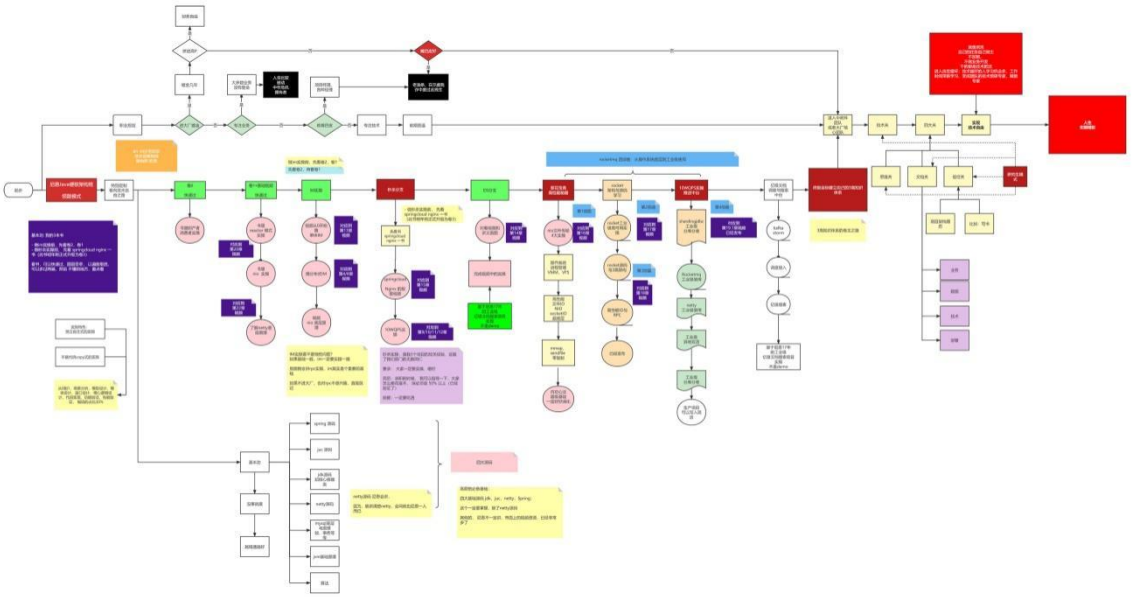
牛逼的职业发展之路

40 岁老架构尼恩用一张图揭秘：Java 工程师的高端职业发展路径，走向食物链顶端的之路

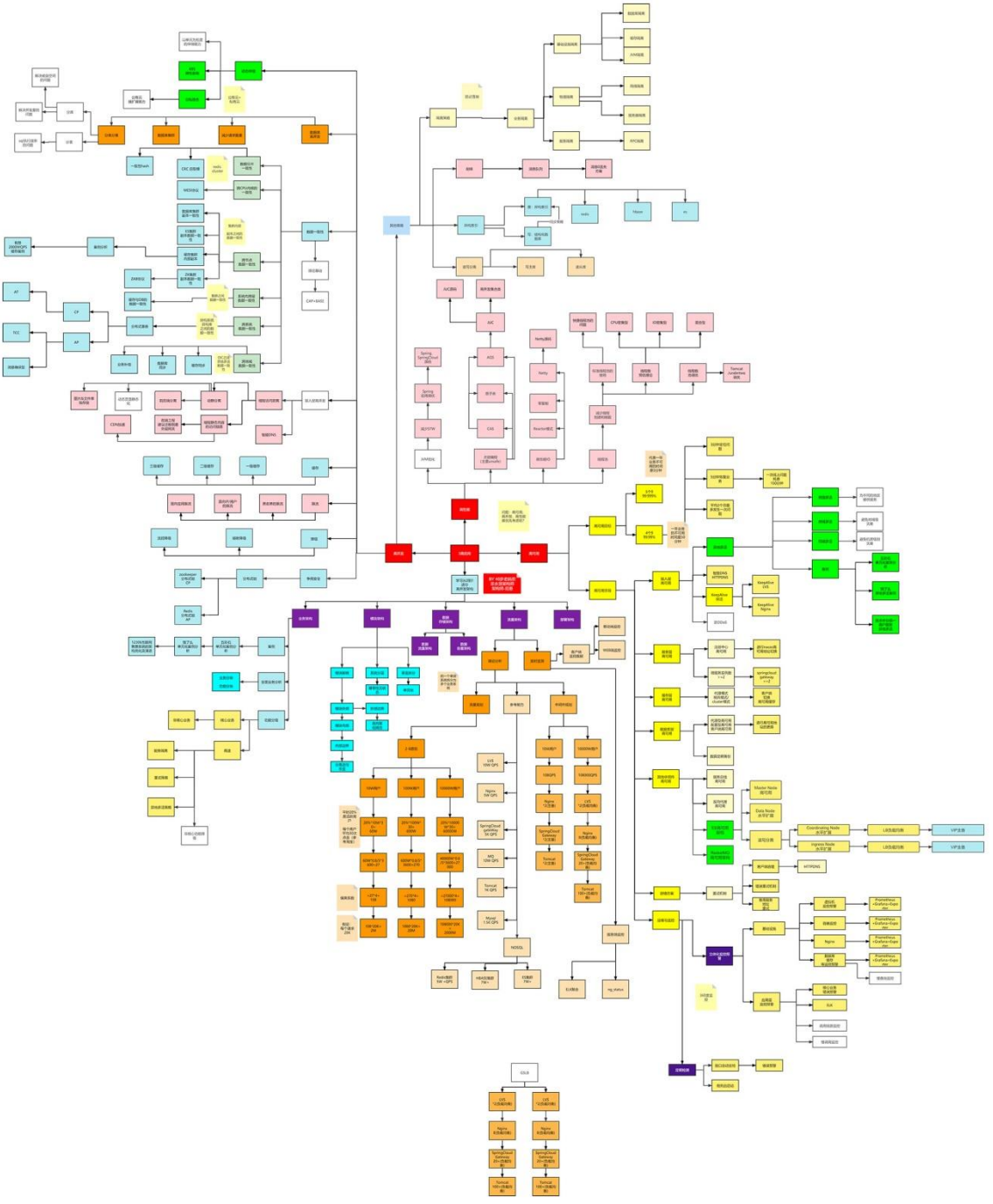
链接：https://www.processon.com/view/link/618a2b62e0b34d73f7eb3cd7



史上最全：价值10W的架构师知识图谱

此图梳理于尼恩的多个 3 高生产项目：多个亿级人民币的大型 SAAS 平台和智慧城市项目

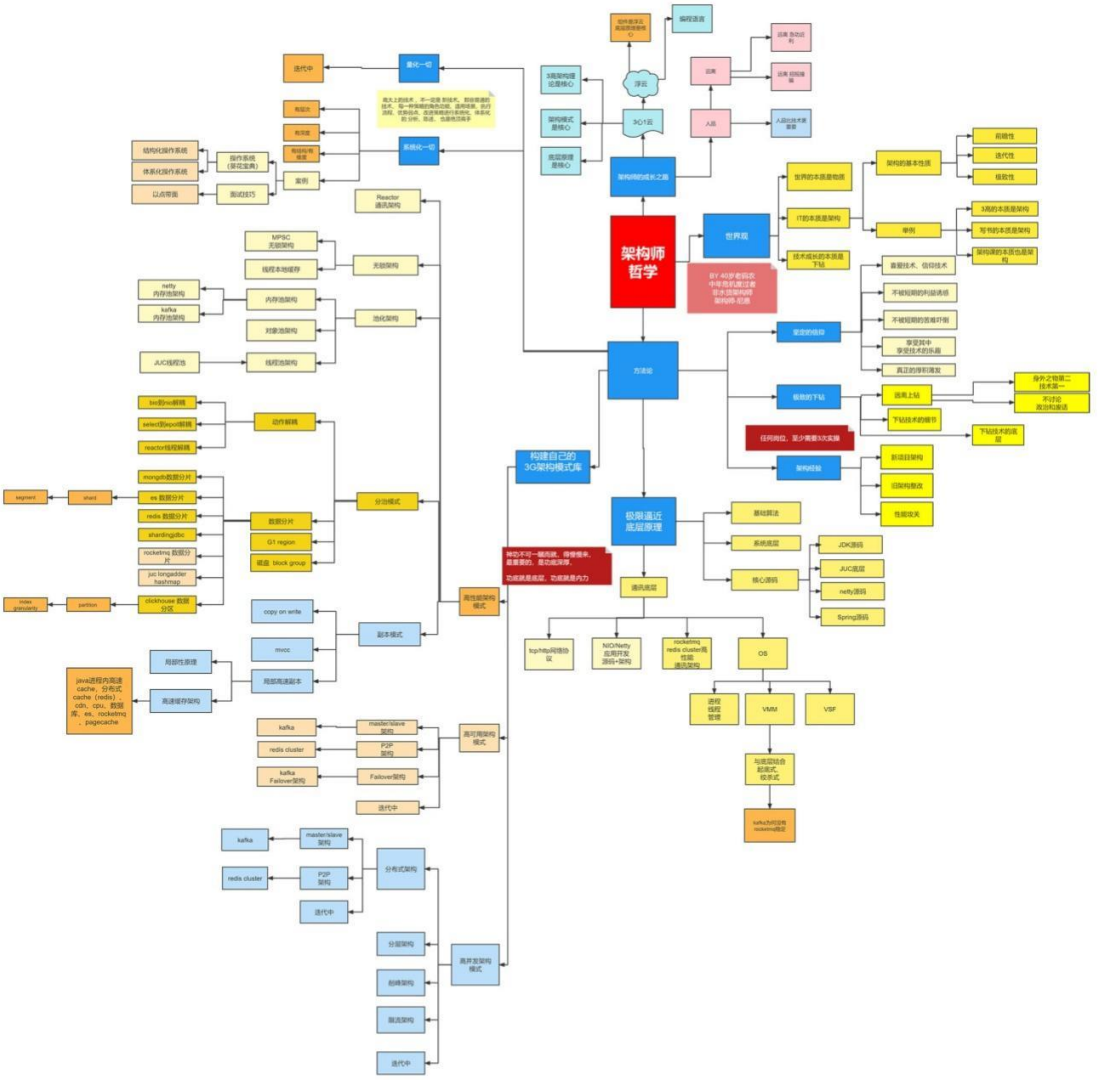
链接：https://www.processon.com/view/link/60fb94216376897l9d246739



牛逼的架构师哲学

40 岁老架构师尼恩对自己的 20 年的开发、架构经验总结

链接：https://www.processon.com/view/link/616f801963768961e9d9aec8



牛逼的3高架构知识宇宙

尼恩 3 高架构知识宇宙，帮助大家穿透 3 高架构，走向技术自由，远离中年危机

链接：https://www.processon.com/view/link/635097d2e0b34d40be778ab4

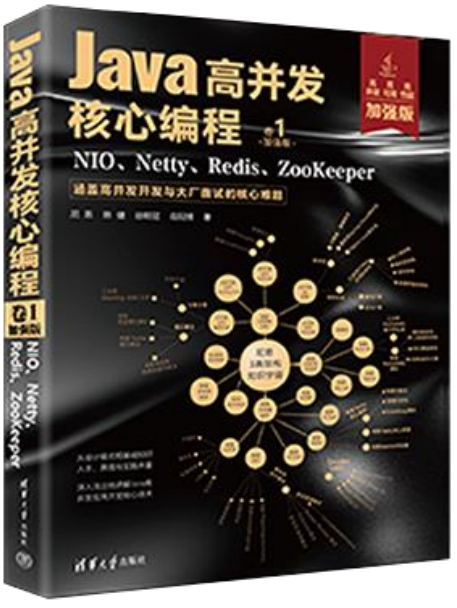
尼恩Java高并发三部曲（卷1加强版）

老版本：《Java 高并发核心编程 卷 1：NIO、Netty、Redis、ZooKeeper》（已经过时，不建议购买）

新版本：《Java 高并发核心编程 卷 1 **加强版**：NIO、Netty、Redis、ZooKeeper》

- 由浅入深地剖析了高并发 IO 的底层原理。
- 图文并茂的介绍了 TCP、HTTP、WebSocket 协议的核心原理。
- 细致深入地揭秘了 Reactor 高性能模式。
- 全面介绍了 Netty 框架，并完成单体 IM、分布式 IM 的实战设计。
- 详尽地介绍了 ZooKeeper、Redis 的使用，以帮助提升高并发、可扩展能力

详情：https://www.cnblogs.com/crazymakercircle/p/16868827.html



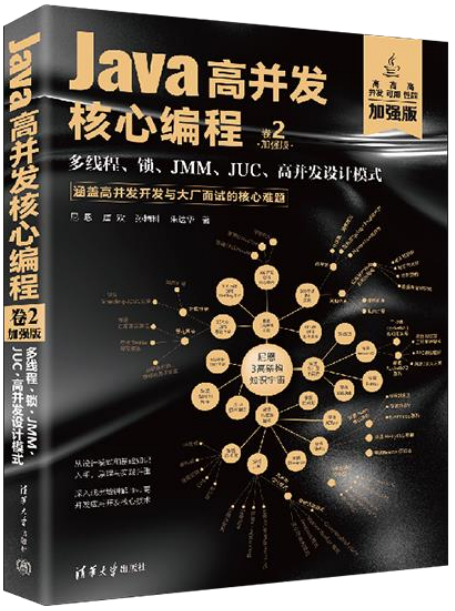
尼恩Java高并发三部曲（卷2加强版）

老版本：《Java 高并发核心编程 卷 2：多线程、锁、JMM、JUC、高并发设计模式》（已经过时，不建议购买）

新版本：《Java 高并发核心编程 卷 2 **加强版**：多线程、锁、JMM、JUC、高并发设计模式》

- 由浅入深地剖析了 Java 多线程、线程池的底层原理。
- 总结了 IO 密集型、CPU 密集型线程池的线程数预估算法。
- 图文并茂的介绍了 Java 内置锁、JUC 显式锁的核心原理。
- 细致深入地揭秘了 JMM 内存模型。
- 全面介绍了 JUC 框架的设计模式与核心原理，并完成其高核心组件的实战介绍。
- 详尽地介绍了高并发设计模式的使用，以帮助提升高并发、可扩展能力

详情参阅：https://www.cnblogs.com/crazymakercircle/p/16868827.html



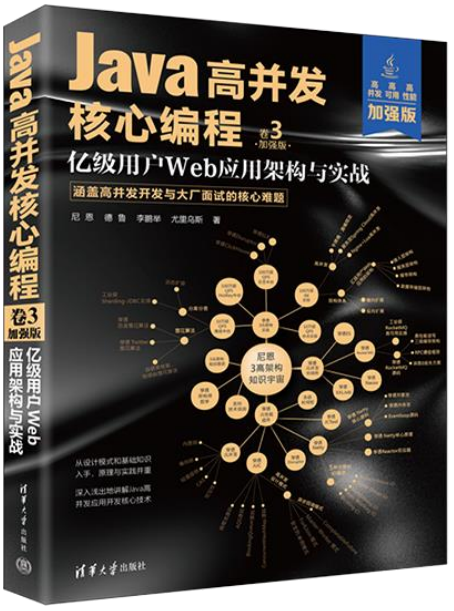
尼恩Java高并发三部曲（卷3加强版）

老版本：《SpringCloud Nginx 高并发核心编程》（已经过时，不建议购买）

新版本：《Java 高并发核心编程 卷 3 **加强版**：亿级用户 Web 应用架构与实战》

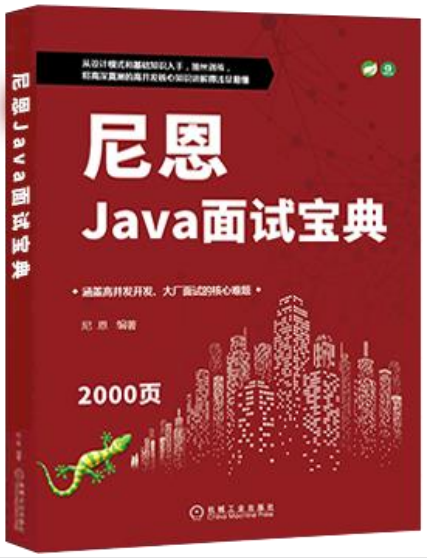
- 在当今的面试场景中，3 高知识是大家面试必备的核心知识，本书基于亿级用户 3 高 Web 应用的架构分析理论，为大家对 3 高架构系统做一个系统化和清晰化的介绍。
- 从 Java 静态代理、动态代理模式入手，抽丝剥茧地解读了 Spring Cloud 全家桶中 RPC 核心原理和执行过程，这是高级 Java 工程师面试必备的基础知识。
- 从Reactor 反应器模式入手，抽丝剥茧地解读了Nginx 核心思想和各配置项的底层知识和原理，这是高级 Java 工程师、架构师面试必备的基础知识。
- 从观察者模式入手，抽丝剥茧地解读了 RxJava、Hystrix 的核心思想和使用方法，这也是高级 Java 工程师、架构师面试必备的基础知识。

详情：https://www.cnblogs.com/crazymakercircle/p/16868827.html



尼恩Java面试宝典

35 个专题（卷王专供+ 史上最全 + 2023 面试必备）
详情：https://www.cnblogs.com/crazymakercircle/p/13917138.html



网盘 › 面试 › 尼恩Java面试宝典

名称

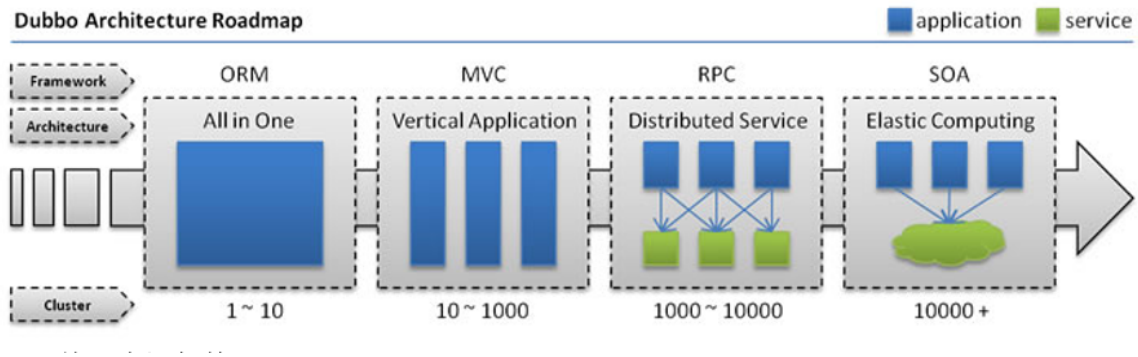
专题01：JVM面试题（卷王专供 + 史上最全 + 2023面试必备）-V10-from-尼恩Java面试宝典-release.pdf
专题02：Java算法面试题（卷王专供 + 史上最全 + 2023面试必备）-V2 -from-Java面试红宝书-release.pdf
专题03：Java基础面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书 -release.pdf
专题04：架构设计面试题（卷王专供+ 史上最全 + 2023面试必备）-V10-from-Java面试红宝书-release.pdf
专题05：Spring面试题_专题06：SpringMVC_专题07：Tomcat面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-尼恩面试宝典-release.pdf
专题08：SpringBoot面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题09：网络协议面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题10：TCPIP协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题11：JUC并发包与容器类（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题12：设计模式面试题 （卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题13：死锁面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题14：Redis 面试题 （卷王专供+ 史上最全 + 2023面试必备）-V5-from-Java面试红宝书-release.pdf
专题15：分布式锁 面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题16：Zookeeper 面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题17：分布式事务面试题（卷王专供 + 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题18：一致性协议 （卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题19：Zab协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题20：Paxos 协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题21：raft 协议（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题22：Linux面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题23：Mysql 面试题（卷王专供+ 史上最全 + 2023面试必备）-V11-from-Java面试红宝书-release.pdf
专题24：SpringCloud 面试题（卷王专供+ 史上最全 + 2023面试必备）-V12-from-Java面试红宝书-release.pdf
专题25：Netty 面试题 （卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题26：消息队列面试题：RabbitMQ、Kafka、RocketMQ（卷王专供+ 史上最全 + 2023面试必备）-V10-from-Java面试红宝书-release.pdf
专题27：内存泄漏 内存溢出（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题28：JVM 内存溢出 实战（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题29：多线程面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题30：HR面试题：过五关斩六将后，小心阴沟翻船！（史上最全、避坑宝典）-V2-from-Java面试红宝书-release.pdf
专题31：Hash连环炮面试题（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题32：大厂面试的基本流程和面试准备（阿里、腾讯、网易、京东、头条.....）-V2-from-Java面试红宝书-release.pdf
专题33：BST、AVL、RBT红黑树、三大核心数据结构（卷王专供+ 史上最全 + 2023面试必备）-V2-from-Java面试红宝书-release.pdf
专题34：Elasticsearch面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-Java面试红宝书-release.pdf
专题35：Mybatis面试题（卷王专供+ 史上最全 + 2023面试必备）-V3-from-尼恩Java面试宝典-release.pdf

1、为什么需要 Dubbo?

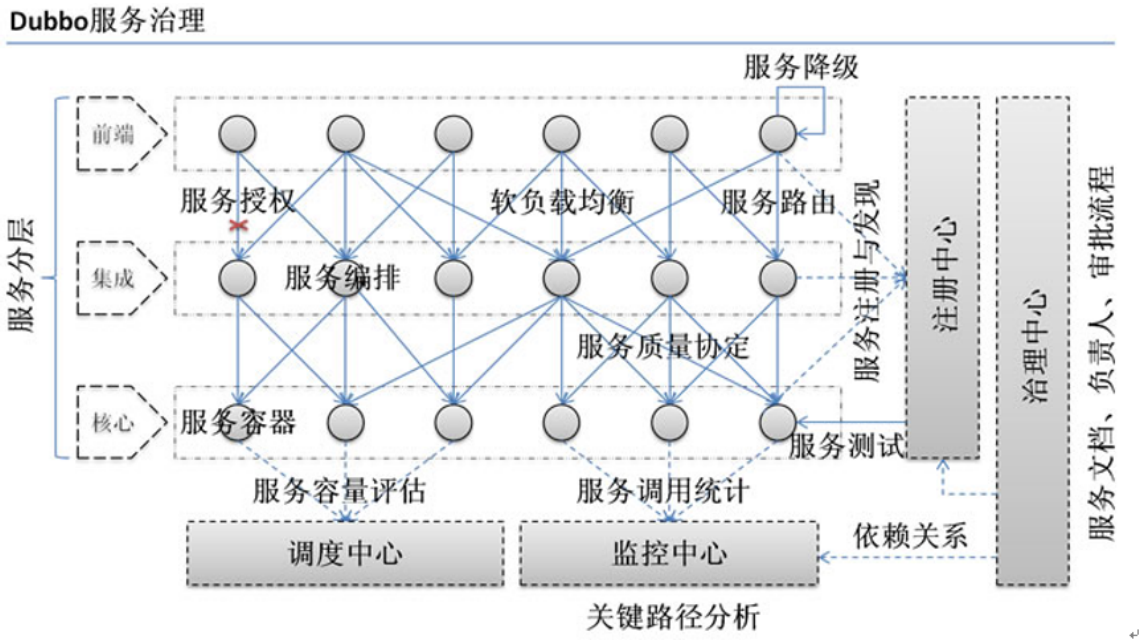
因为是阿里开源项目，国内很多互联网公司都在用，已经经过很多线上考验。内部使用了 Netty、Zookeeper，保证了高性能高可用性。

使用 Dubbo 可以将核心业务抽取出来，作为独立的服务，逐渐形成稳定的服务中心，可用于提高业务复用灵活扩展，使前端应用能更快速的响应多变的市场需求。

下面这张图可以很清楚的诠释，最重要的一点是，分布式架构可以承受更大规模的并发流量。



下面是 Dubbo 的服务治理图。



2、Dubbo 的主要应用场景?

通常有四点，如下：

- RPC 分布式服务，拆分应用进行服务化，提高开发效率，调优性能，节省竞争资源配置管理，解决服务的地址信息剧增，配置困难的问题
- 服务依赖，解决服务间依赖关系错踪复杂的问题
- 服务扩容，解决随着访问量的不断增大，动态扩展服务提供方的机器的问题

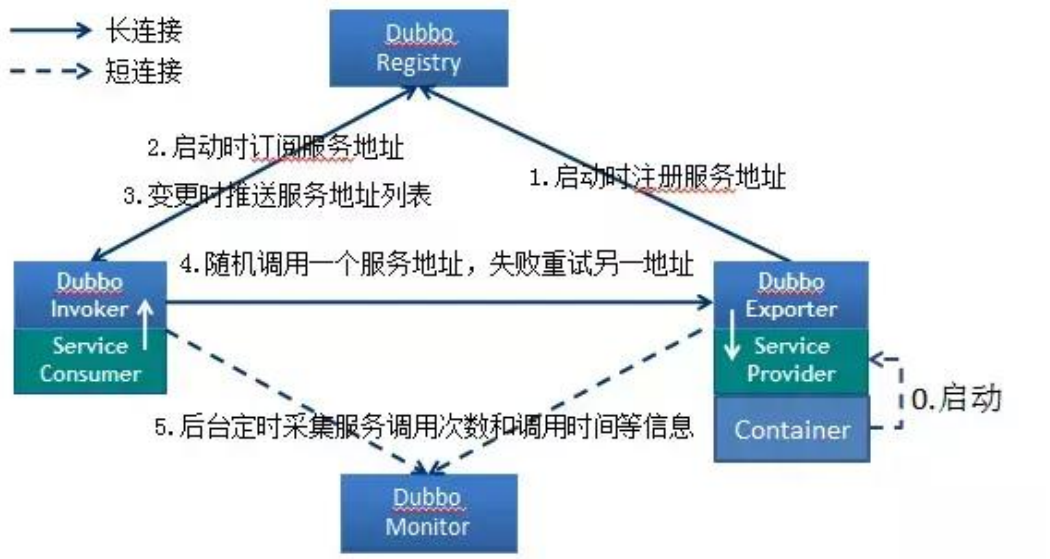
3、Dubbo 的核心功能?

Dubbo主要就是如下 3 个核心功能：

（1）Remoting：网络通信框架，提供对多种 NIO 框架抽象封装，包括“同步转异步”和“请求-响应”模式的信息交换方式。

（2）Cluster：服务框架，提供基于接口方法的透明远程过程调用，包括多协议支持，以及软负载均衡，失败容错，地址路由，动态配置等集群支持。

（3）Registry：服务注册，基于注册中心目录服务，使服务消费方能动态的查找服务提供方，使地址透明，使服务提供方可以平滑增加或减少机器。



4、Dubbo 服务注册与发现的流程?

1、流程说明：

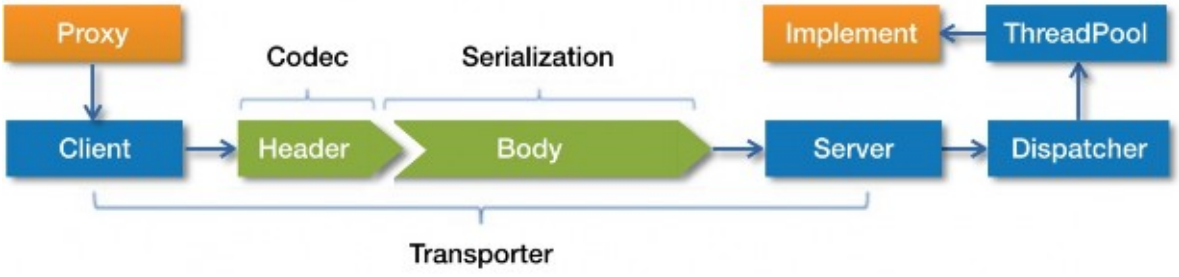
- Provider(提供者)绑定指定端口并启动服务
- 指供者连接注册中心，并发本机IP、端口、应用信息和提供服务信息发送至注册中心存储
- Consumer(消费者），连接注册中心，并发送应用信息、所求服务信息至注册中心
- 注册中心根据 消费 者所求服务信息匹配对应的提供者列表发送至Consumer 应用缓存。
- Consumer 在发起远程调用时基于缓存的消费者列表择其一发起调用。
- Provider 状态变更会实时通知注册中心、在由注册中心实时推送至Consumer

2、设计的原因：

- Consumer 与Provider 解偶，双方都可以横向增减节点数。
- 注册中心对本身可做对等集群，可动态增减节点，并且任意一台宕掉后，将自动切换到另一台去中心化，双方不直接依懒注册中心，即使注册中心全部宕机短时间内也不会影响服务的调用服务提供者无状态，任意一台宕掉后，不影响使用

5、Dubbo 的服务调用流程?

下图为dubbo官方的一张调用图



1) 从client到server经历了编码，序列化，反序列化，解码 的正常网络调用流程, 在nettyServer中处理

2) client采用代理的机制

3) server处理请求的方式通常为分发请求到线程池，同步阻塞或异步非阻塞返回结果

6、Dubbo 支持哪些协议，每种协议的应用场景、优缺点?

协议名程	传输	序列化	连接	使用场景
dubbo默认	mina、netty、grizzy	dubbo、hessian2(默认) 、java、json	dubbo缺省采用单一长连接和NIO异步通讯	1.传入传出参数数据包较小 2. 消费者 比提供者多 3. 常规远程服务方法调用 4.不适合传送大数据量的服务，比如文件、传视务
rni	Java RMI	Java 标准序列化	连接个数: 多连接 连接方式: 短连接 传输协议: TCP/IP 传输方式: BIO	1.常规RFC调用 2.与原RMI客户端互操作 3.可传文件 4.不支持防火墙穿透
hessian	Servlet容器	hessian二进制序列化	连接个数:多连接 连接方式: 短连接 传输协议: HTTP	1. 提供者比消费者多 2.可传文件 3.跨语言传输
http	Servlet容器	表单序列化	连接个数:多连接 连接方式: 短连接 传输协议: HTTP 传输方式:同步传输	1.提供者多余消费者 2.数据包混合
webservice	HTTP	SOAP文件序列化	连接个数: 多连接 连接方式: 短连接 传输协议: HTTP	1.系统集成 2.跨语言调用
thrift	rift RPC实现集成，并在基础上修改了报文头		长连接、NIO异步传输	

还有三种，混个眼熟就行：Memcached 协议、Redis 协议、Rest 协议。

上图基本上把序列化的方式也罗列出来了。

详细请参考：Dubbo 官网：http://dubbo.apache.org/zh-cn/docs/user/references/protocol/dubbo.html

7、Dubbo 有些哪些注册中心?

推荐使用 Zookeeper 作为注册中心，还有 Redis、Multicast、Simple 注册中心，但不推荐。

8、Dubbo 如何实现服务治理?

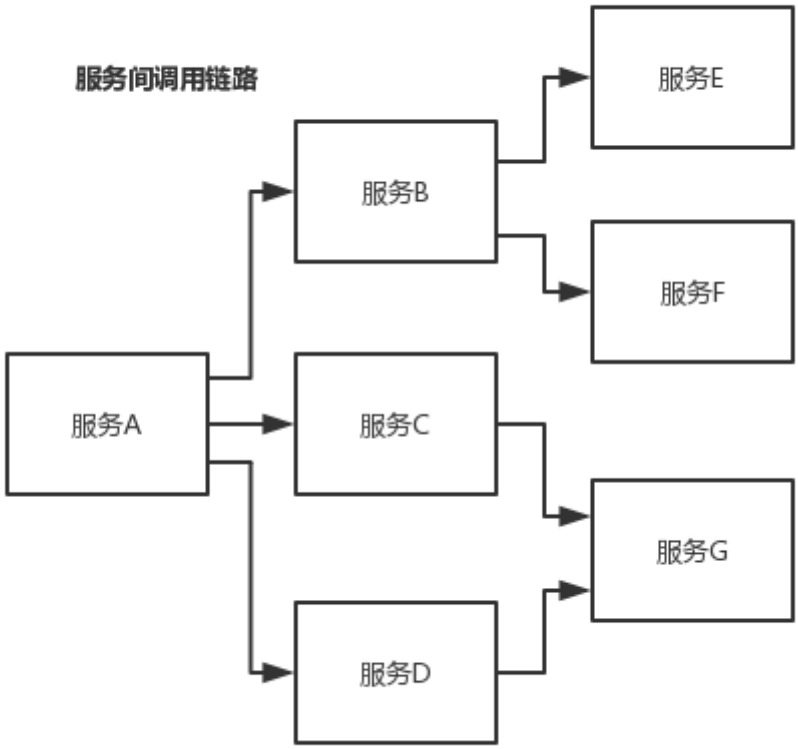
Dubbo服务治理，我们可以分成3个点：

1.调用链路自动生成
一个大型的分布式系统，或者说是用现在流行的微服务架构来说吧，分布式系统由大量的服务组 成。

那么这些服务之间互相是如何调用的？调用链路是啥？

说实话，几乎到后面没人搞的清楚了，因为服务实在太多了，可能几百个甚至几千个服务。

那就需要基于 dubbo 做的分布式系统中，对各个服务之间的调用自动记录下来，然后自动将各个服务之间的依赖关系和调用链路生成出来，做成一张图，显示出来，大家才可以看到对吧。



2.服务访问压力以及时长统计

需要自动统计各个接口和服务之间的调用次数以及访问延时，而且要分成两个级别。

一个级别是接口粒度，就是每个服务的每个接口每天被调用多少次，TP50/TP90/TP99，三个档次的请求延时分别是多少；

- TP50：表示满足百分之五十的网络请求所需的最低耗时。
- TP90：表示满足百分之九十的网络请求所需的最低耗时。
- TP99：表示满足百分之九十九的网络请求所需的最低耗时。

举个例子：有四次请求耗时分别为：

10ms，1000ms，100ms，2ms

那么我们可以这样计算TP99：4次请求中，99%的请求数为4*0.99，进位取整也就是4次，满足这全部4次请求的的最低耗时为1000ms，也就是TP99的答案是1000ms。

第二个级别是从源头入口开始，一个完整的请求链路经过几十个服务之后，完成一次请求，每天全链路走多少次，全链路请求延时的 TP50/TP90/TP99，分别是多少。

这些东西都搞定了之后，后面才可以来看当前系统的压力主要在哪里，如何来扩容和优化啊。

3.其它

服务分层（避免循环依赖）调用链路失败监控和报警 服务鉴权

每个服务的可用性的监控（接口调用成功率？几个 9？99.99%，99.9%，99%）

9、Dubbo 的注册中心集群挂掉，如何正常消费?

可以的，消费者在启动时，消费者会从zk拉取注册的生产者的地址接口等数据，缓存在本地。

每次调用时，按照本地存储的地址进行调用。

消费者本地有一个生产者的列表，他会按照列表继续工作，倒是无法从注册中心去同步最新的服务列表，短期的注册中心挂掉是不要紧的，但一定要尽快修复。

挂掉是不要紧的，但前提是你没有增加新的服务（节点没有变动），如果你要调用新的服务，则是不能办到的。

10、Dubbo 集群提供了哪些负载均衡策略?

负载均衡策略	说明
Random LoadBalance	随机，按权重设置随机概率(默认)
RoundRobin LoadBalance	轮询，按公约后的权重设置轮询比率
LeastActive LoadBalance	最少活跃调用数，相同活跃数的随机
ConsistentHash LoadBalance	一致性 Hash，相同参数的请求总是发到同一提供者

Dubbo 的集群容错方案有哪些?

集群容错方案	说明
Failover Cluster	失败自动切换，自动重试其它服务器(默认)
Failfast Cluster	快速失败，立即报错，只发起一次调用
Failsafe Cluster	失败安全，出现异常时，直接忽略
Failback Cluster	失败自动恢复，记录失败请求，定时重发
Forking Cluster	并行调用多个服务器，只要一个成功即返回
Broadcast Cluster	广播逐个调用所有提供者，任意一个报错则报错

11、Dubbo 支持哪些序列化方式?

在 Dubbo RPC 中，同时支持多种序列化方式：

1.Dubbo 序列化：阿里尚未开发成熟的高效 Java 序列化实现，阿里不建议在生产环境使用它

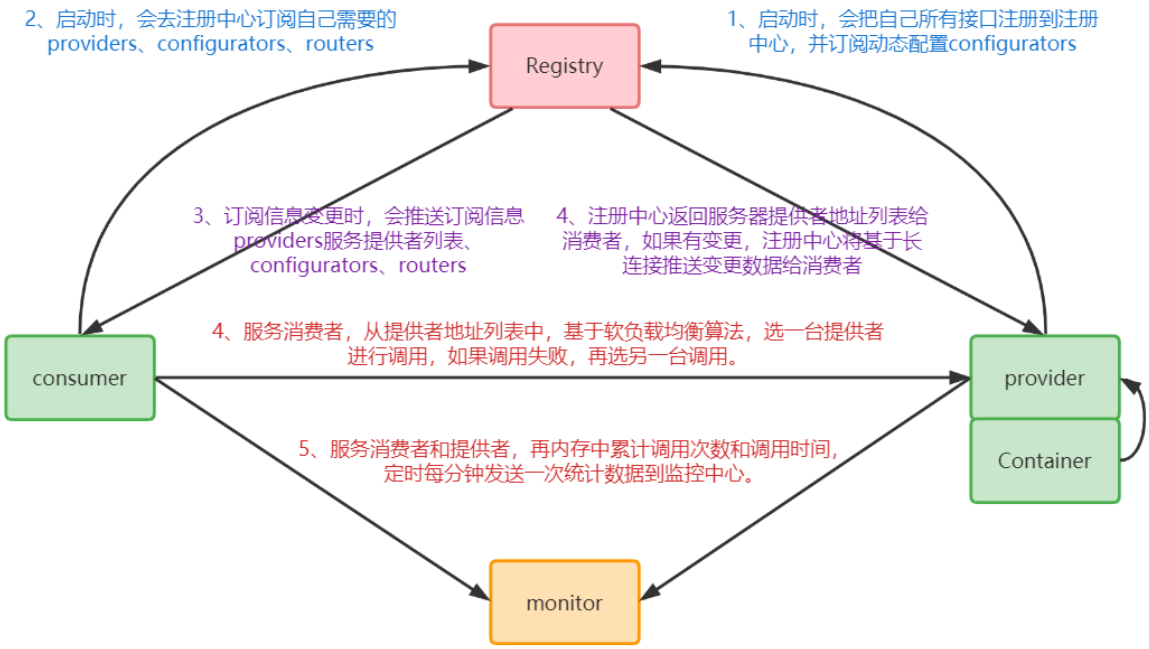
2.Hessian2 序列化：Hessian 是一种跨语言的高效二进制序列化方式。但这里实际不是原生的 Hessian2 序列化，而是阿里修改过的Hessian Lite，它是 Dubbo RPC 默认启用的序列化方式

3.Json 序列化：目前有两种实现，一种是采用的阿里的 Fastjson 库，另一种是采用 Dubbo中自己实现的简单 Json库，但其实现都不是特别成熟，而且 Json 这种文本序列化性能一般不如上面两种二进制序列化。

4.Java 序列化：主要是采用 JDK 自带的 Java 序列化实现，性能很不理想。

12、说说一次 Dubbo 服务请求流程?

基本工作流程：

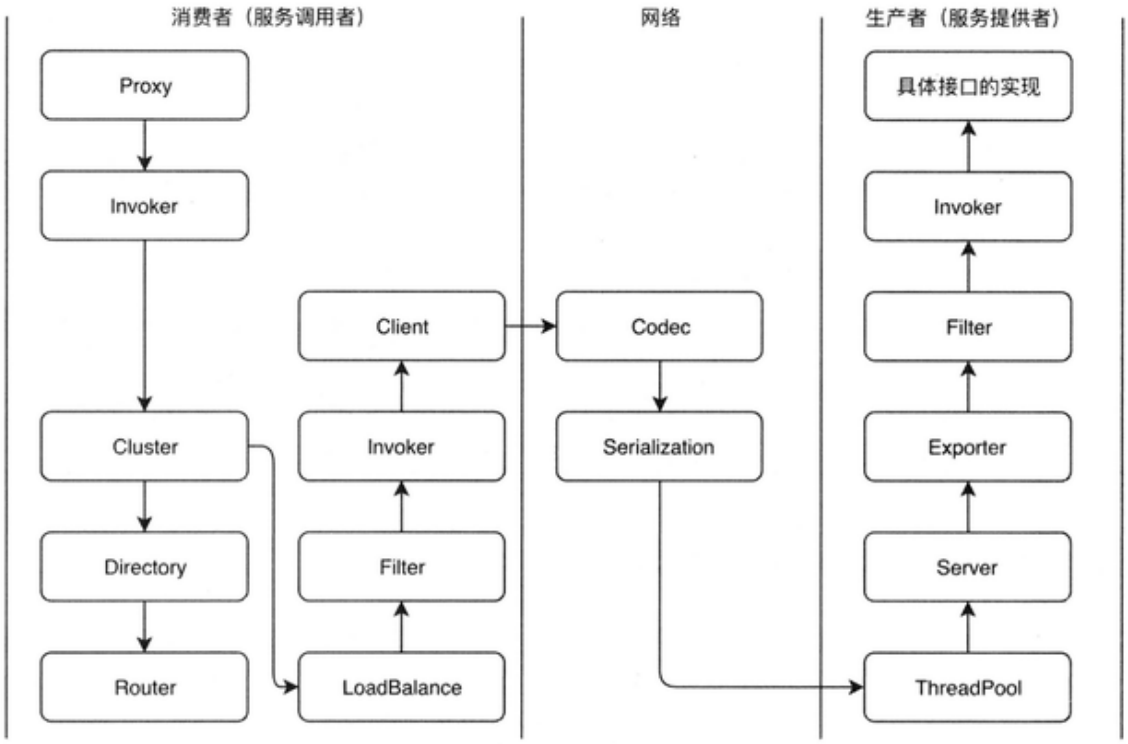


上图中角色说明：

节点	角色明
Provider	暴露服务的服务提供方
Consumer	调用远程服务的服务消费方
Registry	服务注册与发现的注册中心
Monitor	统计服务的调用次数和调用时间的监控中心
Container	服务运行容器

13.能说下Dubbo的总体的调用过程吗?

调用过程图：



1.Proxy持有一个Invoker对象，**使用Invoker调用**

2.之后通过**Cluster进行负载容错**，失败重试

3.调用Directory**获取远程服务的Invoker**列表

4.负载均衡用户**配置了路由规则**，则根据路由规则过滤获取到的Invoker列表用户没**有配置路由规则或配置路由后还有很多节点**，则使用LoadBalance方法做负载均衡，选用一个可以调用的Invoker

5.**经过一个一个过滤器链**，通常是处理上下文、限流、计数等。

6.会**使用Client做数据传输**

7.**私有化协议的构造**(Codec)

8.进行**序列化**

9.服务端收到这个Request请求，将其**分配到ThreadPool**中进行处理

10.**Server来处理这些Request**

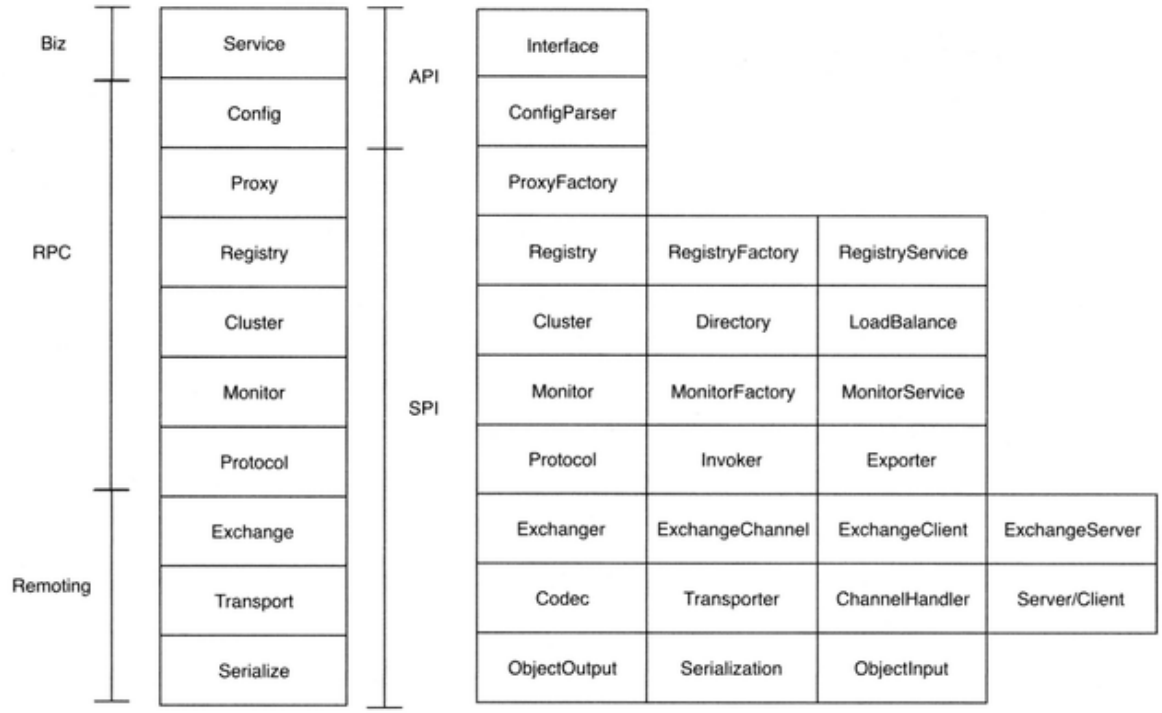
11.根据**请求查找对应的Exporter**

12.之后**经过**一个服务提供者端的**过滤器链**

13.然后找到接口实现并**真正的调用**，将请求结果返回

13.说说Dubbo的分层?

分层图：



从大的范围来说，dubbo分为三层

business业务逻辑层由我们自己来提供，接口和实现还有一些配置信息

RPC层就是真正的RPC调用的核心层，封装整个RPC的调用过程、负载均衡、集群容错、代理

remoting则是对网络传输协议和数据转换的封装。

从API、SPI角度来说，dubbo分为2层

Service和Config两层可以认为是**API**层，主要提供给**API使用者**，使用者只需要配置和完成业务代码就可以了。

后面所有的层级是**SPI**层，主要提供给扩展者使用主要是用来做**Dubbo的二次开发**扩展功能。

再划分到更细的层面，就是图中的10层模式。

13、说说 Dubbo 工作原理

工作原理分 10 层：

第一层：service 层，接口层，给服务提供者和消费者来实现的（留给开发人员来实现）；

第二层：config 层，配置层，主要是对 Dubbo 进行各种配置的，Dubbo 相关配置；

第三层：proxy 层，服务代理层，透明生成客户端的 stub 和服务单的 skeleton，调用的是接口，实现类没有，所以得生成代理，代理之间再进行网络通讯、负责均衡等；

第四层：registry 层，服务注册层，负责服务的注册与发现；

第五层：cluster 层，集群层，封装多个服务提供者的路由以及负载均衡，将多个实例组合成一个服务；

第六层：monitor 层，监控层，对 rpc 接口的调用次数和调用时间进行监控；第七层：protocol 层，远程调用层，封装 rpc 调用；

第八层：exchange 层，信息交换层，封装请求响应模式，同步转异步；

第九层：transport 层，网络传输层，抽象 mina 和 netty 为统一接口；

第十层：serialize 层，数据序列化层。

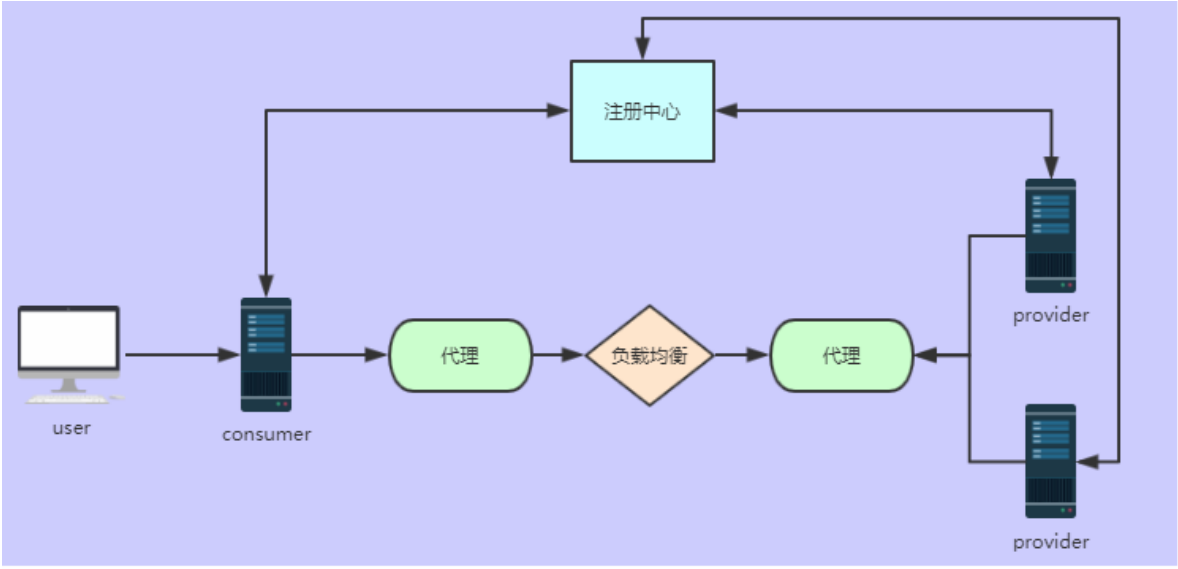
这是个很坑爹的面试题，但是很多面试官又喜欢问，你真的要背么？你能背那还是不错的，我建议不要背，你就想想 Dubbo 服务调用过程中应该会涉及到哪些技术，把这些技术串起来就 OK 了。

14、注册中心挂了，consumer 还能不能调用 provider?

可以。

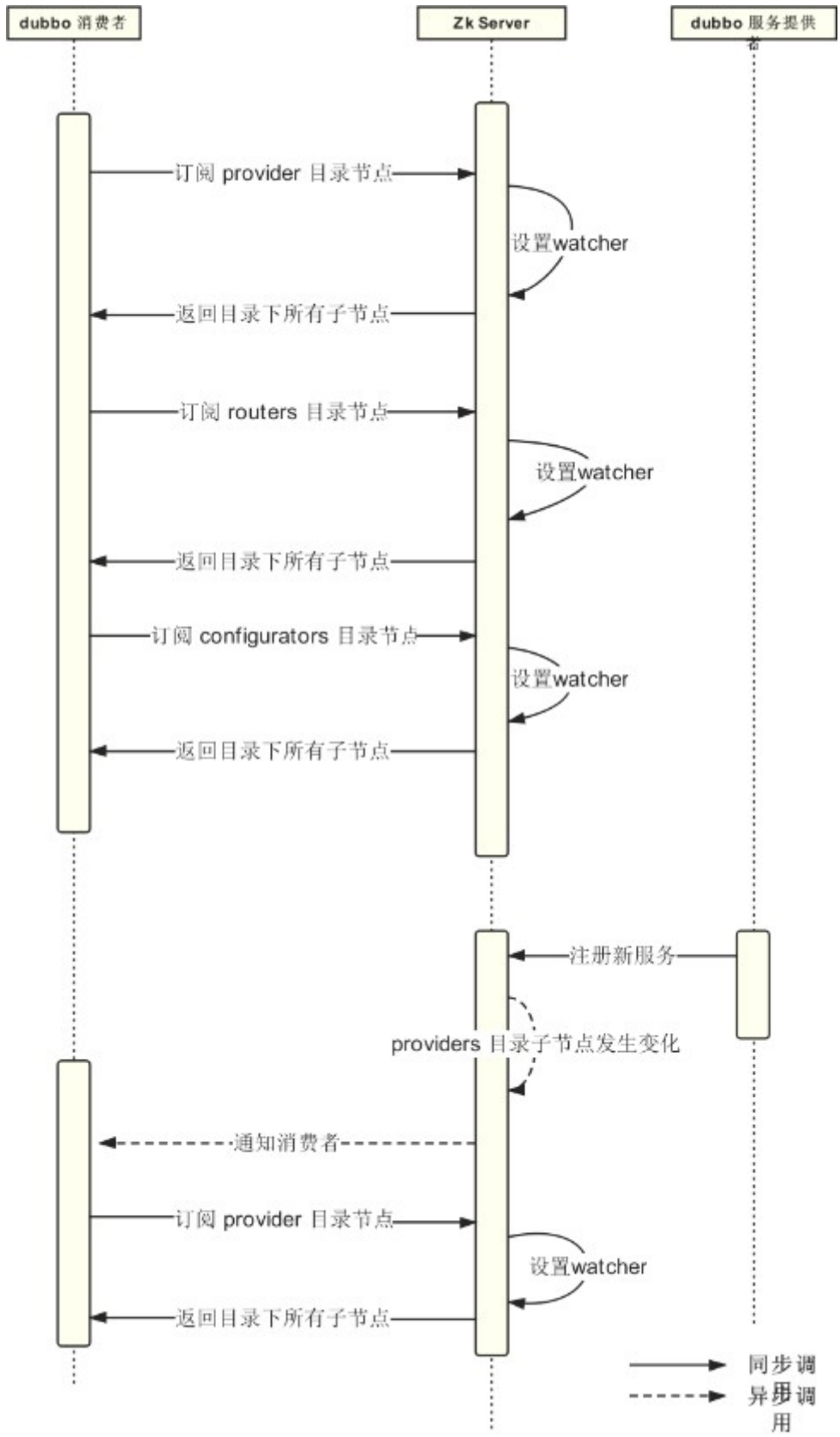
因为刚开始初始化的时候，consumer 会将需要的所有提供者的地址等信息拉取到本地缓存，所以注册中心挂了可以继续通信。

但是 provider 挂了，那就没法调用了。



PS：consumer 本地缓存服务列表。

15、怎么实现动态感知服务下线的呢?



服务订阅通常有 pull 和 push 两种方式：

- pull 模式需要客户端定时向注册中心拉取配置；
- push 模式采用注册中心主动推送数据给客户端。

Dubbo ZooKeeper 注册中心采用是事件通知与客户端拉取方式。

服务第一次订阅的时候将会拉取对应目录下全量数据，然后在订阅的节点注册一个 watcher。

一旦目录节点下发生任何数据变化， ZooKeeper 将会通过 watcher 通知客户端。

客户端接到通知，将会重新拉取该目录下全量数据，并重新注册 watcher。

利用这个模式，Dubbo 服务就可以做到服务的动态发现。

注意：ZooKeeper 提供了“心跳检测”功能，它会定时向各个服务提供者发送一个请求（实际上建立的是一个 socket 长连接），如果长期没有响应，服务中心就认为该服务提供者已经“挂了”，并将其剔除。

16、服务提供者没挂，但在注册中心里看不到?

首先，确认服务提供者是否连接了正确的注册中心，不只是检查配置中的注册中心地址，而且要检查实际的网络连接。

其次，看服务提供者是否非常繁忙，比如压力测试，以至于没有CPU片段向注册中心发送心跳，这种情况减小压力将自动恢复。

17、说说Dubbo的优先级配置

配置优先级别

1.以timeout为例，显示了配置的查找顺序，其他retries，loadbalance等类似。

(1) 方法级优先，接口级次之，全局配置在次之

(2) 如果级别一样，则消费方优先，提供方次之

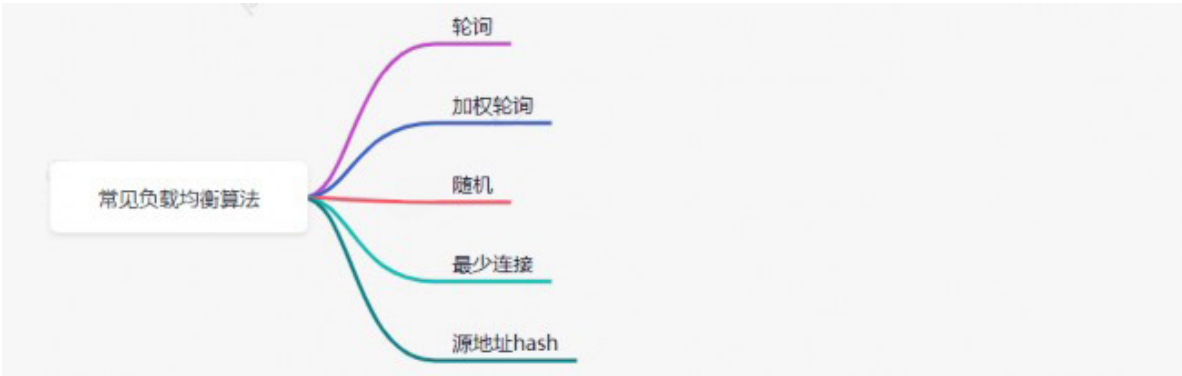
(3) 其中，服务提供方配置，通过URL经由注册中心传递给消费方

2.建议由服务提供方设置超时，因为一个方法需要执行多长时间，服务提供方更清楚，如果一个消费方同时引用多个服务，就不需要关心每个服务的超时设置。

18、负载平衡的意义什么?

在计算中，负载平衡可以改善跨计算机，计算机集群，网络链接，中央处理单元或磁盘驱动器等多种计算资源的工作负载分布。负载平衡旨在优化资源使用，最大化吞吐量，最小化响应时间并避免任何单一资源 的过载。使用多个组件进行负载平衡而不是单个组件可能会通过冗余来提高可靠性和可用性。负载平衡通常涉及专用软件或硬件，例如多层交换机或域名系统服务器进程。

19、常见负载均衡算法有哪些?



20、你知道哪些限流算法?

限流算法有四种常见算法：

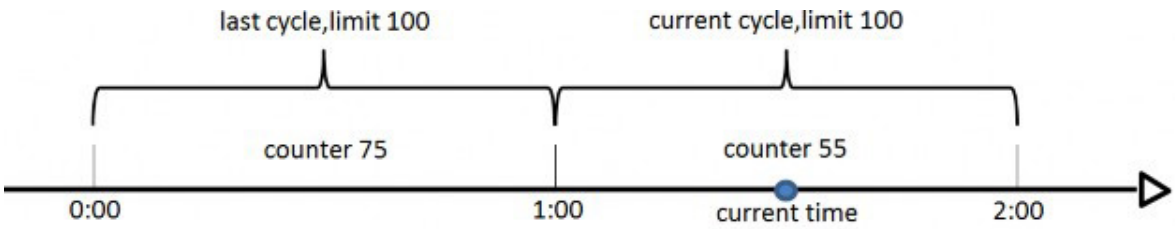
- 计数器算法（固定窗口）滑
- 动窗口
- 漏桶算法
- 令牌桶算法

21、说说什么是计数器（固定窗口）算法

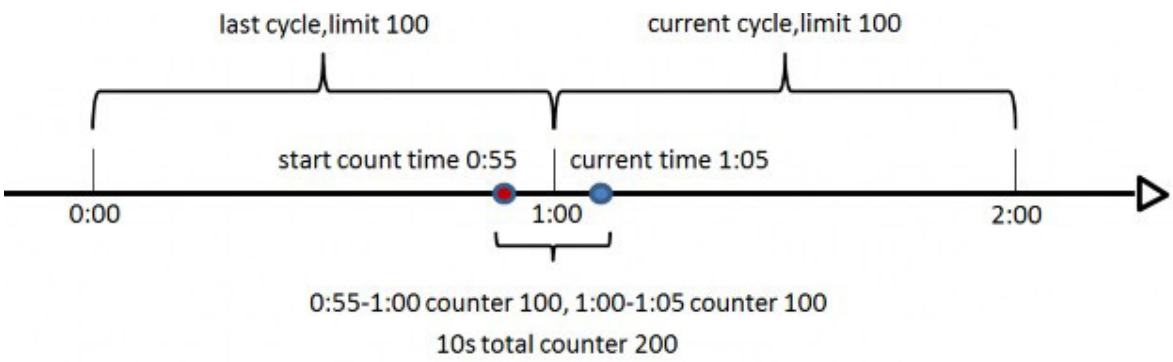
计数器算法是使用计数器在周期内累加访问次数，当达到设定的限流值时，触发限流策略。

下一个周期开始时，进行清零，重新计数。

此算法在单机还是分布式环境下实现都非常简单，使用redis的incr原子自增性和线程安全即可轻松实现。



这个算法通常用于QPS限流和统计总访问量，对于秒级以上的时间周期来说，会存在一个非常严重的问题，那就是临界问题，如下图：



假设1min内服务器的负载能力为100，因此一个周期的访问量限制在100，然而在第一个周期的最后5秒和下一个周期的开始5秒时间段内，分别涌入100的访问量，虽然没有超过每个周期的限制 量，但是整体上10秒内已达到200的访问量，已远远超过服务器的负载能力，

由此可见，计数器算法方式限流对于周期比较长的限流，存在很大的弊端。

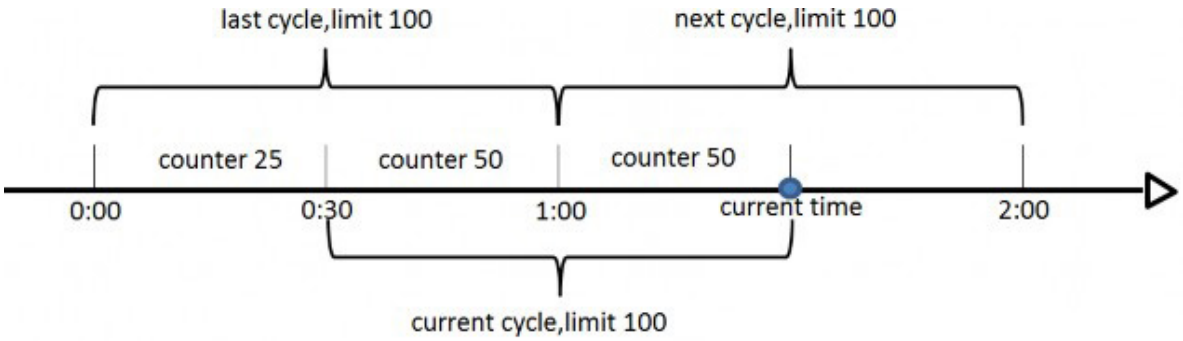
22、说说什么是滑动窗口算法

滑动窗口算法是将时间周期分为N个小周期，分别记录每个小周期内访问次数，并且根据时间滑动删除过期的小周期。

如下图，假设时间周期为1min，将1min再分为2个小周期，统计每个小周期的访问数量，则可以看到，

第一个时间周期内，访问数量为75，

第二个时间周期内，访问数量为100，超过100的访问则被限流掉了



由此可见，当滑动窗口的格子划分的越多，那么滑动窗口的滚动就越平滑，限流的统计就会越精确。

此算法可以很好的解决固定窗口算法的临界问题。

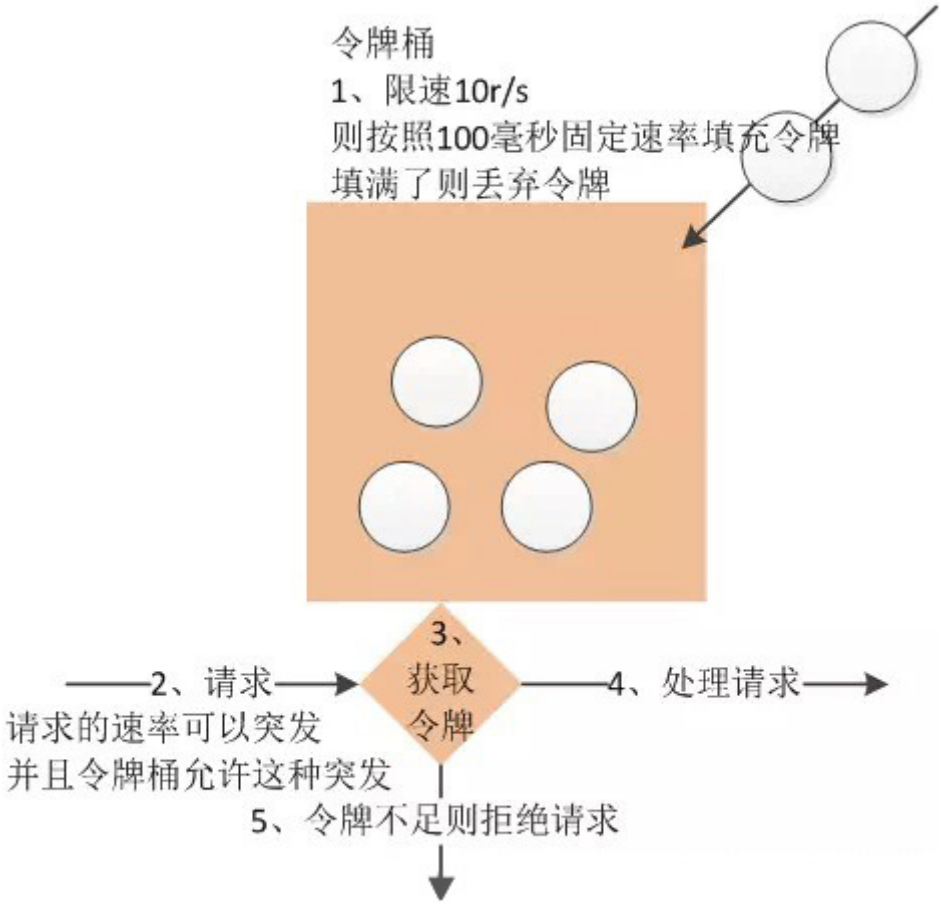
23、说说什么是漏桶算法

漏桶算法是访问请求到达时直接放入漏桶，如当前容量已达到上限（限流值），则进行丢弃（触发限流策略）。

漏桶以固定的速率进行释放访问请求（即请求通过），直到漏桶为空。

24、说说什么是令牌桶算法

令牌桶算法是程序以r（r=时间周期/限流值）的速度向令牌桶中增加令牌，直到令牌桶满，请求到达时向令牌桶请求令牌，如获取到令牌则通过请求，否则触发限流策略



25.Dubbo 支持哪些协议？每种协议的应用场景及其优缺点?

1.dubbo协议，这是一种单一的长连接和 NIO 异步通讯协议，适合高并发小数据量的调用，也适用于消费者数量远大于提供者的场景，这种协议是最常用的，其中传输协议用到了TCP，采用异步通讯和 Hessian 序列化的方式。

2.rmi协议，其中采用 JDK里的rmi 协议实现，传输参数传递时，需要传递的对象实现Serializable 接口，通讯时使用阻塞式短连接，适用于消费者和提供者个数差不多的场景，也可传输文件。

3.webservice协议，这是基于 比较老的WebService 远程调用协议，提供和原生 WebService 的操作接口。适用于多个短连接的场景，是用HTTP 传输的同步方式传输数据，适用于系统集成和跨语言调用的场景。

4.http协议是基于Http 表单提交的远程调用协议，比较适用于应用程序和浏览器 JS 之间的调用。

5.hessian协议是基于 HTTP 通讯，采用 Servlet 暴露服务，使用场景是，传入参数较大，提供者大于消费者，提供者压力较大等场景，通过该协议可文件。

6.memcache协议，这是基于memcached缓存实现的 RPC 协议。

7.redis协议，这是基于基于redis缓存实现的 RPC 协议。

总体来说，目前大多数Dubbo使用场景，都会用Dubbo协议，毕竟大多数使用场景是高并发场景，而且不大会传文件，如果真要传，那还不如用FTP等协议，就别用dubbo组件来传。

26. 说下你知道的Dubbo组件中用到的设计模式?

1 责任链模式

Dubbo的调用链是用责任链模式串连起来的。

比如上文提到的Filter链就是职责链模式，通过这种模式，能通过同一条链路上的多个Filter实现监控、写日志、和安全等方面的需求功能。

2 观察者模式

比较典型的例子是RegistryService。消

费者在初始化调用对象时会回调subscribe方法，从而注册一个观察者，如果观察者对应的的服务地址或端口列表有变更，会通过NotifyListener通知消费者，这样就达到了动态服务治理的效果。

3 修饰器模式

Dubbo还用到了修饰器模式，

比如ProtocolFilterWrapper类是对Protocol类的修饰。

4 工厂模式

比如CacheFactory类的的实现方式是用到了工厂模式。

CacheFactory接口定义getCache方法，然后再定义一个AbstractCacheFactory抽象类实现CacheFactory，并将实际创建cache对象的createCache方法以抽象方法的形式提取。

这样cache对象的创建工作就会通过对应的子类去完成。

5 适配器模式

为了能让用户自己选定日志组件，Dubbo 定义了Logger 接口，并为常见的log4j和slf4j等日志组件提供了适配器，并在此基础上用工厂模式提供一个 LoggerFactory。

这样用户就能在Dubbo里定义Logger，而不用关心实际的日志组件类型。

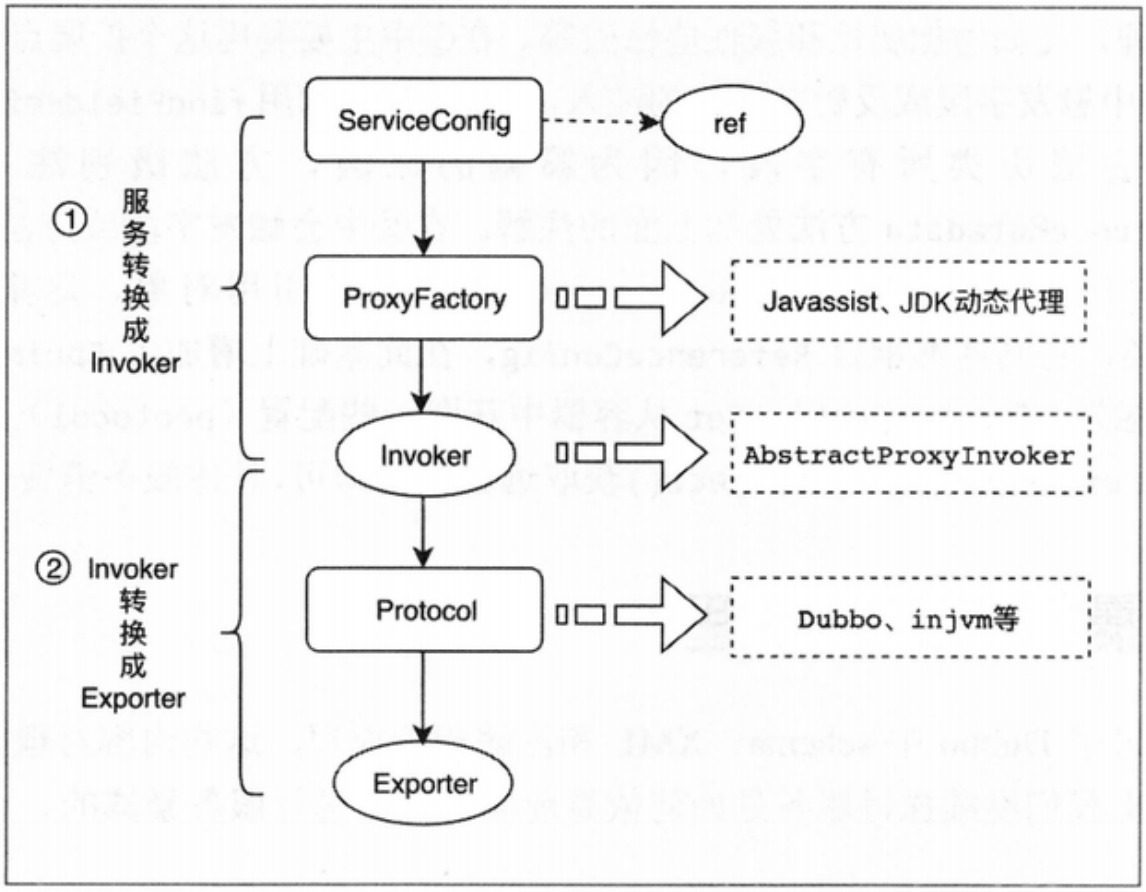
6 代理模式

这是Dubbo的表现形式，比如Dubbo的消费者在使用Proxy类创建远程服务的本地代理时，

在本地代理中需要实现和远程服务一样的接口方法，这样能屏蔽针对远程方法调用的网络通信细节，使得用户能像调用本地方法那样调用远程方法。

27.通过流程图，说明下Dubbo的服务暴露流程?

大致的流程如下所示。



1.通过ServiceConfig对象解析标签，并创建dubbo的标签解析器对象来解析dubbo标签，随后通过触发ContextRefreshEvent事件的回调方法开始暴露服务的动作。

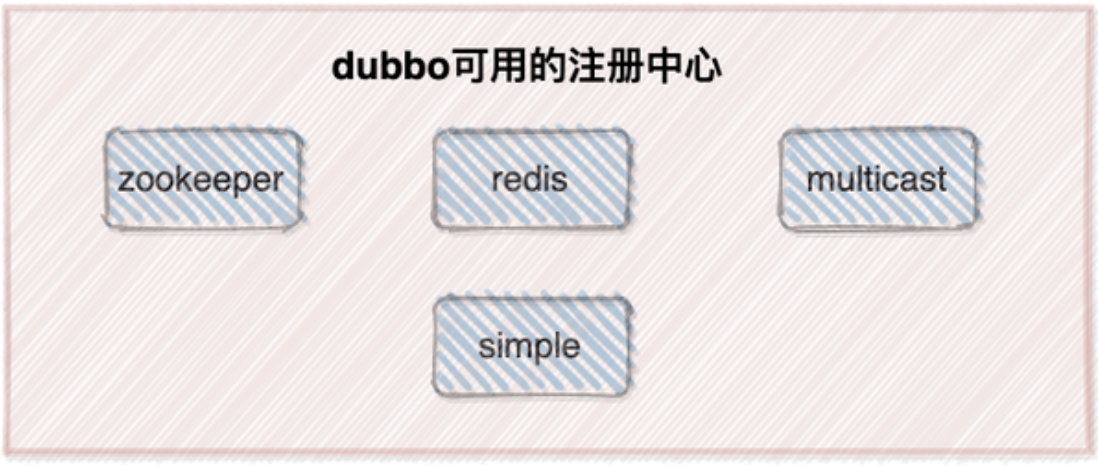
2.通过调用proxyFactory对象的getInvoker方法，并用javassist或DdkProxyFactory来进行动态代理，把服务暴露接口封装成invoker对象，在该对象里包含需要执行的方法名、参数和对应的URL地址。

3.通过DubboProtocol的实现类，把包装后的invoker转换成exporter对象。随后启动服务器端的server来监听端口，等待服务调用的到来。

4.通过RegistryProtocol对象，保存URL地址和invoker之间的映射关系，同时把这层映射关系注册到服务中心，比如Zookeeper里。

上文中的流程如下图所示。

29.Dubbo的注册中心有哪些?



Zookeeper、Redis、Multicast、Simple 等都可以作为Dubbo的注册中心

30.聊聊Dubbo SPI机制?

SPI(Service Provider Interface)，是一种**服务发现机制**，其实就是将结构的实现类写入配置当中，在服务加载的时候将配置文件独处，加载实现类，这样就可以在运行的时候，**动态的帮助接口替换实现类。**

Dubbo的SPI其实是对java的SPI进行了一种增强,可以按需加载实现类之外，增加了 IOC 和 AOP 的特性，还有**自适应扩展**机制。

SPI在dubbo应用很多，包括协议扩展、集群扩展、路由扩展、序列化扩展等等。

Dubbo对于文件目录的配置分为了**三类。**

1.META-INF/services/ 目录：

该目录下的 SPI 配置文件是为了用来兼容 Java SPI 。

2.META-INF/dubbo/ 目录：

该目录存放用户自定义的 SPI 配置文件。

```
key=com.xxx.xxx
```

3.META-INF/dubbo/internal/ 目录：

该目录存放 Dubbo 内部使用的 SPI 配置文件。

31.Dubbo的SPi和JAVA的SPI有什么区别?

Java Spi

Java SPI 在查找扩展实现类的时候，遍历 SPI 的配置文件，并且将实现类**全部实例化**

Dubbo Spi

1，对 Dubbo 进行扩展，不需要改动 Dubbo 的源码

2，延迟加载，可以一次**只加载自己想要加载的**扩展实现。

3，增加了对扩展点 IOC 和 AOP 的支持，一个扩展点可以直接 setter 注入其它扩展点。

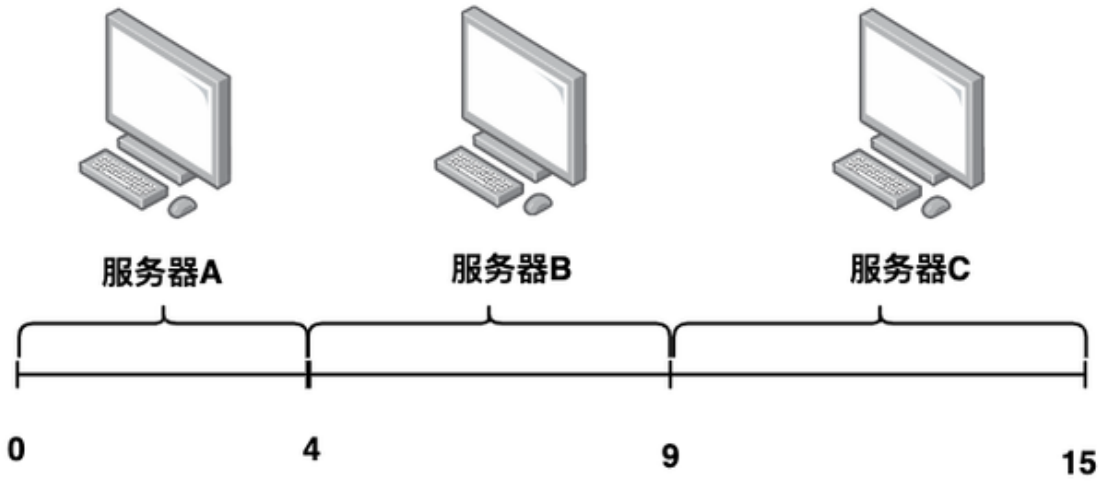
4，Dubbo 的扩展机制能很好的支持第三方 IoC 容器，默认支持 Spring Bean。

32.有哪些负载均衡策略?

1.加权随机:

比如我们有三台服务器[A, B, C]，给他们设置权重为[4, 5, 6]，然后讲这三个数平铺在水平线上,和为15。

然后在15以内生成一个随机数，0～4是服务器A，4～9是服务器B，9～15是服务器C

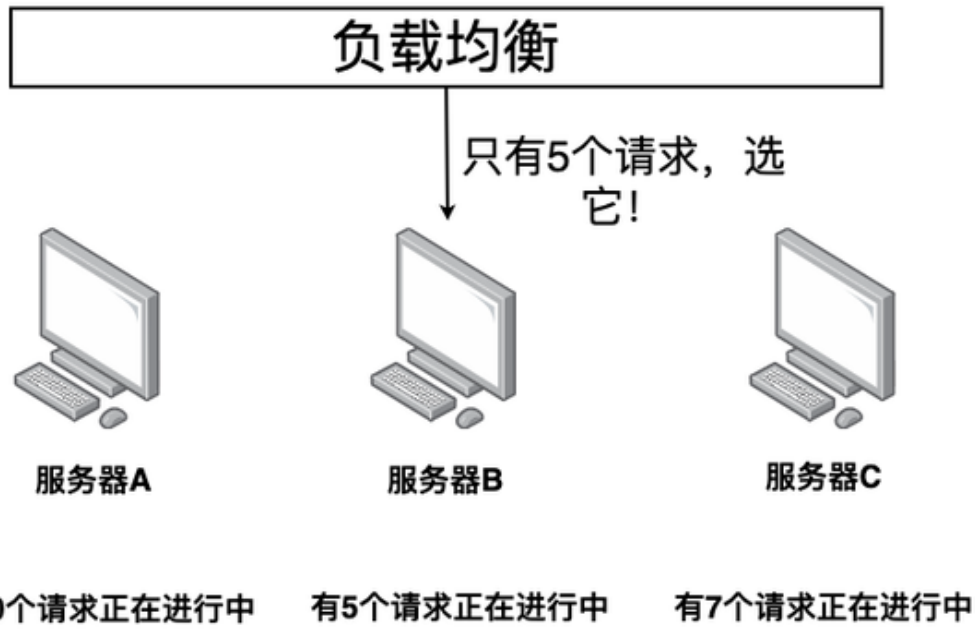


2.最小活跃数:

每个服务提供者对应一个活跃数 active，初始情况下，所有服务提供者活跃数均为0。

每收到一个请求，活跃数加1，完成请求后则将活跃数减1。

在服务运行一段时间后，性能好的服务提供者处理请求的速度更快，因此活跃数下降的也越快，此时这样的服务提供者能够优先获取到新的服务请求。



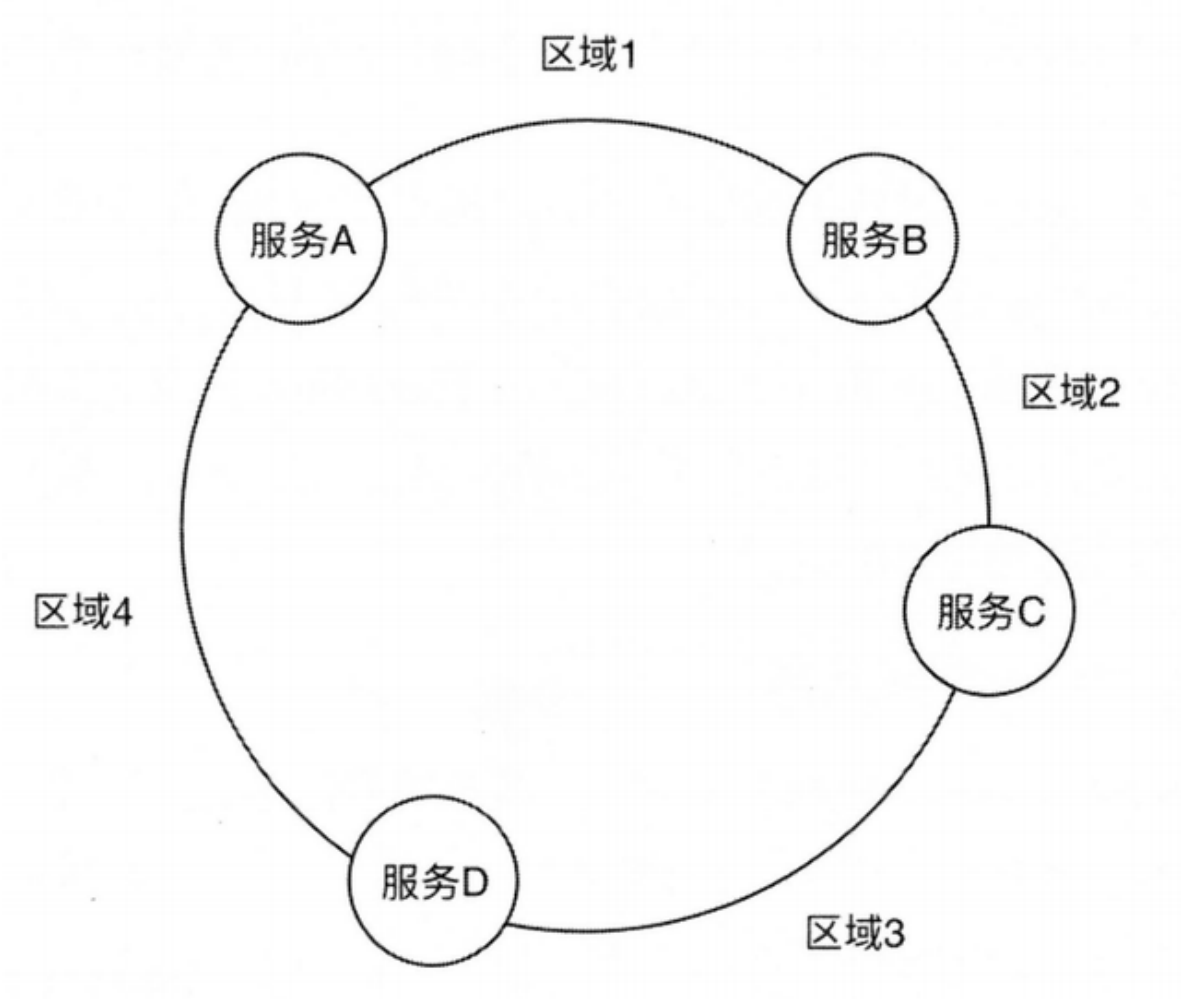
3.一致性hash：

首先求出memcached服务器（节点）的哈希值，并将其配置到0～232的圆（continuum）上。

然后采用同样的方法求出存储数据的键的哈希值，并映射到相同的圆上。

然后从数据映射到的位置开始顺时针查找，将数据保存到找到的第一个服务器上。

如果超过232仍然找不到服务器，就会保存到第一台memcached服务器上。



4.加权轮询:

比如我们有三台服务器[A, B, C]，给他们设置权重为[4, 5, 6]，那么假如总共有15次请求，那么会有4次落在A服务器，5次落在B服务器，6次落在C服务器。

33.集群容错方式有哪些?

1.**Failover Cluster失败自动切换**:

dubbo的默认容错方案，当调用失败时自动切换到其他可用的节点，具体的重试次数和间隔时间可用通过引用服务的时候配置，默认重试次数为1是只调用一次。

2.**Failback Cluster失败自动恢复**：

在调用失败，记录日志和调用信息，然后返回空结果给consumer，并且通过定时任务每隔5秒对失败的调用进行重试

3.**Failfast Cluster快速失败**：

只会调用一次，失败后立刻抛出异常

4.**Failsafe Cluster失败安全**：

调用出现异常，记录日志不抛出，返回空结果

5.**Forking Cluster并行调用多个服务提供者**：

通过线程池创建多个线程，并发调用多个provider，结果保存到阻塞队列，只要有一个provider成功返回了结果，就会立刻返回结果

6.**Broadcast Cluster广播模式**：

逐个调用每个provider，如果其中一台报错，在循环调用结束后，抛出异常。

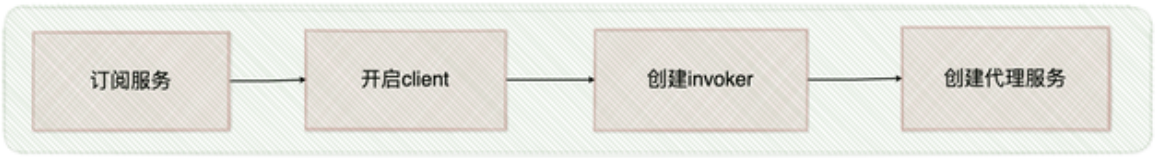
34.说下Dubbo的服务引用的流程。

1.Dubbo客户端根据config文件里的信息从注册中心里订阅服务，并缓存到本地，后续的服务相关信息的会动态更新到本地。

2.DubboProtocol根据provider的地址和接口连接到服务端server，开启客户端client，再创建invoker。

3.用invoker为服务接口生成代理对象，这个代理对象是用来远程调用。

相关流程如下图所示。



35.服务提供者能实现失效踢出是什么原理?

服务失效踢出基于**Zookeeper的临时节点**原理。

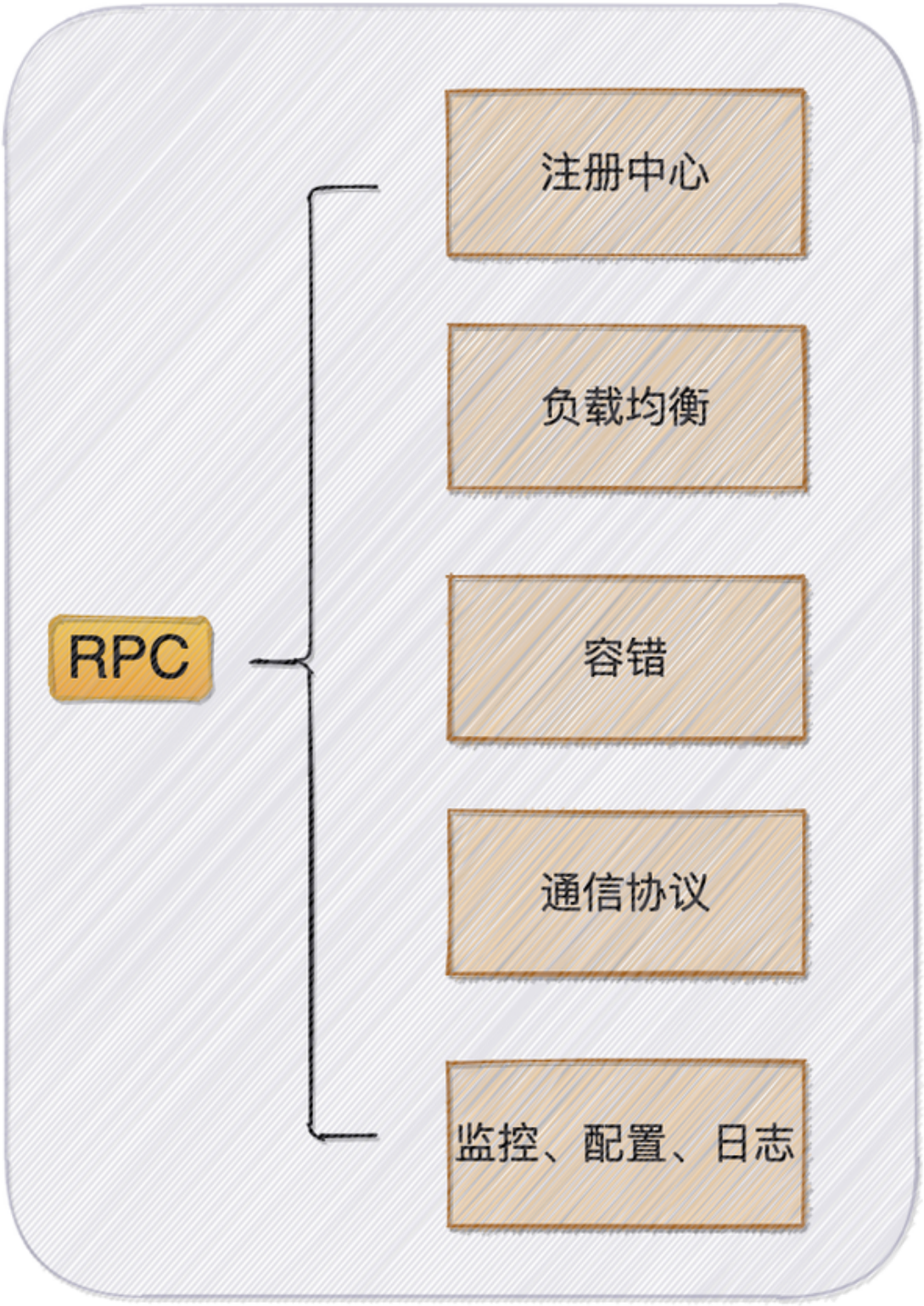
Zookeeper中节点是有生命周期的，具体的生命周期取决于节点的类型，节点主要分为**持久节点**(Persistent)和**临时节点**(Ephemeral) 。

36.为什么要通过代理对象通信？？

其实主要就是为了将调用细节封装起来，将调用远程方法变得和调用本地方法一样简单，还可以做一些其他方面的增强，比如负载均衡，容错机制，过滤操作，调用数据的统计。

37.怎么设计一个RPC框架?

关于这个问题，其实核心考察点就是你**对于RPC框架的理解**，一个成熟的RPC框架**可以完成哪些功能**，其实当我们看过一两个RPC框架后，就可以对这个问题回答个七七八八了，我们来举个例子。



1.首先我们得需要一个**注册中心**，去管理消费者和提供者的节点信息，这样才会有消费者和提供才可以去订阅服务，注册服务。

2.当有了注册中心后，可能会有很多个provider节点，那么我们肯定会有一个**负载均衡**模块来负责节点的调用，至于用户指定路由规则可以使一个额外的优化点。

3.具体的调用肯定会需要牵扯到通信协议，所以需要一个模块来对**通信协议进行封装**，网络传输还要考虑序列化。

4.当调用失败后怎么去处理？所以我们还需要一个**容错模块**，来负责失败情况的处理。

5.其实做完这些一个基础的模型就已经搭建好了，我们还可以有更多的优化点，比如一些请求**数据的监控，配置信息的处理，日志信息的处理**等等。

这其实就是一个比较基本的RPC框架的大体思路，大家有没有get到?

39、说说 Dubbo 与 Spring Cloud 的区别?

两个没关联，如果硬要说区别，有以下几点。

1) 通信方式不同

Dubbo 使用的是 RPC 通信，而 Spring Cloud 使用的是 HTTP RESTFul 方式。

2) 组成部分不同

组件	Dubbo	Spring Cloud
服务注册中心	Zookeeper	Spring Cloud Netflix Eureka
服务监控	Dubbo-monitor	Spring Boot Admin
断路器	不完善	Spring Cloud Netflix Hystrix
服务网关	无	Spring Cloud Netflix Gateway
分布式配置	无	Spring Cloud Config
服务跟踪	无	Spring Cloud Sleuth
消息总线	无	Spring Cloud Bus
数据流	无	Spring Cloud Stream
批量任务	无	Spring Cloud Task
…	…	…

40、简述一下什么是Nginx，它有什么优势和功能?

Nginx是一个web服务器和方向代理服务器，用于HTTP、HTTPS、SMTP、POP3和IMAP协议。因它的稳定性、丰富的功能集、示例配置文件和低系统资源的消耗而闻名。

Nginx---Ngine X，是一款免费的、自由的、开源的、高性能HTTP服务器和反向代理服务器；也是一个IMAP、POP3、SMTP代理服务器；Nginx以其高性能、稳定性、丰富的功能、简单的配置和低资源消耗而闻名。

也就是说Nginx本身就可以托管网站（类似于Tomcat一样），进行Http服务处理，也可以作为反向代理服务器 、负载均衡器和HTTP缓存。

Nginx 解决了服务器的C10K（就是在一秒之内连接客户端的数目为10k即1万）问题。它的设计不像传统的服务器那样使用线程处理请求，而是一个更加高级的机制—事件驱动机制，是一种异步事件驱动结构。

优点：

(1) **更快** 这表现在两个方面：一方面，在正常情况下，单次请求会得到更快的响应；另一方面，在高峰期（如有数以万计的并发请求），Nginx可以比其他Web服务器更快地响应请求。

(2) **高扩展性，跨平台** Nginx的设计极具扩展性，它完全是由多个不同功能、不同层次、不同类型且耦合度极低的模块组成。因此，当对某一个模块修复Bug或进行升级时，可以专注于模块自身，无须在意其他。而且在HTTP模块中，还设计了HTTP过滤器模块：一个正常的HTTP模块在处理完请求后，会有一串HTTP过滤器模块对请求的结果进行再处理。这样，当我们开发一个新的HTTP模块时，不但可以使用诸如HTTP核心模块、events模块、log模块等不同层次或者不同类型的模块，还可以原封不动地复用大量已有的HTTP过滤器模块。这种低耦合度的优秀设计，造就了Nginx庞大的第三方模块，当然，公开的第三方模块也如官方发布的模块一样容易使用。 Nginx的模块都是嵌入到二进制文件中执行的，无论官方发布的模块还是第三方模块都是如此。这使得第三方模块一样具备极其优秀的性能，充分利用Nginx的高并发特性，因此，许多高流量的网站都倾向于开发符合自己业务特性的定制模块。

(3) **高可靠性**：用于反向代理，宕机的概率微乎其微 高可靠性是我们选择Nginx的最基本条件，因为Nginx的可靠性是大家有目共睹的，很多家高流量网站都在核心服务器上大规模使用Nginx。 Nginx的高可靠性来自于其核心框架代码的优秀设计、模块设计的简单性；另外，官方提供的常用模块都非常稳定，每个worker进程相对独立，master进程在1个worker进程出错时可以快速“拉起”新的worker子进程提供服务。

(3) **低内存消耗** 一般情况下，10 000个非活跃的HTTP Keep-Alive连接在Nginx中仅消耗2.5MB的内存，这是Nginx支持高并发连接的基础。

(4) **单机支持10万以上的并发连接** 这是一个非常重要的特性！随着互联网的迅猛发展和互联网用户数量的成倍增长，各大公司、网站都需要应付海量并发请求，一个能够在峰值期顶住10万以上并发请求的Server，无疑会得到大家的青睐。理论上，Nginx支持的并发连接上限取决于内存，10万远未封顶。当然，能够及时地处理更多的并发请求，是与业务特点紧密相关的。

(6) **热部署** master管理进程与worker工作进程的分离设计，使得Nginx能够提供热部署功能，即可以在7×24小时不间断服务的前提下，升级Nginx的可执行文件。当然，它也支持不停止服务就更新配置项、更换日志文件等功能。

(7) **最自由的BSD许可协议** 这是Nginx可以快速发展的强大动力。BSD许可协议不只是允许用户免费使用Nginx，它还允许用户在自己的项目中直接使用或修改Nginx源码，然后发布。这吸引了无数开发者继续为Nginx贡献自己的智慧。 以上7个特点当然不是Nginx的全部，拥有无数个官方功能模块、第三方功能模块使得Nginx能够满足绝大部分应用场景，这些功能模块间可以叠加以实现更加强大、复杂的功能，有些模块还支持Nginx与Perl、Lua等脚本语言集成工作，大大提高了开发效 率。这些特点促使用户在寻找一个Web服务器时更多考虑Nginx。 选择Nginx的核心理由还是它能在支持高并发请求的同时保持高效的服务。

41、Nginx是如何处理一个HTTP请求的呢?

Nginx 是一个高性能的 Web 服务器，能够同时处理大量的并发请求。它结合多进程机制和异步机制，异步机制使用的是异步非阻塞方式 ，接下来就给大家介绍一下 Nginx 的多线程机制和异步非阻塞机制。

1、多进程机制

服务器每当收到一个客户端时，就有 服务器主进程 （ master process ）生成一个 子进程（worker process ）出来和客户端建立连接进行交互，直到连接断开，该子进程就结束了。

使用进程的好处是各个进程之间相互独立，不需要加锁，减少了使用锁对性能造成影响，同时降低编程的复杂度，降低开发成本。其次，采用独立的进程，可以让进程互相之间不会影响 ，如果一个进程发生异常退出时，其它进程正常工作， master 进程则很快启动新的 worker 进程，确保服务不会中断，从而将风险降到最低。

缺点是操作系统生成一个子进程需要进行 内存复制等操作，在资源和时间上会产生一定的开销。当有大量请求时，会导致系统性能下降 。

2、异步非阻塞机制

每个工作进程 使用 异步非阻塞方式 ， 可以处理 多个客户端请求 。

当某个 工作进程 接收到客户端的请求以后，调用 IO 进行处理，如果不能立即得到结果，就去 处理其他请求 （即为 非阻塞 ）； 而 客户端 在此期间也 无需等待响应 ， 可以去处理其他事情（即为 异步 ）。

当 IO 返回时，就会通知此 工作进程 ； 该进程得到通知，暂时 挂起 当前处理的事务去 响应客户端请求 。

硬核推荐：尼恩Java硬核架构班

又名疯狂创客圈社群 VIP

详情：https://www.cnblogs.com/crazymakercircle/p/9904544.html


规划中
《架构师实操篇：基于netty 手写 rpc 框架- 参考 dubbo、seata rpc框架》
《架构师实操篇：go语言学习，以及基于 go 手写 rpc 框架》
《架构师实操篇：千万级任务调度平台 架构与实操- 基于尼恩17年的亿级搜索项目》
《架构师实操篇：工业级 亿级文档搜索 平台 架构与实操- 基于尼恩17年的亿级搜索项目》
特色
会员制
提供技术方向指导，
职业生涯指导，少躺坑，少弯路
简历指导
这个很重要，
对于挪窝涨薪来说
实操性
以上项目，都是老架构师
在生产上实操过的项目
非水货
40岁老架构师，不是水货架构师
《Java高并发三部曲》为证

架构班（社群 VIP）的起源：

最初的视频，主要是给读者加餐。很多的读者，需要一些高质量的实操、理论视频，所以，我就围绕书，和底层，做了几个实操、理论视频，然后效果还不错，后面就做成迭代模式了。

架构班（社群 VIP）的功能：

提供高质量实操项目整刀真枪的架构指导、快速提升大家的:

- 开发水平
- 设计水平
- 架构水平

弥补业务中 CRUD 开发短板，帮助大家尽早脱离具备 3 高能力，掌握：

- 高性能
- 高并发
- 高可用

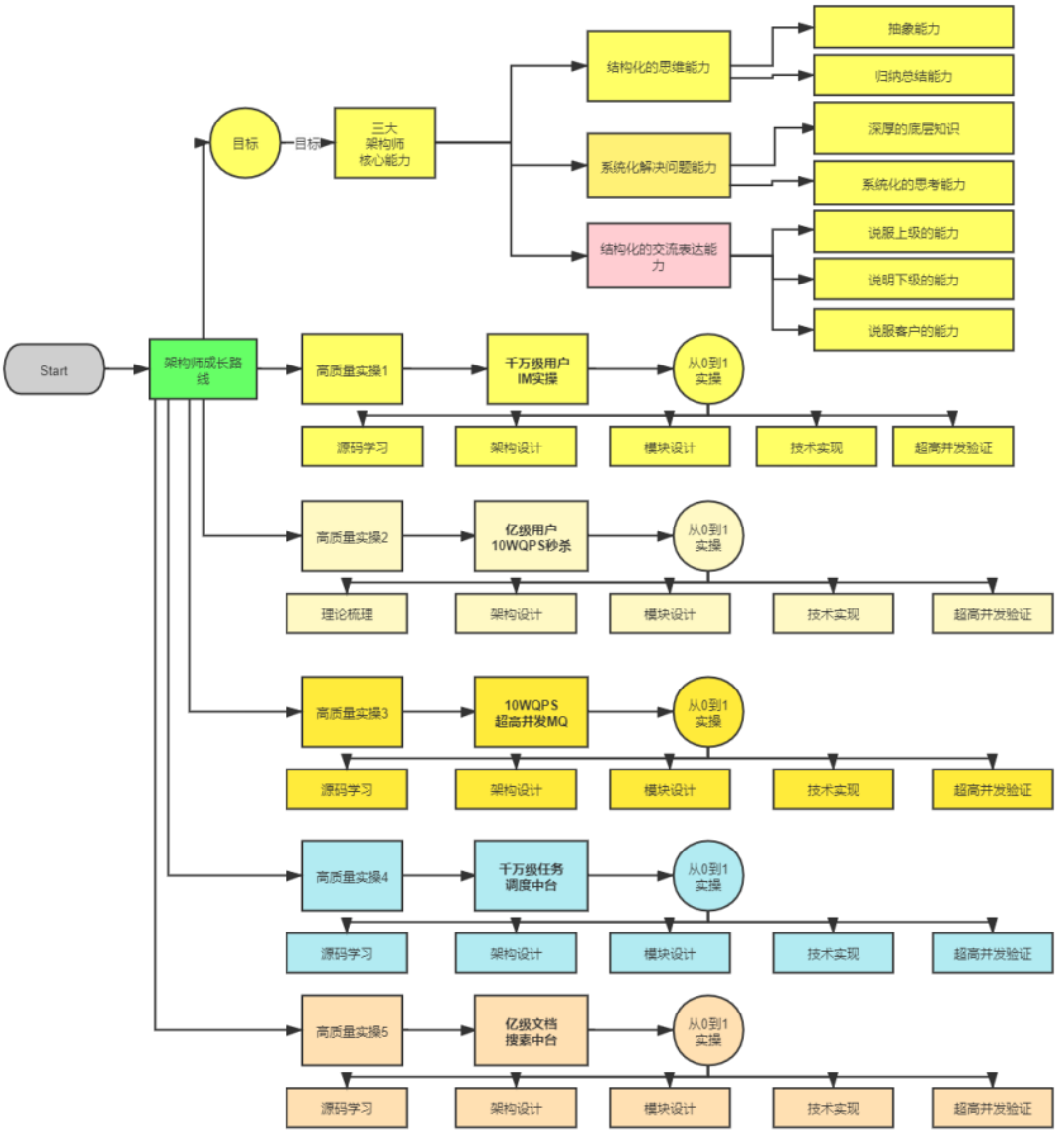
作为一个高质量的架构师成长、人脉社群，把所有的卷王聚焦起来，一起卷：

- 卷高并发实操
- 卷底层原理
- 卷架构理论、架构哲学
- 最终成为顶级架构师，实现人生理想，走向人生巅峰

架构班（社群 VIP）的目的：

- 高质量的实操，大大提升简历的含金量，吸引力，增强面试的召唤率
- 为大家提供九阳真经、葵花宝典，快速提升水平
- 进大厂、拿高薪
- 一路陪伴，提供助学视频和指导，辅导大家成为架构师
- 自学为主，和其他卷王一起，卷高并发实操，卷底层原理、卷大厂面试题，争取狠卷 3 月成高手，狠卷 3 年成为顶级架构师

N 个超高并发实操项目：简历压轴、个顶个精彩



【样章】第 17 章:横扫全网 Rocketmq 视频第 2 部曲: 工业级 rocketmq 高可用(HA)底层原理和实操

工业级 rocketmq 高可用底层原理，包含：消息消费、同步消息、异步消息、单向消息等不同消息的底层原理和源码实现；消息队列非常底层的主从复制、高可用、同步刷盘、异步刷盘等底层原理。

工业级 rocketmq 高可用底层原理和搭建实操，包含：高可用集群的搭建。

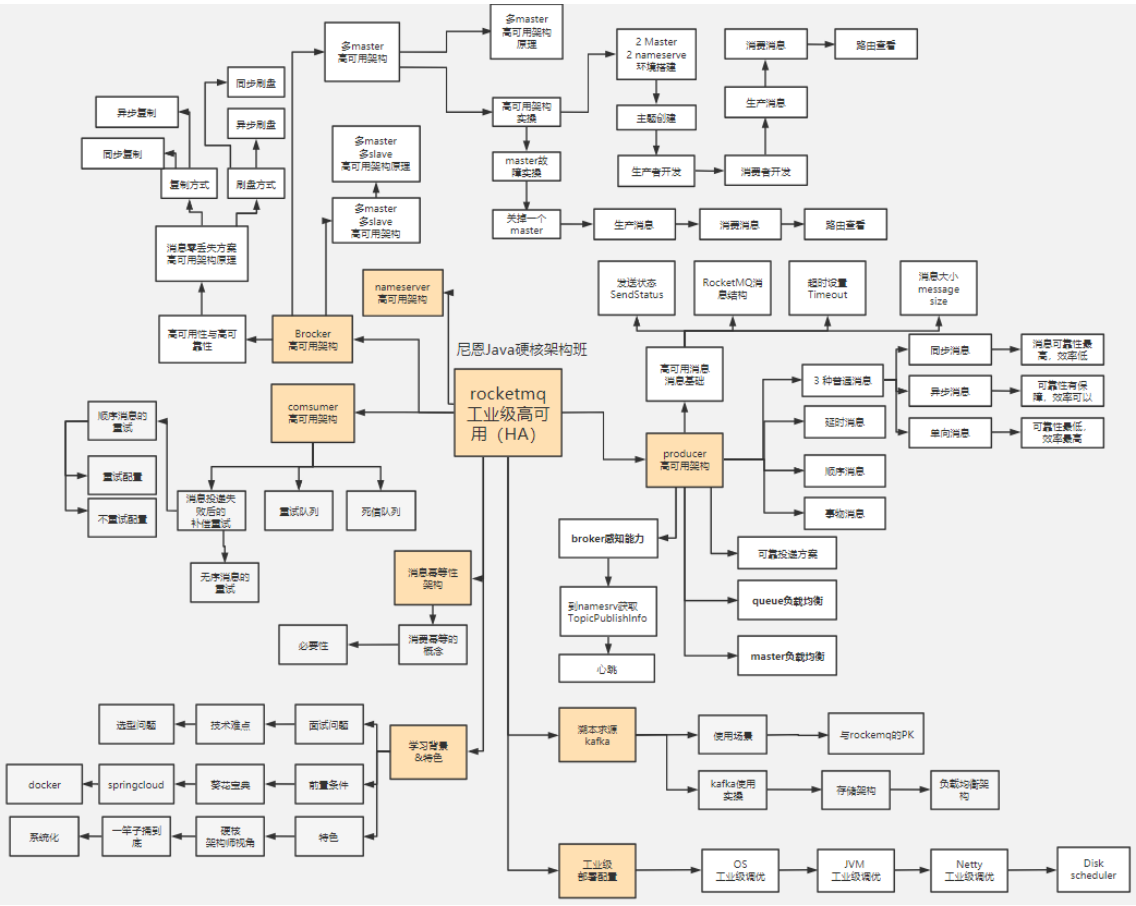
解决以下难题：

1、技术难题：RocketMQ 如何最大限度的保证消息不丢失的呢？RocketMQ 消息如何做到高可靠投递？

2、技术难题：基于消息的分布式事务，核心原理不理解

3、选型难题： kafka or rocketmq ，该娶谁？

下图链接：https://www.processon.com/view/6178e8ae0e3e7416bde9da19



成功案例：2 年翻 3 倍，35 岁卷王成功转型为架构师

详情：http://topcoder.cloud/forum.php?mod=forumdisplay&fid=43&page=1

最新　最后发表　热门　精华

成功案例：[1057号卷王] 3年小伙拿到外企offer，薪酬涨了200%
卷王1号　超级版主　前天 17:41

成功案例：[645号卷王] 4年经验卷王逆袭，被毕业后，反涨24W
卷王1号　超级版主　2022-9-21

成功案例：[878号卷王] 小伙8年经验，年薪60W
卷王1号　超级版主　2022-8-13

年薪70W案例：通过尼恩的指导，小伙伴年薪从40W涨到70W
卷王1号　超级版主　2022-2-11

成功案例：[493号卷王] 5年小伙拿满意offer，就业寒冬季逆涨30%
卷王1号　超级版主　前天 17:43

成功案例：[250号卷王] 就业极寒时代，收offer 涨25%
卷王1号　超级版主　前天 17:38

成功案例：[612号卷王] 就业极寒时代，从外包到自研
卷王1号　超级版主　前天 17:15

成功案例：[913号卷王] 热烈祝贺6年经验卷王，年薪40W
卷王1号　超级版主　2022-9-21

成功案例：[959号卷王] 4年经验卷王，喜获百度、Boss直聘等N个优质offer，最高涨100%
卷王1号　超级版主　2022-9-21

成功案例：[529号卷王] 5年经验卷王喜收2大offer，最高涨5K
卷王1号　超级版主　2022-9-21

成功案例：[811号卷王] 热烈祝贺7年经验卷王，薪酬涨30%
卷王1号　超级版主　2022-9-21

成功案例：[287号卷王] 不惧大寒潮，卷王逆市收4 offer，涨30%，可喜可贺
卷王1号　超级版主　2022-5-30

成功案例：[1002号卷王] 5月份"被毕业"，改简历后，斩获顶级央企Offer，涨薪7000+
卷王1号　超级版主　2022-7-5

成功案例：[7号卷王] 热烈祝贺小伙伴涨薪120%
卷王1号 超级版主 2022-8-13

成功案例：[134号卷王] 大三小伙卷1年，斩获顶级央企Offer，成功逆袭
卷王1号 超级版主 2022-7-6

成功案例：[1008号卷王] 5年经验卷王收42W offer，月涨8000，可喜可贺
卷王1号 超级版主 2022-5-30

成功案例：[453号卷王] 非全日制 6年卷王喜提3 offer，年薪30W，可喜可贺
卷王1号 超级版主 2022-5-21

成功案例：[924号卷王] 6年卷王喜提4 offer，最高涨薪9000，可喜可贺
卷王1号 超级版主 2022-5-21

成功案例：[15号卷王] 4年卷王入职 微软，涨薪50%，可喜可贺
卷王1号 超级版主 2022-5-12

成功案例：[527号卷王] 4年卷王喜提2 offer，涨薪50%，可喜可贺
卷王1号 超级版主 2022-5-13

成功案例：[788号卷王] 3年卷王喜提优质Offer，涨薪60%
卷王1号 超级版主 2022-5-11

成功案例：热烈祝贺：非全日制卷王，喜提2个心仪offer，面3家过2家
卷王1号 超级版主 2022-4-21

成功案例：[693号卷王] 二线城市6年卷王喜提4大优质Offer，含央企offer，最高薪酬35W
卷王1号 超级版主 2022-4-16

成功案例：[85号卷王] 双非2本小伙，春招大捷，喜提9个offer，最高薪酬近30万
卷王1号 超级版主 2022-4-14

成功案例：[741号卷王] 卷王逆袭！6年小伙从很少面试机会到搞定35K*14薪Offer
卷王1号 超级版主 2022-4-12

成功案例：[642号卷王] 热烈祝贺，6年卷王喜提优质国企offer
卷王1号 超级版主 2022-4-7

成功案例：[796号卷王] 热烈祝贺，36岁卷王喜提52万优质offer
卷王1号 超级版主 2022-3-25

成功案例：[15号卷王] 小伙卷1年，涨薪9K+，喜收ebay等多个优质offer

卷王1号 超级版主 2022-3-24

成功案例：[821号卷王] 小伙狠卷3个月，喜提10多个offer

卷王1号 超级版主 2022-3-21

成功案例：[736号卷王] 3年半经验收22k offer，但是小伙志存高远，冲击25k+

卷王1号 超级版主 2022-3-20

成功案例：热烈祝贺1群小卷王offer拿到手软，甚至拒了阿里offer

卷王1号 超级版主 2022-3-16

简历案例：简历一改，腾讯的邀请就来了！热烈祝贺，小伙收到一大堆面试邀请

卷王1号 超级版主 2022-3-10

成功案例：祝贺我圈两大超级卷王，一个过了阿里HR面，一个过了阿里2面

卷王1号 超级版主 2022-3-10

成功案例：小伙伴php转Java，卷1.5年Java，涨薪50%，喜收多个优质offer

卷王1号 超级版主 2022-3-10

成功案例：4年小伙狠卷半年，拿到 移动、京东 两大顶级offer

尼恩 超级版主 2022-3-5

成功案例：[267号卷王] 助力3年经验卷王，拿到蜂巢的17k x 14薪的offer

卷王1号 超级版主 2022-2-27

成功案例：[143号卷王] 二本院校00后卷神，毕业没到一年跳到字节，年薪45W

卷王1号 超级版主 2022-2-27

成功案例：[494号卷王] 尼恩分布式事务助力卷王拿到 中信银行offer

卷王1号 超级版主 2022-2-27

成功案例：[76号卷王] 2线城市卷王，狠卷1.5年，喜收22K offer

卷王1号 超级版主 2022-2-27

成功案例：[429号卷王] 小伙伴在社群卷5个月，涨8k+

卷王1号 超级版主 2022-2-27

成功案例：[154号卷王] 双非学校毕业卷王，连拿 京东到家&滴滴 两个大厂Offer

卷王1号 超级版主 2022-2-27

成功案例：[232号卷王] 涨薪10K，继续卷向食物链顶端

卷王1号 超级版主 2022-2-27

成功案例：狠卷1年技术，喜收 腾讯、阿里、微软三大Offer，最高年薪56W

卷王1号 超级版主 2022-2-27

成功案例：[449号卷王] 应届毕业卷王喜收 滴滴offer, 年薪33W

卷王1号 超级版主 2022-2-27

成功案例：[551号卷王] 小伙伴学完后，成功进入大厂，并且推荐自己的朋友加VIP学习

卷王1号 超级版主 2022-2-10

成功案例：[214号卷王] 助力2年经验卷王，成功拿到17K月薪

卷王1号 超级版主 2022-2-10

成功案例：[92号卷王] 课程实操助力社群小伙伴喜收 喜马拉雅Offer

卷王1号 超级版主 2022-2-10

成功案例：社群卷王小伙伴成功过了滴滴三面 获滴滴Offer

卷王1号 超级版主 2022-2-10

[612号卷王]滴滴小伙伴，蹲点考察半年，觉得靠谱后加入 疯狂创客圈

卷王1号 超级版主 2022-2-10

成功案例：[732号卷王] 尼恩助力3年经验卷王收获 京东offer, 年薪35W

卷王1号 超级版主 2022-2-27

成功案例：[558号卷王] 2年经验卷王，喜收 网易和阿里子公司两个优质offer

卷王1号 超级版主 2022-2-27

成功案例：[569号卷王] 双非应届生卷王, 喜收字节跳动实习offer

卷王1号 超级版主 2022-2-25

成功案例：[420号卷王] 狠卷1年，卷王涨薪80%，涨薪12000元!

卷王1号 超级版主 2022-2-25

成功案例：[76号卷王] 通过尼恩1年半的指导，专科学历小伙伴从0.8K涨到22K

卷王1号 超级版主 2022-2-10

简历优化后的成功涨薪案例（VIP 含免费简历优化）


1.5年小伙搞定15K offer
就业寒冬涨100%
5月7日改简历
11月21日晒offer
秘诀:
简历指导+狠狠卷
多谢啦
你好，好多公司都觉得我的简历太简单了，我感觉是不是没法优化了
简历.pdf
397.4 KB
好的
晚上指导你改
推送中台需要加不
我还没做啊，准备八股文暂时没时间搞这些，入职了会开始搞
OK
还有其他的项目吗
一个项目，太少啦
那种练习项目也行
他那个不止两倍吧，我才是两倍，原先7k，现在15k
那也很牛
都翻倍了，都是牛人
正常就30%
慢慢来
下一步可以瞄准25k
其实我工作一年
那你就更牛逼拉
一年经验，搞定15k offer，舍你其谁


卷王逆袭成功案例
6年小伙从很少面试机会到
搞定35K*14薪
3月5日改简历
4月11日拿offer
面试邀请少
小伙很苦恼
理想很丰满
目标35K
我要努力争取35+
一个月拿到了
理想的offer
面试法宝
rocketmq四部曲

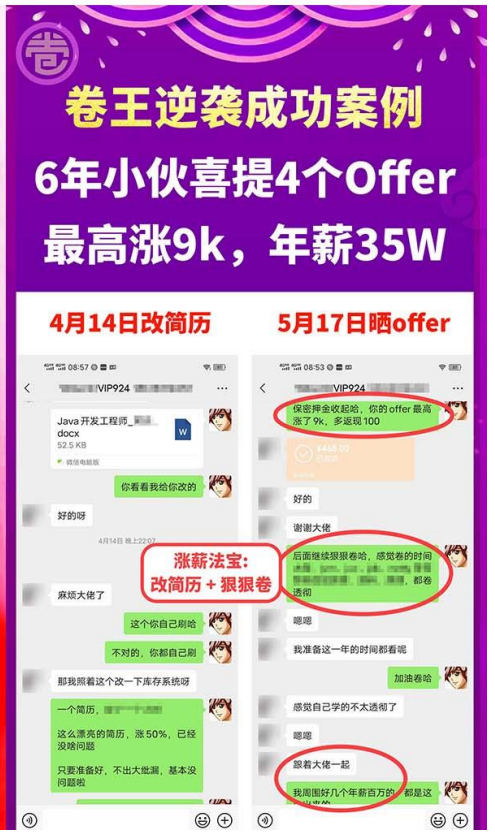

6年 经验小伙伴
喜收25K offer
3月12日改简历
12月1日晒offer
秘诀:
简历指导+狠狠卷
老师
咱们以这个项目为模板改哈
嗯好
项目有多少人，你带领多少人？
你工作几年了？
6年了
项目有多少人，你带领多少人？
语音交流一下
好
你方便吗
OK
不知道往哪个方向发展
改简历挪窝，20 涨到 25 k，高级开发，P7带的后端团队
任务重，道路远，并不是没有方向


7年经验卷王
薪酬涨30%
7月11日改简历
9月1日晒offer
秘诀:
改简历＋狠狠卷
咱们开始修改哈
要语音还是打字也行，还没到家
解决过最复杂的难题是啥？说3个哈
看看，有没有问题？
没问题
只能挪窝
下次你喊足一点哦
只要本事好，拿offer比较容易的
喊了，但是我流水太差了。。。我这家是补贴，银行不认这玩意，只能在上家的基础上加到30%封顶（特别优秀除外，我只是不错。。。）
入职了，最终并不能大幅涨薪，还是旧的太低，只涨了30%
恭喜一下
现在不比往年
能涨30%是大牛
拿到offer都算大牛
现在很多人投几百发，连个电话都没有
你已经很


4年经验卷王逆袭
被毕业后，反涨24W
7月改简历
8月30日晒offer
秘诀：
改简历＋狠狠卷

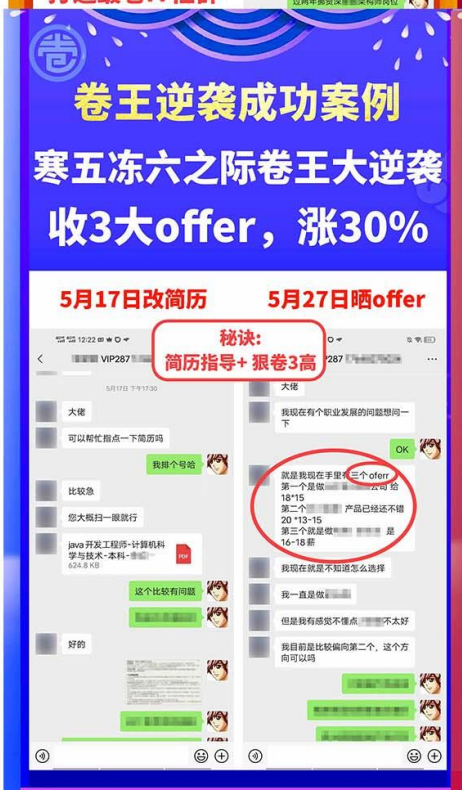

小伙5月份"被毕业"，改简历后
斩获顶级央企Offer
涨薪7000+
5月29日改简历
7月5日晒offer
秘诀：
简历指导＋狠卷3高


卷王逆袭成功案例
武汉6年喜收4个优质offer
最高的年薪35W
2月9日改简历
4月15日晒offer
面试法宝：
改简历＋实操


卷王逆袭成功案例
6年小伙喜提4个Offer
最高涨9k，年薪35W
4月14日改简历
5月17日晒offer
涨薪法宝：
改简历＋狠狠卷


卷王逆袭成功案例
5年经验小伙收2个offer
最高涨薪8k，年薪42W
5月9日改简历
5月30日晒offer
秘诀:
简历指导+ 狠卷3高
以此为样
大家狠狠卷
打造最卷IT社群


卷王逆袭成功案例
非全日制 6年经验卷王
喜提3个Offer，年包30W
5月9日改简历
5月18日晒offer
面试法宝:
改简历 + 狠狠卷


卷王逆袭成功案例
寒五冻六之际卷王大逆袭
收3大offer，涨30%
5月17日改简历
5月27日晒offer
秘诀:
简历指导+ 狠卷3高


卷王逆袭成功案例
4年卷王入职微软，涨50%
3月7日改简历
5月12日晒offer
涨薪法宝:
改简历 + 狠狠卷


4年小伙喜收百度、Boss直聘
等N个顶级Offer
最高涨幅100%
6月27日改简历
9月19日晒offer
秘诀:
改简历＋狠狠卷


卷王逆袭成功案例
4年卷王入收2个offer，涨50%
3月23日改简历
5月12日晒offer
涨薪法宝:
改简历＋狠狠卷


小伙大三暑期很焦虑
跟着尼恩卷一年
校招斩获顶级央企Offer
去年5月19日加入VIP群
今年7月5日晒offer
秘诀:
狠狠卷 书+视频


小伙高中学历
薪酬涨120%
5月6日改简历
7月22日晒offer
秘诀:
改简历＋狠狠卷


卷王逆袭成功案例
非全日制卷王 面试3家
收2个offer 涨薪30%
4月13日改简历
4月21日晒offer
面试法宝：
改简历＋面试题


5年卷王喜收2大Offer
最高涨5K
5月19日改简历
9月13日晒offer
秘诀：
改简历＋狠狠卷


卷王逆袭成功案例
3年经验卷王，涨60%
4月16日改简历
5月11日晒offer
涨薪法宝：
改简历＋狠狠卷


卷王逆袭成功案例
双非二本小伙春招大翻身
喜提9大offer
2月22日改简历
4月13日晒offer
面试法宝：
改简历＋IM实操
9大offer 最高年薪30万

修改简历找尼恩（资深简历优化专家）

- 如果面试表达不好，尼恩会提供 简历优化指导
- 如果项目没有亮点，尼恩会提供 项目亮点指导
- 如果面试表达不好，尼恩会提供 面试表达指导

作为 40 岁老架构师，尼恩长期承担技术面试官的角色：

- **从业以来，“阅历”无数，对简历有着点石成金、改头换面、脱胎换骨**的指导能力。
- **尼恩指导过刚刚就业的小白，也指导过 P8 级的老专家，都指导他们上岸。**

如何联系尼恩。尼恩微信，请参考下面的地址：

语雀：https://www.yuque.com/crazymakercircle/gkkw8s/khigna
码云：https://gitee.com/crazymaker/SimpleCrayIM/blob/master/疯狂创客圈总目录.md