比赛地址：
https://tianchi.aliyun.com/programming/introduction.htm?spm=5176.100066.0.0.23fc33afZPeRwX&raceId=231657

赛题介绍：

Apache Dubbo (incubating) 是阿里巴巴中间件团队开源的一款高性能 Java RPC 通讯框架。在分布式应用场景下，服务间通讯是非常重要的能力，通常由服务提供者暴露服务，由服务消费者调用服务。在 Dubbo 服务整合能力的支持下，使用 RPC 可以像使用本地调用一样轻松、便捷。但是在异常复杂的系统环境下，服务间调用也会变得非常复杂，如果没有一套完善的、经过大规模生产环境验证的服务治理能力的话，系统将会处于非常危险的境地。因此，从另一个方面来讲，Dubbo 不只是单纯的服务通讯框架，更是一套完备的服务治理框架。

提到服务治理能力就不能不说一下微服务。微服务不光是创造性的将曾经的单体系统拆分为若干个独立的微服务系统，更重要的是其为这些服务的和谐运行提供了最佳实践和解决方案。服务注册、服务发现、服务治理、负载均衡、服务监控、流量管控、服务降级、服务熔断和服务安全等等，这些能力都是一个可用和可靠的微服务系统所不可或缺的。微服务的一大问题在于改造过程必须深入服务内部，拿使用 Dubbo 来说，所有接入的微服务系统都必须引入 Dubbo 组件，并暴露或消费相关的服务。

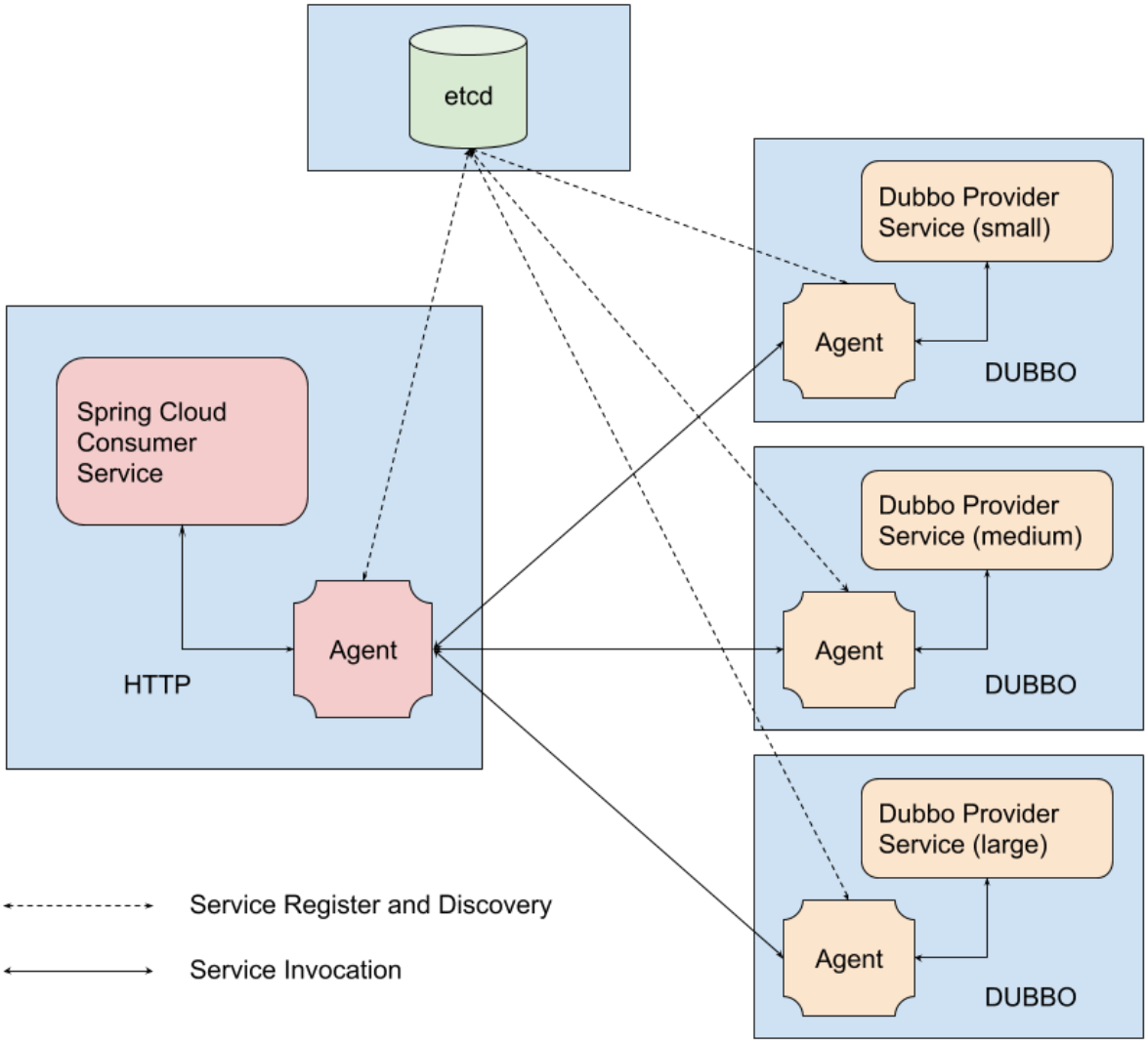
而 Service Mesh 则另辟蹊径，其实现服务治理的过程不需要改变服务本身。通过以 proxy 或 sidecar 形式部署的 Agent，所有进出服务的流量都会被 Agent 拦截并加以处理，这样一来微服务场景下的各种服务治理能力都可以通过 Agent 来完成，这大大降低了服务化改造的难度和成本。而且 Agent 作为两个服务之间的媒介，还可以起到协议转换的作用，这能够使得基于不同技术框架和通讯协议建设的服务也可以实现互联互通，这一点在传统微服务框架下是很难实现的。

要求：

实现一个高性能的 Service Mesh Agent 组件，并包含如下一些功能：

1. 服务注册与发现
2. 协议转换
3. 负载均衡

系统架构



（图片摘自官网）

根据题意可知，我们要一套服务治理组件，该组件的目的是服务于系统中 Consumer（以下称之为 C）与 Provider（以下称之为 P），以 agent（以下称之为 CA 或者 PA）方式作为 C 与 P 之间的信使，为 C 与 P 传递消息。整理后信息如下：

C 入栈出栈均为 HTTP 协议

P 入栈出栈均为 Dubbo 协议

P 性能各不相同，性能比例为 S:M:L 为 1：2：3

Etcd 为注册中心

所以我们的 CA 要具备如下功能：

- 监听 Etcd 节点信息，增加或删除负载节点

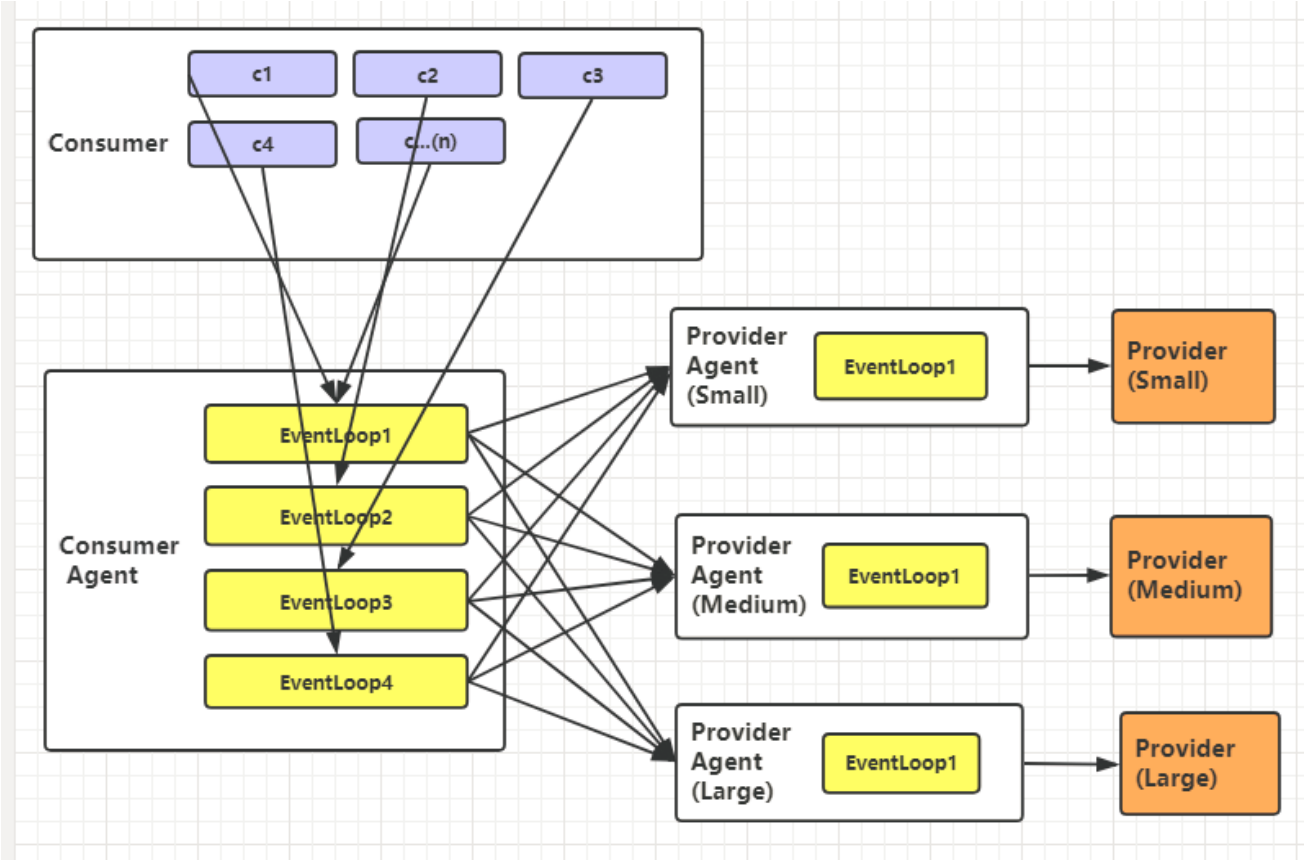
- 解析 http 协议，并将其转化为自定义协议发送给 PA
- 根据 P 的性能动态选择负载节点
- 如果所有 P 均满负载时返回当前服务不可用给 C
- 接收 PA 返回的消息回复给 C

我们的 PA 要具备如下功能：

- 注册节点信息到 Etcd
- 解析 CA 发送过来的消息，并将其转化为 Dubbo 协议发送给 P
- 接收 P 返回的消息回复给 CA

通讯工具选择，准备选择 netty 或者 baseio，后来想了下还是使用 baseio，希望通过这次比赛，试探一下 baseio 的性能如何。

为减少线程切换，我们使用线程模型如下



如图所示：

C 中每个 c 代表一条连接（目前比赛中使用的是 512）

CA 中启动 4 条 EventLoop

PA 中启动 1 条 EventLoop

在这里我们把 CA 中每条 EventLoop 作为一个小的 CA（以下称之为 CAs），它具备与 CA 完全一致的功能，这样做的目的是可以隔离线程减少线程间的数据访问以及线程切换，各条 EventLoop 独立运行互不干扰，每条 EventLoop 会与环境内所有的 PA 建立连接，不要担心这里会因为建立过多的连接而创建很多 EventLoop，在这里所有的连接都与 CAs 复用同一条 EventLoop，也就是说每个 CAs 均为单线程应用，没有线程切换（说没有不严谨），极少的内存共享，大部分的数据访问会落在自己 CPU 所在的缓存内，同样 PA 也是仅创建一条 EventLoop，CA 与 PA 的连接以及 PA 与 P 的连接均复用该 EventLoop。

还有一个好处是单线程对与计算 rt 非常友好，所以我们所使用的负载方式为根据 rt 全动态负载。

一些细节处理：

因为 PA 与 P 仅有一条连接，而 PA 与 CA 却是多条连接，这里涉及一个消息回溯问题，即如何将消息返回给正确的 channel，这里我们在 PA 维护了一张 channelId 与 msgId 的关系表，该表使用 IntObjectHashMap 存储，key 为 msgId，V 为 channel。