**RPC-Wheel的整体架构图**

Client

Service

ZK Cluster

服务发现

Remote Proxy

服务注册

Local proxy

RPC调用

路由调用

**RPC-Wheel由三个部分组成**

组成一：Client

组成二：Local Proxy

组成三：Global Proxy

组成四：Service

Client的主要职责

1：负责request的序列化

2：与Proxy的通信

3：负责Response的反序列化

Local Proxy的主要职责

1：负载均衡

{

1.1：随机算法(优先级高)

{

优点:算法简单,无状态,可以很好的支持权重

}

1.2：轮训算法(优先级中)

{

效果和随机算法类似,有状态,需要记着上次call的哪台机器,权重算法不太好设计

(预先构造环,随机打乱)

}

1.3：一致性Hash算法(优先级中)

{

支持权重设计,构造静态Hash环

}

1.4：最小负载(优先级低)

{

用于解决服务器之间能力不均的情况,权重算法支持不易,暂不支持权重

}

1.5：需要支持权重

1.5.1：压测引流

1.5.2：灰度发布

}

2：服务的容错

{

2.1：FailOver,失败自动切换其他的服务器,通常用于读操作,但是重试次数多了,延迟会高

2.2：FailFast,快速失败,只发起一次调用,失败立即报错,通常用于”非幂等”的写

2.3：FailSafe,失败安全,出现异常的时候,直接忽略,通常用于审计日志的写入

2.4：FailBack,失败自动恢复,失败后记录请求,后台定时发送

2.5：Forking,并行调用,同时调用多个服务器,有一个成功就返回,用于实时性高的读请求

}

3：服务的限流(单机限流)

{

服务端的限流

3.1：服务级别的限流

3.2：方法级别的限流

客户端的限流

客户端并不知道服务端的情况,所以设置限流不太实用

}

4：服务的熔断

{

域级别的熔断和方法级别的熔断

}

5：服务降级

{

人工降级,由服务方统一降级,至于客户端的降级处理,需要客户端自己捕获异常

}

6：服务的发现

7：服务的路由

{

按方法或者客户端IP/域进行分流

[{"method":"add,del","value":120},{"ip":"xxx","value":120}]

6.1：读写分离

6.2：灰度发布

}

8：负责与Server端的通信

9：子集划分(优先级低)

{

描述:

如果有100个client,100个server,两者以笛卡尔积建立连接,这样会造成海量的连接数量,心跳包数量,所以进行子集划分,每个client随机选择100个连接中的20个,作为自己的子集

缺点:

server间的负载不均衡

}

10：流量摘除

{

通过web页面修改配置中心,然后由配置中心秒级通知LocalProxy

通过调整实例的权重为0,对于不支持权重的”负载均衡算法”也需要特别处理权重为0的情况

}

Global Proxy的主要职责

作为Local Proxy的备份,用于一组Local Proxy的备份,这样Local Proxy宕机以后可以确保Client端正常工作

Server的主要职责

1：注册服务

2：提供RPC服务

3：优雅关闭

{

步骤:

3.1 : 服务端主动取消服务实例的注册(将自己在配置中心下线),proxy不再请求该实例

3.2 : 不再接收新连接(已建立的长连接依然有效)

3.3 : 关闭业务线程池,等待所有已接收的请求处理完毕

3.4 : 关闭Spring Application Context,触发应用自身的清理工作

3.5 : 断开与所有客户端的连接

3.6 : 退出进程

如果到达某个阈值(默认一分钟),进程还没有退出,直接调用kill -9 {pid}停止进程

}

超时与重试：

Local Proxy

ZK Cluster

Server B

200ms

默认10S

Server A

Client

默认400ms

超时的配置:

通过JSON

[{"method":"add","value":120},{"method":"del","value":120},{"value":500}]

add()-120ms，del()-120ms，其它-500ms

**模块的介绍：**

1：rpc-client

客户端,负责与proxy通信

2：rpc-communication

底层的通信模块,默认基于Netty

3：rpc-container

启动server的容器

4：rpc-core

5：rpc-proxy

代理,包含了核心的服务治理相关的功能,负责与Server通信

6：rpc-registry

注册中心

7：rpc-serialization

序列化模块

常见参数配置

1：连接情况

client->local proxy : cpu core / 2

local proxy - > each server : 2

2：线程模型

IO线程池 : cpu core / 2

业务线程池 : 4 \*cpu core ~ 40 \* cpu core