**从0到1整体认知分布式系统**

**概要：**

1. 分布式架构的发展历史与背景
2. **如何着手架构一套分布示式系统**
3. 分布式架构所带来的成本与风险

**分布式架构的发展历史与背景**

场景一：

一家做政务OA系统的公司老板发现跟竞争对手比发现自己的系统的架构不是分布示的，找到技术负责人问，把系统架构升级成分布示架构要多长时间？技术负责人网上查了查 dubbo官网看了看 Demo 这不很简单吗，拍着胸脯一个月能升级好。

现在我的问题是：这位技术理在改造过程中可能会遇到什么风险和问题？

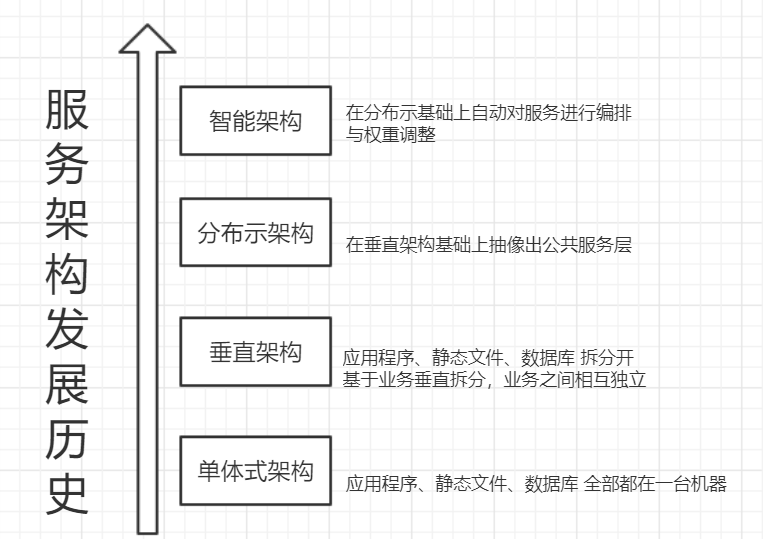
1. 新功能和旧BUG的问题
2. 业务完整性的问题
3. 团队协作方式转变
4. 开发人员技能提升
5. 系统交付方式转变

这些问题解决涉及业务部门及整个技术部门（开发、测试、运维）协商与工作标准的制定。业务相关问题暂不做讨论,技术架构上应该要清楚自己的职责是，如何通过技术手段把业务波动降至最低、开发成本最低、实施风险最低？

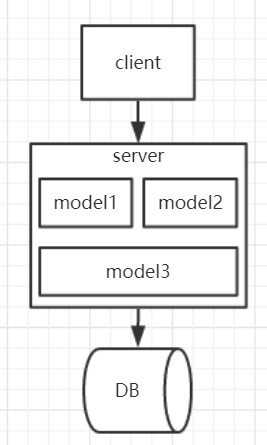
要解决这些问题的前提之一就是要对分布式架构有整体的认知，今天课程就通过三个知识点讲解来提高大家对分布示架构的整体认知：

1. 系统架构的发展历史
2. 一套分布式系统的组成
3. 分布式架构所带来的成本与风险

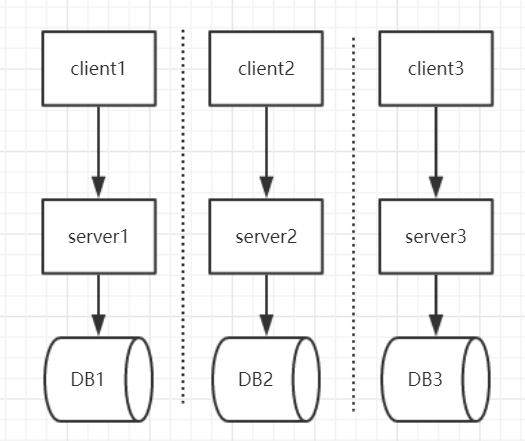
**架构的发展历史：**



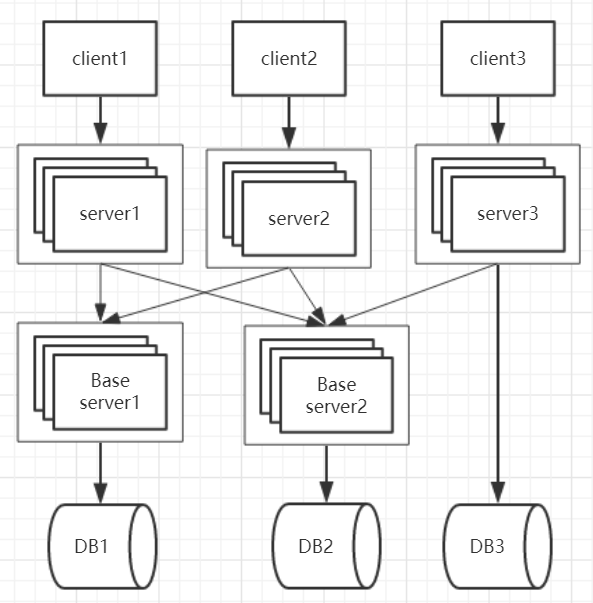
**单体式架构：**



**垂直架构:**



**分布示架构：**



**如何着手架构一套分布示式系统？**

提问：实现一个分布示框架最核心功能是什么?

**RPC远程调用技术：**

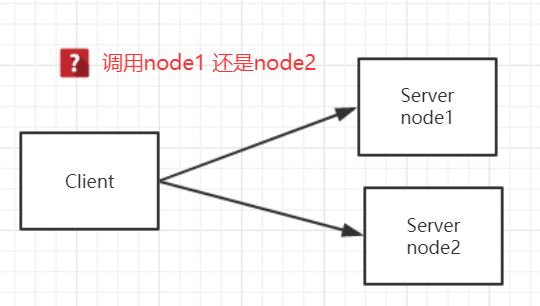


大家知道的 有哪些远程调用的 方式？拿几个大家比较熟悉的来举例：RMI 、Web Service、Http

|  |  |  |  |
| --- | --- | --- | --- |
| 协议 | 描述 | 优点 | 缺点 |
| RMI | JAVA 远程方法调用、使用原生二进制方式进行序列化 | 简单易用、SDK支持，提高开发效率 | 不支持跨语言 |
| Web Service | 比较早系统调用解决方案 ，跨语言, 其基于WSDL 生成 SOAP 进行消息的传递。 | SDK支持、跨语言 | 实现较重，发布繁琐 |
| Http | 采用htpp +json 实现 | 简单、轻量、跨语言 | 不支持SDK |

基于比较上述比较，大家会选择哪个方案，综合考虑 RMI是比较合适的方案，基本没有学习成本。而跨语言问题基本可以勿略。

如果服务端不是单个的话，这个方案差点我就用了。实际上服务端是多个的 ，好了新的问题又来了。



**2、负载均衡：**这么多个机器调用哪一台?

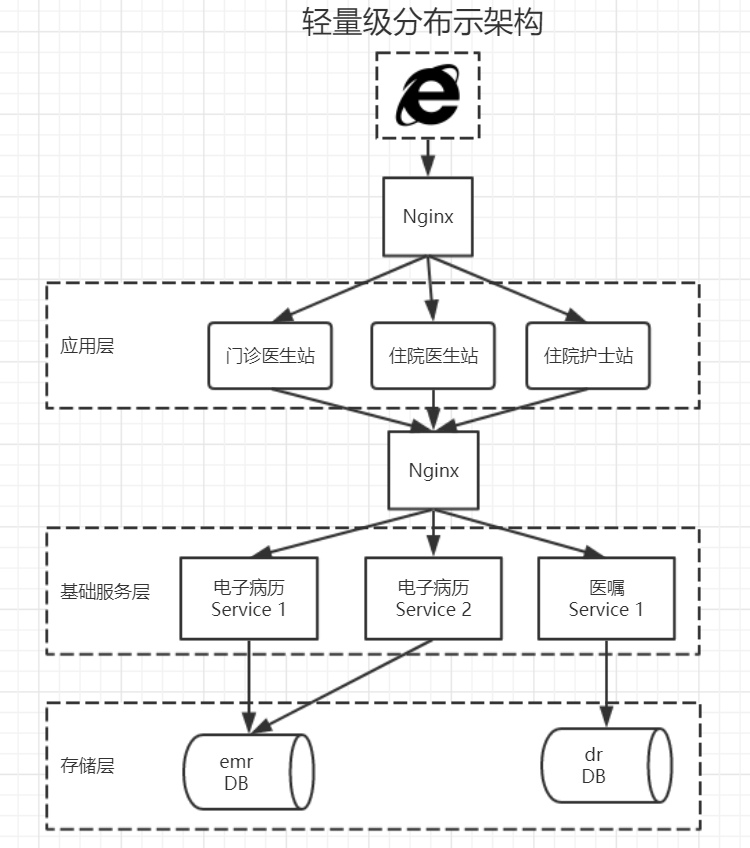
**3、容错：**如果调用其中一台调用出错了怎么办？

**4、服务配置：**服务端地址配置在哪里？

**5、健康检测：**服务关宕机或恢复后怎么办？

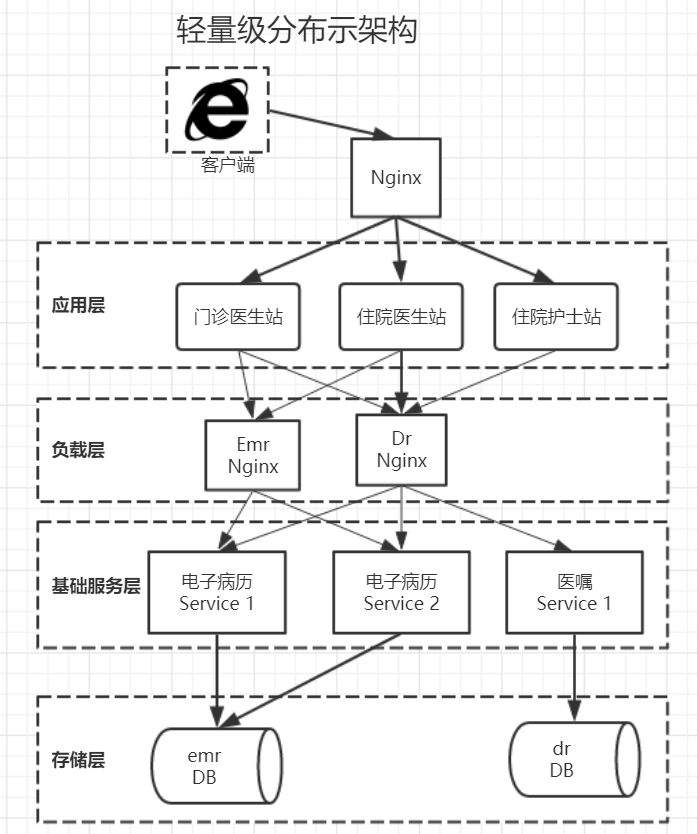
这些功能怎么解决呢？一个一个的去编码实现么？。有没有现成的方案可以直接实现这些功能？Nginx完全支持这些功能。所以企业在做轻量级架构 会采用 Http+Nginx 方式。

【图】：



同学可能就要问了，Nginx那不就单点了吗？Nginx本身还可以根据业务进行垂直拆分，变更之后如下图。

【图】：



**Http+Nginx 方案总结：**

**优点：**简单快速、几乎没有学习成本

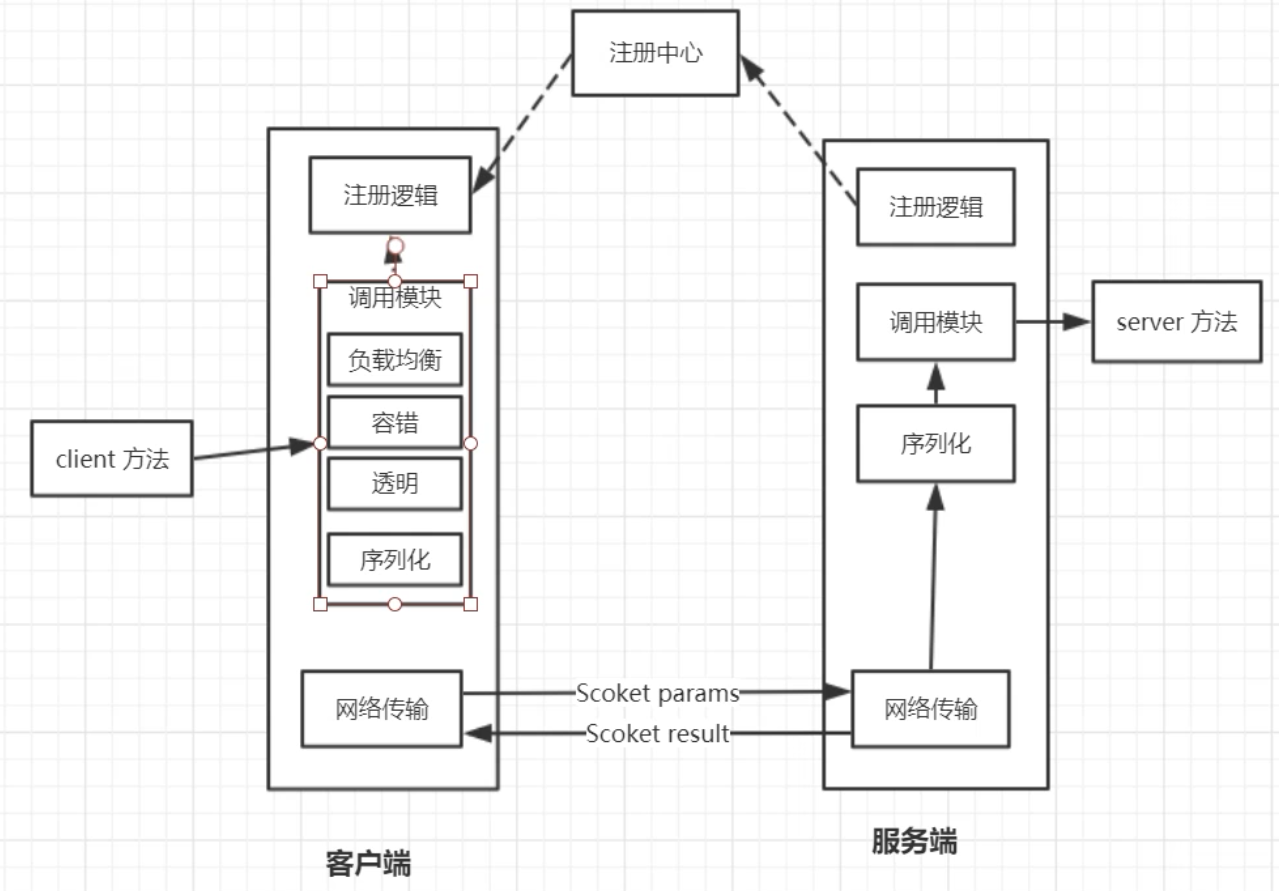
**适用场景：**轻量级分布式系统、局部分布式架构。

**瓶颈：**Nginx中心负载、Http传输、JSON序列化、开发效率、运维效率。

基于瓶颈考虑大型系统需要一个更加专业的方案，该方案必须做到以下几点：

1. 去中心化，客户端直连服务端
2. 动态注册和发现服务
3. 软负载均衡实现
4. 高效稳定的网络传输
5. 高效可容错的序列化

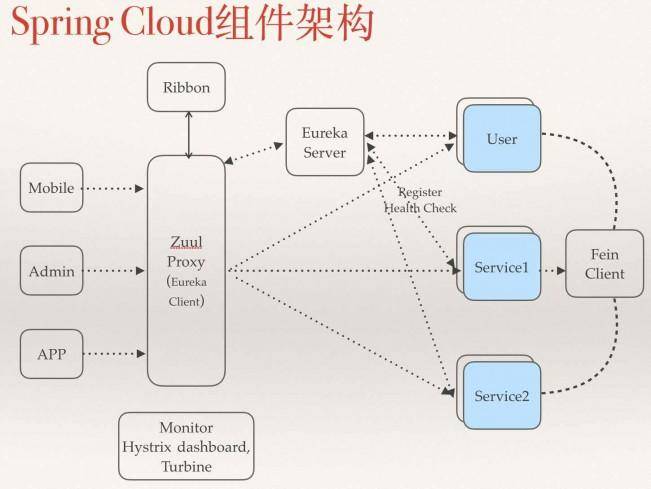
**其对应的功能组件如下：**



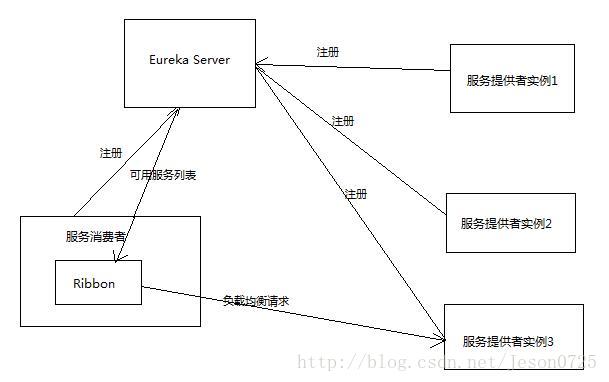
1. 注册中心逻辑：
2. 服务端动态注册服务提供者信息
3. 客户端从注册中心接收服务提供者信息，并存储至本地缓存
4. 注册中心实时监听提供者状态，如果变更将即时通知客户端
5. 调用逻辑：
6. 负载均衡
7. 容错
8. 对服务调用者透明
9. 传输模块：
10. minal,servlet 容器、netty
11. 序列化模块：
12. kryo、hessian、java、protobuf、JSON、XML

**主流的分布式框架比较**

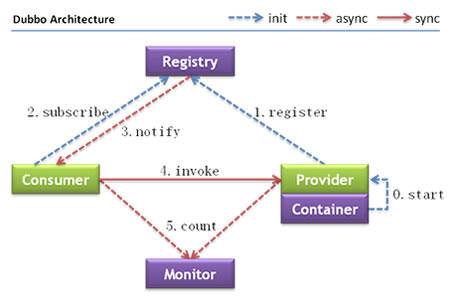
**Spring Cloud Zuul**



Spring Cloud Ribbon



**Dubbo**



**设计一个RPC框架**

基于公司背景来设计你的框架：

初发点：这个框架的应用背景，及好处

**框架结构图：**包含哪些模块？

**结构图描述**：文件描述结构图中的元素

**协作流程图**：初始化流程、调用流程？

调用流程的描述

分布式架构所带来的成本与风险

**分布式事物：**

分布式事物是指一个操作，分成几个小操作在多个服务器上执行，要么多成功，要么多失败这些分布事物要做的

**不允许服务有状态（stateless service）**

无状态服务是指对单次请求的处理，不依赖其他请求，也就是说，处理一次请求所需的全部信息，要么都包含在这个请求里，要么可以从外部获取到（比如说数据库），服务器本身不存储任何信息。

**服务依懒关系复杂**

服务 A --> B--> C 那和服务C 的修改 就可能会影响 B 和C，事实上当服务越来 越多的时候，C的变动将会越来越困难。

**部署运维成本增加**

不用说了，相比之前几个节点，运维成本的增加必须的。

**源码管理成本增加：**

原本一套或几套源码现在拆分成几十个源码库，其中分支、tag都要进行相应管理。

**如何保证系统的伸缩性：**

伸缩性是指,当前服务器硬件升级后或新增服务器处理能力就能相对应的提升。

**分布式会话：**

此仅针对应用层服务，不能将Session 存储在一个服务器上。

**分布式JOB**

通常定时任务只需要在一台机器上触发执行，分布式的情况下在哪台执行呢？

**Dubbo学习路线：**

简单入门与基本概念--->常规应用与架构原理--->特殊应用与扩展实现