

MyRPC清单

改进

- JSON格式编码（跨语言）
- send超时没有处理的问题
- 负载均衡一致性hash策略
- 怎么在外部终止一个协程，防止协程泄露

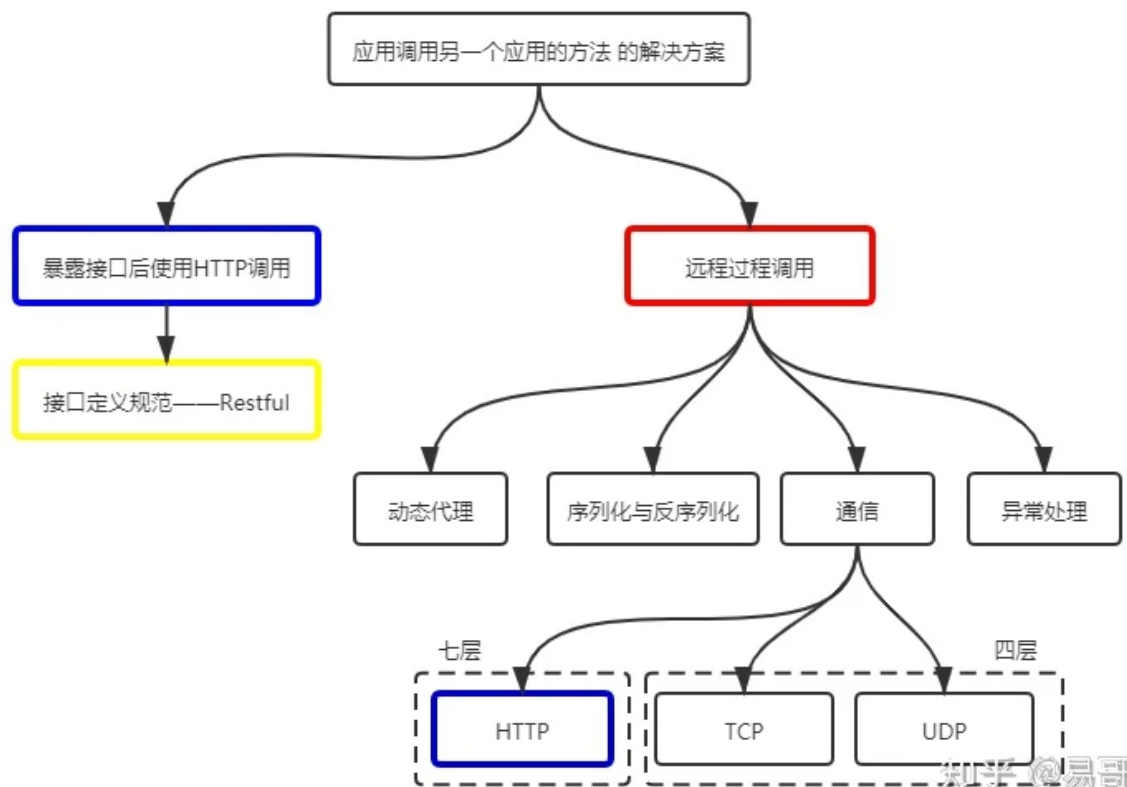
问题

- RPC 和 HTTP 的关系
- RPC over HTTP 和 RPC over TCP 什么区别
- net/rpc中为什么要提供HTTP服务
- 什么是服务发现
- 负载均衡有哪些方法
- 什么是注册中心
- 为什么需要注册中心

问题解释

| RPC和HTTP的关系

RPC指的是远程调用，HTTP指的是通信协议。RPC通信可以用HTTP协议，也可以用TCP、UDP或者说自定义协议，不做约束。



远程过程调用，那为什么不使用HTTP直接进行。Restful不可以吗？

HTTP协议的优点在于，可读性好，且可以得到防火墙的支持，跨语言的支持。问题在于，HTTP协议在应用层，有很多信息是没有用的，效率低。

RPC的优势在于效率更高。但所谓的效率优势是针对 http1.1协议 来讲的，http2.0协议已经优化编码效率问题，像 grpc 这种 rpc 库使用的就是 http2.0协议。

那为什么还要用RPC调用？

因为良好的RPC调用是面向服务的封装，针对服务的可用性和效率等都做了优化。单纯使用http调用则缺少了这些特性。

| RPC over HTTP 和 RPC over TCP 什么区别

最大的区别在于效率。但是HTTP2.0已经优化了编码效率的问题，这时候还用RPC更多是因为服务治理等特性。

| net/rpc中为什么要提供HTTP服务

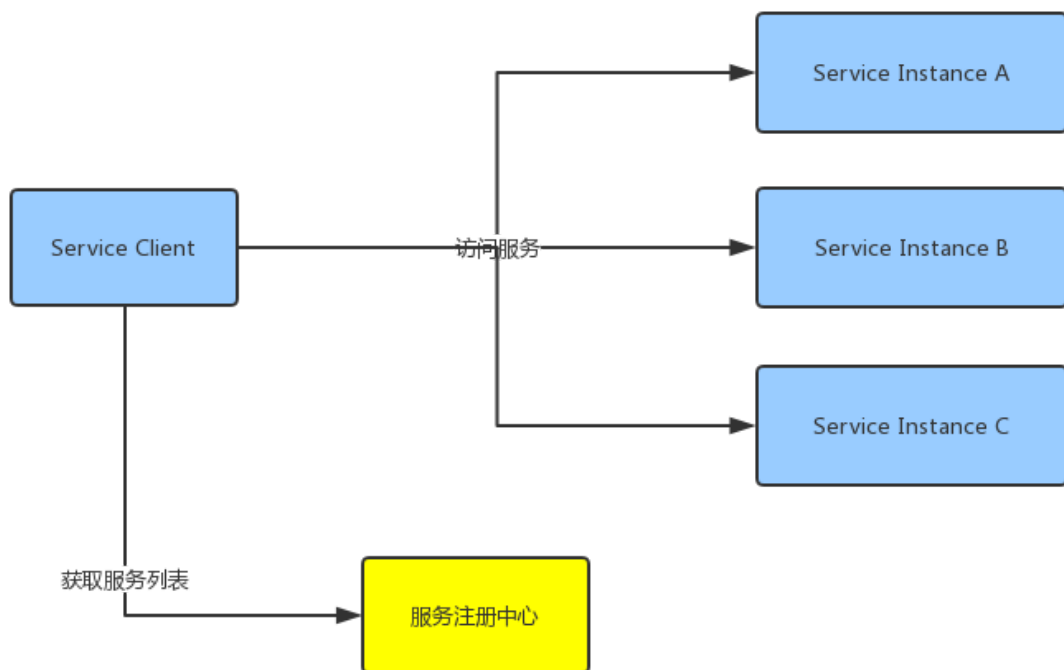
和这的理由是一样的，可以支持浏览器的使用，还可以跨语言之类的。不同路径可以提供不同服务。

| 什么是服务发现

目前，服务发现主要存在有两种模式，客户端模式与服务端模式，两者的本质区别在于，**客户端是否保存服务列表信息**。下面用两个图来表示客户端模式与服务端模式。

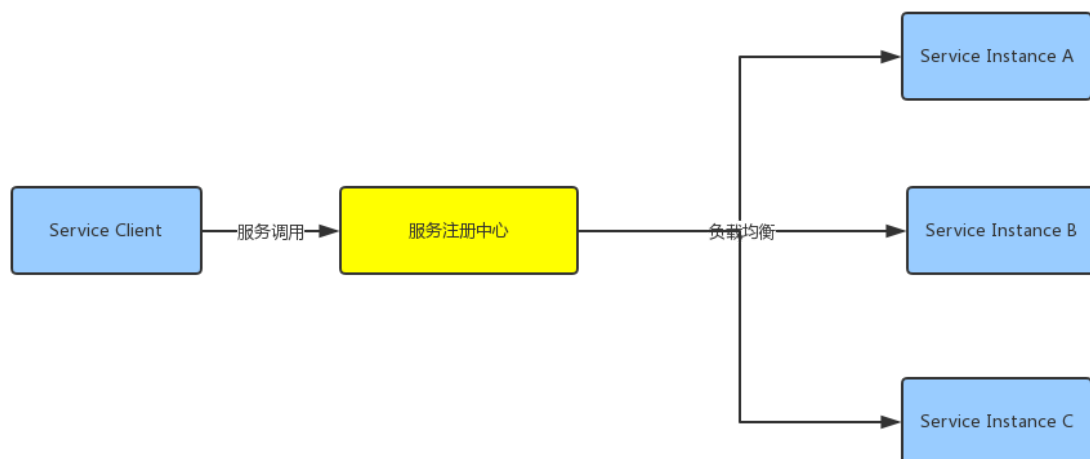
本项目采用的是客户端模式的服务发现。

客户端模式服务发现：



http://blog.csdn.net/Mr_SeaTurtle_

服务端模式的服务发现：



http://blog.csdn.net/Mr_SeaTurtle_

客户端模式	服务端模式
只需要周期性获取列表，在调用服务时可以直接调用少了一个RT。但需要在每个客户端维护获取列表的逻辑	简单，不需要在客户端维护获取服务列表的逻辑
可用性高，即使注册中心出现故障也能正常工作	可用性由路由器中间件决定，路由中间件故障则所有服务不可用，同时，由于所有调度以及存储都由中间件服务器完成，中间件服务器可能会面临过高的负载
服务上下线对调用方有影响（会出现短暂调用失败）	服务上下线调用方无感知

| 负载均衡有哪些方法

随机选择策略

从服务列表中随机选择一个

轮询算法

依次调度不同的服务器，每次调度执行 $i = (i + 1) \bmod n$ 。

加权轮询

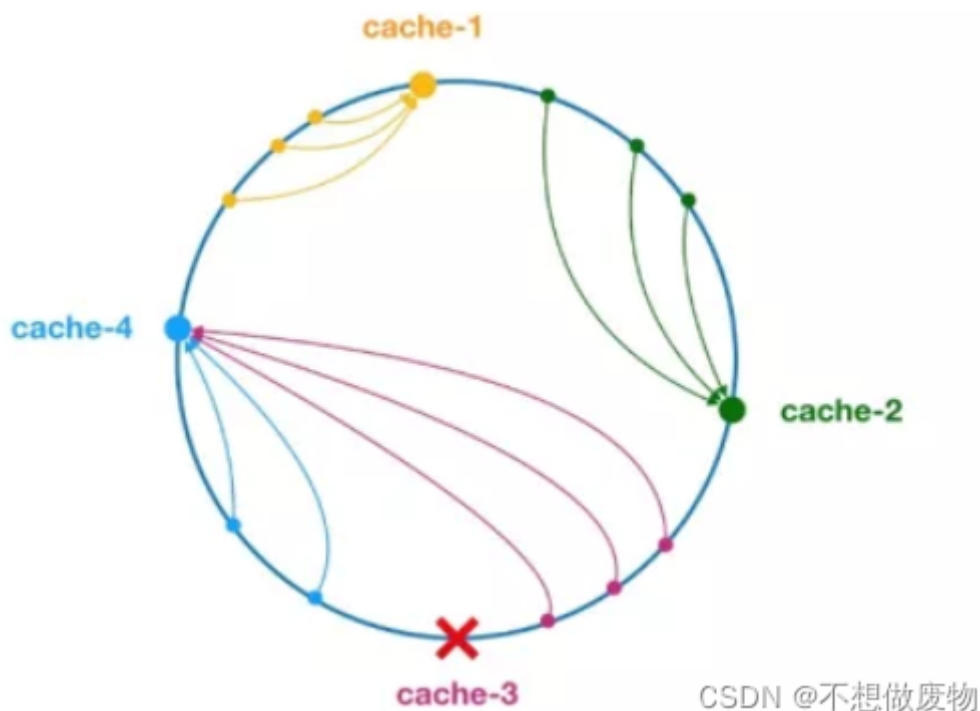
在轮询算法的基础上，为每个服务实例设置一个权重，高性能的机器赋予更高的权重，也可以根据服务实例的当前的负载情况做动态的调整，例如考虑最近5分钟部署服务器的 CPU、内存消耗情况。

哈希/一致性哈希策略

哈希策略

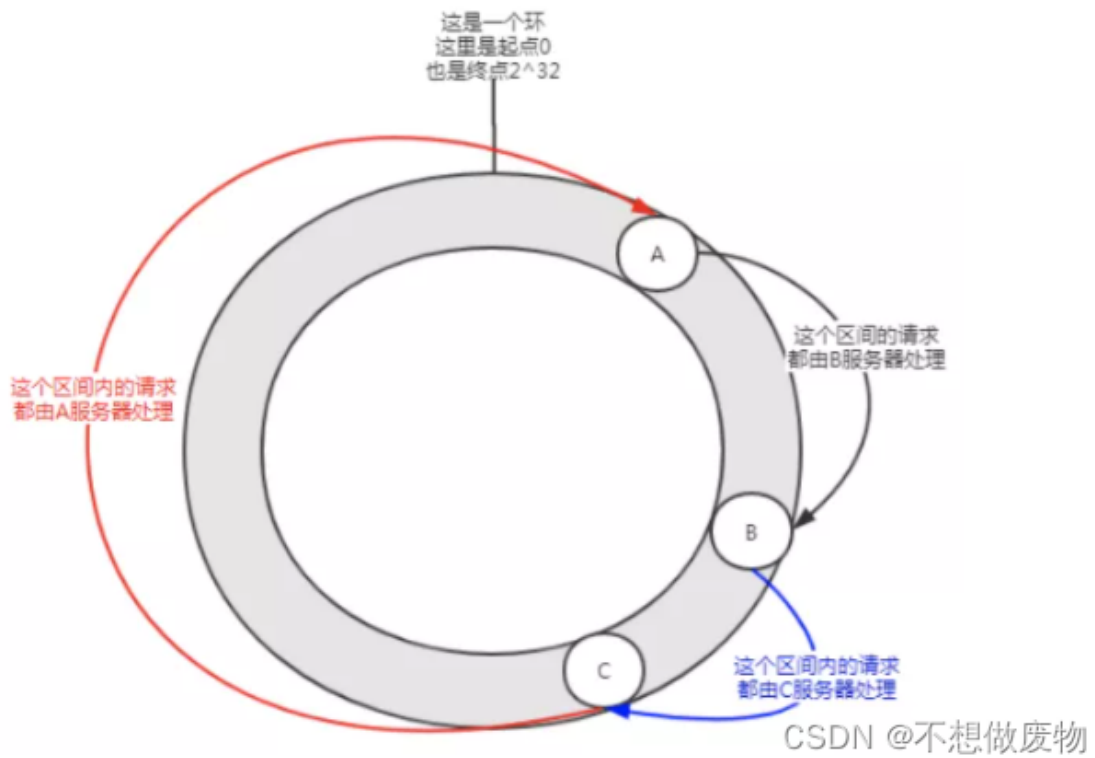
哈希取模。问题是当扩容缩容的时候，需要大量的数据迁移。

一致性哈希策略

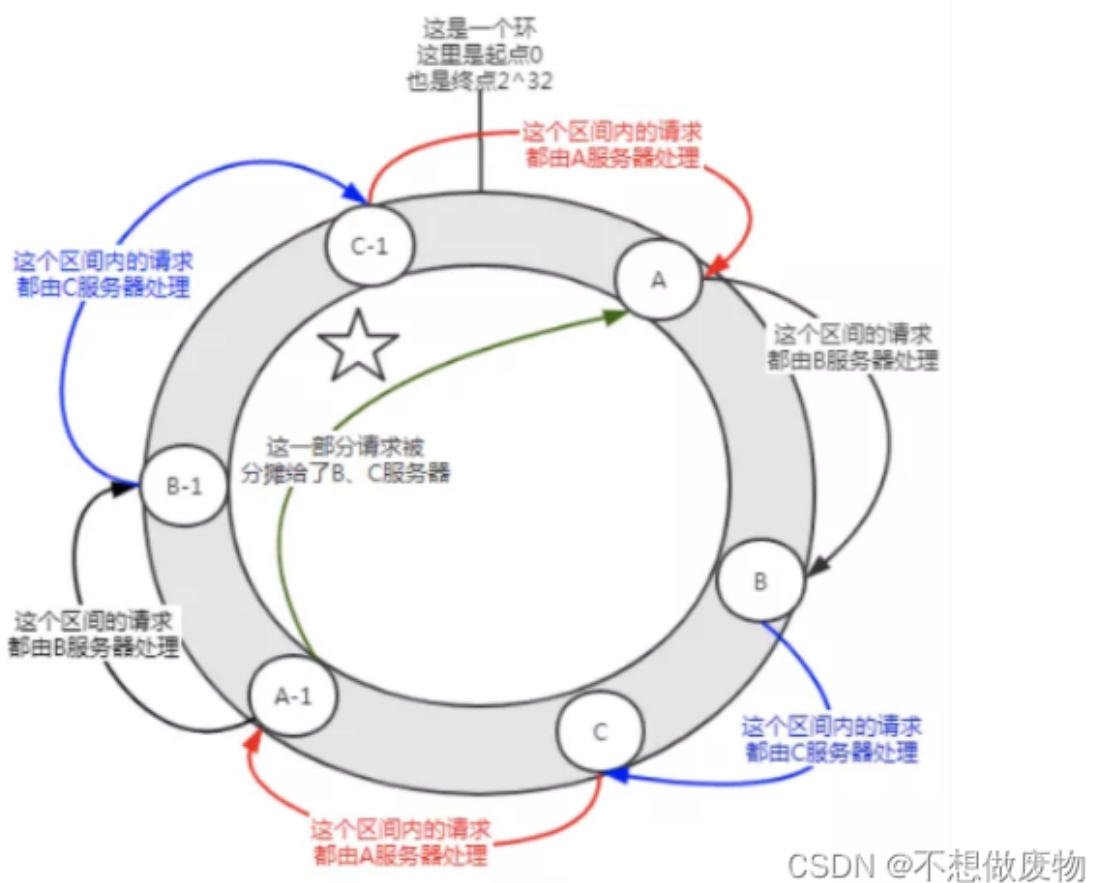


把原来哈希算法点对点的映射转为点对线的映射，它减轻了扩缩容时数据迁移的数量。比如绿色点对应的缓存项将会被存储到 cache-2 节点中，不再像单纯的哈希算法那样，必须映射到对应的点上。还有扩缩容问题，比如由于 cache-3 挂了，原本应该存到该节点中的缓存项最终会存储到 cache-4 节点中。

当然一致性哈希策略也有他的问题。比如数据偏斜问题。即哈希分布不均匀导致大量的请求被发送给同一个服务器。



可以引入**虚拟节点**的方式解决。

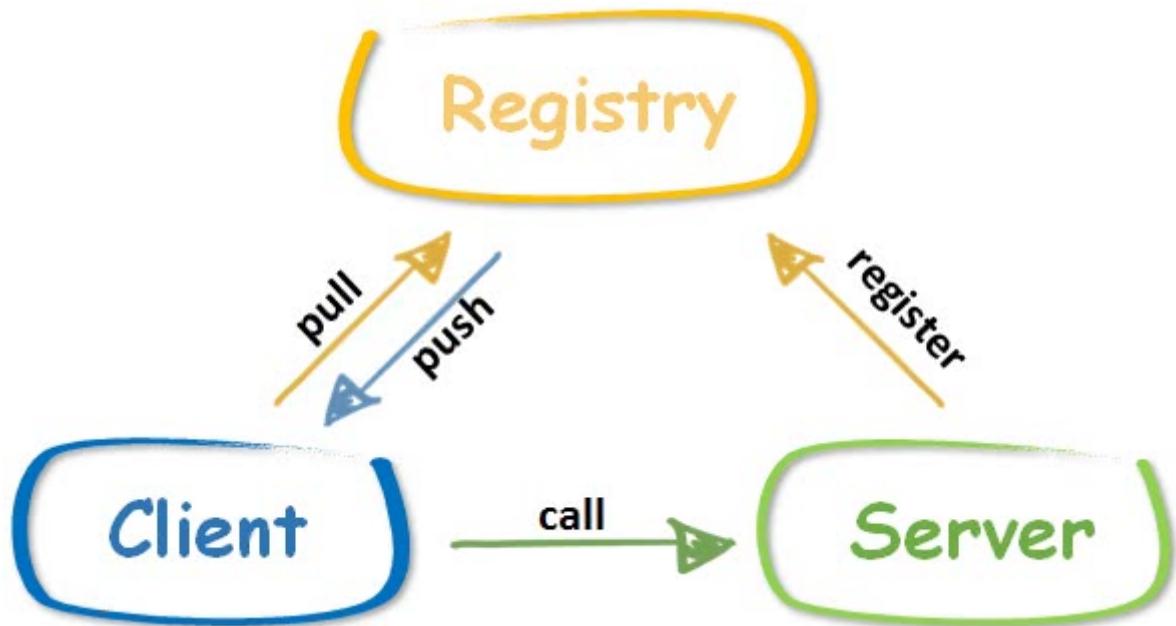


Dubbo负载均衡策略之 一致性哈希

谈谈一致性哈希算法及其 Golang 实现（含负载均衡算法概述）

| 什么是注册中心

你可以把他看成是一个通讯录。它维护了服务器和服务地址的映射关系。服务器会注册到这里，当客户端需要请求服务时，就到这里找到服务器的地址，进行请求。



为什么需要注册中心

在本项目中，如果不用注册中心的话，多个服务器地址硬编码到代码中。不利于维护。

实际项目中，注册中心还要考虑更多的问题，比如：服务注册后，如何被及时发现。服务发现时，如何进行路由。服务宕机后如何及时下线。服务异常时，如何进行降级。