CLOUDWEGO

CLOUDWEGO DAY #2 CLOUDWEGO DAY #2 CLOUDWEGO DAY #2 CLOUDWEGO DAY #2

CloudWeGo

流程引擎使用CloudWeGo的落地实践

CloudWeGo | 稀土掘金 出品

讲师：王晨

2023/03/25

自我介绍

王晨

字节高级开发工程师

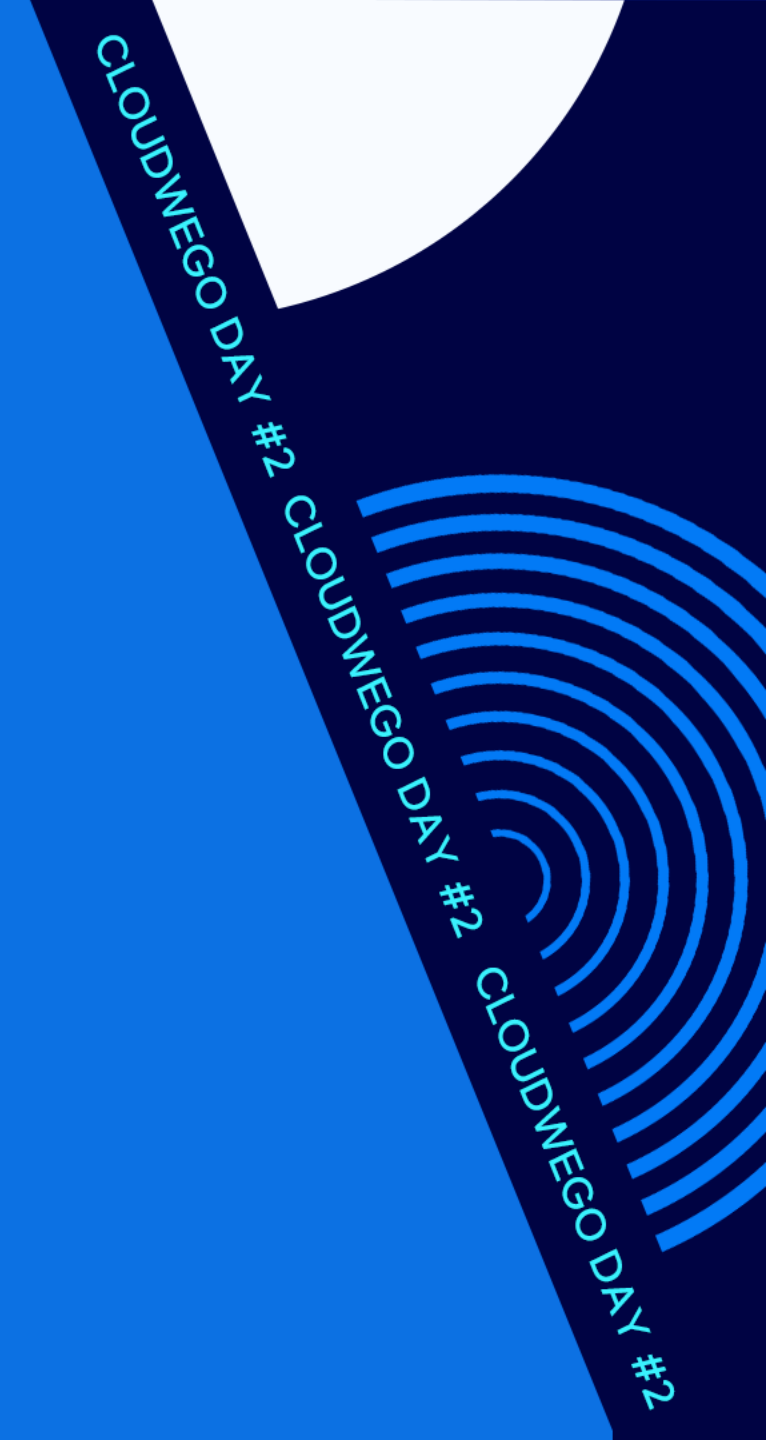
讲师简介:

互联网行业技术老兵，主要从事流程编排、数据中心，客服用户体系的建设。技术栈主要为Java、Golang等。对新技术在行业内的应用持续敏锐。

目录

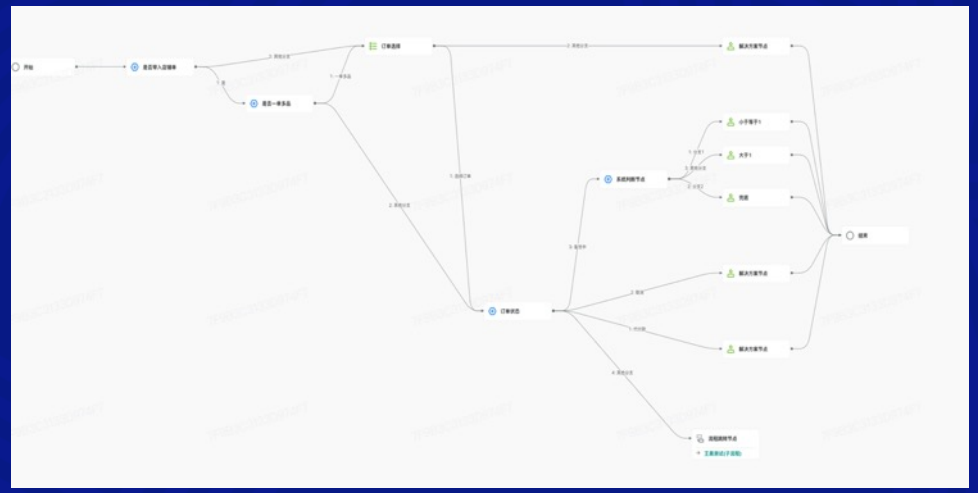
场景一：压测分流

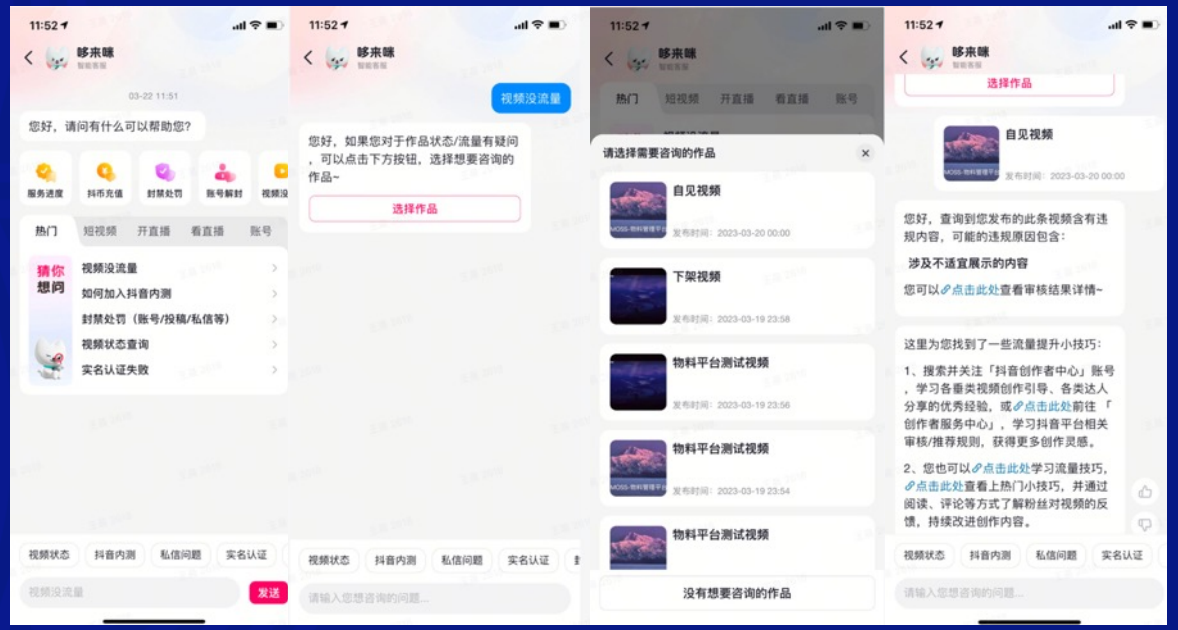
流程引擎使用 Gorm + Kitex + ByteMesh 落地实践



项目简要介绍

- 流程引擎: 主要负责流程编排和流程执行。
- 应用场景: 智能客服的多轮会话和人工客服的坐席辅助、审批场景。
- 线上部署流程 1000+，每天流程拉起量 300W+，流程节点数平均 100+。





当时遇到了什么问题

- 预览测试--流程在正式发布上线前，需要寻找大量的历史数据验证流程的正确性。
- 预览数据需要和线上真实数据隔离，避免影响线上数据统计。
- 业务代码改动尽量小的情况下，一套代码支持上述两种场景。

技术选型-Gorm

- 业界常用的方案是采用影子表隔离线上流量和压测流量。
- Gorm 是一个使用 Go 语言编写的 ORM 框架。它文档齐全，对开发者友好，支持主流数据库，支持压测影子表。

```
bytedgorm.WithStressTestSupport(),
```

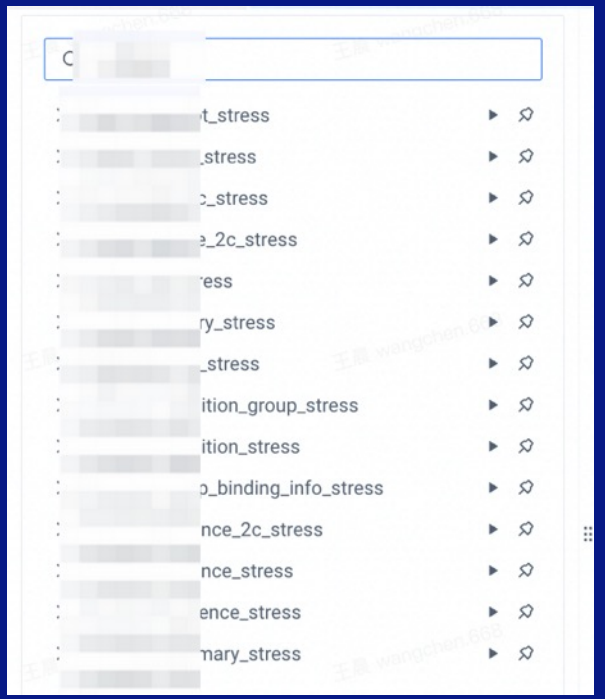
```
func (s *stressTest) stressTestCallback(readMode bool) func(db *gorm.DB) {
    return func(db *gorm.DB) {
        if db.DryRun || !s.isTestRequest(db.Statement.Context) {
        if readMode && shouldSkipStressTestsForRead(db.Statement.Context) {
        // Raw SQL, for example:
        //   DB.Raw("select sum(age) from users where name = ?", "name").Scan(&ages)
        // Skip shadow table when SELECT SQL & ContextSkipStressForRead is true
        rawSQL := db.Statement.SQL.String()
        if shouldSkipStressTestsForRead(db.Statement.Context) && len(rawSQL) > 6 && strings.EqualFold(rawSQL[:6],
            return
        }
        // Use shadow table
        if rawSQL != "" {
            db.Statement.SQL.Reset()
            db.Statement.SQL.WriteString(s.replaceWithShadowTable(db, rawSQL))
            return
        }
        if db.Statement.TableExpr != nil {
            // Irregular Table, for example:
            //   DB.Table("users as u").Find(&users)
            //   DB.Table("`users` as u").Find(&users)
            //   DB.Table("(?) as u", db.Model(User{}).Select("Name")).Find(&users)
            db.Statement.TableExpr.SQL = s.replaceWithShadowTable(db, db.Statement.TableExpr.SQL)
            return
        }
        if db.Statement.Table != "" {
            db.Statement.Table = s.getSuffixedTableName(db, db.Statement.Table, s.StressTestTableSuffix)
        }
    }
}
```

技术选型-Gorm

- 使用非常简单，只需要识别是压测流量，然后在 Context 中增加压测标示。

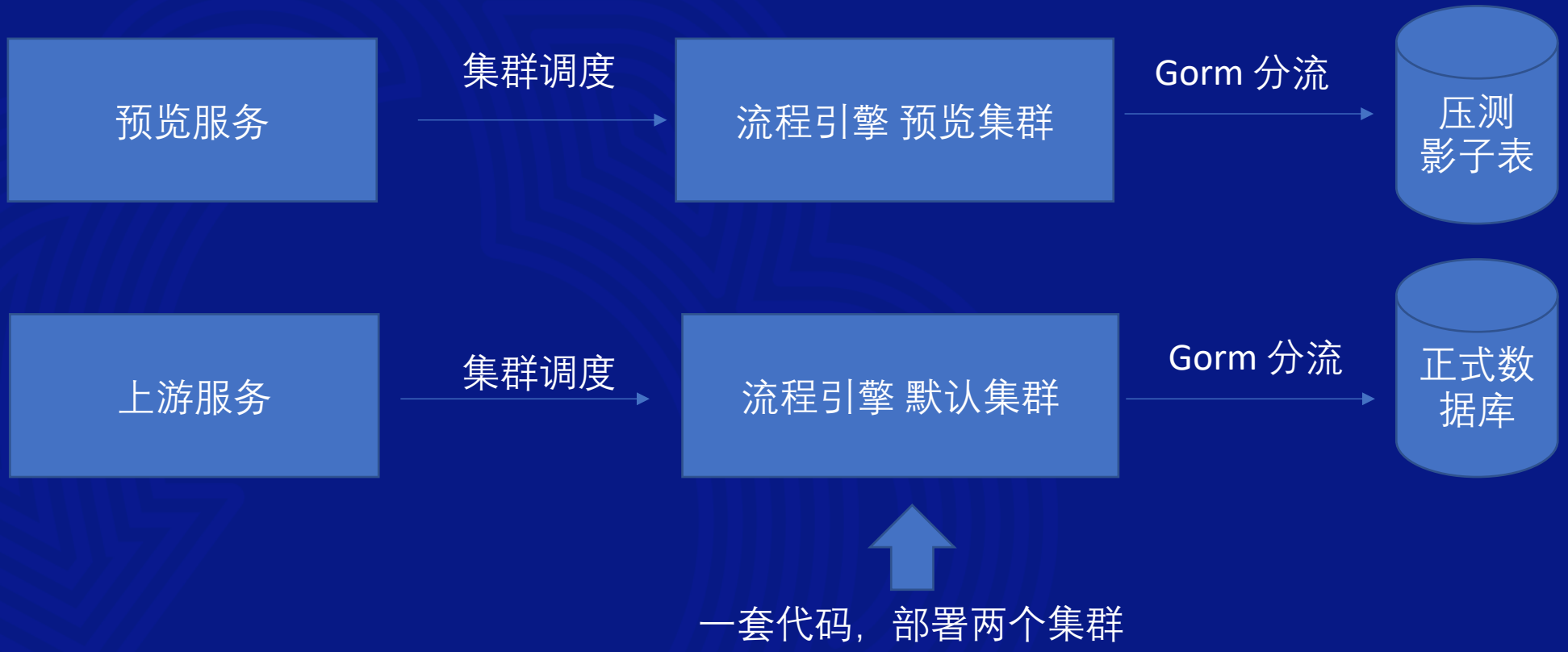
```go
func ChangeContext(ctx context.Context) context.Context {  14 usages  wangchen
    if IsPreviewRequest(ctx) {
        logs.CtxInfo(ctx, format: "ChangeContext")
        if _, ok := ctx.Value( key: "K_STRESS").(string); !ok {
            logs.CtxInfo(ctx, format: "add stress tag")
            return context.WithValue(ctx, key: "K_STRESS", val: "true")
        }
    }
    return ctx
}
```

- Gorm 框架识别压测标示后，动态修改 SQL 语句，将数据存储在影子表。
- 影子表默认标示是 ${TableName}_stress，后缀支持自定义。



技术选型-Gorm

- 基于 Gorm，很容易就实现了以下功能，可以更专注在业务逻辑上。



技术选型-Kitex + ByteMesh 流量调度很方便

- ByteMesh

它实现了一个高性能多协议的代理和一个灵活可扩展的控制服务，将他们与云平台原生集成，最终通过轻量级的 RPC 框架来输出能力。

场景二：频繁迭代，快速试错

用户体系数据迁移使用 FAAS 落地实践

当时遇到了什么问题

- 客服用户体系，需要重构，字段多(80+)，涉及两个服务 10张表，存在关联关系，历史的数据较乱，测试环境和线上数据存在较大差异。
- 新系统开发、数据迁移、双写需要同步进行。
- 任务重，时间紧(2个月完成)，数据校对和迁移部分风险小(老表只读)，需要快速试错。

技术选型–函数式编程+分布式任务平台

- FaaS 即 **F**unction **a**s **a** **S**ervice，是事件驱动的全托管计算服务。它遵循服务函数化理念，支持用户一键创建和部署函数，支持快速构建任何类型的应用和服务。

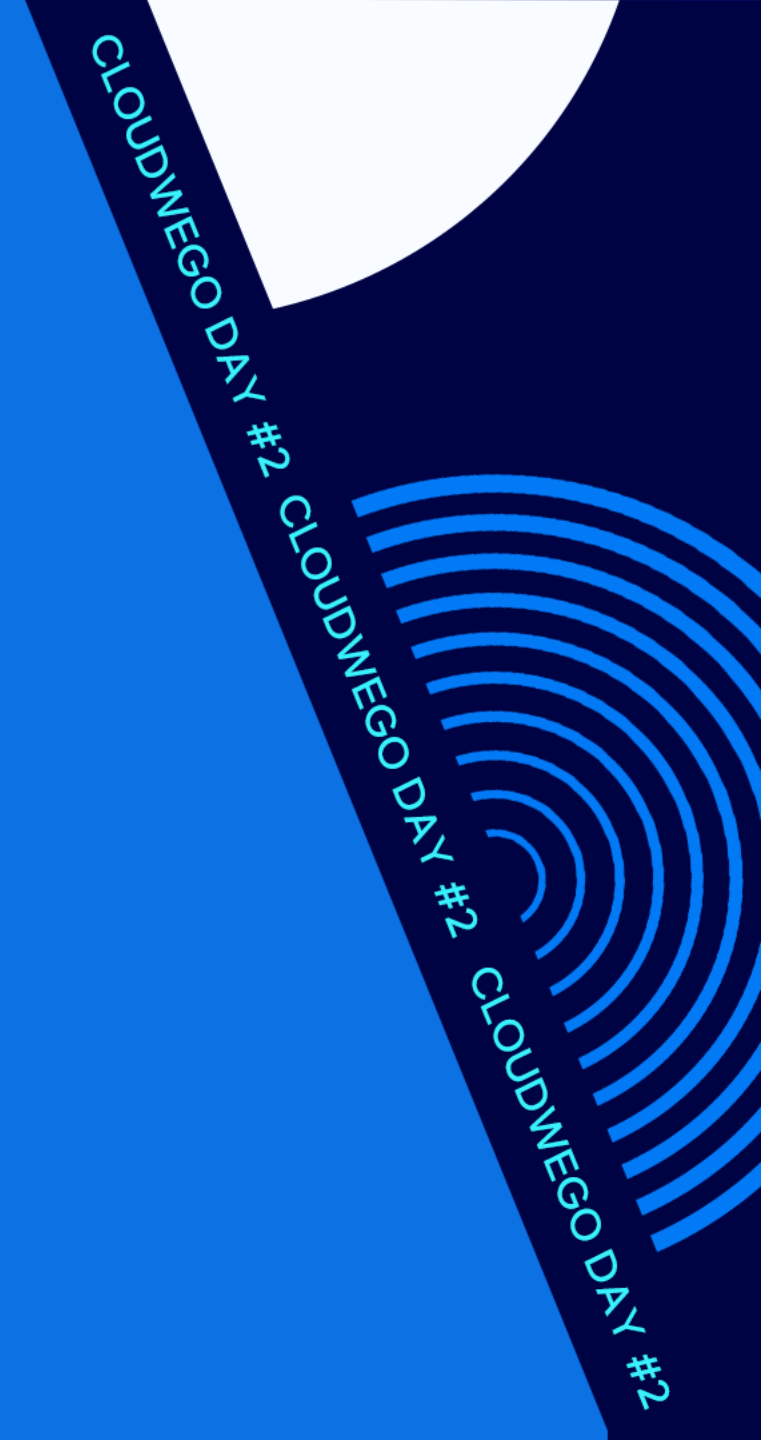
- 分布式任务平台 [Cronjob] 可以支持镜像以及自定义镜像方式运行定制化任务，提供了定时运行、API 触发运行以及任务依赖执行三种调度方式。

技术选型–函数式编程+分布式任务平台

- 快速试错，部署100 多次。
- 顺利完成了项目上线。体验非常好，减少因为编译部署的打断。

场景三：大流量高并发场景

数据中心依赖 Kitex 的落地实践



当时遇到了什么问题

- 客服系统需要接入的信息量大，调用方多【400+】，字段多【2000+】。

数据动态获取

数据裁剪

二次加工

数据聚合

泛化调用

动态确定执行计划

自定义脚本

请求自定义DSL

流量大

稳定性要求高
(影响面大)

信息接入成本低

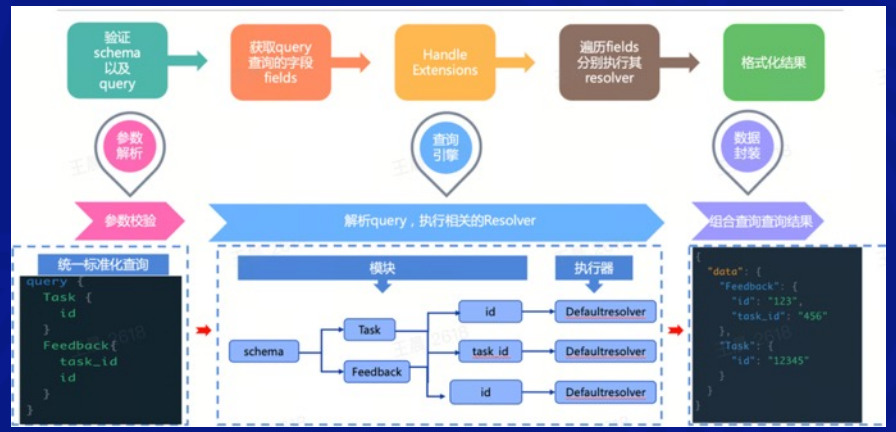
技术选型

- Golang 的协程模式和 Kitex 的 NIO 提供了很好的性能基础。
- 开源的技术积累提供了很好的参考。Dynamic Thrift 、GQL、Gengine、GoValuate、GoRule。
- 基于 Kitex + ByteMesh 可以轻松实现限流、超时、重试的管理。

入流量

Byte Mesh

Kitex

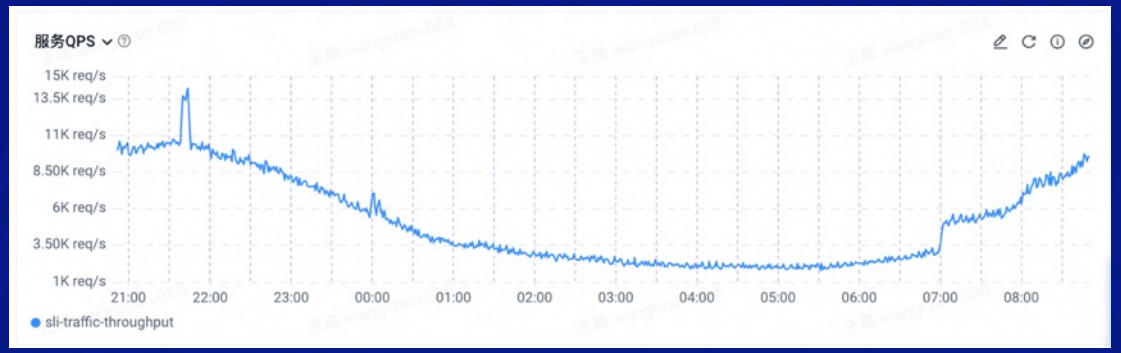


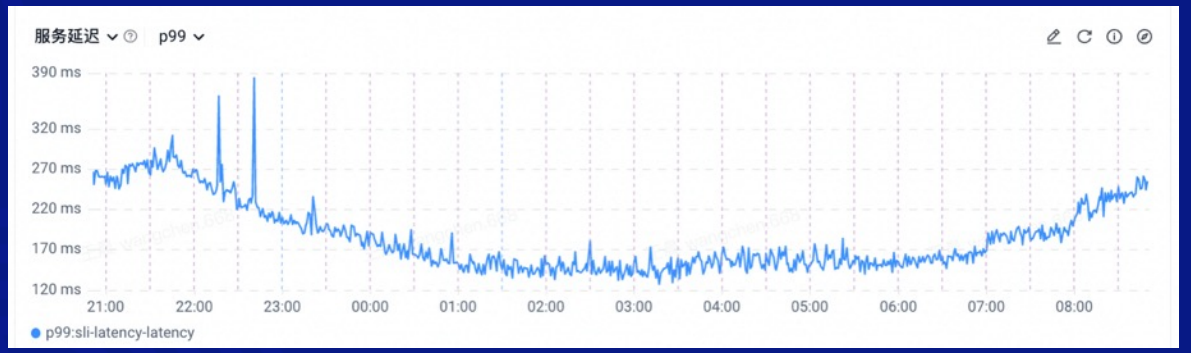
出流量

Byte Mesh

技术结果

- QPS 最高 1.3 W
- 支撑了客服平台绝大部分的信息接入场景，服务PCT 99 平均200ms





服务技术选型的考虑点

1. 基础设施是否完善
   - 服务发现、服务治理、服务稳定性手段、服务可见性
2. 生态是否完善
   - 平台支持程度【分布式定时任务、FAAS等】
   - SDK丰富程度【不用从 HTTP 开始写起】
3. 性能
   - RPC 性能
   - 编译和启动性能【可以减少研发分心】
   - 内存占用【QPS 越大效果越好】
4. 扩展性
   - 是否可定制、可扩展
5. 学习成本&&新人是否友好

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- 接口

  如果某个东西长得像鸭子，像鸭子一样游泳，像鸭子一样嘎嘎叫，那它就可以被看成是一只鸭子

  Go 语言作为一门静态语言，它通过通过接口的方式完美支持鸭子类型。

```go
type IGreeting interface {
    sayHello()
}

func sayHello(i IGreeting) {
    i.sayHello()
}
```

```go
type Go struct {}
func (g Go) sayHello() {
    fmt.Println("Hi, I am GO!")
}

type PHP struct {}
func (p PHP) sayHello() {
    fmt.Println("Hi, I am PHP!")
}
```

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- 大小写

  Go 中根据首字母的大小写来确定可以访问的权限。如果首字母大写，则可以被其他的包访问；如果首字母小写，则只能在本包中使用。

- 循环依赖
  - 原生 Go 不支持循环依赖，这一点和 Spring 框架有很大不同。

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- For Range 副本覆盖

```go
func doTestForRange() {  1 usage  new *
    messages := make([]Message, 0)
    messages = append(messages, Message{MessageType: "1"})
    messages = append(messages, Message{MessageType: "2"})

    msgTypes := make([]*string, 0)
    for _, msg := range messages {
        msgTypes = append(msgTypes, &msg.MessageType)
    }
    for _, msgType := range msgTypes {
        fmt.Println(*msgType)
    }
}
```

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- Context VS Thread-Local
  - Java 中传递日志 ID 需要通过 Thread-Local，如果启动新的线程，需要使用方处理[除非包装创建线程的方法]。
  - GO 中日志 ID 通过 Context 对象传递，跨线程不存在丢失的问题。

```
asyncExecutor.execute(
        () -> {
            try {
                RequestIDUtil.setCurrentRequestID(logId);
                org.slf4j.MDC.put( key: "LOG_ID", logId);
```

```
public class BasicMDCAdapter implements MDCAdapter {

    private InheritableThreadLocal<Map<String, String>> inheritableThreadLocal = new Inheritab
        @Override
        protected Map<String, String> childValue(Map<String, String> parentValue) {
            if (parentValue == null) {
                return null;
            }
            return new HashMap<String, String>(parentValue);
        }
    };
```

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- 优雅的资源释放-使用defer

```java
InputStream fin = null;

try {
    // open file and read
fin = new FileInputStream(...);
    String line = fin.readLine();
    System.out.println(line);
} catch (Exception e) {
    // handle exception
} finally {
    fin.close();
}
```

```go
tosObject, err := client.GetObject(context.Background(), fileKey)
defer tosObject.R.Close()
```

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- gopkg

  由 CloudWeGo 与字节跳动的语言团队合作维护，里面包含也了对 Golang 标准库能力的增强。

  https://github.com/bytedance/gopkg

```
slicex.DistinctBy(sendContent.SendItems, func(t *SendItem) string {
    return t.InfoKey
})
sendContent.SendItems = slicex.Filter(sendContent.SendItems, func(t *SendItem) bool {
    return t.QueryCount > 10
})
sort.Sort(sendContent.SendItems)
```

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- Kitex 和 Hertz 定制化

```
svr := sopcheckerservice.NewServer(new(SopcheckerServiceImpl), server.WithMiddleware(mw.AccessLogMW))
```

```
func AccessLogMW(next endpoint.Endpoint) endpoint.Endpoint {  1 usage  qiuxuliang
    return func(ctx context.Context, request, response interface{}) error {
        logs.CtxInfo(ctx,  format: "[AccessLogMW] request: %v", printer.SPrintStruct(request))
        err := next(ctx, request, response)
        logs.CtxInfo(ctx,  format: "[AccessLogMW] response: %v", printer.SPrintStruct(response))
        return err
    }
}
```

```
func InitHttpServer() {  1 usage  wangchen *
    byted.Init()
    r := byted.Default()
    customizeRegister(r)
    r.Spin()
}

// customizeRegister register customize routers.
func customizeRegister(r *server.Hertz) {  2 usages  wangchen *
    baseAPI := r.Group( relativePath: "/approval_runner_api")
    baseAPI.GET( relativePath: "/ping", handler.Ping)

    approvalApi := baseAPI.Group( relativePath: "/api")
    approvalApi.Use(login.DoorgodMiddleware)
    approvalApi.GET( relativePath: "/doApproval", handler.GetApprovalPage)
    approvalApi.POST( relativePath: "/doApproval", handler.DoApproval)

    botApi := baseAPI.Group( relativePath: "/bot")
    botMap, err := dal.GetApprovalBotConfig(context.TODO())
    for path, _ := range botMap {
        botApi.POST(path, handler.LarkCallBack)
        logs.Info( format: "customizeRegister path:%+v", path)
    }
```

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- 常用的脚手架脚本放在MakeFile 中。

```
PSM                     =
BOOSTRAP_SCRIPT         =       output/bootstrap.sh
BUILD_SCRIPT            =       build.sh

IDL_DIR=${GOPATH}/src/code.byted.org/cpputil/service_rpc_idl

gen-thrift:
    rm -rf kitex_gen
    kitex -module code.byted.org/ies-cs/staff -service ${PSM} idl/ies_kefu_staff.thrift
    git add .

gen-db:
    comments_build_tools --target=orm_v3 --ddlpath=app/db/tables/ddl.sql --package=tables > app/db
    git add .
```

Go 语言 && Kitex && Hertz 使用小例子&&工具包推荐

- *CloudWeGo 官网：https://www.cloudwego.io/*
- *Kitex https://github.com/cloudwego/kitex*
- *Hertz https://github.com/cloudwego/hertz*
- *gopkg https://github.com/bytedance/gopkg*
- *gorm https://gorm.io*

Java转GO的一些思考

1. 对上/对下达成一致，志在必得。
2. 时间要选对，项目不要太忙的时候。
3. 快速试错，小步快跑，战线尽量缩短。

GO语言学习	开始干，写demo	定位调试
2W 产出各种 本地小程序	Kitex、Hertz、 Gorm、Redis、 MQ	Panic 小助手 PPROF
多尝试 坑点自行体验	善用代码 生成工具	多分析，问题 暴露越早越好

4 个后端 1个月完成项目【流程引擎】开发。

谢谢大家

CLOUDWEGO

CLOUDWEGO DAY #2 CLOUDWEGO DAY #2 CLOUDWEGO DAY #2 CLOUDWEGO DAY #2

CloudWeGo

THANKS

CloudWeGo | 稀土掘金 出品

2023/03/25

扫码关注公众号