Api -> 机器需要绑定

建立socket

Socket -> api

通信流程

远程调用

serverA serverB 方法

serverA serverB提供了哪些方法 serverA直接调用

1. 服务治理 socket function 哈希映射的
2. 支持多语言
3. 服务发现

Thrift 多语言 thrift

Tars 多语言 tails web log register

Grpc php java c++ 等 protobuffer

Dubbo java

grpc 基于protobuffer序列化工具 http2 协议

.proto 协议的序列化工具 http2 与 http1.x比较

1. 支持二进制协议
2. 多路复用的流 http1.x 一个流对应一个tcp http2 实现一个tcp复用多个流 设置优先级
3. 头部压缩 减少数据包的大小 性能
4. 服务器推送 请求回应 长连接 推送
5. 流量控制 设置流量窗口值

IDL protobuffer

描述远程调用的过程

A 调用 B func1

response = client\_stub.SayHello(helloworld\_pb2.HelloRequest(

name='you',

request\_initiation=request\_timestamp,// 为了超时)

grpc 四种调用方式：

1. 简单调用 一个请求参数 + 一个回应的参数

rpc SayHello (HelloRequest) returns (HelloReply) {}

1. 服务端流调用 一个请求参数 + 多个参数通过stream的回应

rpc sayHello (HelloRequest) returns (stream HelloReply) {}

B:

For {

Stream.send(...)

}

A: for {

Int t = Stream.recv()

}

1. 客户端流调用 多个参数通过一个stream参数 + 一个返回的回应

A: for {

Stream.send(...)

}

B: for {

Stream.recv()

}

4. 双向流调用

Rpc 与 rest 比较 web开发 RESTful

Rest

Re 变现层 status 状态 t传输

Rest 通过url来定位资源 用http动词描述操作

GET 获取

POST 新建或者更新

PUT 更新

DELETE 删除

GET /tasks 获取所有的任务

POST /tasks 创建一个新的任务

GET /tasks/id 获取一个 id 的任务

PUT /tasks/id 更新一个 id的任务

Delete /tasks/id 删除一个id的任务

1. http1.x 明文 json/xml http2 二进制 protobuffer
2. Rest 不需要关注服务端具体实现 面向资源 rpc 面向方法 类名 方法名 参数 返回值
3. HTTP2 tcp / udp/ http

Rest 客户端 与 服务端 cs

Rpc 微服务之间的调用 自己开发 细节 请求回应

survey模式

消息队列 Zeromq 也是解决 服务间调用的问题

1. Sub/pub 监听发布模型 广播
2. Req/rep
3. Push/poll
4. pair

消息队列

1. 解耦 关心核心流程 通知而非处理
2. 最终一致性 两个系统的状态保持一致的
3. 广播
4. 错峰流控 各个服务处理能力 数据库处理能力 通过消息队列操作 写操作 读

异步 （同步）

Rpc 强一致性 call 线程

Rpc 同步阻塞的 请求回应 （异步）callback rpc 在 线程池来解决的

同步 （异步）callback

Qt 单片机 金融 客户端

想从事 1-3

c++ 后台开发：

1. 数据结构、算法以及设计模式

红黑树 nginx linux内核 openresty（nginx + luajit）shared\_dict epoll

比较的方式：

排序 redis 跳表 红黑树

Get delete update o(logn)

Zrange(10,20) 红黑树很难做到

定时器 nginx 红黑树 linux内核 时间轮

一个真实时间片的数组 + 多个循环队列 来实现

10ms 256

1 2 3 256 近期不会过期的事件 数组来实现 %64

多线程环境 为什么不适用 红黑树呢 删除一个节点 无法控制节点移动

保证锁的粒度

排序 （快排， 希尔 归并） 稳定

Sort（快排） 不稳定的

12 ，22.1，22.2，11

11, 12, 22,22

布隆过滤器 （redis击穿） 添加一些经常访问的 key

B/B+ mysql索引 设计模式

责任链 过滤器 观察者（发布订阅）pub 代理 工厂 tars 单例

1. 中间件编程

数据库交互 （mysql redis mongo）

数据库驱动 mysqlclient c 驱动 hiredis c驱动 阻塞的库 连接池的问题

根据当前项目网络框架来实现 协议 redis.get( callback) 不会阻塞网络线程 契合当前框架

Nginx 配置 过滤器 模块 openresty lua web

Zeromq 库

1. 网络io

网络模型 select，poll，epoll

同步与异步

阻塞与非阻塞

Libevent libuv nodejs callback 嵌套回调

Reactor 和 proactor

Epoll io event

内核 callback

1. linux系统
2. 编辑 vi sed awk grep
3. 查看日志 tail multitail
4. 磁盘 dd umount
5. 网络 netstat tcpdump
6. shell脚本
7. 系统 iostat uptime
8. 开源框架
9. 配置 xml json lua
10. 制定协议 protobuffer thrift messagepack flatbuffer
11. 压缩 zlib snappy
12. 加密 xxtea openssl tls
13. 日志 log4cpp boost.log
14. 服务端框架 长连接 skynet Actor模型 nginx lighttpd
15. 协程 libco ntyco
16. 代码工程化

Git svn makefile autoconf cmake

1. 性能测试

测试库 luatest gtest

调试库 gtest boost.test

性能工具 tails web cpucache 流量 错误统计 webbench wrk

1. 源码 nginx redis zeormq
2. 造轮子
3. 集群 zeromq rpc 数据库
4. 性能调优
5. Docker
6. 一线架构分析
7. linux内核 网卡