陌陌服务化架构实践

高飞航 @ GITC 2017

讲师简介

- 高飞航
- 基础软件组负责人
- 中间件 & 基础服务
- 工作经历

关于陌陌

公司成立，iOS与Android客户端上线，查找附近的人、群组
2011
添加会员服务及表情商城，上线代理的首款联运游戏
2013
在纳斯达克交易所挂牌上市，月活跃用户数6930万
2014
广告投放系统上线，陌陌现场上线，实现盈利
2015
推出独立App哈你直播，月度活跃用户达8110万
2016
发布全新短视频功能，月度活跃用户达9130万
2017

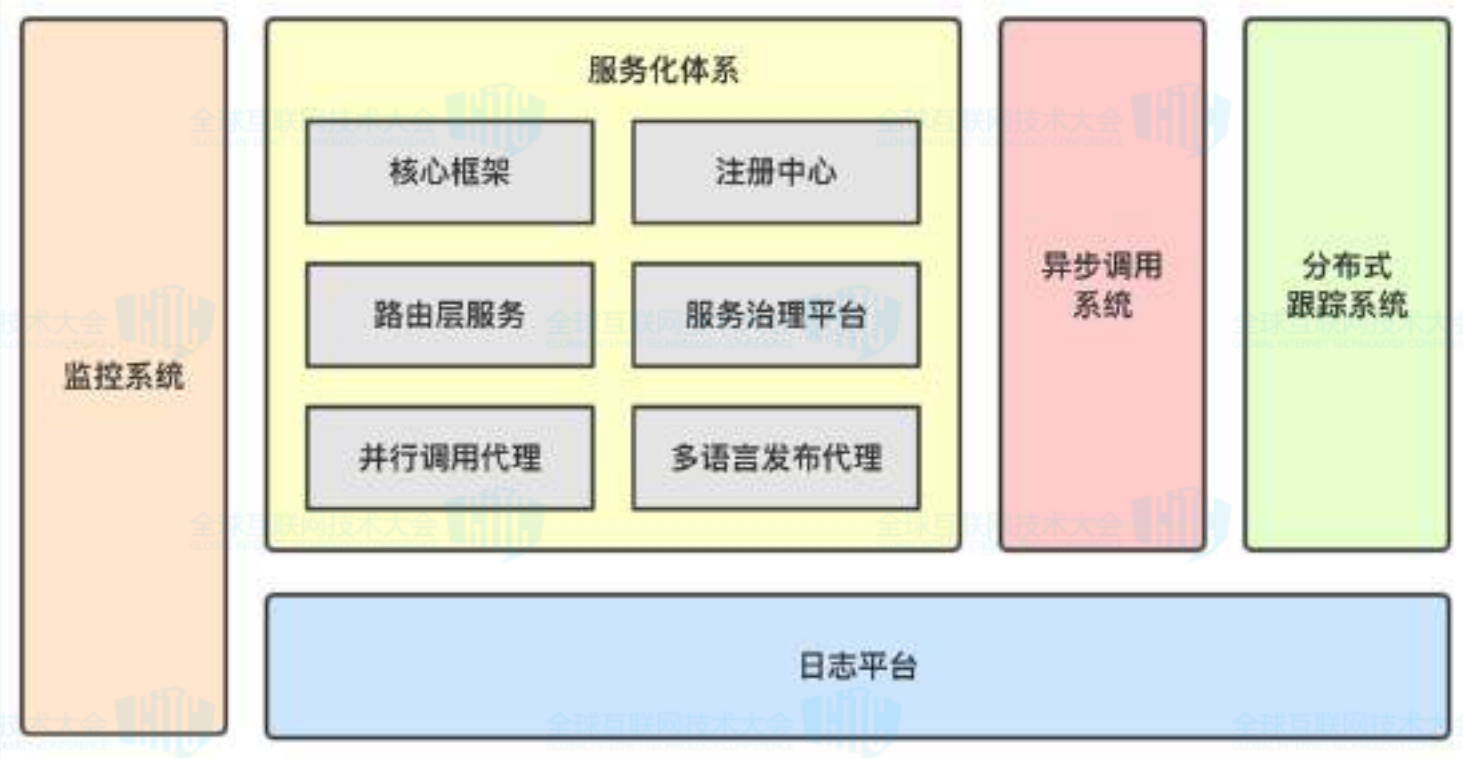
陌陌的服务化

- 起步早
  - 12年底开始转向服务化架构
- 自主研发
  - 核心框架与周边生态产品全部自研
- 架构升级迅速
  - 拥抱微服务与容器化等先进理念

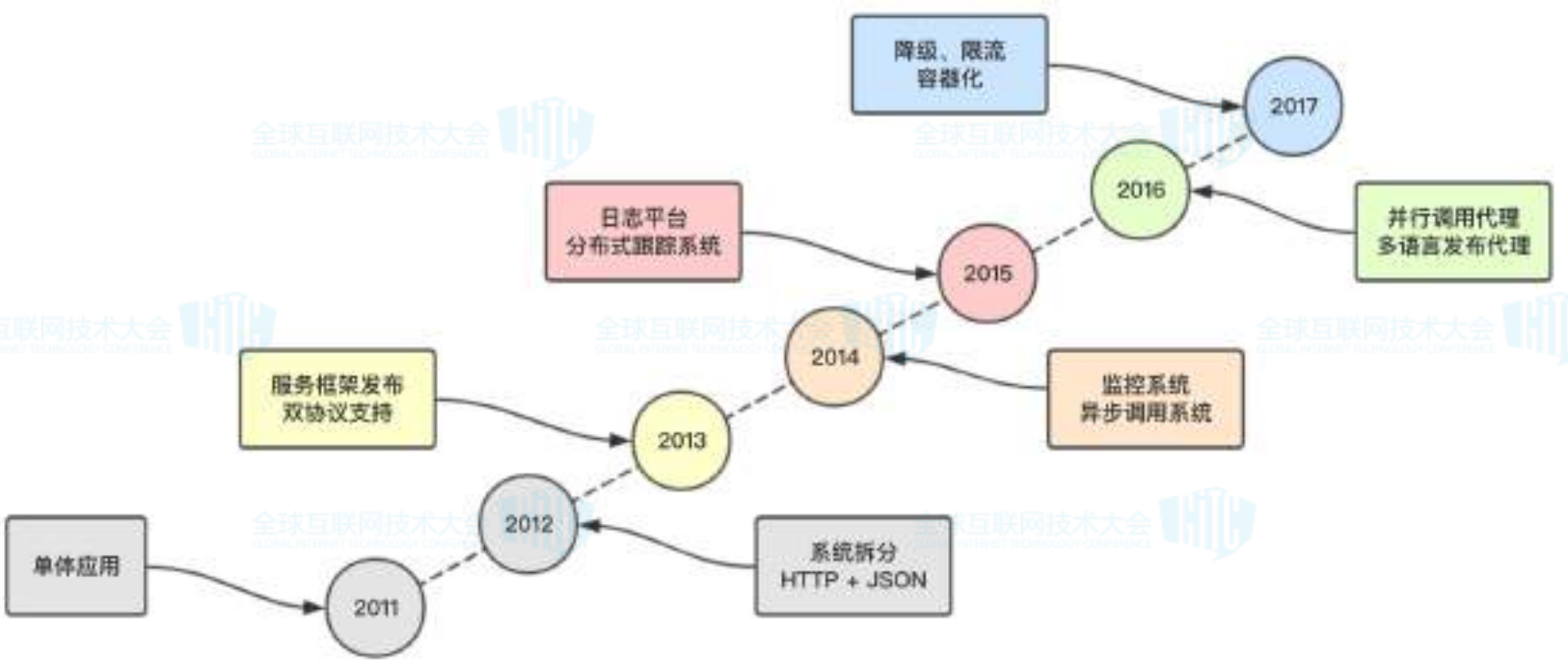
服务化概况

- 1000+ 服务
- 7000+ 实例
- 6000+ 主机（含Docker容器）
- 全天调用总量 1500亿
- 峰值 300万 / 秒

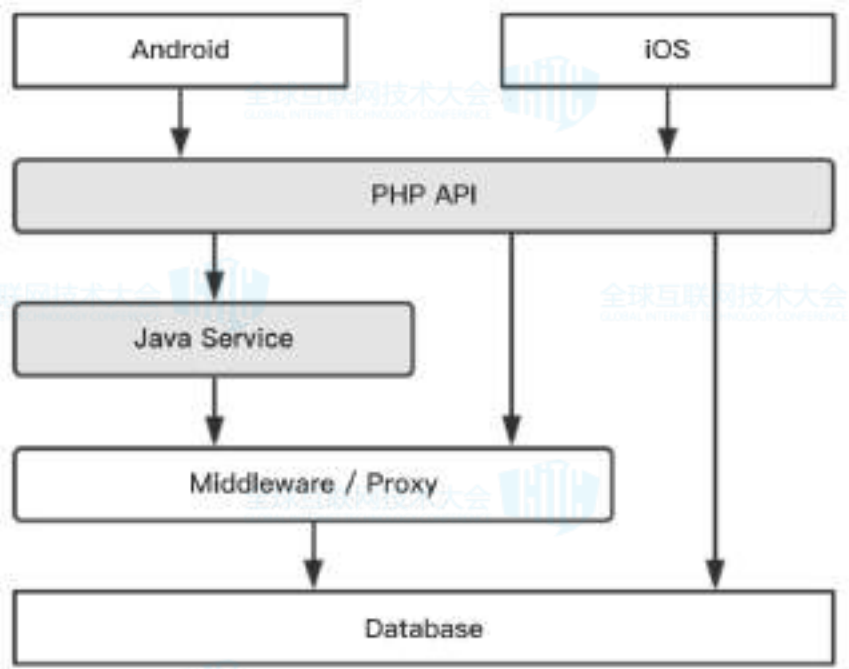
服务化生态


监控系统
服务化体系
核心框架
注册中心
路由层服务
服务治理平台
并行调用代理
多语言发布代理
异步调用
系统
分布式
跟踪系统
日志平台

服务化架构演进


降级、限流
容器化
2017
2016
并行调用代理
多语言发布代理
日志平台
分布式跟踪系统
2015
2014
监控系统
异步调用系统
服务框架发布
双协议支持
2013
2012
系统拆分
HTTP + JSON
单体应用
2011

应用架构体系

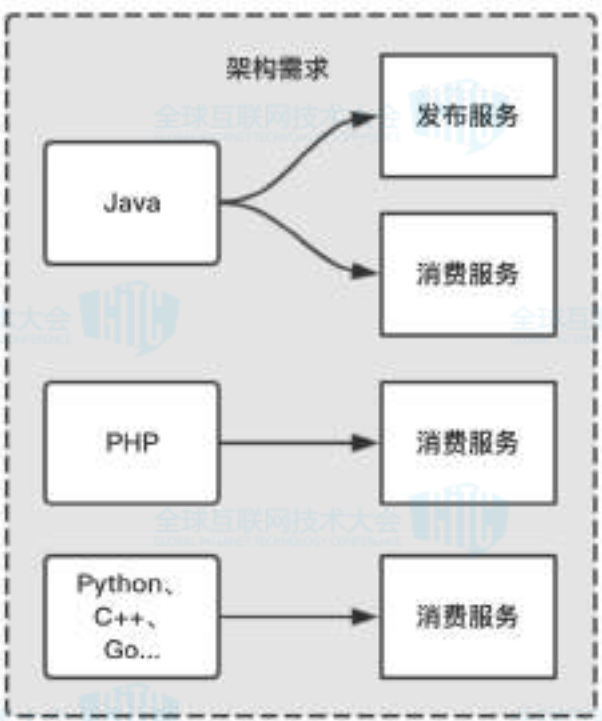


- PHP开发API快速响应业务需求
- Java开发服务层处理复杂运算

服务化前

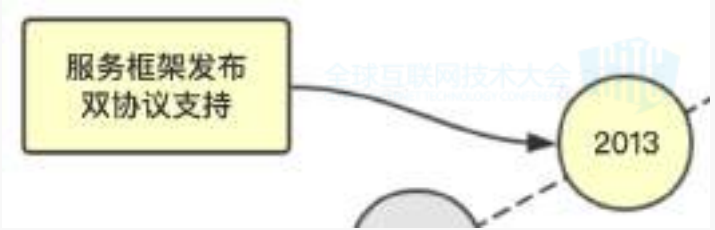
- 从单体应用过渡到垂直应用架构
- 开始出现包含公共逻辑的接口
- 上层PHP调用下层Java应用为主
- HTTP + JSON方式通信
- 使用共享的Command结构进行交互

服务化需求



- Java内部高效通信
- PHP等多种语言仅消费服务
- 统一的服务发现机制

服务化体系 v1.0



Java内部

Hessian序列化
自定义传输协议
连接复用

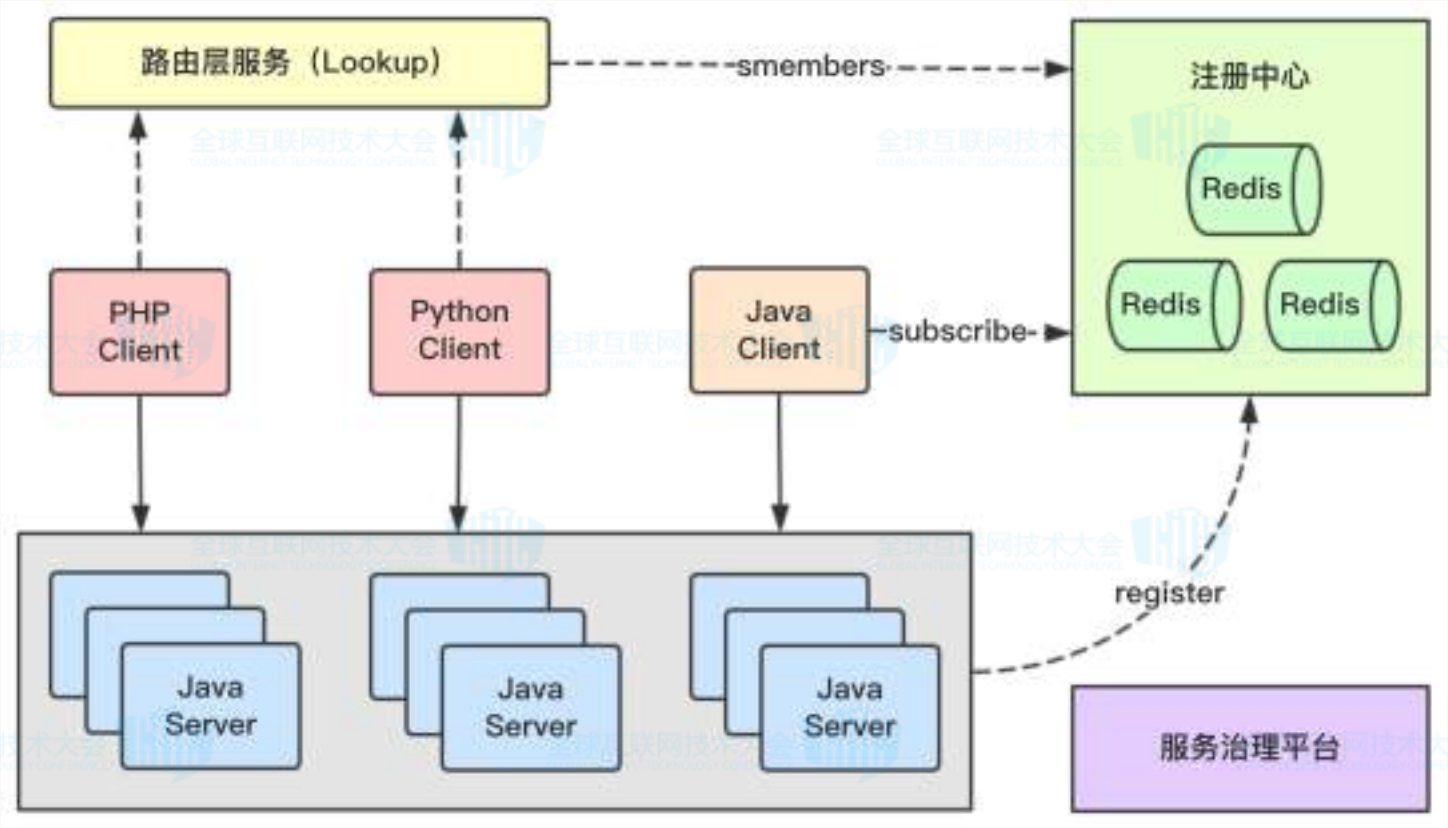
跨语言

JSON序列化
Redis传输协议
路由层服务

注册中心

Redis底层存储
Pub-Sub变更通知
主从结构高可用

服务化体系 v1.0 架构


路由层服务（Lookup）
smembers
注册中心
Redis
Redis
Redis
PHP Client
Python Client
Java Client
subscribe
Java Server
Java Server
Java Server
register
服务治理平台

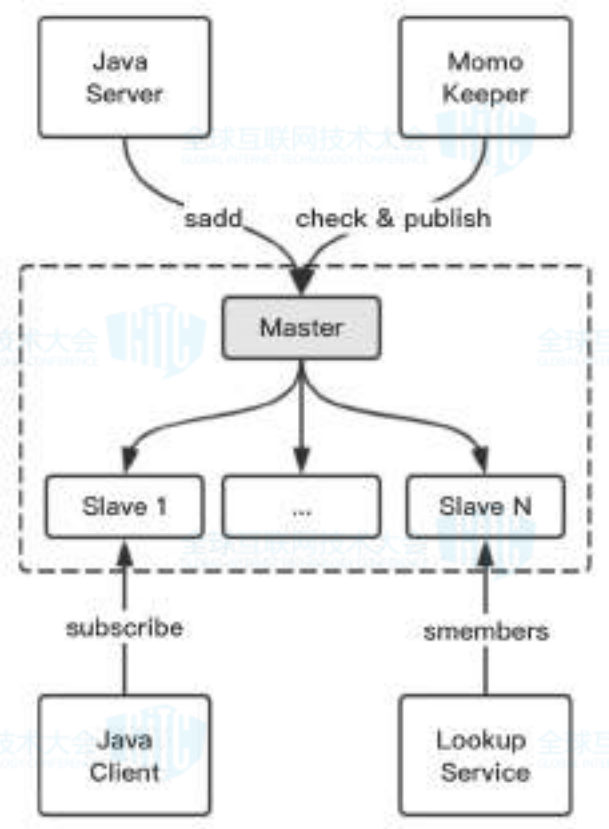
跨语言协议

```
{
    "service": "/service/helloworld",
    "method": "hello",
    "args": ["world"]
}
```

```
*2
$3
GET
$14
RequestMessage
```

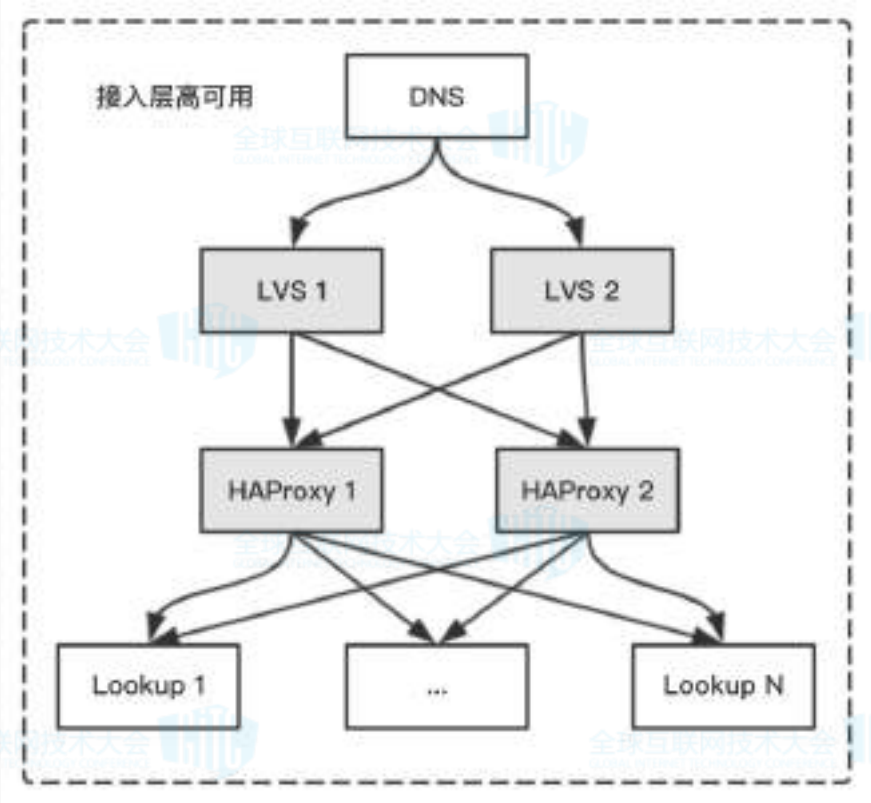
- JSON序列化协议
  - 贴近原有数据结构与交互方式
  - 灵活性、可读性
  - 业务接口复杂度可控
- Redis传输协议
  - 相比HTTP更加简洁高效
  - 多语言客户端丰富、稳定
  - 节省传输层开发成本

注册中心



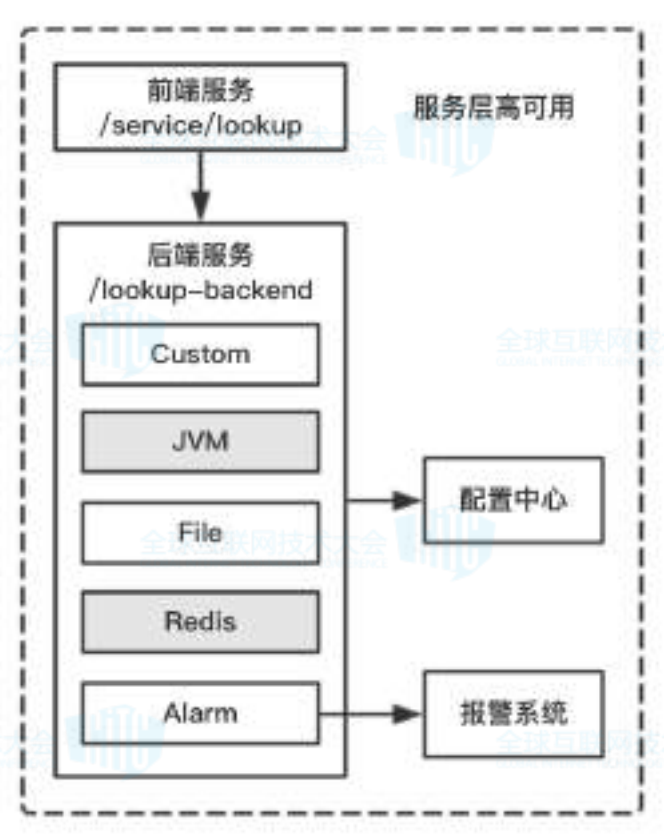
- Redis vs ZooKeeper
  - 满足变更通知与高可用的需求
  - 使用和运维经验更加丰富
- 架构优化
  - 扩展一主多从分担压力
  - 路由层服务增加缓存减少QPS
  - 订阅不同服务使用单链接多Channel

路由层服务



- 通过域名访问服务
- DNS解析到LVS服务器
- 加入HAProxy解决网段受限的问题

路由层服务

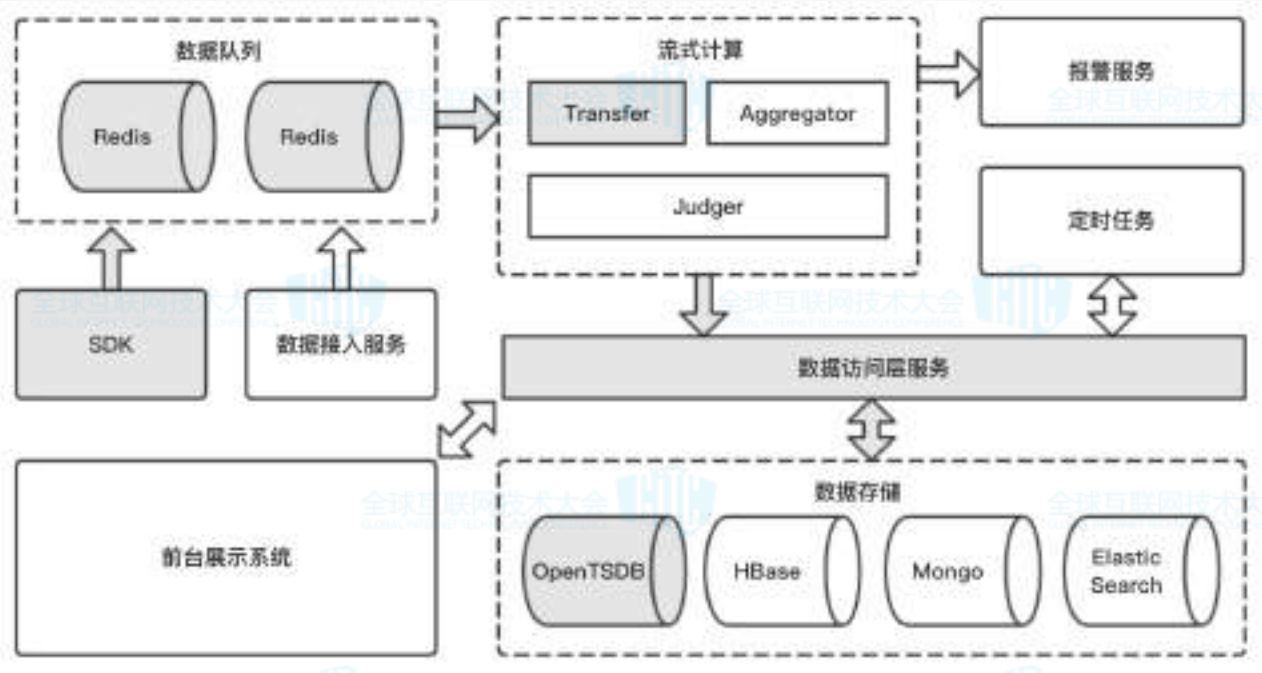


- 地址来源分层，组成责任链结构
- 开关灵活控制功能开启

服务化生态 v1.0

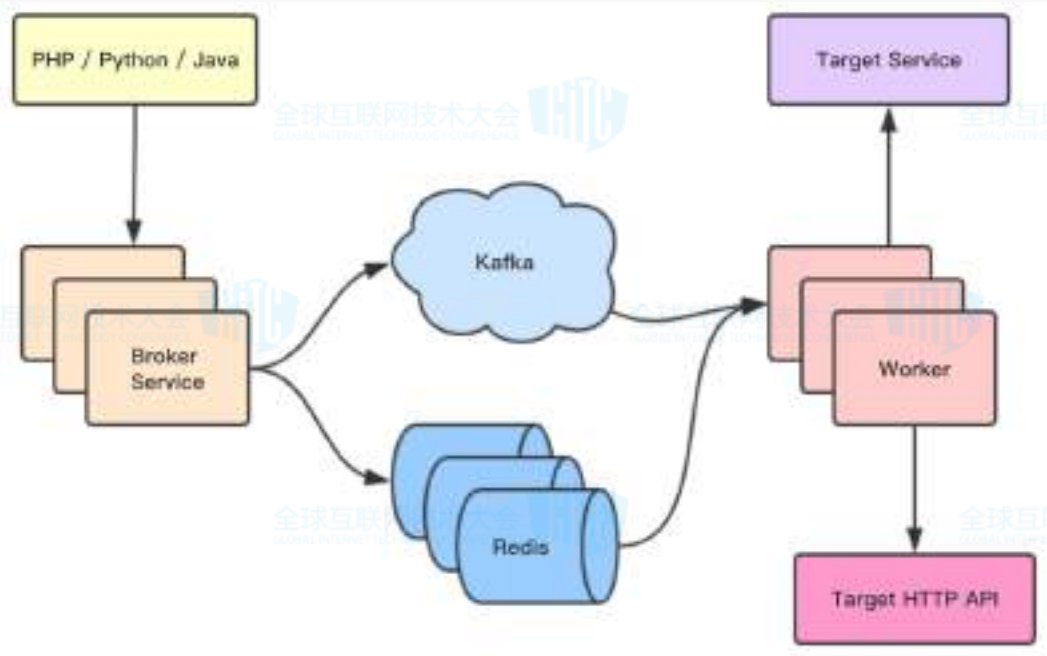
- 完善监控机制
  - 仅有本地日志，没有统一的监控系统
- 异步调用支持
  - 框架内部实现功能有限，使用场景日趋复杂

监控系统



- 数据采集
  - 通过SDK统计
  - 分钟为周期提交
  - Redis队列传输
- 监控项
  - 精确到方法
  - 百分位时间
  - 线程、GC

异步调用系统

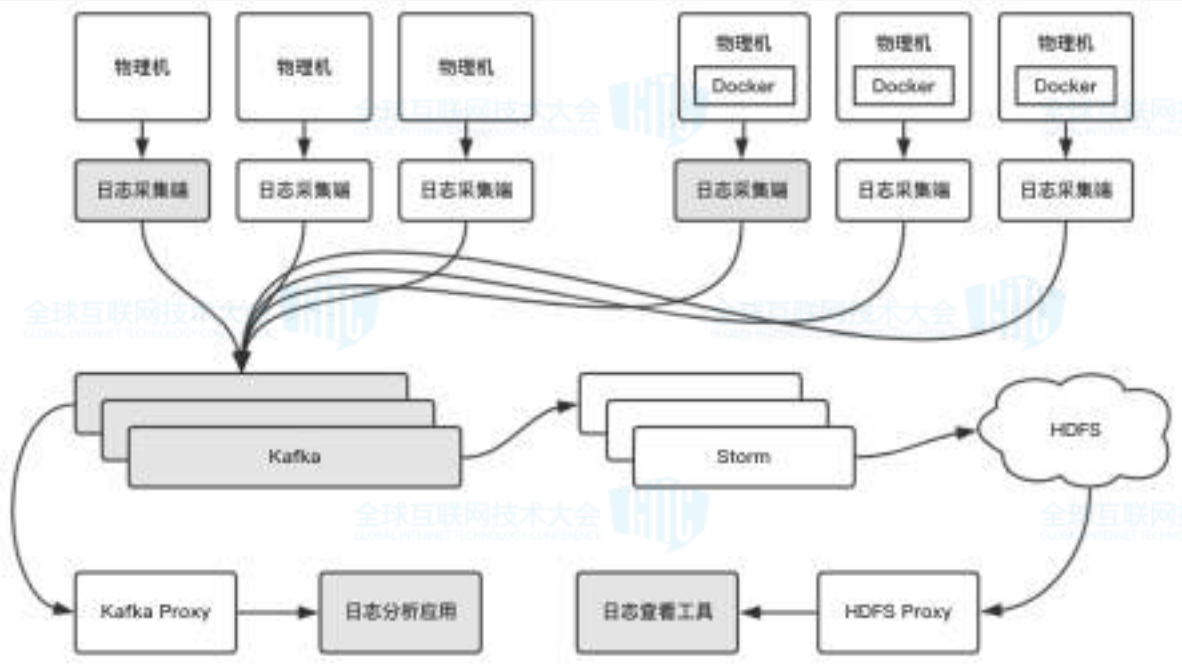


- 针对异步调用需求开发独立产品
- Broker、队列、Worker三种角色
- 支持调用Service与HTTP接口
- 增加Kafka队列模式解决请求堆积问题

服务化生态 v2.0

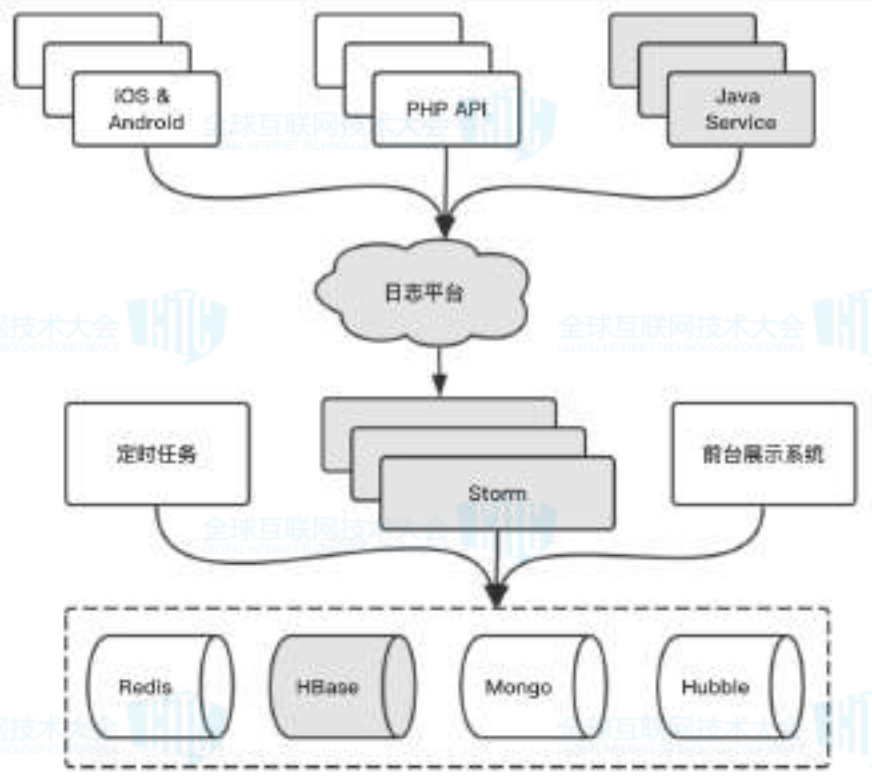
- 日志平台
  - 服务器数量增多，日志查找成本剧增
- 分布式跟踪系统
  - 调用链路复杂，依赖关系难以梳理

日志平台



- 原理
  - Agent采集
  - Kafka、Storm中转
  - HDFS存储
- 场景
  - 多服务器日志查看
  - 滚动冲刷日志回溯
  - 分析数据采集

分布式跟踪系统

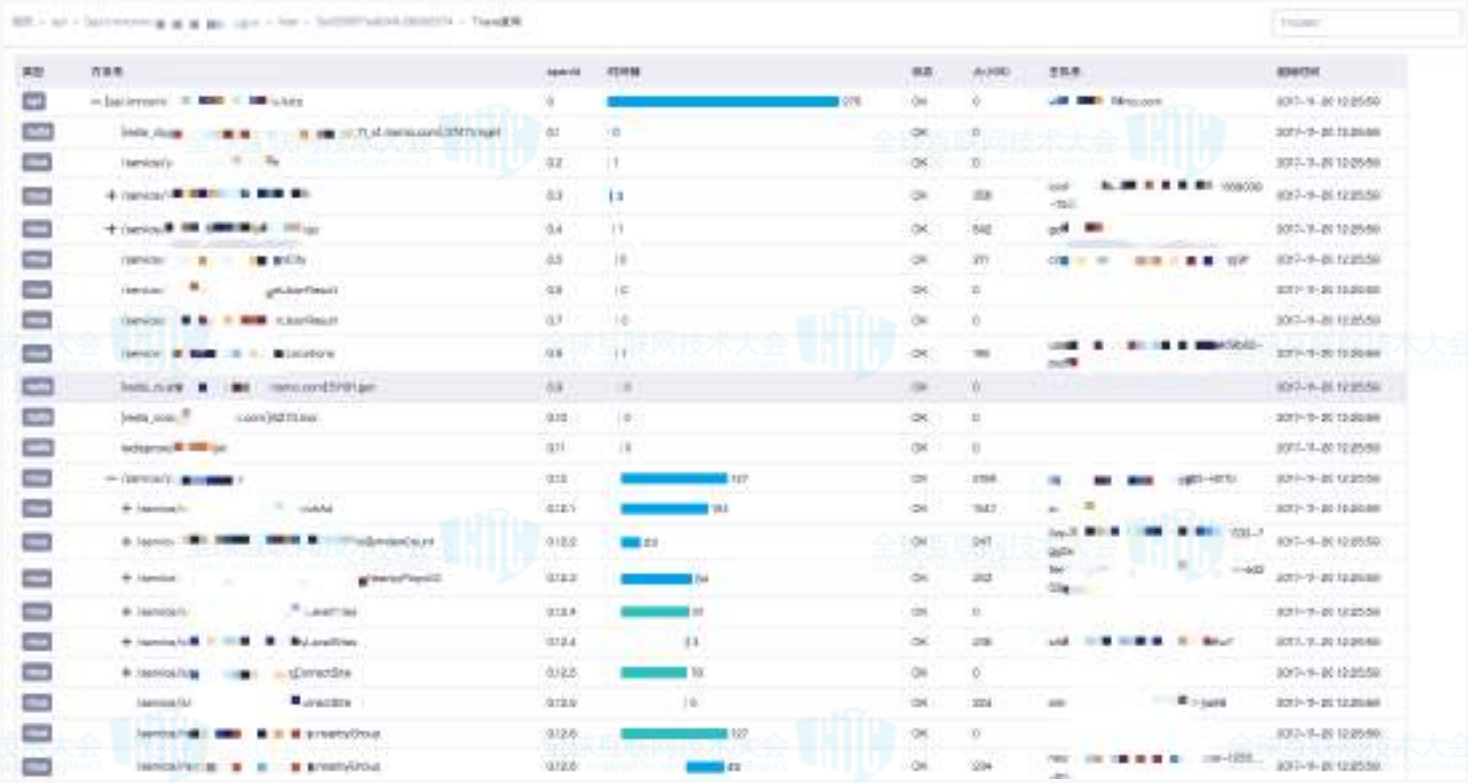


- 原理
  - 制定链路日志规范
  - 分层记录、统一收集
  - 唯一ID关联还原
- 场景
  - 调用链路分析
  - 单次慢请求分析
  - 链路异常报警

调用链路分析

类型	方法名	请求量	请求比例	平均耗时	耗时占比	依赖程度	标记	最新采集时间
moa	— /service/[redacted]rbyPlays	1875	-	41	-	-		2017-11-01 18:47:59
moa	— /service/[redacted]edListByIds	1319	1	10.3	26%	97%	强依赖	2017-11-01 18:47:59
redis	redis_cluster_[redacted]_a.mget	28077	21.5	4.5	15%	99%	强依赖	2017-11-01 18:47:59
moa	— /service/[redacted]nFeeds	74	1	27	49%	36%		2017-11-01 18:47:59
redis	redis_cluster_[redacted]_a.mget	-	-	-	-	-		-
moa	/service/[redacted]TagsWithTagIdsV2	1319	1	1.7	4%	97%	强依赖	2017-11-01 18:47:59
redis	redis_cluster_[redacted]_a.del	1150	1	1.1	2%	14%		2017-11-01 18:47:48
redis	redis_cluster_[redacted]_a.expire	1150	1	0.6	1%	14%		2017-11-01 18:47:48
redis	redis_cluster_[redacted]_a.zadd	3450	3	3.9	10%	19%		2017-11-01 18:47:45

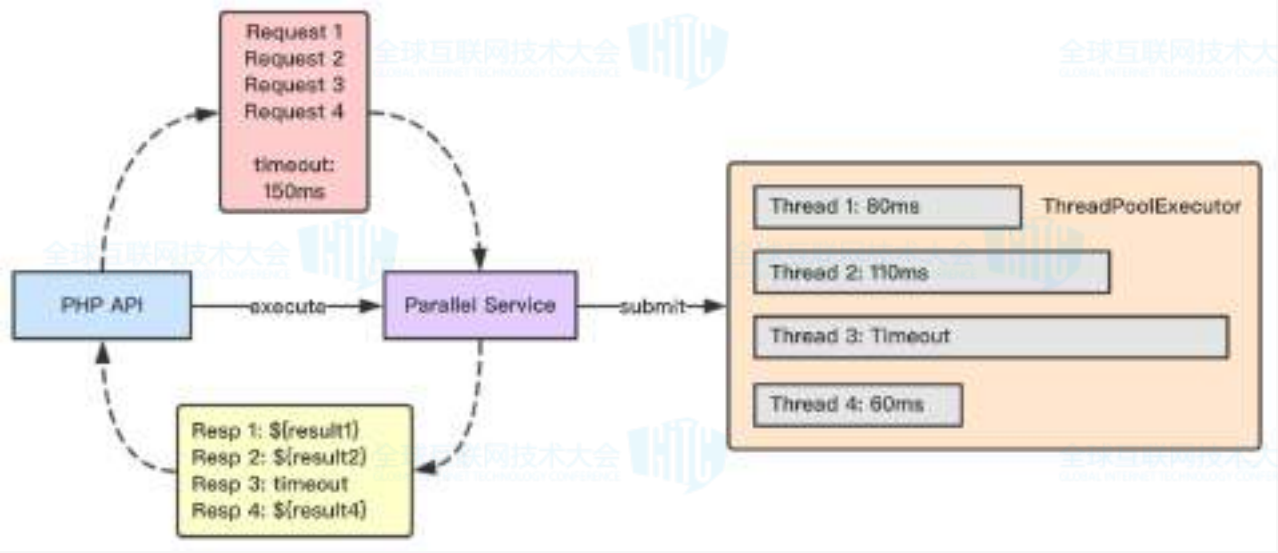
单次请求分析



服务化生态 v3.0

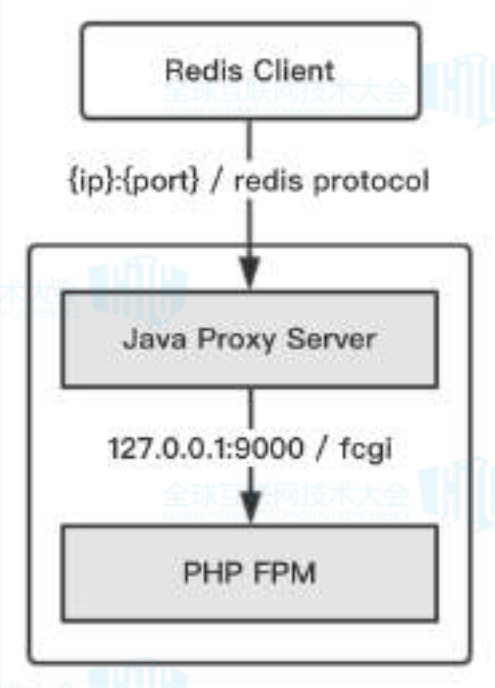
- 并行调用代理
  - PHP作为大量服务的调用方，无法实现并行调用
- 多语言发布代理
  - 其他语言开始有抽取公共服务接口的需求

并行调用代理服务



- Java开发代理服务
- 请求打包发送至代理服务
- 代理并行执行，打包返回响应
- 标记失败请求，由调用方处理

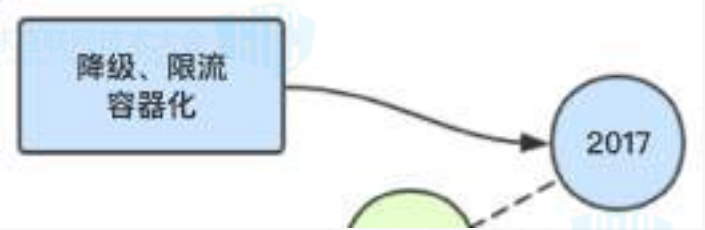
多语言服务发布代理



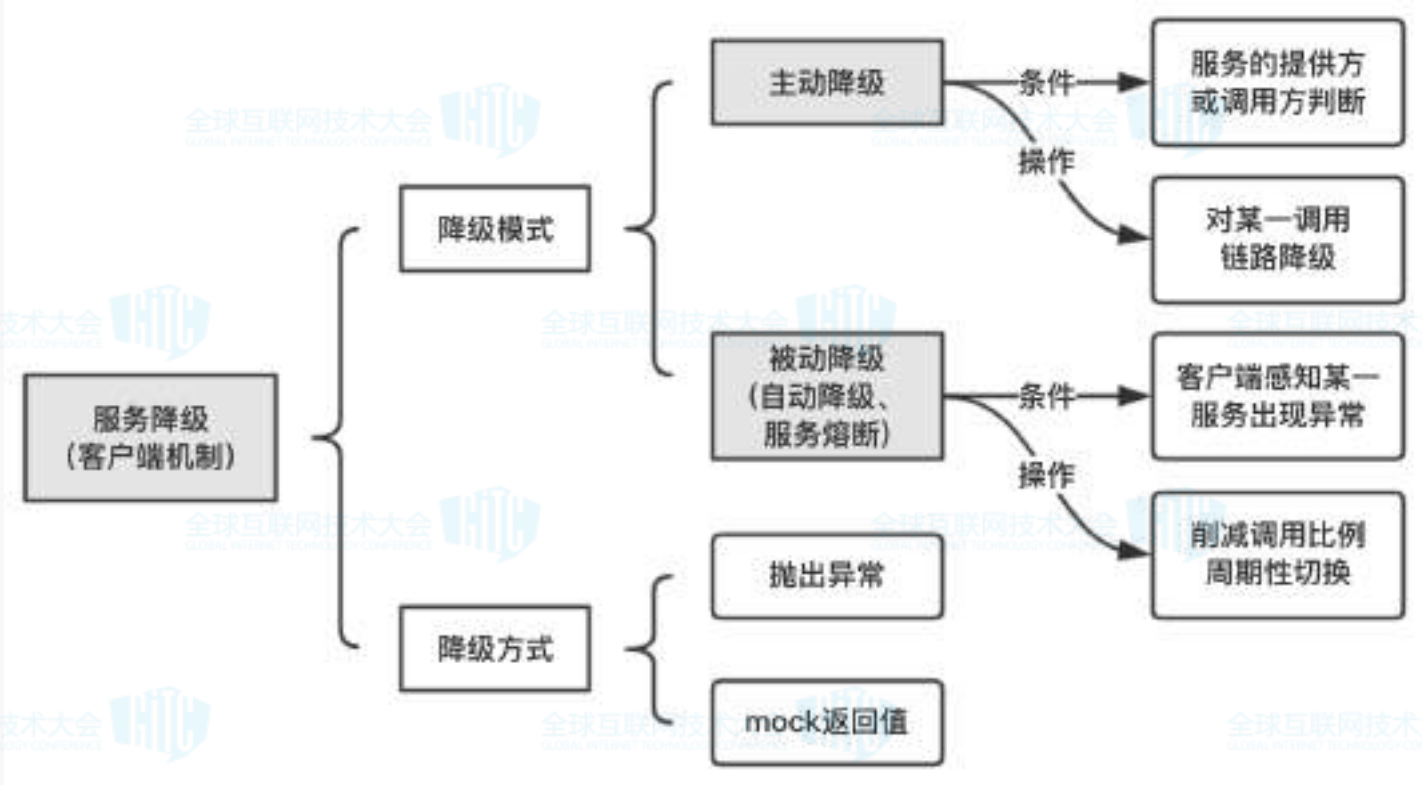
- 服务框架改造成Proxy Server
- 同时部署Proxy与业务进程
- 以公有协议（Fast CGI）传递Request
- 复用框架原有的服务发现、服务治理、监控等机制

服务化生态 v4.0

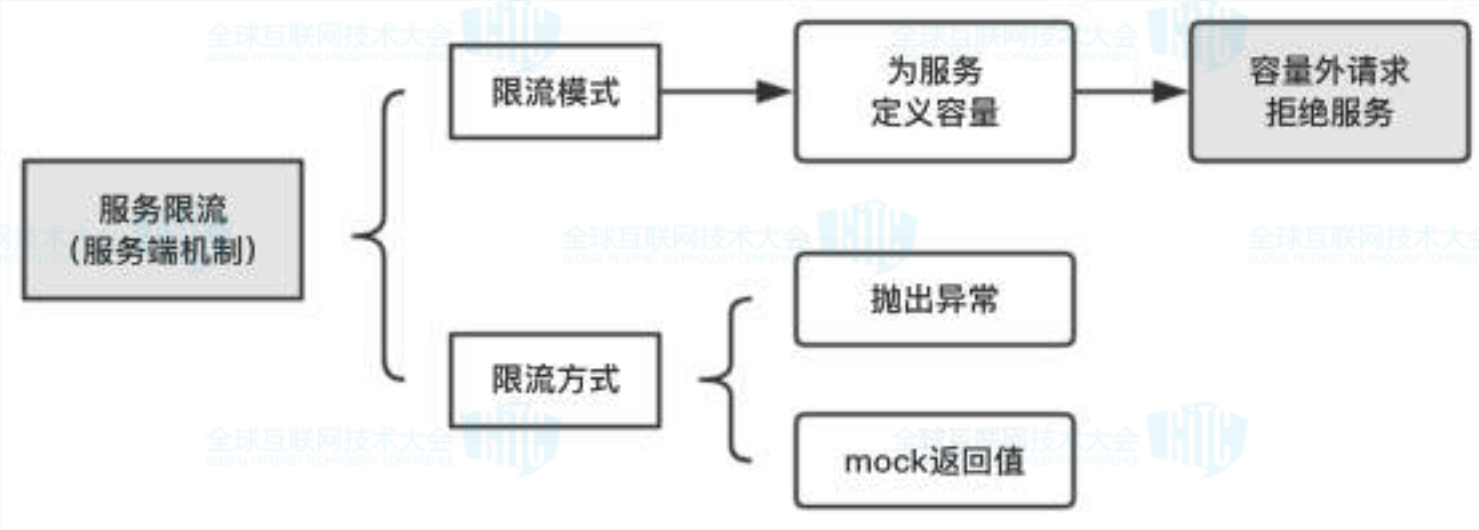
- 降级、限流
  - 系统稳定性成为首要目标，快速从异常中恢复
- 容器化
  - 实现资源合理利用、灵活调度



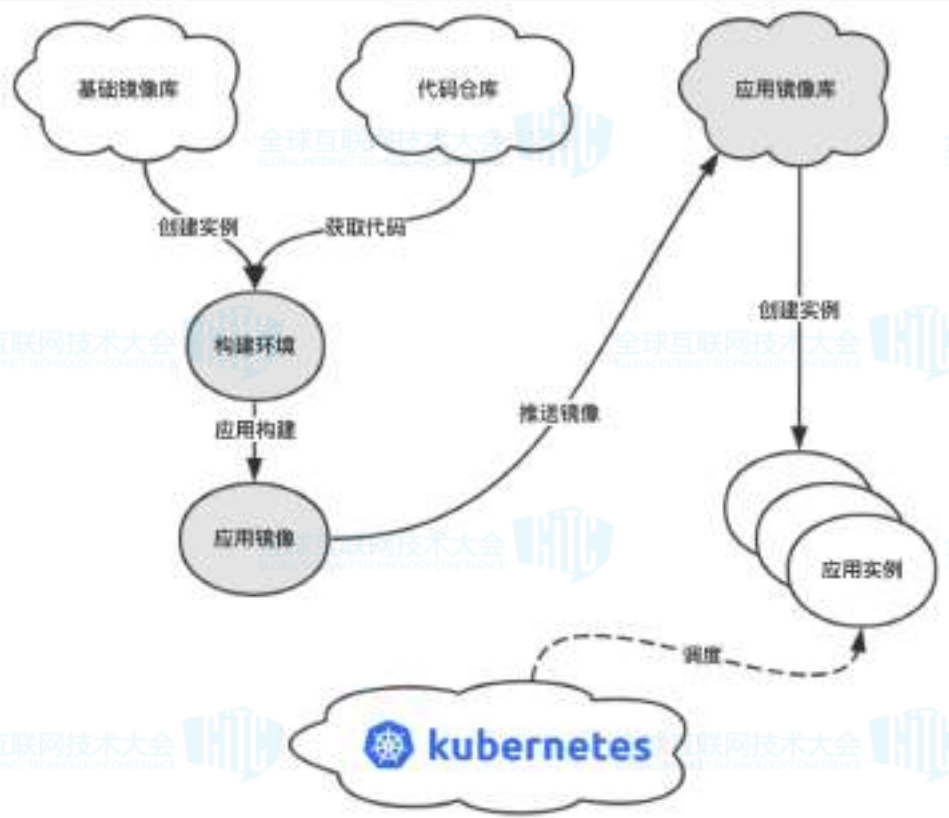
服务降级


服务降级
（客户端机制）
降级模式
主动降级
条件
服务的提供方
或调用方判断
操作
对某一调用
链路降级
被动降级
（自动降级、
服务熔断）
条件
客户端感知某一
服务出现异常
操作
削减调用比例
周期性切换
降级方式
抛出异常
mock返回值

服务限流



容器化



- 由基础镜像创建构建环境
- 拉取代码构建应用
- 创建应用镜像推送至应用仓库
- 由应用镜像创建应用实例
- 应用实例由k8s进行调度

服务部署

服务化架构的未来方向

- 微服务（Microservice）
  - 细粒度、独立运行、轻量通信
- 服务网格（Service Mesh）
  - 服务间的通信机制作为更底层的设施
  - 预装轻量的通信代理，构成通信网格

Q & A

Thank you!

Welcome to join us!

Mail To: gao.feihang@immomo.com