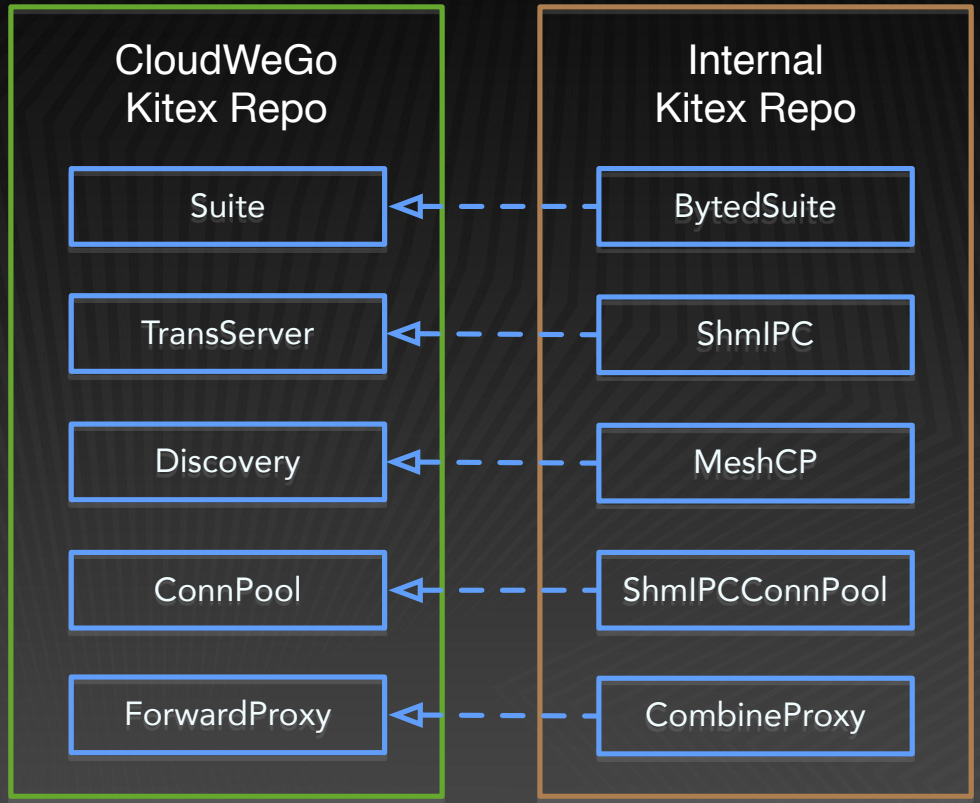
合并微服务调用



Remote Call

Local IPC

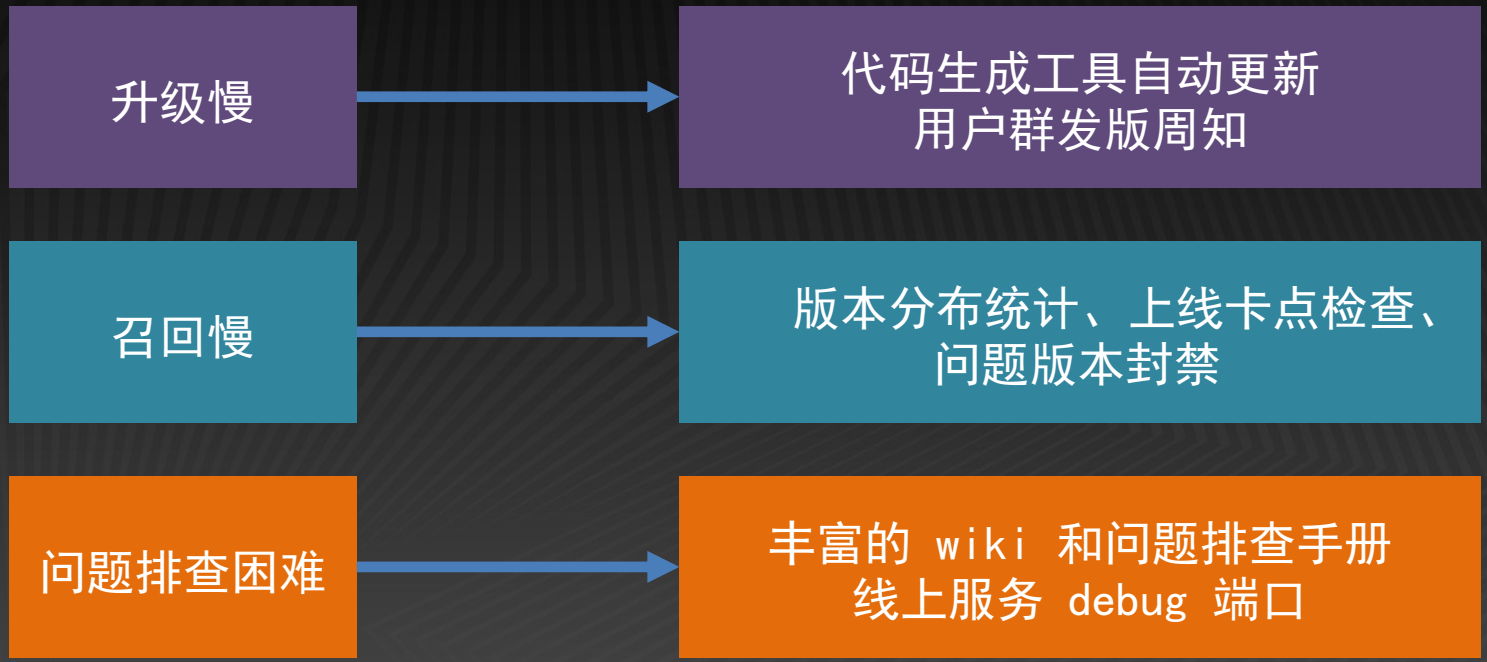
内部落地的典型案例：合并部署



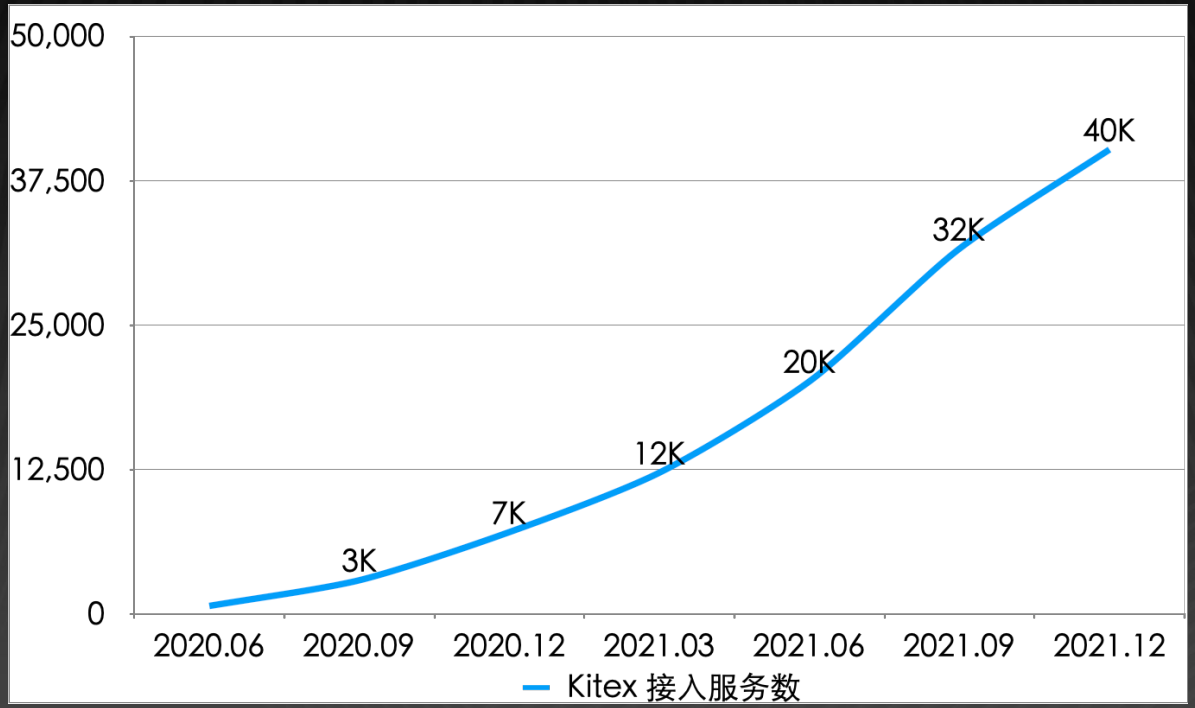
- 中心化的部署调度和流量控制
- 基于共享内存的通信协议
- 定制化的服务发现和连接池实现
- 定制化的服务启动和监听逻辑

某抖音服务，30% 合并流量，服务端 CPU 19%↓ ，延迟TP99 29%↓

微服务框架推进的痛点



Kitex 在字节内部的发展



四、Kitex 的开源实践

开源筹备

1. 代码层面
2. 文档方面
3. 社区运营

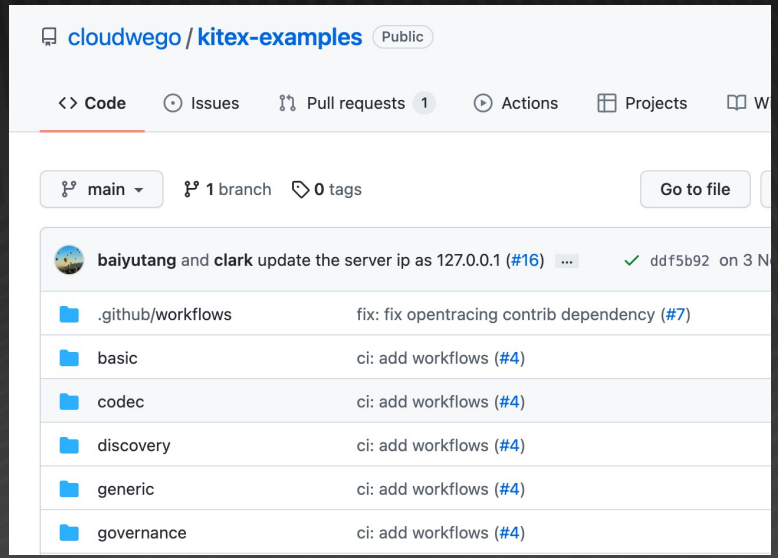
开源筹备：代码层面

1. 代码拆分、脱敏
2. 内部仓库引用开源仓库，避免内外多副本同时维护
3. 在开源过程中确保内部用户平滑切换、体验无损

CloudWeGo

开源筹备：文档方面

1. 组织用户文档，覆盖方方面面
2. 建设用例仓库（cloudwego/kitex-examples）



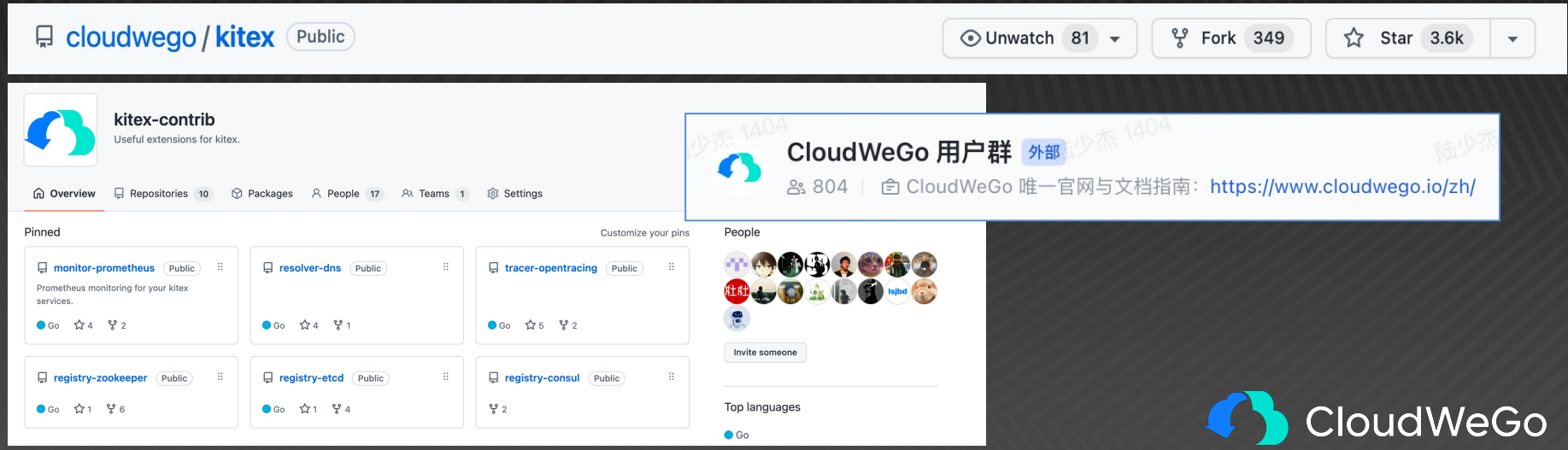
CloudWeGo

开源筹备：社区运营

1. 官网建设
2. 用户群维护，答疑解惑
3. 飞书机器人对接 issue 管理、PR 管理，快速响应
4. 优秀贡献者激励

社区建设的成果

- cloudwego/kitex 收获 3.5k+ stars
- kitex-contrib 获得多个外部用户贡献的仓库
- cloudwego 飞书用户群近 800 个用户
- ……



CloudWeGo

未来的展望

- 持续向开源社区反馈最新的技术进展
  - thrift JIT
  - protobuf 编解码性能优化
  - 支持 xDS
  - ……
- 拓展更多开源组件
- 对接更多公有云基础设施

需求驱动，反馈开源，共建生态

CSDN

Thanks

cloudwego.io

CloudWeGo 用户飞书群

火山引擎开发者社区

CloudWeGo

成就一亿技术人

成为技术人交流和成长的家园

用户为本 | 求真求是 | 协作共赢 | 极客精神 | 结果导向

CSDN