



# 大规模企业级 HTTP 框架设计和实践

高文举

August, 2022

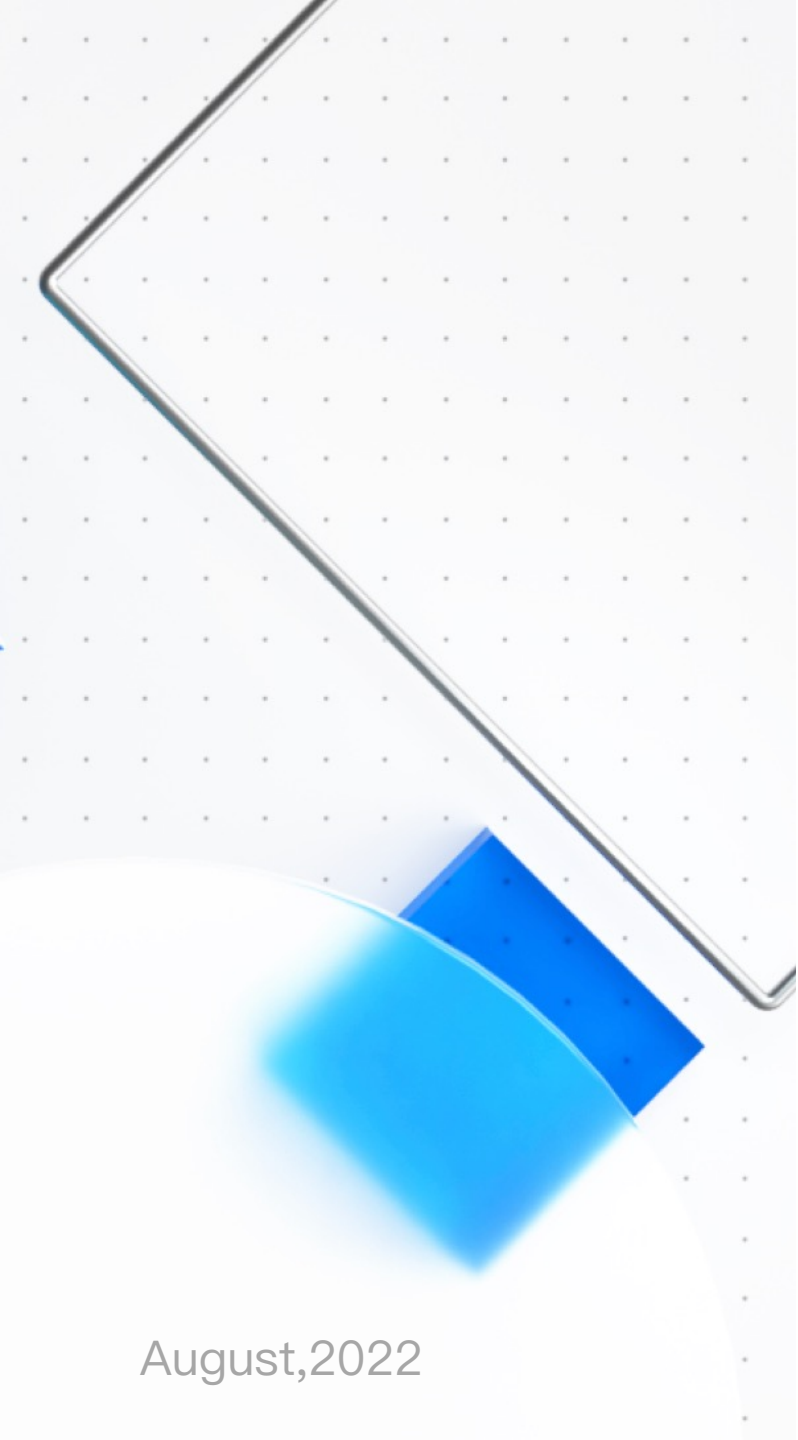
## 01\_ 自我介绍



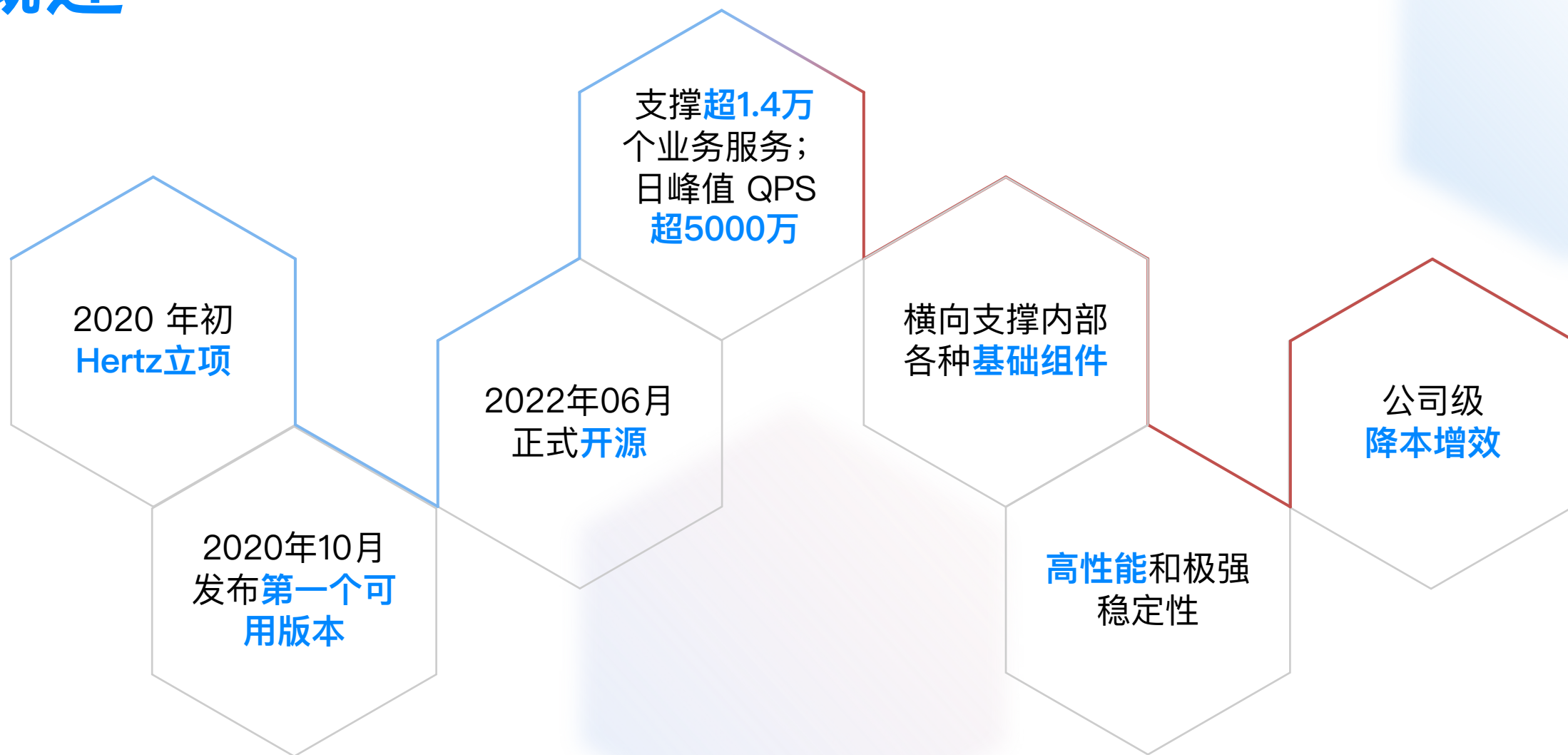
### 高文举

CloudWeGo – Hertz 项目负责人，字节跳动基础架构服务框架资深研发工程师

17年开始接触微服务，云原生等话题，曾就职于百度，负责金融级网关、百度云服务网格等研发工作。19年加入字节跳动，负责 Hertz 框架研发，推动公司级 Golang HTTP 框架 Ginex 向 Hertz 成功转型。

- 
- 0 字节跳动内部 Go HTTP 框架的变
  - 1 迁
  - 02 企业级 HTTP 框架的设计考量和落地思路
  - 03 Hertz 的核心特
  - 04 点
  - 04 未来规划和挑战
  - 05 总结

# 概述



August, 2022

## 01\_字节跳动内部 Go HTTP 框架的变迁

Ginex:  
基于Gin的  
一个封装

问题显现

各种魔改  
框架萌芽

# 01\_字节跳动内部 Go HTTP 框架的变迁-基于Gin封装



## 开荒的时代

公司大规模面向 Golang 语言  
转型

## 框架伴随业务野蛮生长

补齐基础设施后便直接交付业  
务，迭代迅速

## “大力出奇迹”

先解决需求，其他的往后放

## 业务共同参与维护

框架和业务侧共同维护一个代  
码仓库

## 01\_字节跳动内部 Go HTTP 框架的变迁

Ginex:  
基于Gin的  
一个封装

问题显现

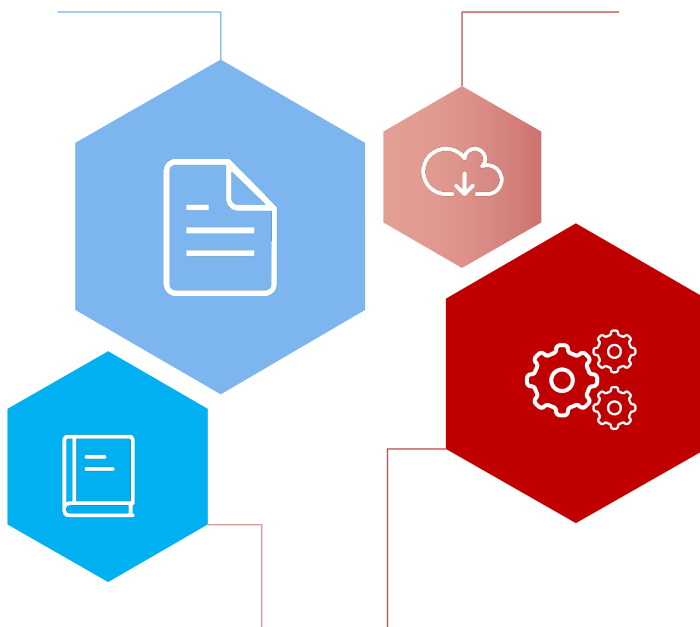
各种魔改  
框架萌芽

# 01\_字节跳动内部 Go HTTP 框架的变迁-问题显现



迭代受开源项目限制

无法满足性能敏感业务需求



代码混乱膨胀  
维护困难

无法满足不同场景的功能需求



## 01\_字节跳动内部 Go HTTP 框架的变迁

Ginex:  
基于Gin的  
一个封装

问题显现

各种魔改  
框架萌芽

# 01\_字节跳动内部 Go HTTP 框架的变迁-魔改开源框架



August, 2022

# 01\_字节跳动内部 Go HTTP 框架的变迁-小结

01

**早期基于开源框架封装**

开荒、大力出奇迹

02

**随着时间发展，问题逐渐出现**

框架混乱膨胀、新需求无法得到很好的满足

03

**为了解决问题出现基于另外的开源框架魔改的萌芽**

继续老路、进入怪圈

## 02\_企业级 HTTP 框架设计的考量和落地思路



跳出怪圈



核心痛点梳理



建立框架  
科学发展观

## 02\_企业级 HTTP 框架设计的考量和落地思路-跳出怪圈



### 自主研发

- 框架代码全链路自主可控
- 避免引入任何三方不可控因素
- 建设完备的用户使用手册贯穿开发、测试、发布流程
- 公司级代码生成工具聚焦核心逻辑



### 质量控制

- 单测
- 集成测试
- 压力测试
- 性能测试
- 模糊测试



### 严格准入

- 完善的 Design-Review-Develop-Test-Review 迭代流程
- 统一的需求管理
- 严格的发版准入规范

## 02\_企业级 HTTP 框架设计的考量和落地思路



跳出怪圈

核心痛点梳理

建立框架  
科学发展观

## 02\_企业级 HTTP 框架设计的考量和落地思路-痛点梳理



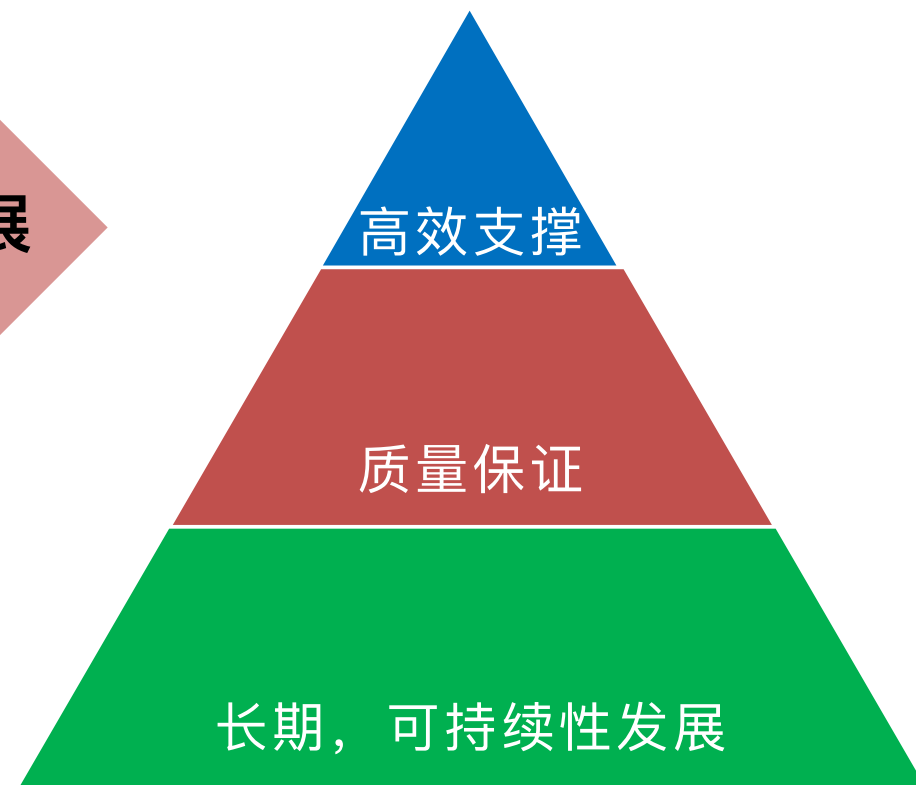
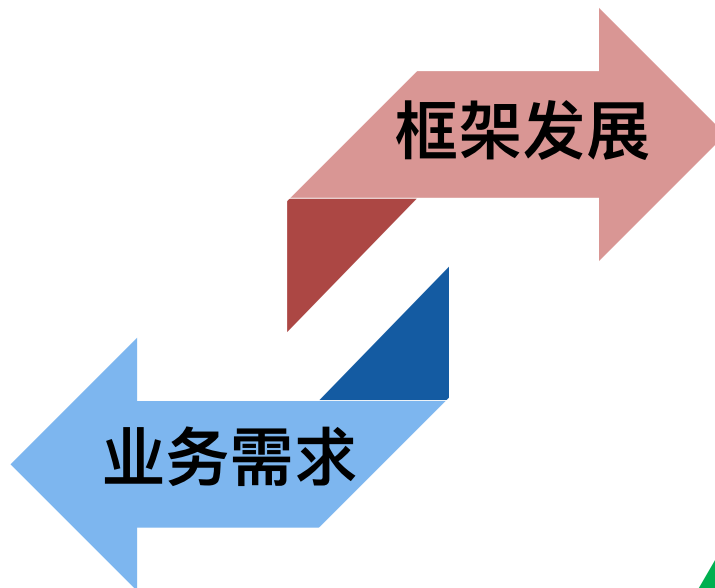
功能需求



性能需求



易用稳定



## 02\_企业级 HTTP 框架设计的考量和落地思路



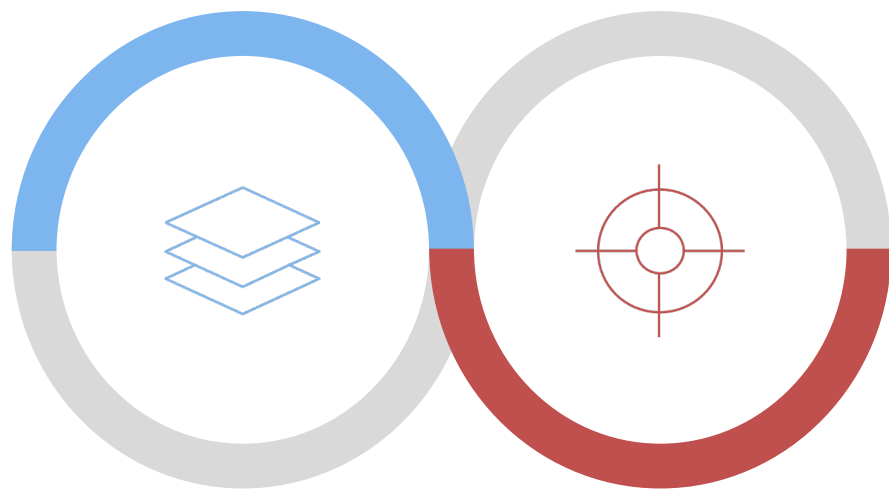
跳出怪圈

核心痛点梳理

建立框架  
科学发展观



## 02\_企业级 HTTP 框架设计的考量和落地思路-框架科学发展观

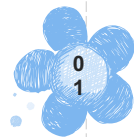


### 多样的需求

支撑各个业务  
线及基础设施  
(横向扩展性)

### 灵活的结构

贯穿HTTP生命  
周期的掌控力  
(纵向模块化)



### 聚类需求：

面向通用能力展开设计



### 跳出局部：

在更大范围寻求最优解

## 02\_企业级 HTTP 框架设计的考量和落地思路

01

**跳出怪圈**

自研、质量控制、严格准入

02

**痛点梳理**

业务需求多样、框架可持续发展

03

**框架科学发展观**

需求聚类、跳出局部

## 03\_Hertz 的核心特点

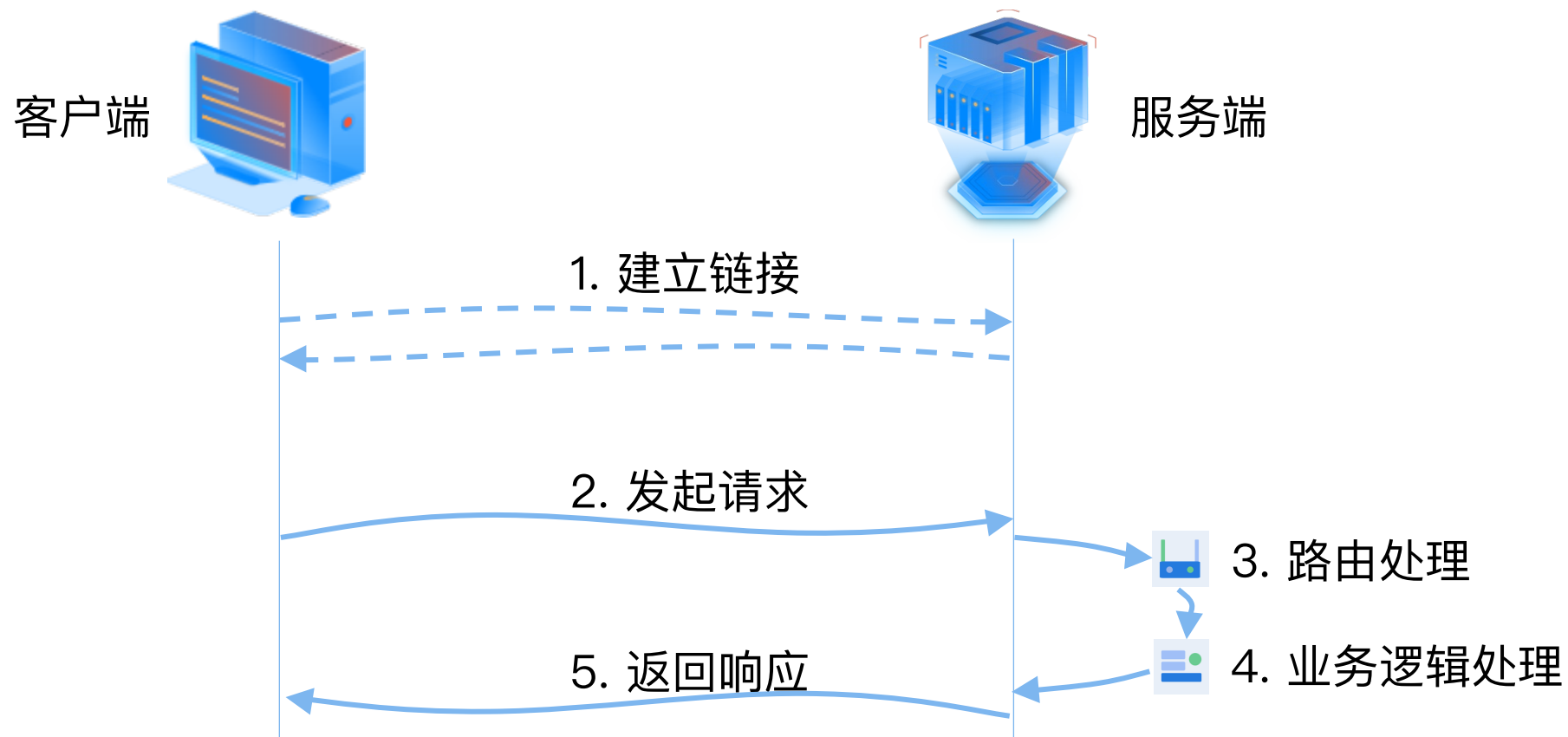


分层抽象

易用  
可扩展

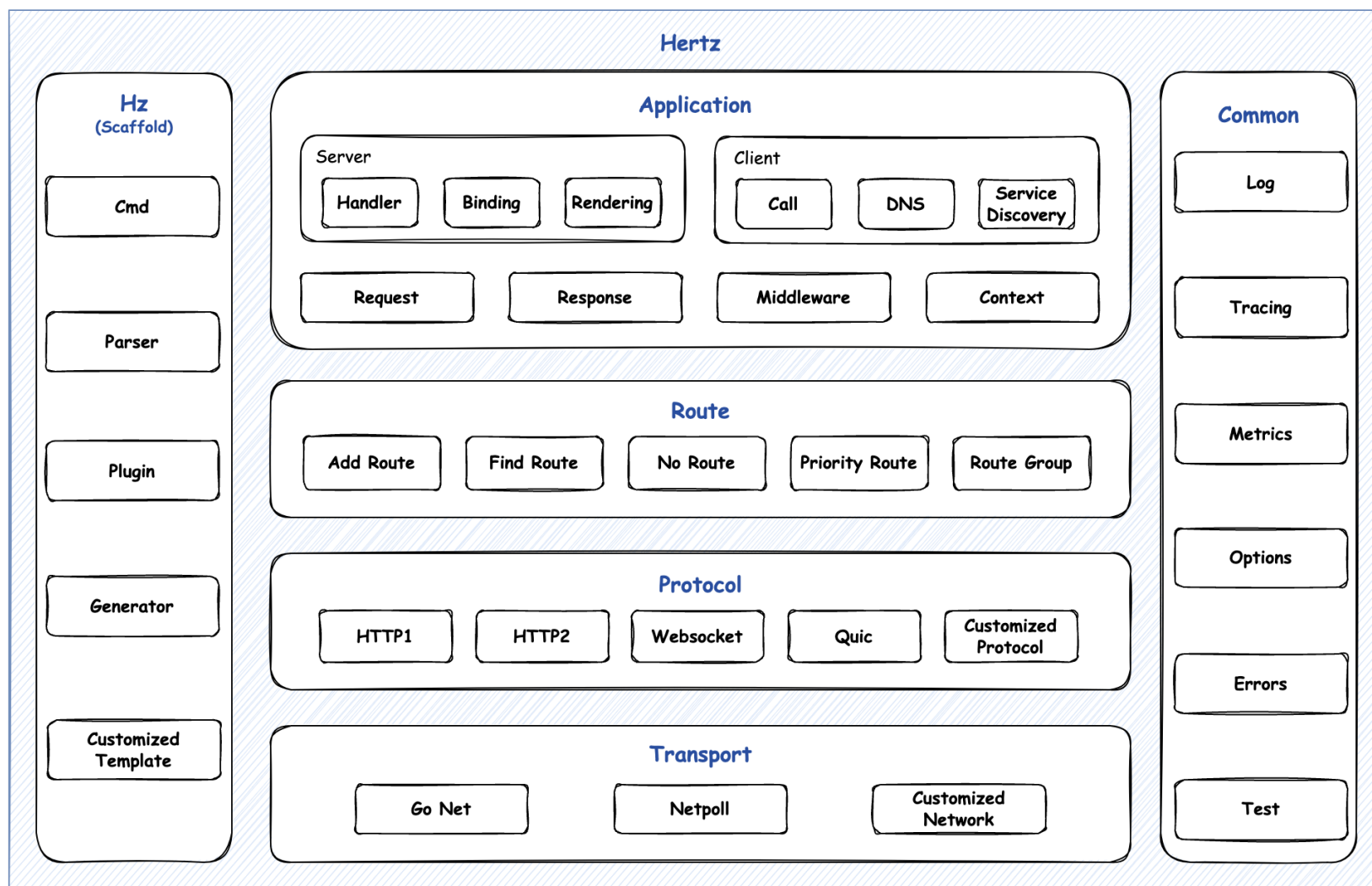
自主可控的  
高性能探索

## 03\_Hertz 的核心特点-分层抽象



一个请求从建立连接到完成请求的全过程

## 03\_Hertz 的核心特点-分层抽象



**传输层：**抽象网络接口  
**协议层：**解析请求，渲染响应  
**路由层：**基于URL进行逻辑分发  
**应用层：**业务直接交互层

**通用层：**提供通用能力和接口

**Hz：**脚手架工具，基于IDL文件快速生成项目骨架

## 03\_Hertz 的核心特点

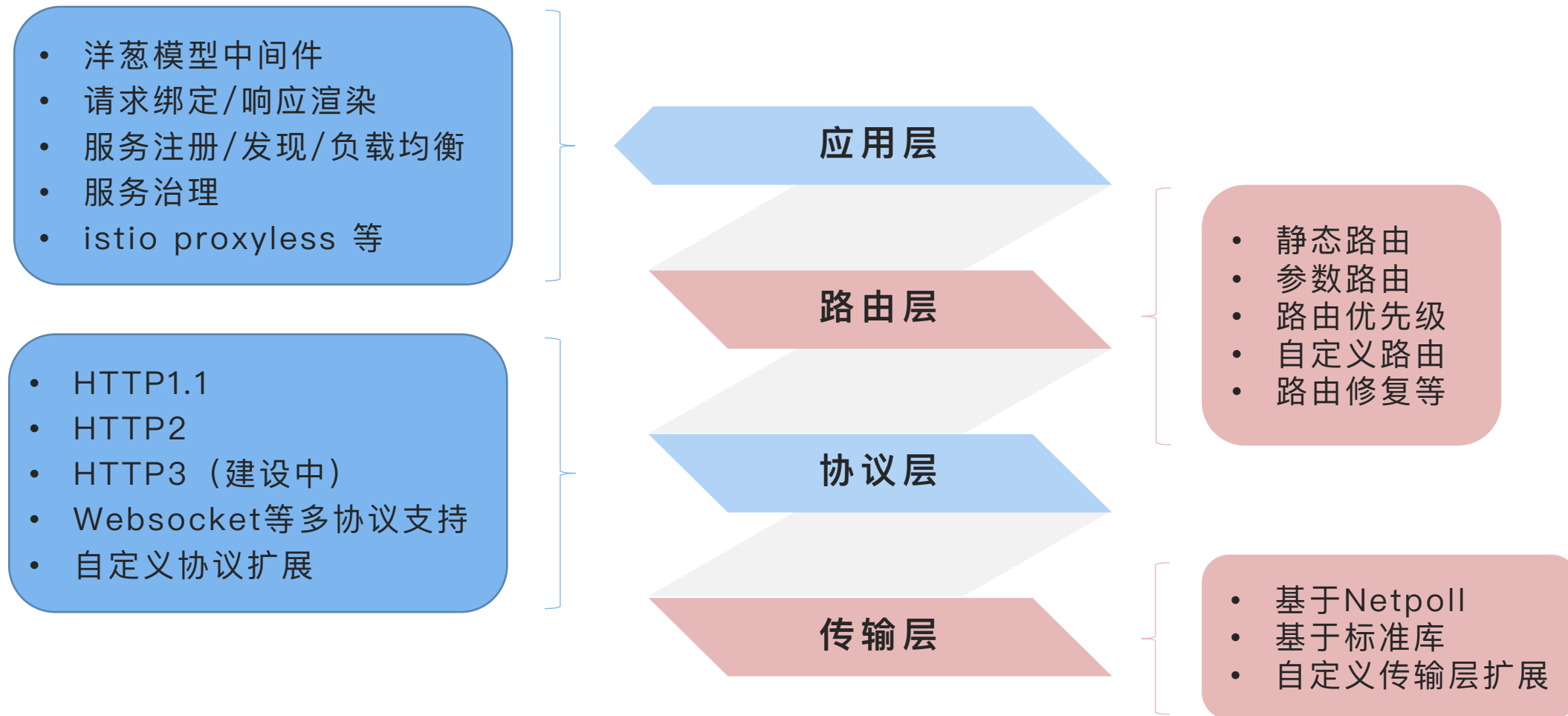


分层抽象

易用  
可扩展

自主可控的  
高性能探索

## 03\_Hertz 的核心特点-易用可扩展



## 03\_Hertz 的核心特点



分层抽象

易用  
可扩展

自主可控的  
高性能探索



## 03\_Hertz 的核心特点-性能探索

### 场景描述

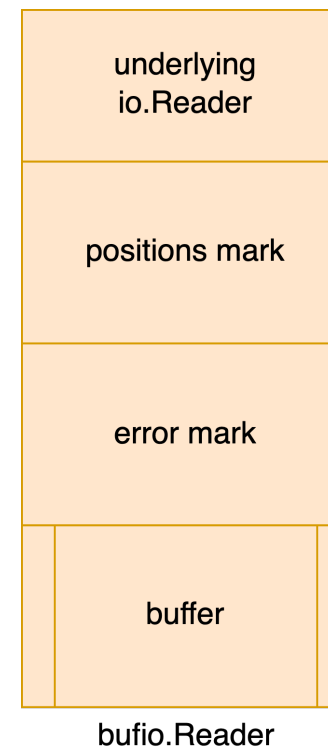
- 减少系统调用次数，提升整体效率，涉及io操作，通常引入带buffer的io数据结构
- HTTP/1.1协议中的Header为不定长数据段，往往需要解析到最后一行，才能够确定是否完成解析

### bufio.Reader的问题

buffer长度固定，超出buffer长度后，.Peek()方法直接报错（ErrBufferFull），无法完成既定语义功能

### 真实使用环境复杂多变

不同的业务，不同的场景，数据规模各异，如何通用且高效的解决这个问题成为一大挑战



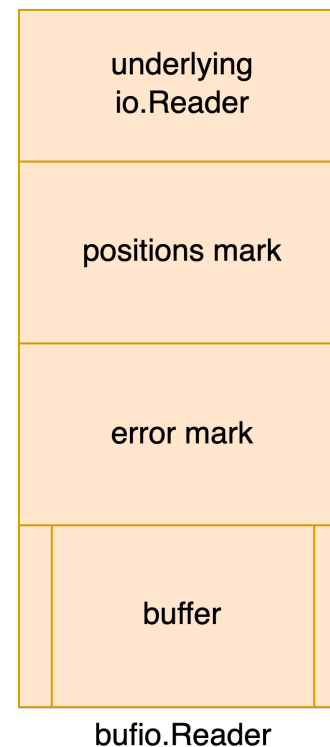
## 03\_Hertz 的核心特点-性能探索

### 一些可能的解

1. 索性直接利用bufio.Reader的局限当做feature，通过buffer大小作为header大小的限制  
(功能受限)
2. header解析带状态，暂存中间数据，通过在上层堆叠额外复杂度的方式突破bufio本身的限制  
(性能受限)

### 企业内部场景复杂多变

不同的业务，不同的场景，数据规模各异，如何通用且高效的解决这个问题成为一大挑战



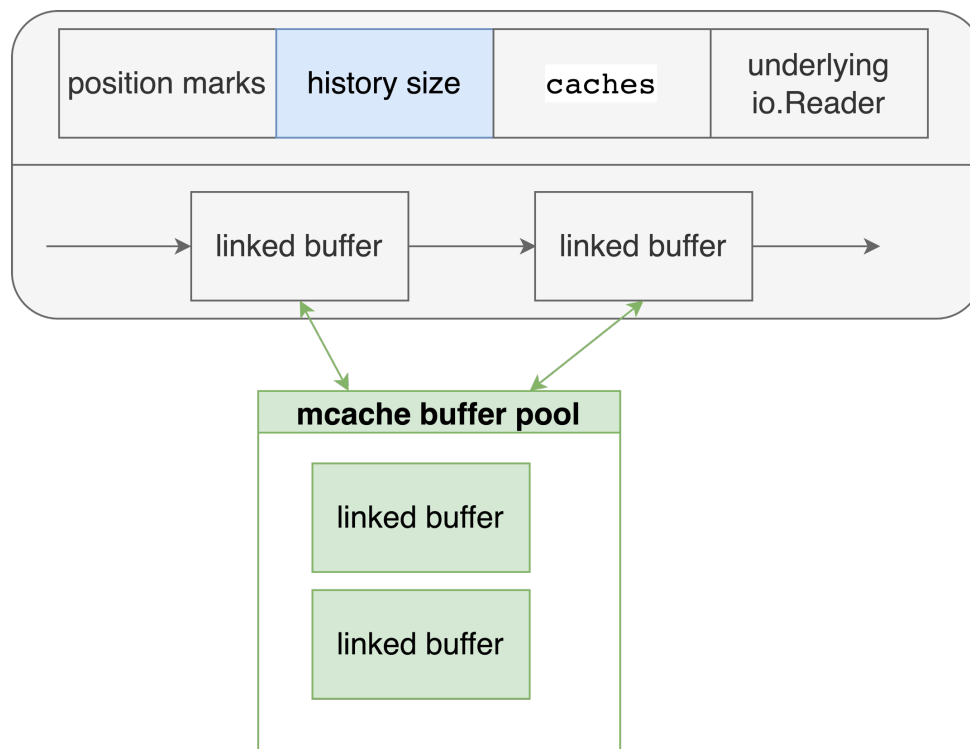
## 03\_Hertz 的核心特点-性能探索

### 自适应linked buffer

- buffer根据真实请求进行动态扩缩容调整
- 传输层LT触发+数据预拷贝
- 协议层最大程度做到零拷贝协议解析

### 核心

- 自适应调整，框架使用者无感
- 核心包大小区间收益明显



## 03\_Hertz 的核心特点-性能探索

针对HTTP/1.1 进行中的优化



### 协议层探索

- Header parser重构
- 传输层预解析

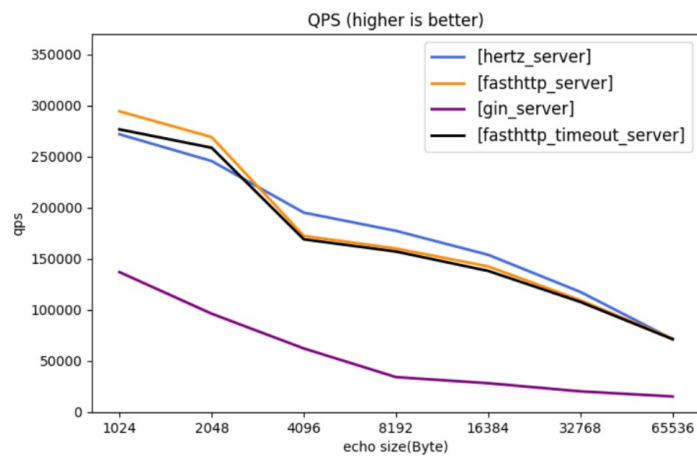


### 传输层探索

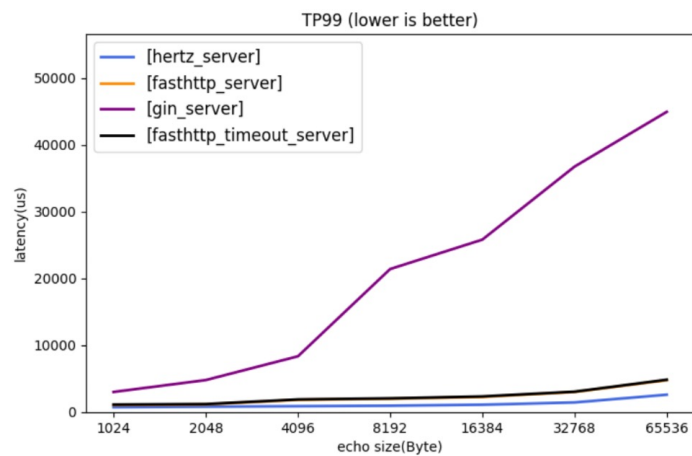
- 使用writev整合发送Header & Body
- 新增接口整合.Peek() + .Skip() 语义

## 03\_Hertz 的核心特点-性能探索

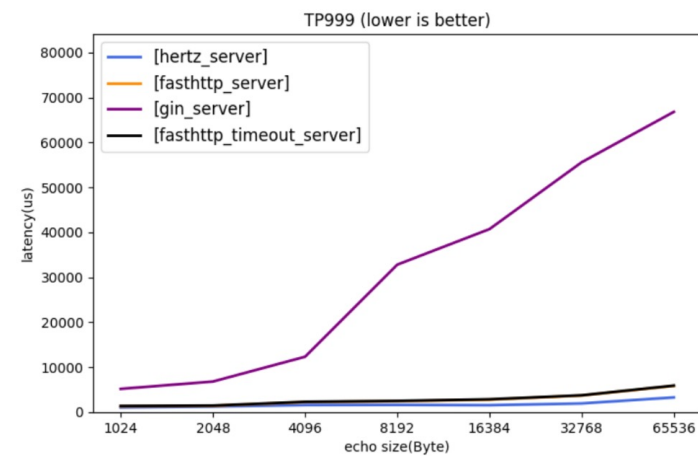
### Hertz benchmark (已开源)



较高的极限QPS



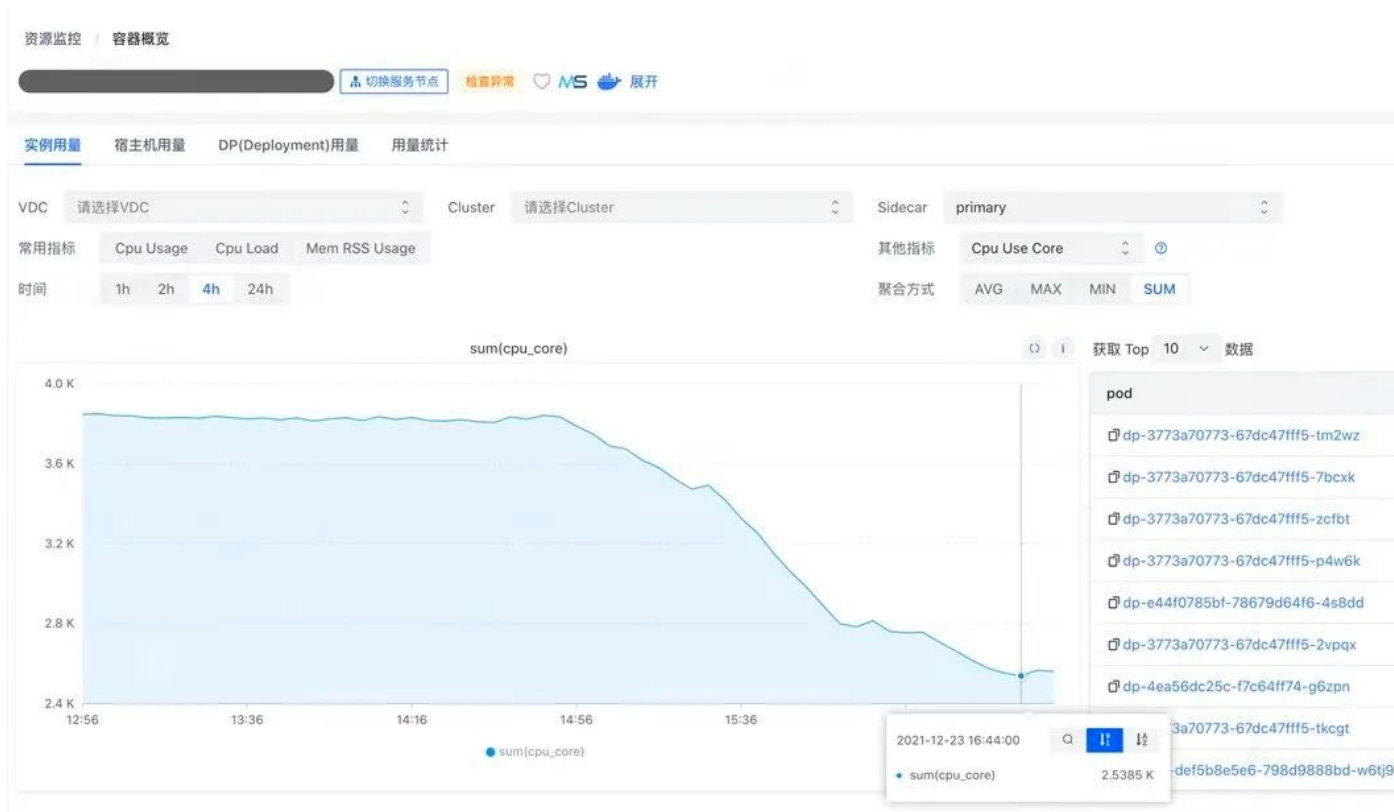
更低的TP99时延



更低的TP999

## 03\_Hertz 的核心特点-性能探索

### 字节跳动服务网格控制面从Gin迁移至Hertz



#### 真实收益:

- CPU: 开销 4k -> 2.5k
- Goroutine: 6w -> 不到100个
- 火焰图: 框架开销占比几乎消失
- 稳定承载线上超 13M QPS 流量

Server 侧不感知但是同样重要的指标:

- 请求 (网络+**框架处理**+业务) 耗时P99 更低, 更稳定

## 03\_Hertz 的核心特点

01

### 分层抽象

解构HTTP框架、分层解耦

02

### 易用可扩展

丰富的API，足够灵活的扩展能力

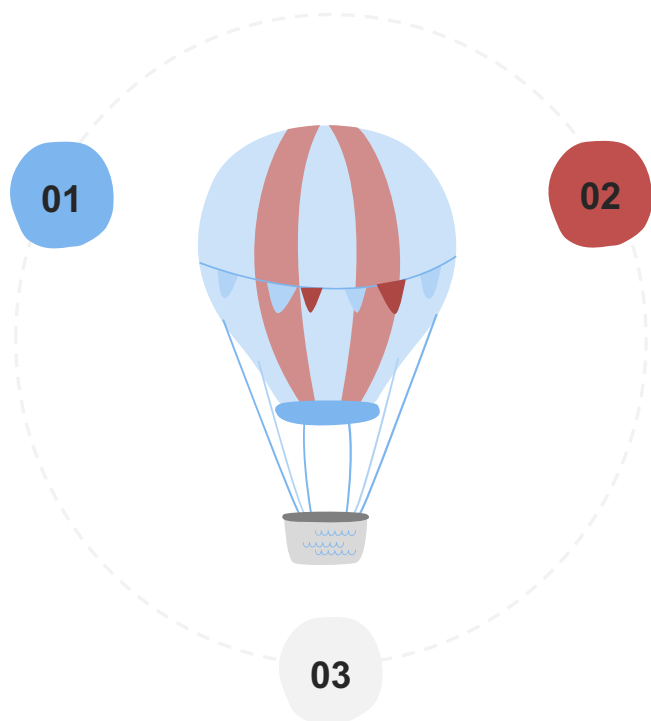
03

### 自主可控的高性能探索

自适应buffer，零拷贝解析，更多探索持续中

## 04\_Hertz 未来规划和挑战

打造泛HTTP框架



助力CloudWeGo

服务更多用户



# 总结

1

字节跳动内部 Go HTTP 框架的变迁：从基于开源封装，到开启自研之路

2

企业级 HTTP 框架的设计考量和落地思路：破圈、需求提炼、框架科学发展观

3

Hertz 核心特点：分层抽象、易用可扩展、自主可控的性能探索

4

Hertz 未来的规划和挑战：框架持续打磨、助力CloudWeGo、服务更多用户



# THANKS

欢迎关注 CloudWeGo



August, 2022