架构设计文档（ADR）

文档概述

文档目的

明确红人广告竞价咨询平台的技术栈选型，指导开发、测试、运维全流程落地；统一技术决策标准，确保架构适配产品核心诉求（AI 报价生成、多平台数据采集、实时数据监控、红人与品牌方双端交互）。

产品核心诉求

- 核心能力：AI 智能报价、红人与品牌方双向匹配、广告视频全链路数据监控；
- 性能要求：高并发多平台 API 调用（数据采集）、毫秒级实时监控数据查询、轻量化部署与弹性扩容；
- 扩展要求：模块解耦，支持 AI 报价算法、数据监控维度的独立迭代；
- 合规要求：合法采集 UCG 平台数据，保障数据传输与存储安全。

选型原则

- 解耦易扩展：核心模块（报价、匹配、监控）独立部署 / 迭代，适配后期功能扩展；
- 高性能：适配高并发数据采集、实时监控场景，保障接口响应速度；
- 轻量化：降低部署成本，支持容器化快速扩容；
- 适配 AI：兼容 Python AI 算法模块的高效调用；
- 合规性：优先对接官方 API，规避数据采集风险。

整体架构选型

架构模式

采用**微服务架构**，核心框架选用「Hertz（字节跳动开源）+ Nacos」（替代 Spring Cloud Alibaba）。

架构对比

备选方案	核心特点	最终选择理由
单体架构	开发成本低、上手快；但模块耦合度高，高并发场景下扩容困难，无法独立迭代 AI 报价 / 监控模块	不选择：产品容，单体架构
Spring Cloud（Java）	生态成熟、组件丰富；但 JVM 占用内存高，部署包体积大，协程处理高并发 API 采集效率低	不选择：对比场景下无优势
Go-Micro/ Hertz（Go）	基于 Go 协程模型，高并发处理能力强；编译后为单二进制文件，部署轻量化；模块解耦易扩展	选择：适配产部署成本低，

架构图

整体微服务拆分如下（附依赖关系）：

代码块

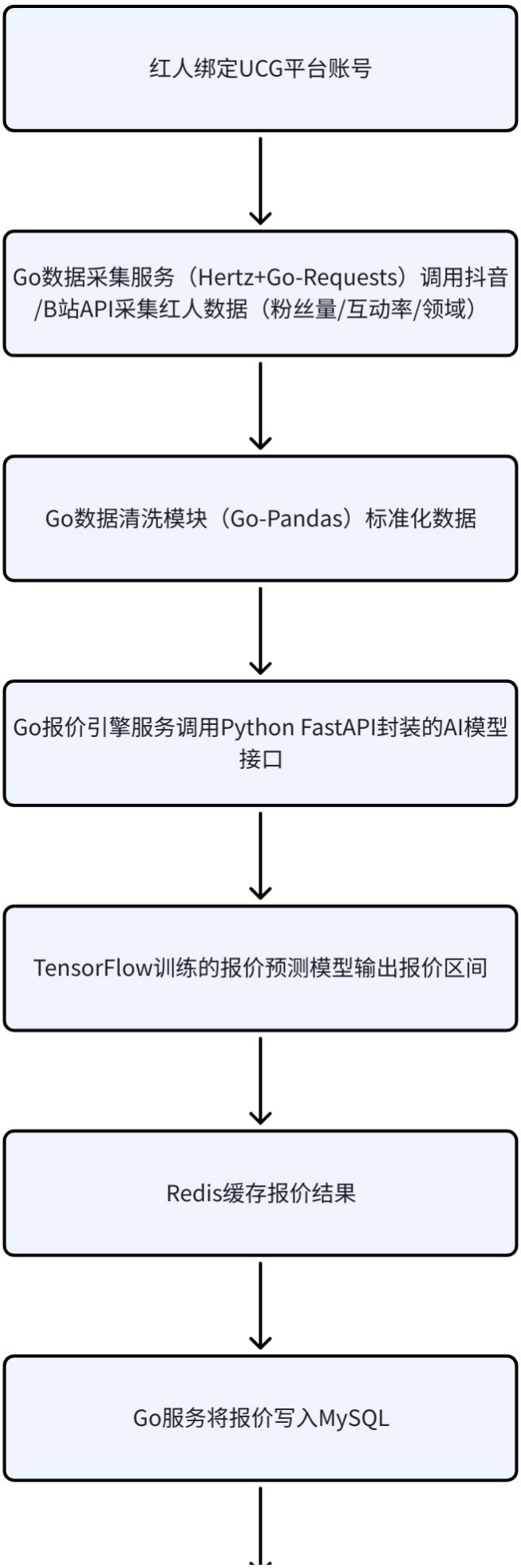
```
微服务集群
├─ 用户中心服务（Go+Hertz）：负责红人与品牌方账号管理、权限校验，为所有模块提供用户数据支撑；
├─ 报价引擎服务（Go+Hertz）：对接AI算法模块，生成红人报价、计算品牌方报价匹配度；
├─ 匹配系统服务（Go+Hertz）：负责红人与品牌方标签匹配、受众重合度分析、咨询消息流转；
├─ 数据采集服务（MVP阶段Python+FastAPI，正式上线Go+Hertz）：对接抖音/B站/快手官方API，采集红人数据、广告视频基础数据；
├─ 数据监控服务（Go+Hertz）：处理实时监控数据（播放量/转化率）、AI内容分析结果存储与查询；
├─ AI算法服务（Python+FastAPI）：训练报价预测模型、识别视频产品功能点，由Go服务调用；
└─ 公共中间件（Nacos/Kafka/Redis/ClickHouse）：支撑服务注册、异步通信、缓存、时序数据存储。
```

各层技术选型

技术分层	选型方案	选型理由（贴合产品场景）
前端层	- PC 端：React + Ant Design Pro（中后台看板适配） - 移动端：UniApp（跨 APP / 小程序） - 可视化：ECharts（数据看板 / 报表）	React 适配复杂数据可视化（ UniApp 一套代码适配双端， ECharts 支持多维度数据图表
后端层	- 核心语言：Go 1.21+ - Web 框架：Gin（RESTful API 开发） - 微服务框架：Hertz（字节开源，服务注册 / 发现 / 配置） - ORM 框架：GORM（MySQL/MongoDB 交互） - 权限框架：Casbin + Gin Middleware	1. Go 协程适配高并发多平台 处理上千个 API 调用）； 2. Gin 路由性能高，支撑实时 3. Hertz 轻量化微服务框架， 积仅 10-20MB； 4. GORM 适配混合数据库架构 结构化数据存储需求； 5. Casbin 轻量适配 RBAC 权
AI 算法层	- 框架：TensorFlow/PyTorch（模型训练） - 工程化：FastAPI（AI 接口封装） - NLP：jieba + 百度 ERNIE（产品功能点识别）	TensorFlow 适配报价预测回 ERNIE 适配视频内容 NLP 分析 FastAPI 轻量化封装 AI 接口，
数据采集层	- 多平台 API 对接：Go-Requests（Go 生态 HTTP 客户端） - 补充采集：Scrapy（Python，合规爬虫） - 数据清洗：Go-Pandas/Go-Spark（轻量清洗）/ Spark（大批量数据）	1. Go-Requests 高性能适配多 更适配后端服务集成； 2. Scrapy 仅作为无官方 API 平 3. 轻量数据用 Go 原生库清洗 Spark。
中间件层	- 注册中心 / 配置中心：Nacos（兼容 Go，与 Hertz 无缝集成） - 缓存：Redis 7.0（集群） - 消息队列：Kafka 3.6（异步数据采集 / 通知） - 搜索引擎：Elasticsearch 8.0（红人 / 品牌方标签检索）	1. Nacos 兼容 Go 生态，支撑 2. Redis 缓存实时监控数据（ 压力； 3. Kafka 适配高并发数据采集 4. Elasticsearch 支撑标签化D 快速检索。
数据库层	- 关系型数据库：MySQL 8.0（主从复制） - 非关系型数据库：MongoDB 6.0 - 时序数据库 / OLAP：ClickHouse 23.0 - 缓存数据库：Redis 7.0（集群）	1. MySQL 存储结构化核心数 从分离提升读性能； 2. MongoDB 存储非结构化数 适配不同平台数据字段差异； 3. ClickHouse 专为时序数据 4. Redis 缓存临时数据（报价 度。
部署运维层	- 容器化：Docker 24.0 + Kubernetes（K8s）1.28 - 监控：Prometheus 2.45 + Grafana 10.0 - 环境：阿里云 ECS / 容器服务（测试 / 生产分离）	1. Docker 封装 Go 编译后的单 性扩容（如大促期品牌方投放 2. Prometheus 监控 Go 服务 监控指标； 3. 测试 / 生产环境分离，保障

核心技术链路

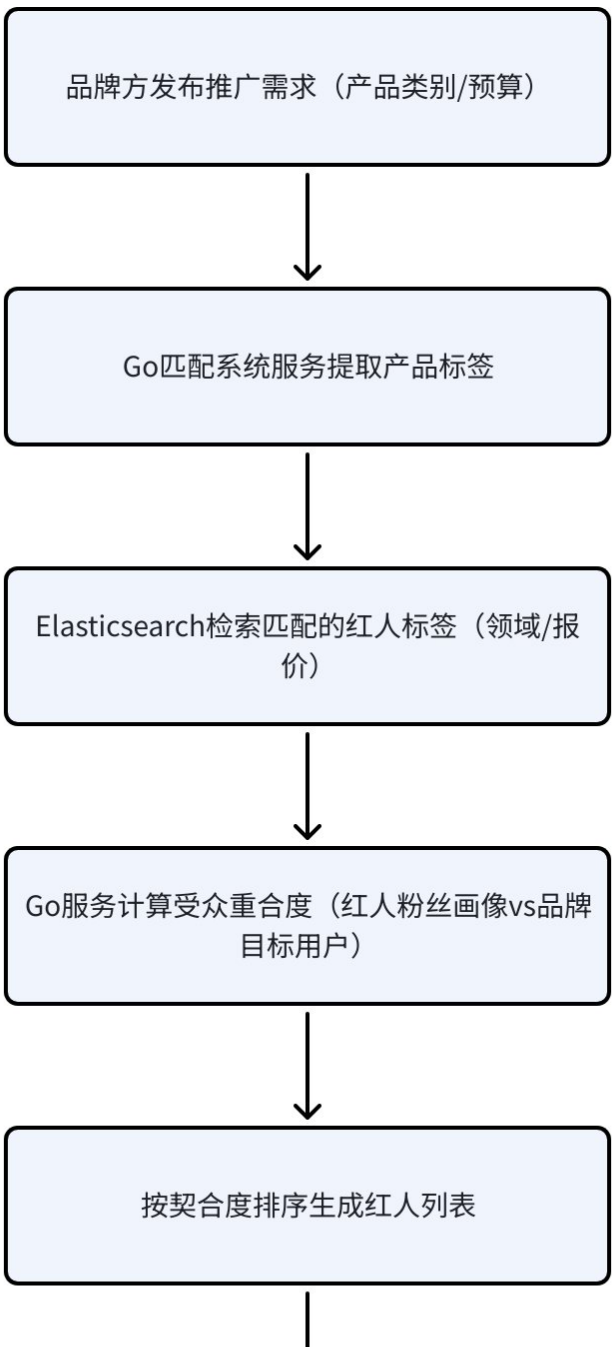
AI 报价生成链路

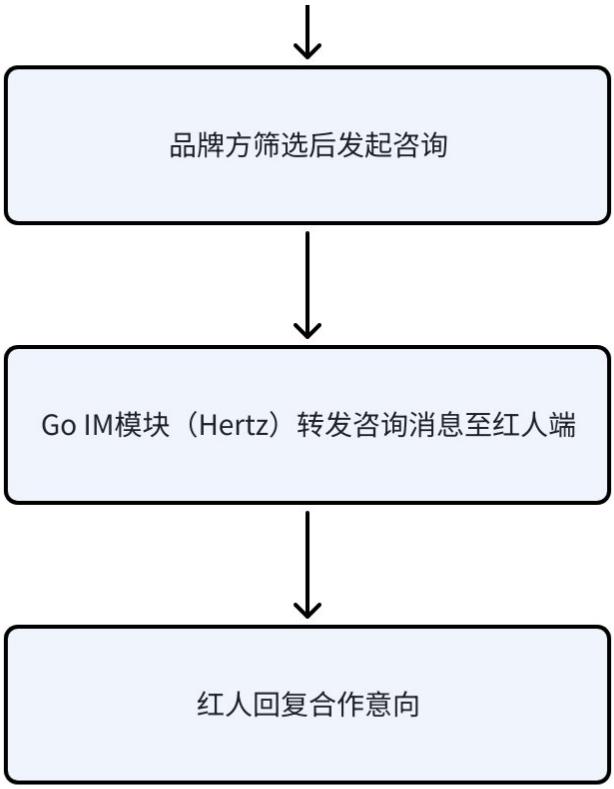

红人绑定UCG平台账号
Go数据采集服务（Hertz+Go-Requests）调用抖音/B站API采集红人数据（粉丝量/互动率/领域）
Go数据清洗模块（Go-Pandas）标准化数据
Go报价引擎服务调用Python FastAPI封装的AI模型接口
TensorFlow训练的报价预测模型输出报价区间
Redis缓存报价结果
Go服务将报价写入MySQL

代码块

```
红人绑定UCG平台账号 → Go数据采集服务（Hertz+Go-Requests）调用抖音/B站API采集红人数据（粉丝量/互动率/领域） → Go数据清洗模块（Go-Pandas）标准化数据 →
Go报价引擎服务调用Python FastAPI封装的AI模型接口 → TensorFlow训练的报价预测模型输出报价区间 → Redis缓存报价结果 → Go服务将报价写入MySQL → 前端（React）展示报价
```

红人 - 品牌方匹配链路

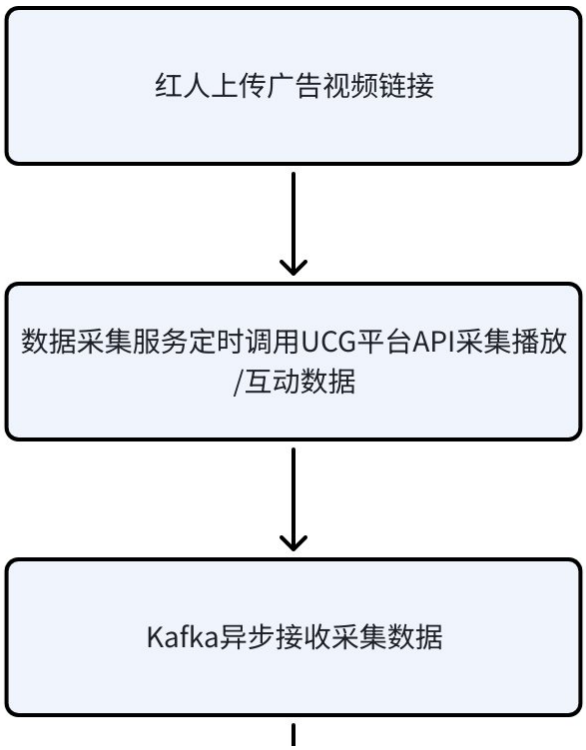


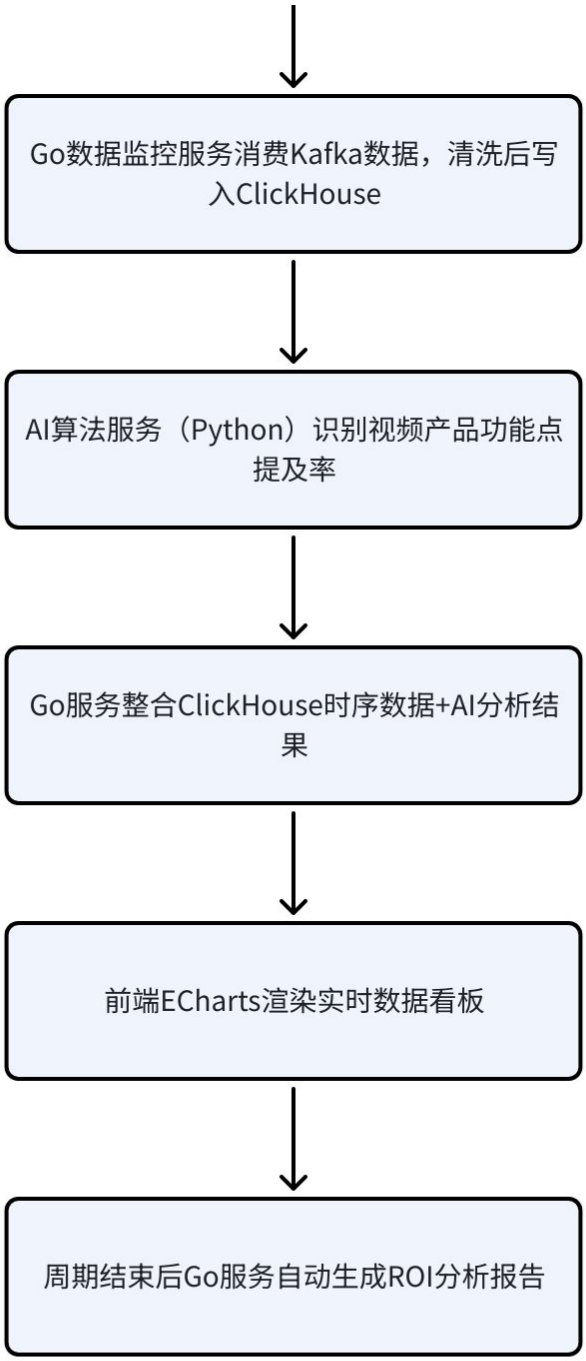


代码块

```
品牌方发布推广需求（产品类别/预算） → Go匹配系统服务提取产品标签 → Elasticsearch检索匹配的红人标签（领域/报价） →
Go服务计算受众重合度（红人粉丝画像vs品牌目标用户） → 按契合度排序生成红人列表 → 品牌方筛选后发起咨询 → Go IM模块（Hertz）转发咨询消息至红人端 → 红人回复合作意向
```

视频数据监控链路





代码块

```
1  红人上传广告视频链接 → Go数据采集服务定时调用UCG平台API采集播放/互动数据 → Kafka异步接收采集数据 →
2  Go数据监控服务消费Kafka数据，清洗后写入ClickHouse → AI算法服务（Python）识别视频产品功能点提及率 →
3  Go服务整合ClickHouse时序数据+AI分析结果 → 前端ECharts渲染实时数据看板 → 周期结束后Go服务自动生成ROI分析报告
```

风险评估&应对

风险点	影响程度	应对方案
抖音 / B 站 API 接口限流	高	1. 基于 Go 协程实现 API 调用限流策略，按平台配额 2. 降级策略：限流时优先返回 Redis 缓存数据，同时 3. 多账号轮询调用，分散限流风险。
Go 团队技术熟练度不足	中	1. 开展 1 周 Gin+Hertz 基础培训，聚焦 “API 开发 + 互” 核心场景； 2. 复用 Python AI 模块，降低 Go 开发复杂度； 3. 编写通用工具类（如 API 调用、Redis 交互），减
AI 报价模型准确率低	中	1. 初期采用 “规则 + 少量数据训练” 的混合报价方案 Python 补充 AI 预测）； 2. 持续沉淀合作数据，迭代训练 AI 模型； 3. 提供红人手动调整报价功能，兜底 AI 误差。
Go 与 MongoDB 交互的灵活性不足	低	1. 结合 GORM Mongo 驱动 + 原生 bson 操作，适配 2. 制定 MongoDB 数据存储规范，统一不同平台红人
实时监控数据延迟	中	1. Kafka 异步采集 + ClickHouse 预计算热点数据，降 2. 前端采用 “定时刷新 + 增量更新” 模式，减少服务 3. 核心监控指标（播放量 / 转化率）缓存至 Redis，

版本/环境规划

技术栈版本

技术组件	版本号	备注
Go	1.21	稳定版，兼容 Hertz 框架
Gin	1.9.1	Go Web 框架核心版本
Hertz	0.9.0	字节开源微服务框架
GORM	1.25.4	支持 MySQL/MongoDB 多驱动
MySQL	8.0	主从复制架构
MongoDB	6.0	副本集部署
ClickHouse	23.0	集群部署，适配时序数据
Redis	7.0	集群模式，支撑高可用
Kafka	3.6	3 节点集群
Docker	24.0	容器化部署核心版本
Kubernetes	1.28	阿里云容器服务 K8s 版本
Python	3.10	AI 算法模块核心版本
FastAPI	0.104.1（当前环境为 0.110.2）	AI 接口封装框架

环境规划

环境类型	部署方式	核心用途
开发环境	本地 Docker + 单机数据库	开发人员调试代码，验证模块功能（如 Go API
测试环境	阿里云 ECS（4 核 8G）+ 测试集群	功能测试、性能测试（高并发 API 调用、数据 测试
生产环境	阿里云容器服务 K8s + 分布式数据库	正式对外提供服务，支撑红人与品牌方的核心 开启主从复制、集群扩容保障高可用

附件

架构拓扑图

（附：微服务集群拓扑图，标注各服务依赖关系、数据流向；数据库集群部署图，标注主从节点、分片规则）

数据库 ER 图

（附：核心表 ER 图，包括红人基础表、品牌方表、合作订单表、监控数据表的字段关联）

API 对接文档

（附：抖音 / B 站 官方 API 对接文档、权限申请流程；Python AI 模块 FastAPI 接口文档）

Go 开发规范

（附：Go 代码规范、微服务接口规范、数据库操作规范，保障团队开发一致性）