

# 从微服务到 Serverless 架构 -享受纯粹的编程乐趣

王晓波

同程艺龙 机票事业群

TABLE OF

# CONTENTS 大纲

---

- 同程实践 Serverless 的背景
- Serverless 与传统架构比较
- 同程的 Serverless 实现了什么
- Serverless 架构在同程的一些实际使用那些业务场景分析
- 我们的下一步在 Serverless 上要做什么

一条 SQL 到一个服务的距离到底有多远

# 环境，框架，依赖，太难搞



- 各种环境的统一性
- 各种开发框架的烦心事
- 各种调用的乱依赖
- 代码以外花了多少时间

# 部署, 运维, 扩容, 太麻烦



- 部署在哪, 多少实例
- 各种配置, 上线, 下线
- 扩个容各种关注点
- 真的可以人人是全栈吗

# 调试, 性能, 安全, 太复杂



- 开发调试最烦对外依赖
- 性能往往做好后才细想
- 安全出了问题才懂了它
- 专业领域还有多少

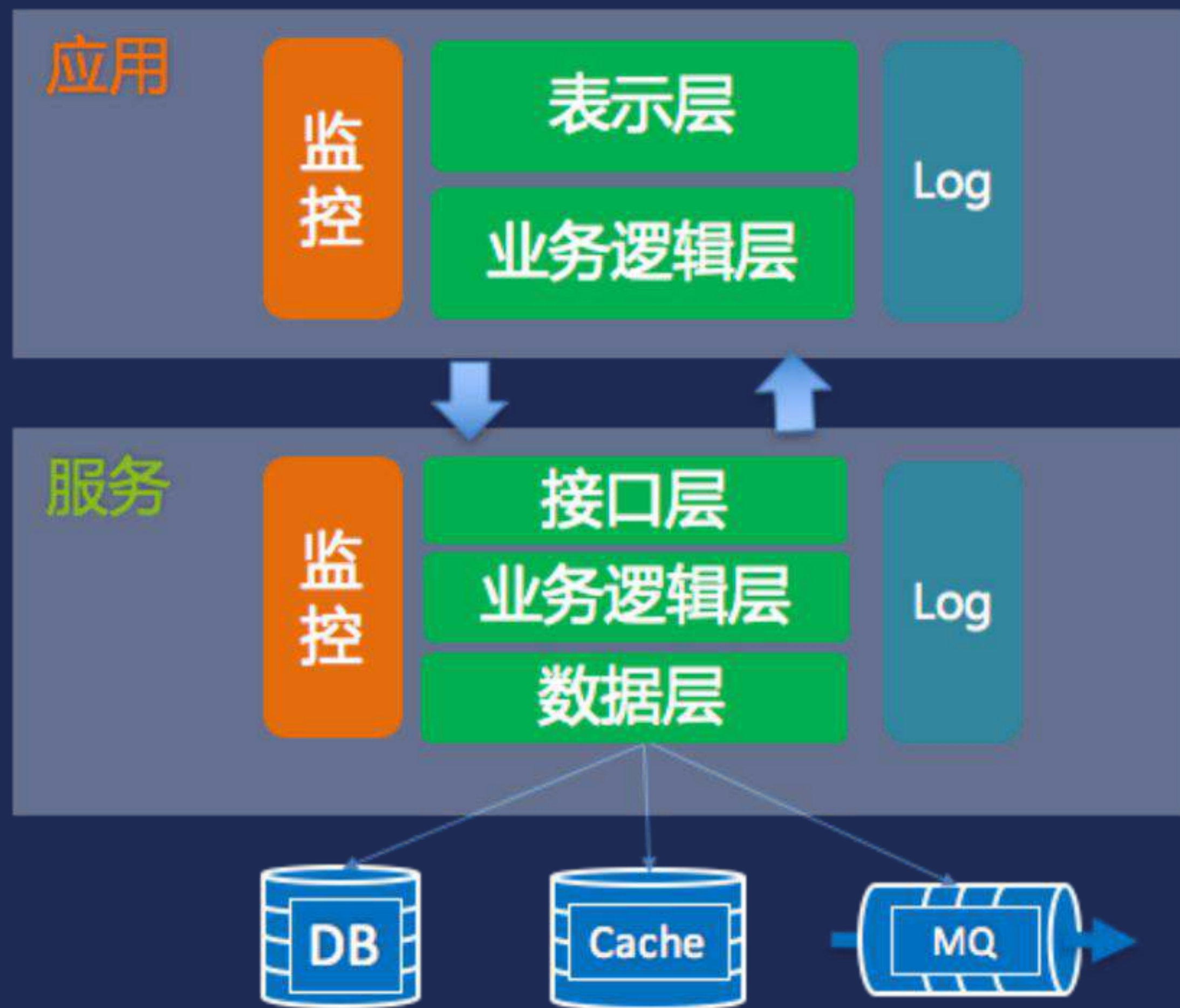
从一条 SQL 到一个可用服务距离真的很远很远

如何安安静静的写代码???

让程序员享受纯粹的编码乐趣



# 传统的系统架构



特点:

应用多年大家多理解  
也有粗粒度的服务展现  
拥有一定的系统伸缩性能力  
重了框架国轻了架构

# 传统架构的缺点

- ★ 一个简单的应用变的不简单了

- ★ **问题:**

不只是个网站

各种应用增加到N千多个，想一想玩不动了

流量动不动就是海量（每天数亿级请求量

原来很简单的一句SQL，现在没法用了

最痛苦的是连缓存也跑不动了

服务器加到不知道放到哪里好

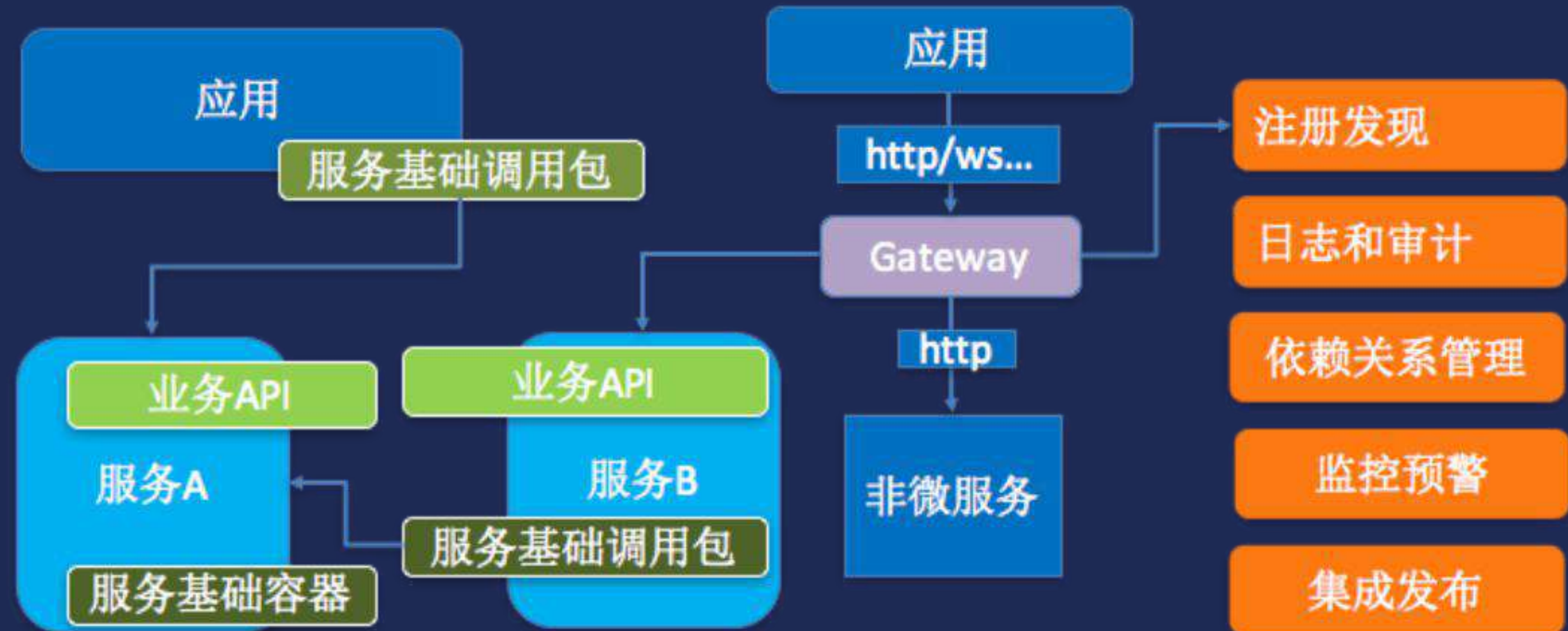
运维最忙的事怎么是各种背不完的锅

总体看来就是又大又乱，一头雾水



# 同程的微服务架构

接口的统一  
容错处理（限流、回退、隔离、熔断）  
全链路的服务监控  
服务注册发现，控制服务的API数量  
统一代码框架，支持多种编程语言  
服务依赖关系管理(服务的分级)  
服务的CI/CD

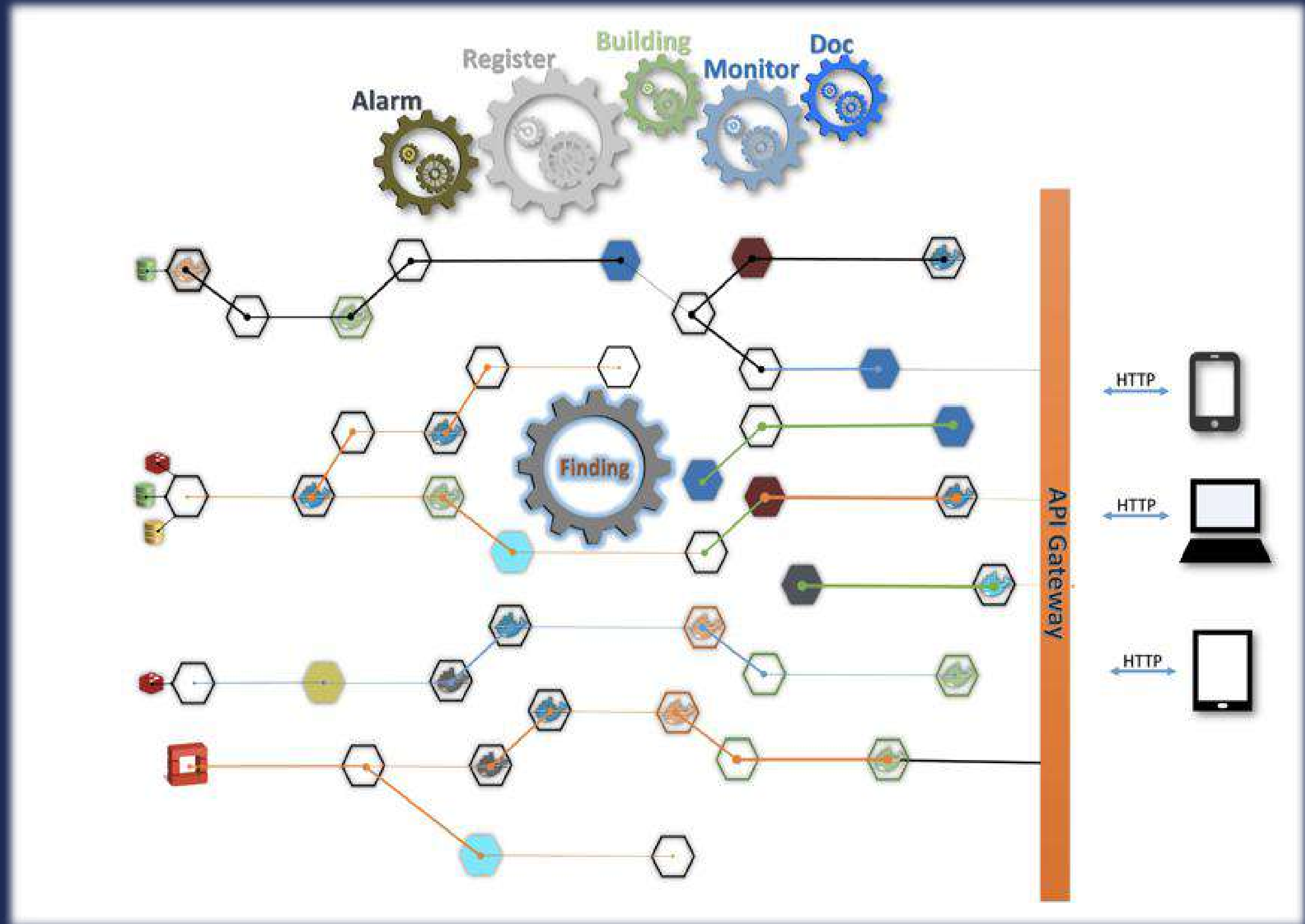


# 微服务架构的缺点

- ★ 微服务整体推进是要时间，不可能一次改完，中间过程需要过渡
- ★ 还有很多为快速打样的做的单体应用，或一些简单应用
- ★ 一个微服务经过长时间迭代更新后也需要新重构

## 问题：

整个体系中并非只有微服务管理，调用多会因为这个不得不变复杂起来了  
写一个微服务的成本也并不小  
开发过程的调试比较麻烦  
服务的数量越来越多虽然跑在**Docker**之上但也是资源  
运维虽然**DEVOPS**了但依然变的有点麻烦



一个代码脚本能不能就是一个微服务

# 同程在微服务化之后的需求

★ 总结起来这一切就是自己挖个坑和自己再填上坑

- ① 用一两个小时写个服务并上线
- ② 不用关心服务器在哪部多少
- ③ 只需要配管人员的配置接下来就完全是代码的事
- ④ 开发人员可以像操作IDE那样完成他不熟悉的技能
- ⑤ 代码没跑最好不要计算资源成本
- ⑥ 不需要人人都是全栈工程师，他就想好好写倾
- ⑦ 运维就是智能的，让系统与算法去解决问题



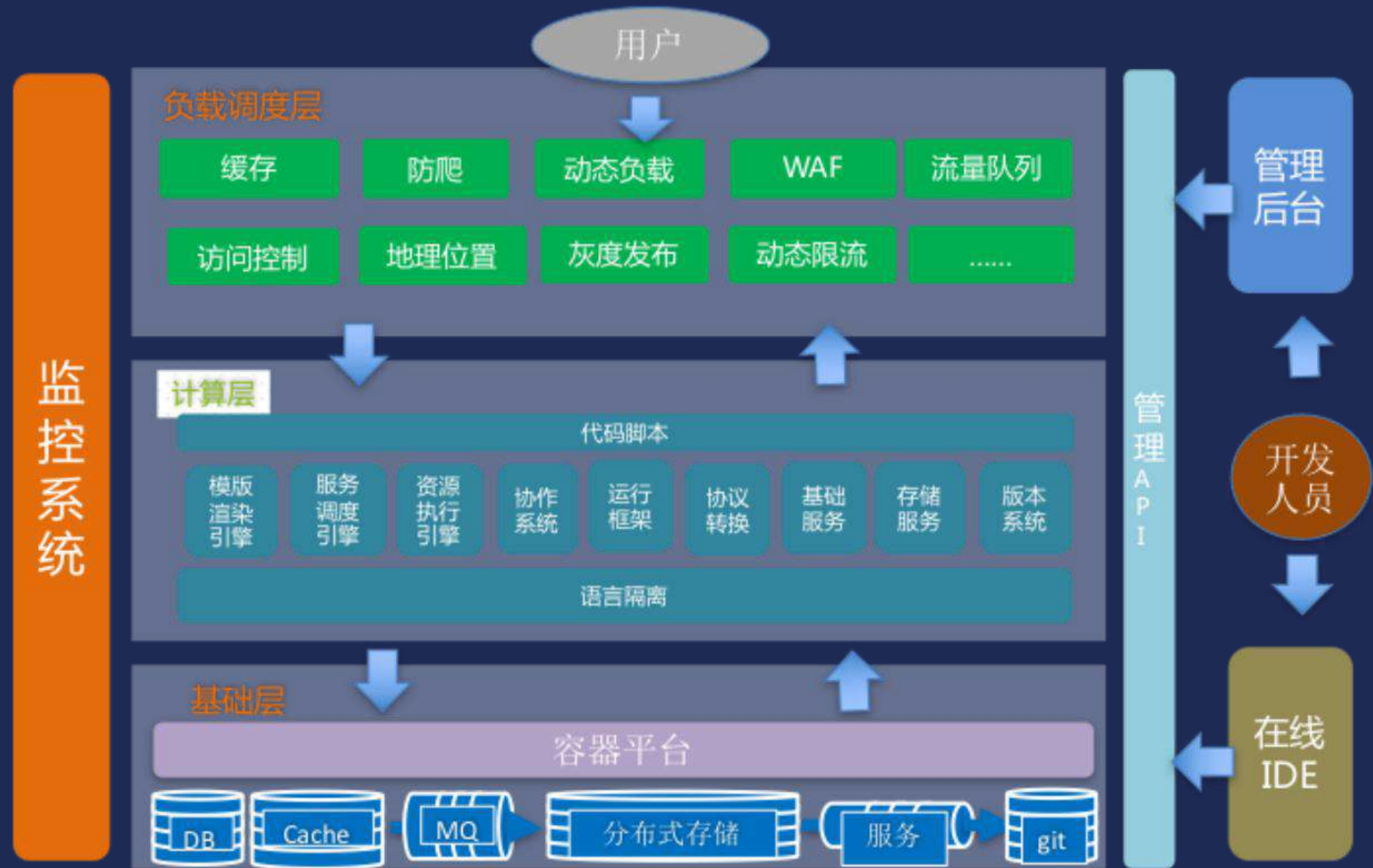
# Serverless架构解决同程的问题

解决我们最新的痛点：  
业务快速变化中如何更快地开发。  
其实在这快速地开发中有很多是快不起来的  
（开发环境的问题，上线部署的问题，应用弹性设计问题，可运维性的问题等等）。



# 同程Serverless架构的结构

居于原有基础服务  
利用原有容器平台  
集成能集成的一切





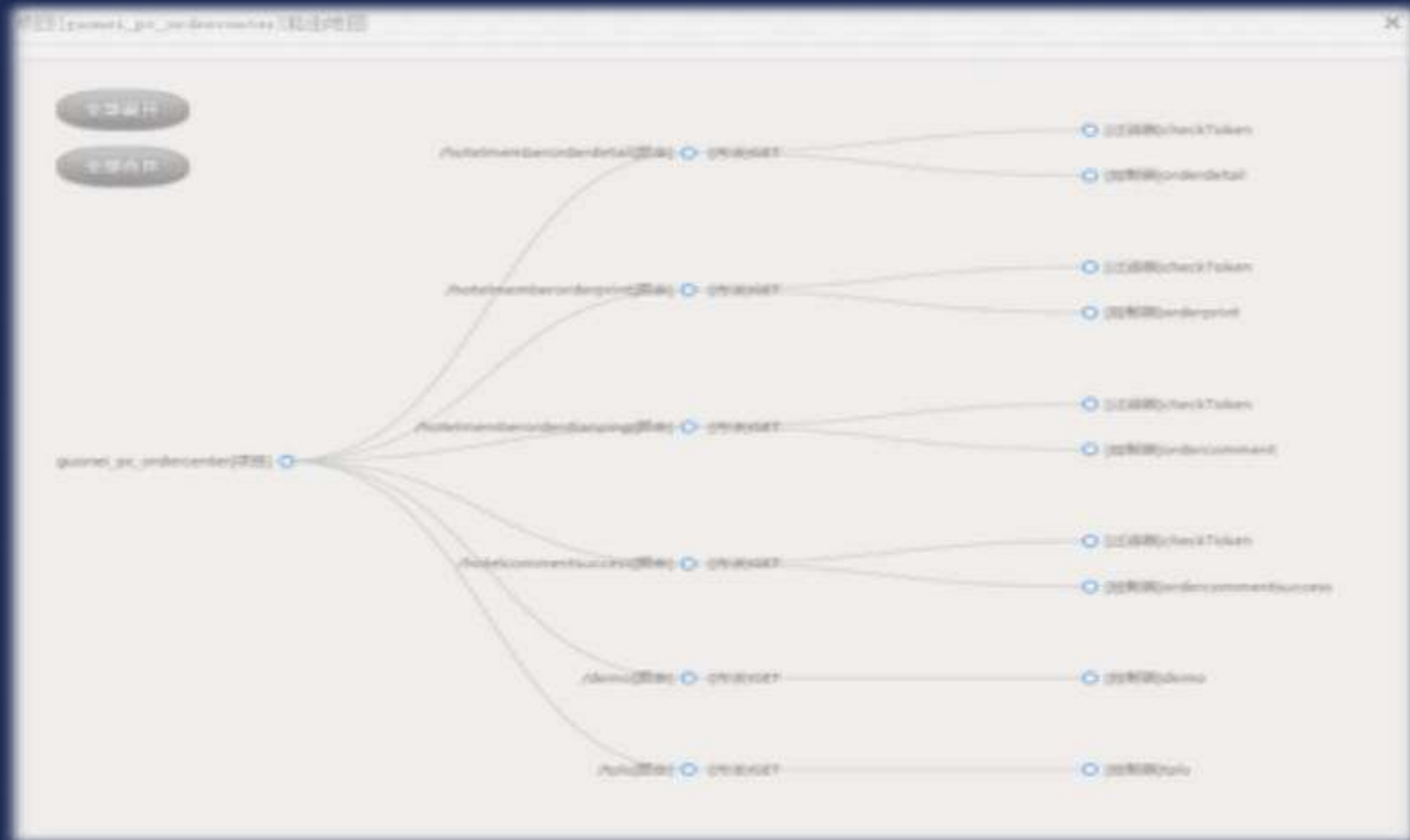
# Serverless架构的隔离，编排与调度

底层容器级隔离

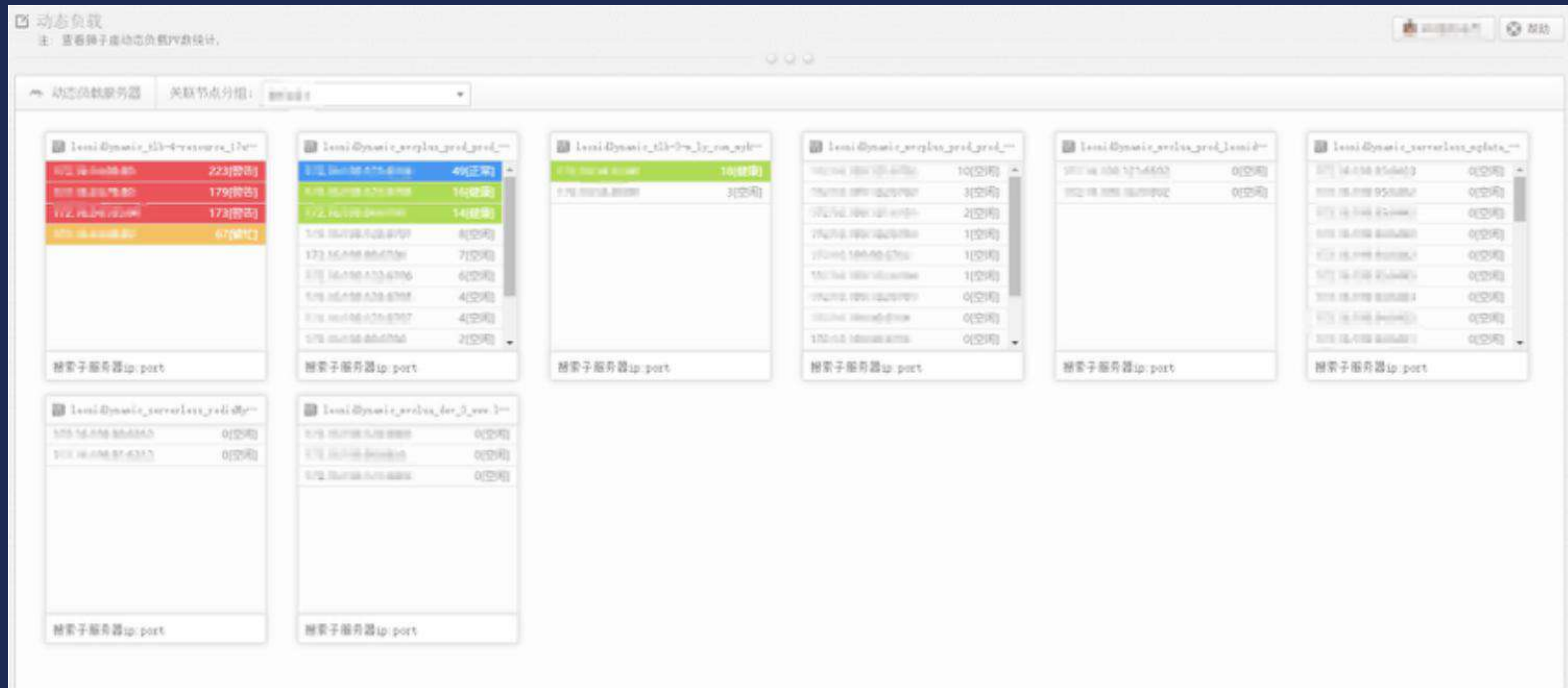
语言级隔离（语言VM） 目前我们只支持Node.Js和LUA

通过Gateway访问到应用

根据流量自动拉起应用和自动扩容部署实例



# Serverless架构的资源利用率



最小单元部署4台物理服务器最多可支持1万个应用



# Serverless平台化建设:动态负载均衡器



## 多种负载均衡模式

分组模式：分组负载      轮询模式：轮询集群中所有机器  
弹性模式：根据访问情况自动增减负载

根据访问量，自动扩容和缩容集群

将应用的分级为6级，根据请求量和后端应用的响应时间，  
动态调整负载机器数量



## 热加载

免去每次对负载均衡的Reload操作，减少系统吞吐量的影响

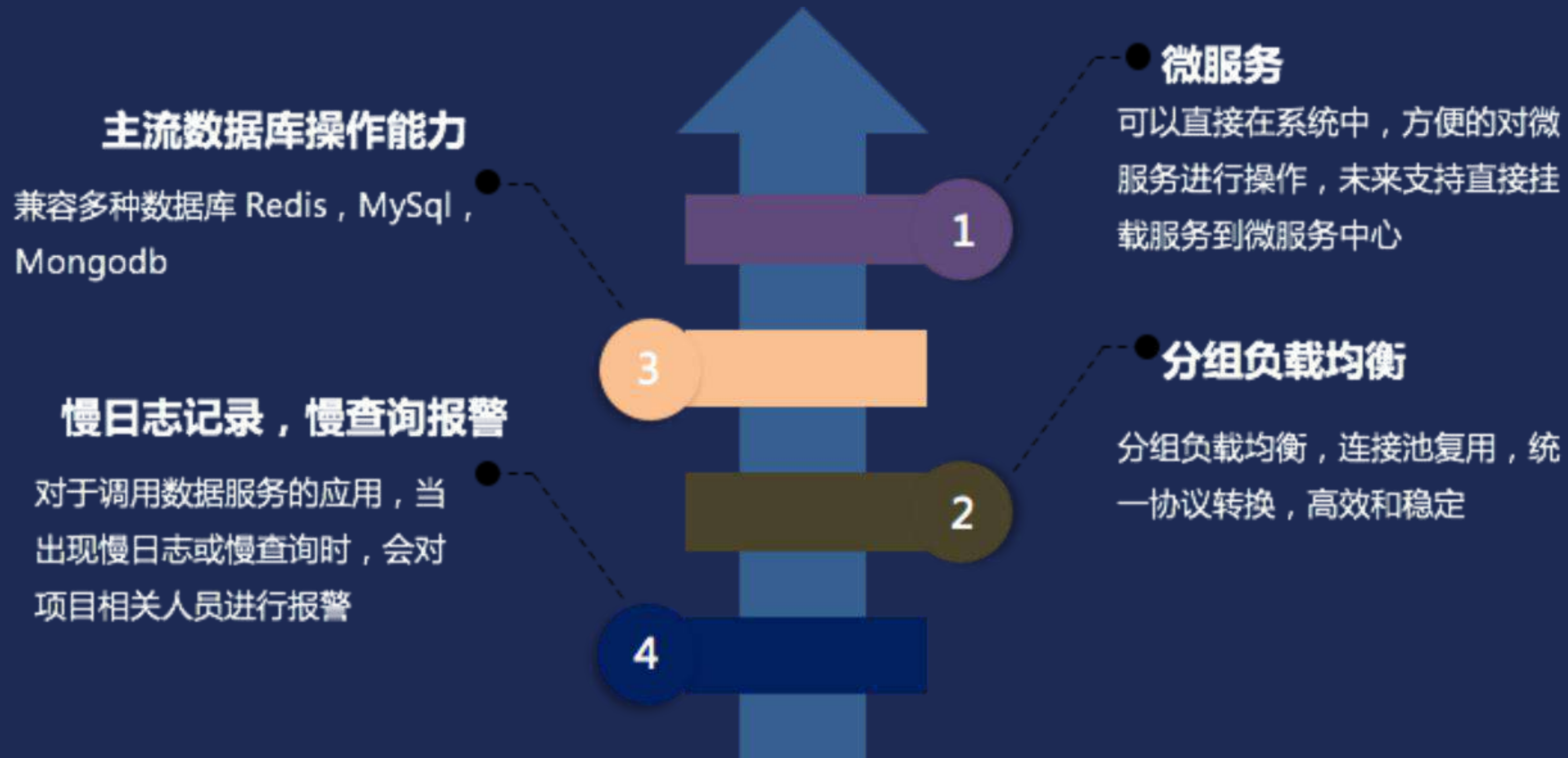
适用任何应用

动态负载均衡器可以适用任何应用，包括：JAVA, IIS, Go 等等



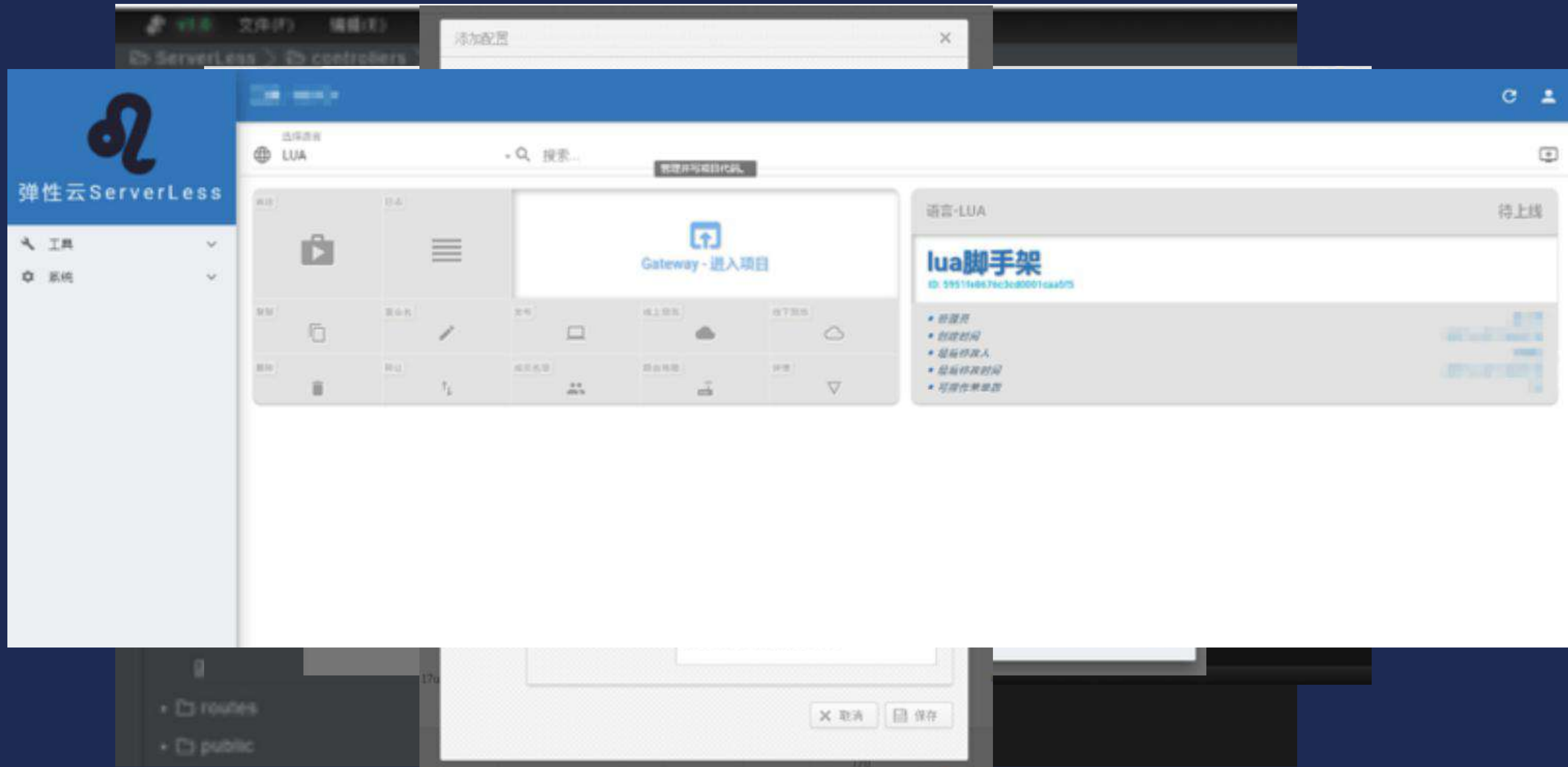
★ Serverless服务拥有毫秒级别的弹性扩容和缩容的能力

# Serverless平台化建设:数据服务系统



★ **Serverless**服务拥有访问常规数据源的能力

# Serverless平台化建设:统一IDE的Web化



- ★ 将所有的功能可视化,可配置化, 不再需要本地环境, 有多少环境不再被关注

# Serverless在同程的情况：开发、发布和运维的效率

90%

**初始化项目脚手架**

申请git

安装框架

安装模块

调试脚手架

系统框架自带

80%

**数据源、路由和日志**

定义编写路由

数据源连接

拦截器、控制器

日志记录

MVC+系统进行配置

40%

**编码开发**

工具函数，常用类库

逻辑编写

程序调试

代码版本控制

云端编辑器编写代码

90%

**发布和运维**

申请应用上线

配置运行环境

申请访问路径

应用运行情况和日志

应用系统一键发布

我们把什么放在Serverless平台跑了

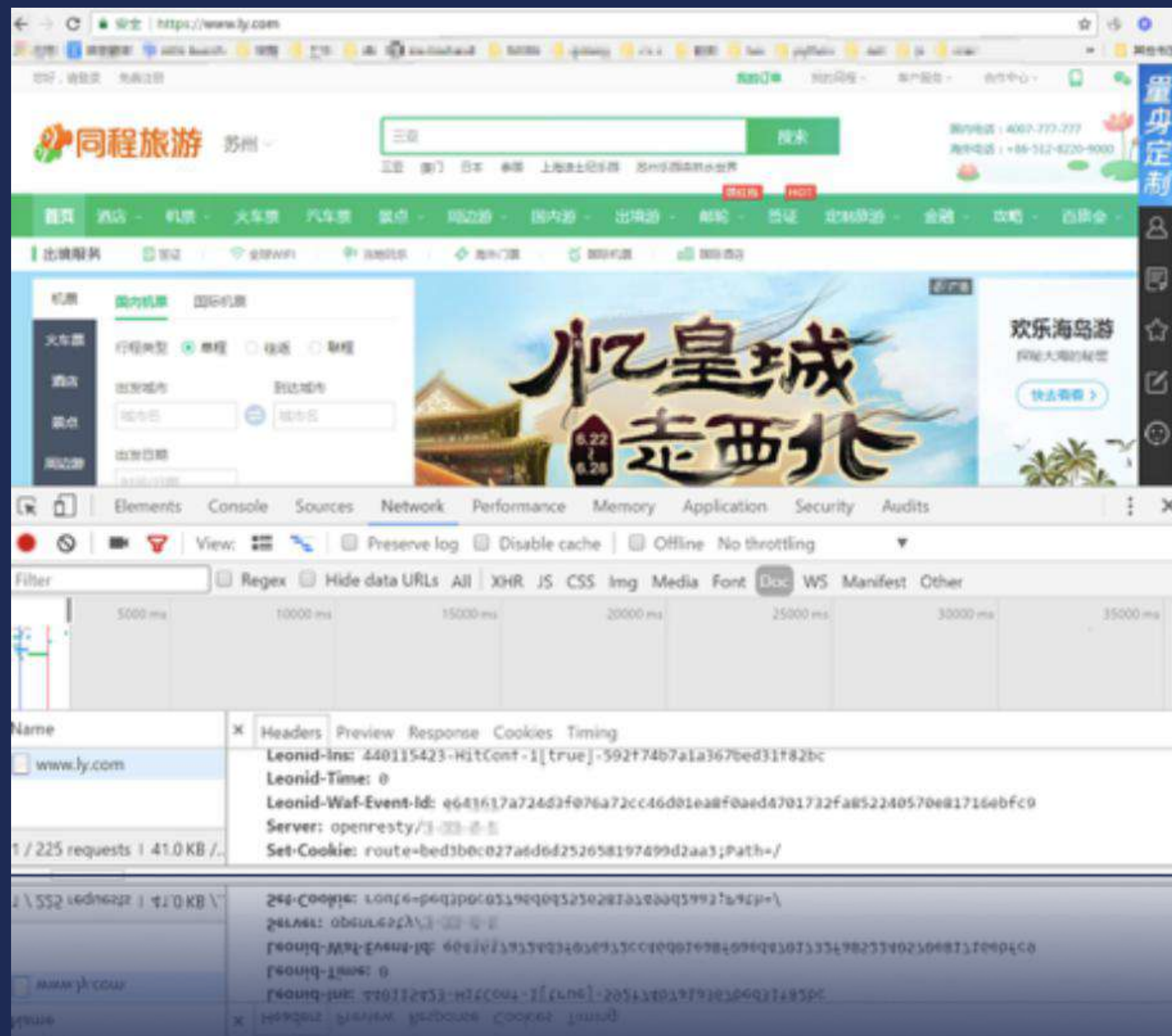


# Serverless在同程的情况：Web应用

所有的网页

所有的活动推广

部分变化量大的后台



# Serverless在同程的情况： 轻型服务

一些逻辑简单的业务服务

一些逻辑变化快的业务服务

大量的临时的小服务

```
ipDefender.lua  f1.lua  main.lua x  index.lua  utils.lua
29
30         console.error("设置黑名单失败 ip:" .. v.ip .. "err:" .. err)
31     end
32 end
33 end
34 end
35 else
36     console.error("未获取到黑名单列表")
37 end
38 console.error("设置黑名单列表完成")
39
40 console.error("设置配置列表完成")
41 }], 10)
42 if err then
43     console.error("定时器启动失败" .. err)
44 end
45
46 local requestText = req.raw_body
47 if requestText and requestText ~= "" then
48     local json, err = global.json_parse(requestText)
49     if err then
50         console.error("代理转发请求失败 json反序列化失败:" .. err)
51         utils.response(res, utils.RspType.Exception, utils.RspCode.RspCode_7000)
52     else
53         local resp, body, err = req.proxy()
54         if resp and resp.status >= 400 then
55             console.error("代理转发请求失败 response status:" .. resp.status)
56             utils.response(res, utils.RspType.Exception, utils.RspCode.RspCode_7000)
57         elseif err then
58             console.error("代理转发请求失败 error:" .. err)
59             utils.response(res, utils.RspType.Exception, utils.RspCode.RspCode_7000)
60         else
61             res.send(body)
62         end
63     end
64 else
65     console.error("代理转发请求失败 未传入请求体")
66     utils.response(res, utils.RspType.Exception, utils.RspCode.RspCode_7000)
67 end
68
69 end
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
```

# Serverless在同程的情况：配套功能集成

The screenshot displays the Trip.com hotel booking interface. At the top, there are navigation tabs for '房型' (Room Type), '酒店信息' (Hotel Information), '交通' (Transportation), and '点评(39)' (Reviews (39)). Below this, the search criteria are shown: '入住 2017-06-28' (Check-in 2017-06-28), '离店 2017-06-29' (Check-out 2017-06-29), and '共1晚' (Total 1 night). There are also buttons for '去筛选' (Go to filter), '房型' (Room type), '立即确认' (Confirm immediately), '取消预订' (Cancel booking), '预订' (Book), and '支付方式' (Payment method).

The main content area features a highlighted '特价单人房(无窗)' (Special Single Room (No Window)) for ¥222. Below this is a table of room options:

名称	房型	床型	房价	取消政策	价格	预订
(同程推荐)	大床	双床	无窗	不可取消	¥222	预订
(双人入住)	大床	双床	无窗	不可取消	¥229	预订
(优惠价-双床)	单人床	双床	无窗	不可取消	¥240	预订
特价单人房(无窗)	大床	无窗	无窗	不可取消	¥273	预订
特价单人房(无窗)	大床	无窗	无窗	不可取消	¥288	预订

At the bottom of the table, there are two small images of the room. To the right of the main content is a map showing the hotel's location and a list of nearby hotels with their prices:

- 北京慕尚酒店: 距离酒店17米, ¥199
- 新HOME连锁酒店(北京五棵松店): 距离酒店124米, ¥160
- 北京茂林国际大酒店: 距离酒店454米, ¥376
- 海友酒店(北京金源酒店): 距离酒店543米, ¥139
- 北京长荣酒店: 距离酒店611米, ¥209
- 北京快捷酒店(北京金源林五棵松地铁站店): 距离酒店711米, ¥226

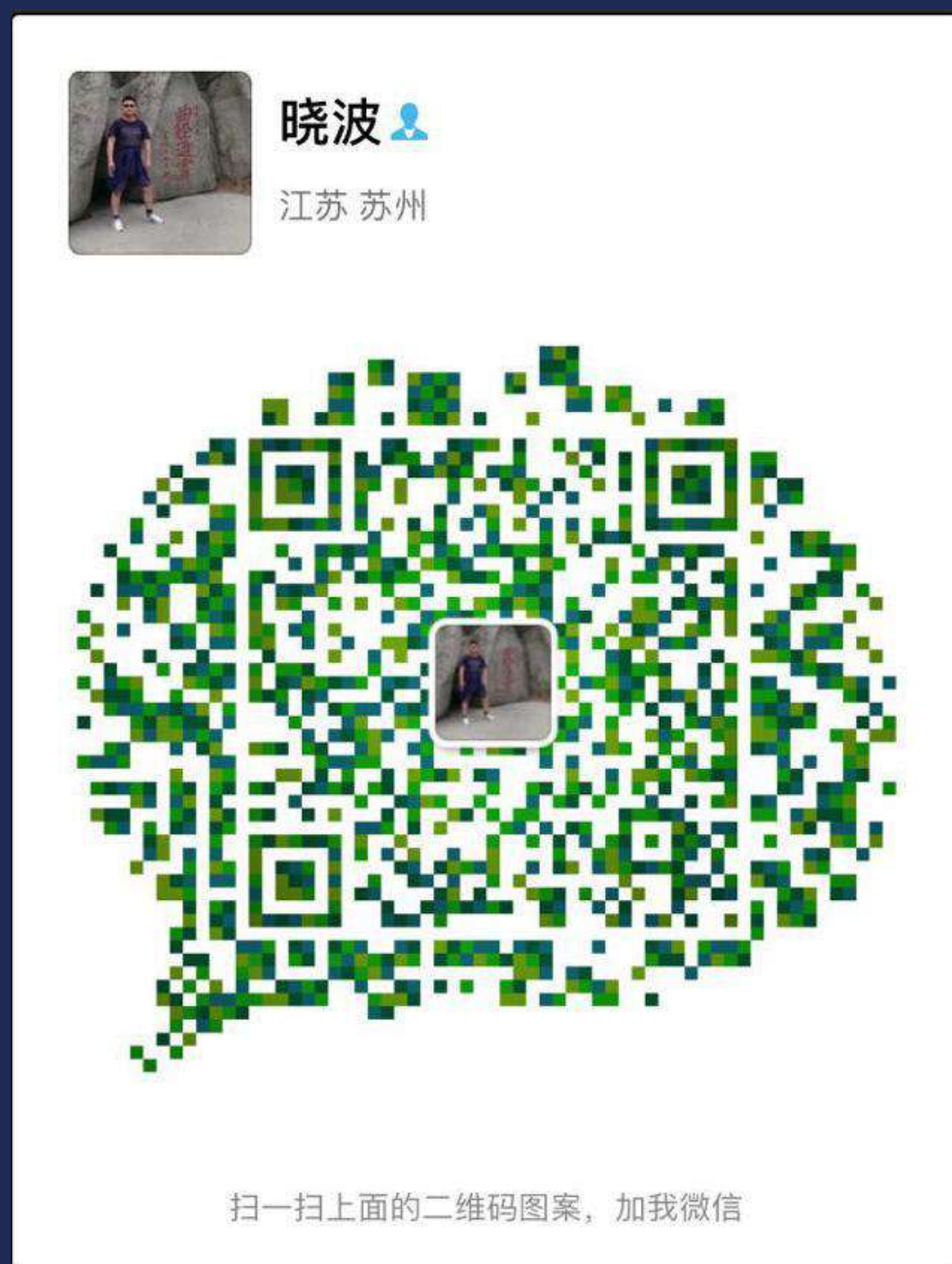
价格实时计算服务

# • 我们下一步在 **Serverless** 上要做什么

- ★ 加入更多的语言
- ★ Web化IDE能力提升
- ★ 更多的技能被配置化
- ★ 将自动测试系统并入



# SPEAKER



## 王晓波

同程艺龙 机票事业群 CTO



# Thanks!

联合主办： 腾讯云 |  **Geekbang**  
极客邦科技

协作伙伴： 腾讯TES |  腾讯课堂  
KE.QQ.COM 学习成就梦想