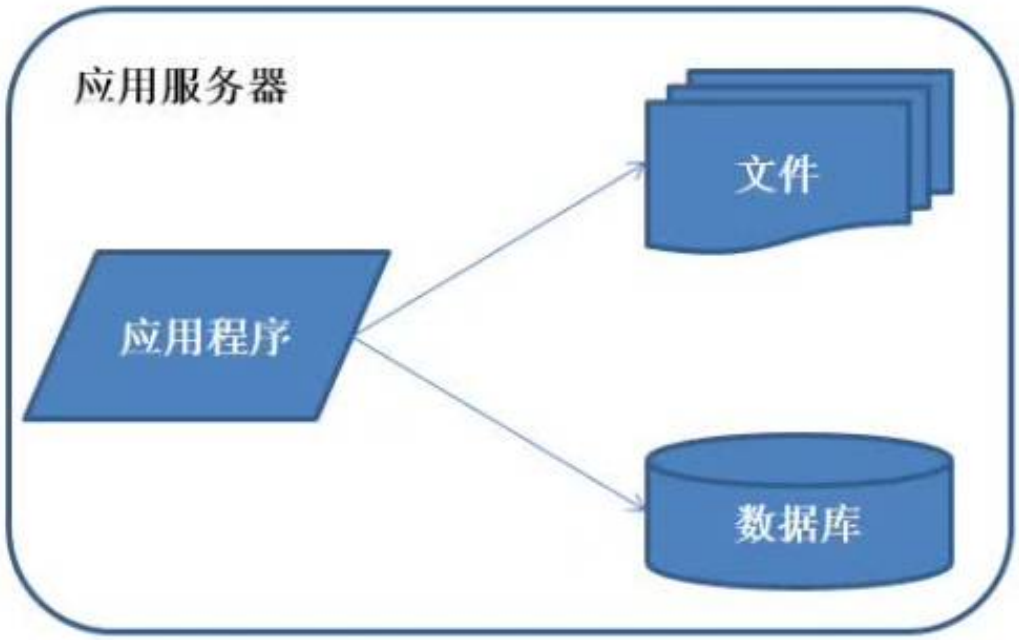
图说分布式架构的发展和演进

佚名 架构师技术联盟

编注: 架构决定的系统的稳定性，扩展性和并发性，架构的演进是从简单到复杂，从单一到复合持续改进的过程，也是经验的积累和技术的结晶。

初始阶段架构

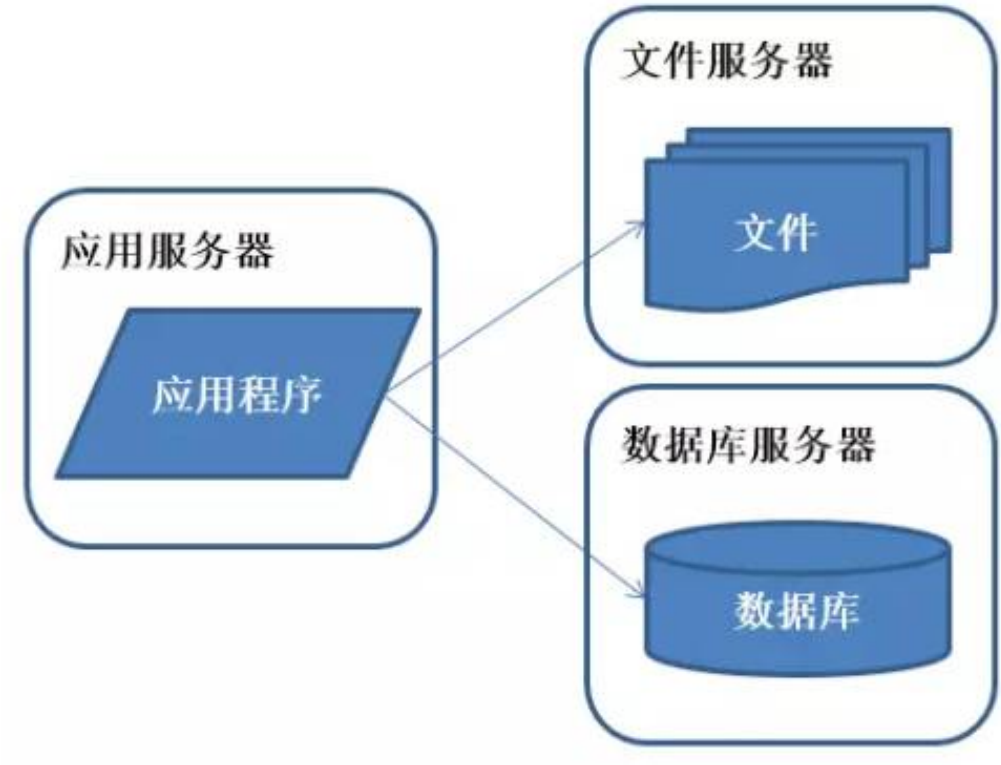


初始阶段的小型系统、应用程序、数据库、文件等所有的资源都在一台服务器上。通俗称为LAMP。

特征：应用程序、数据库、文件等所有的资源都在一台服务器上。

描述：通常服务器操作系统使用linux，应用程序使用PHP开发，然后部署在Apache上，数据库使用Mysql，汇集各种免费开源软件以及一台廉价服务器就可以开始系统的发展之路了。

应用服务和数据服务分离

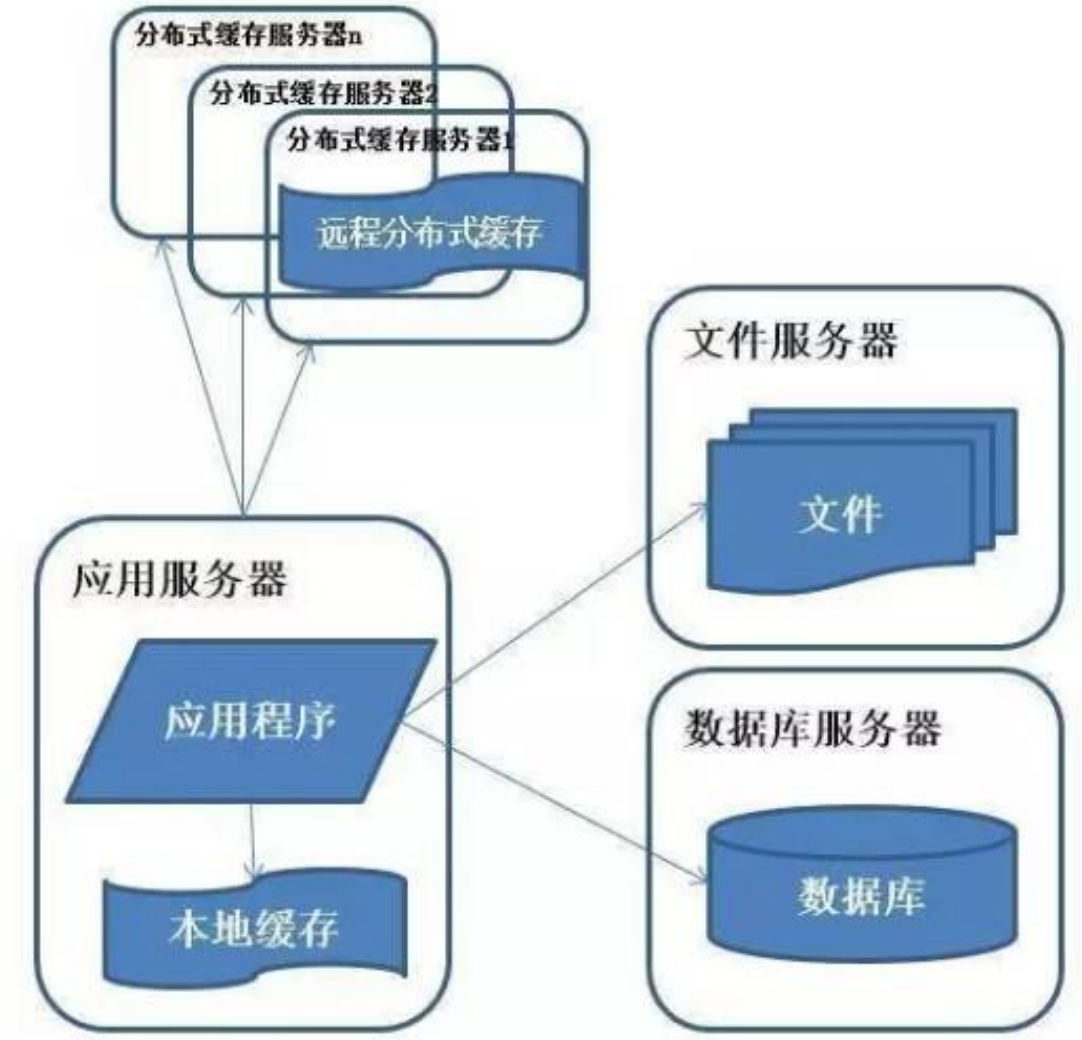


好景不长，发现随着系统访问量的再度增加，webserver机器的压力在高峰期会上升到比较高，这个时候开始考虑增加一台webserver。

特征：应用程序、数据库、文件分别部署在独立的资源上。

描述：数据量增加，单台服务器性能及存储空间不足，需要将应用和数据分离，并发处理能力和数据存储空间得到了很大改善。

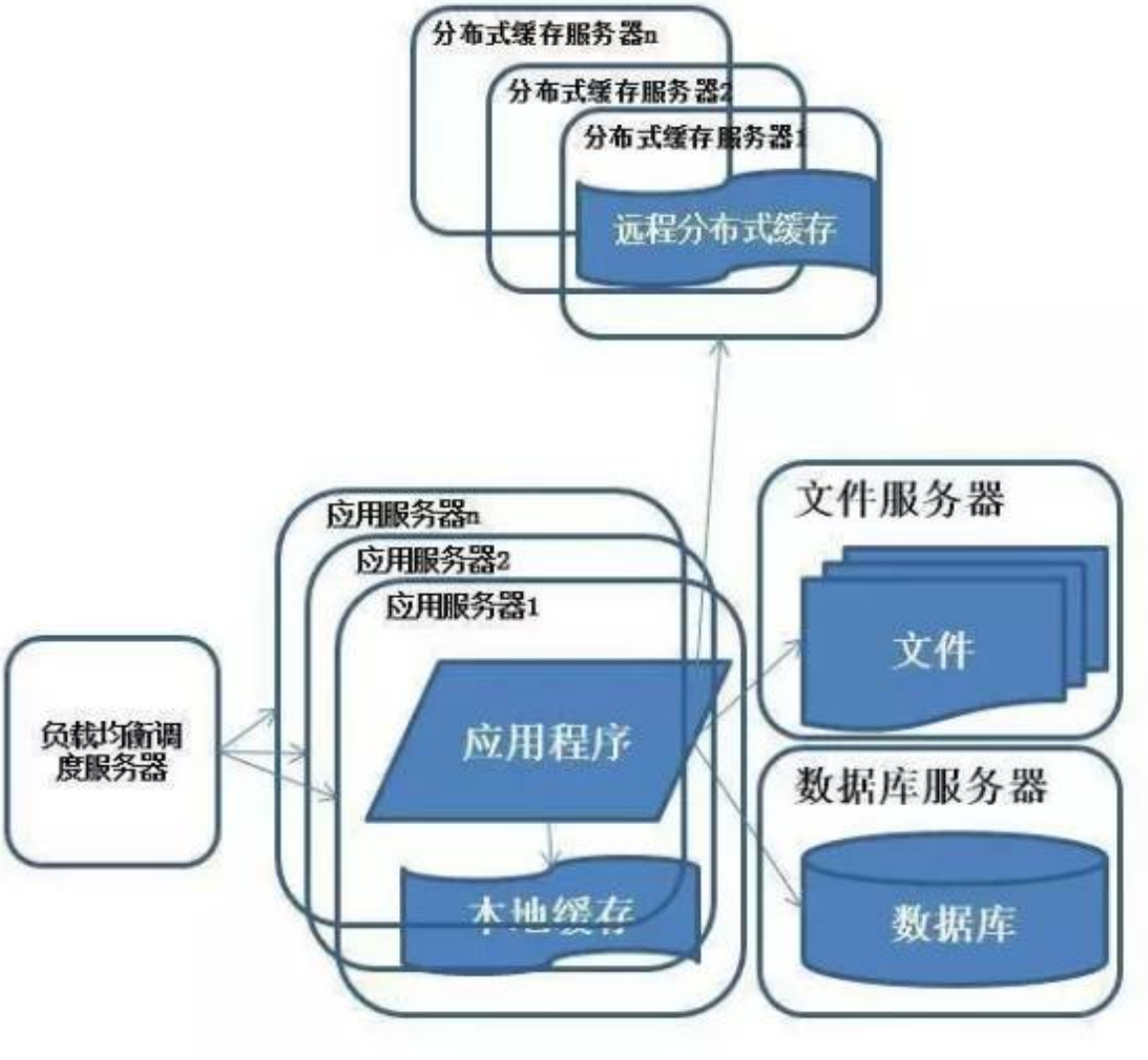
使用缓存改善性能



特征：数据库中访问较集中的一小部分数据存储在缓存服务器中，减少数据库的访问次数，降低数据库的访问压力。

描述：系统访问特点遵循二八定律，即80%的业务访问集中在20%的数据上。缓存分为本地缓存和远程分布式缓存，本地缓存访问速度更快但缓存数据量有限，同时存在与应用程序争用内存的情况。

使用应用服务器集群

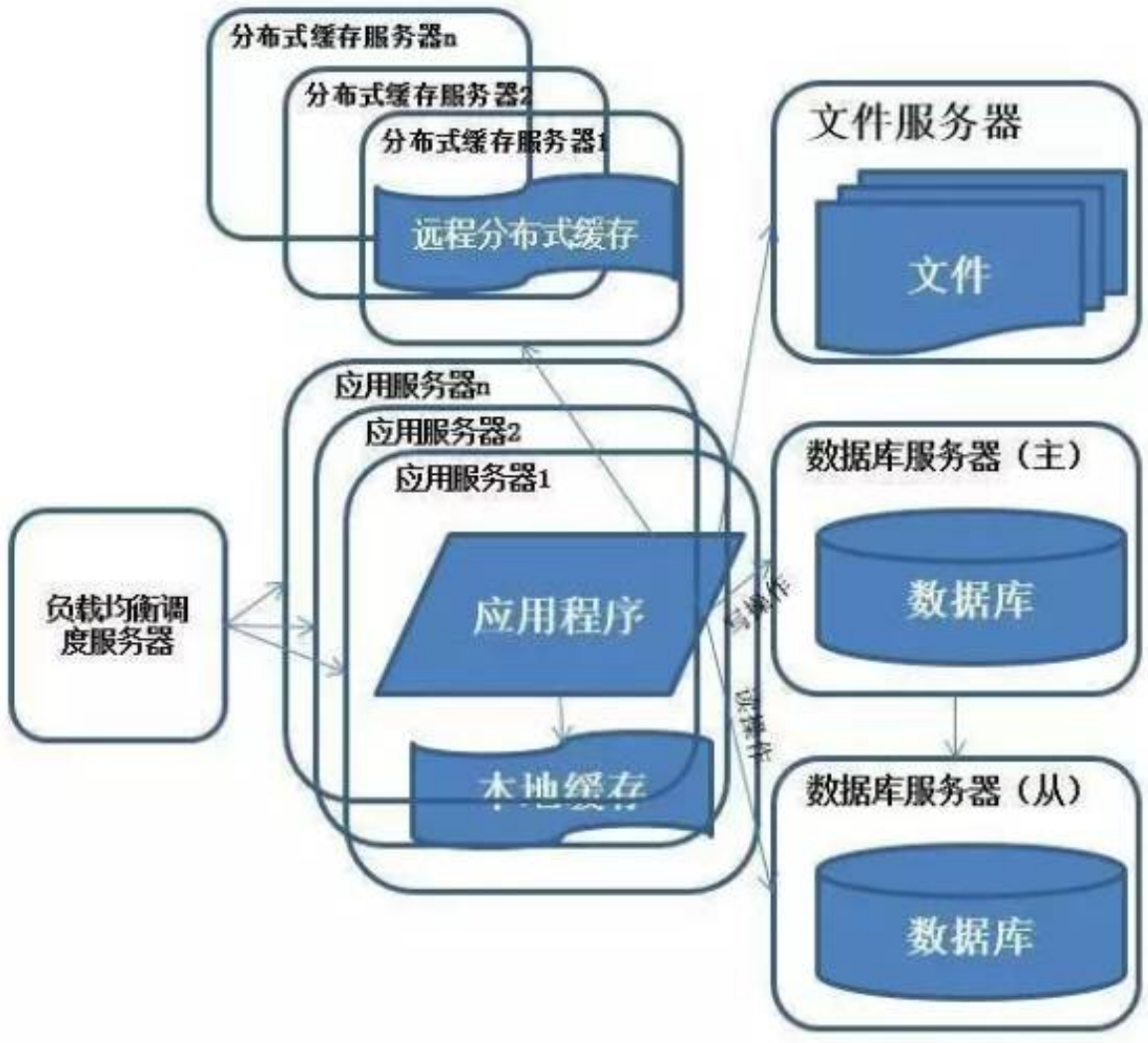


在做完分库分表这些工作后，数据库上的压力已经降到比较低了，又开始过着每天看着访问量暴增的幸福生活了，突然有一天，发现系统的访问又开始有变慢的趋势了，这个时候首先查看数据库，压力一切正常，之后查看webserver，发现apache阻塞了很多的请求，而应用服务器对每个请求也是比较快的，看来是请求数太高导致需要排队等待，响应速度变慢。

特征：多台服务器通过负载均衡同时向外部提供服务，解决单台服务器处理能力和存储空间上限的问题。

描述：使用集群是系统解决高并发、海量数据问题的常用手段。通过向集群中追加资源，提升系统的并发处理能力，使得服务器的负载压力不再成为整个系统的瓶颈。

数据库读写分离

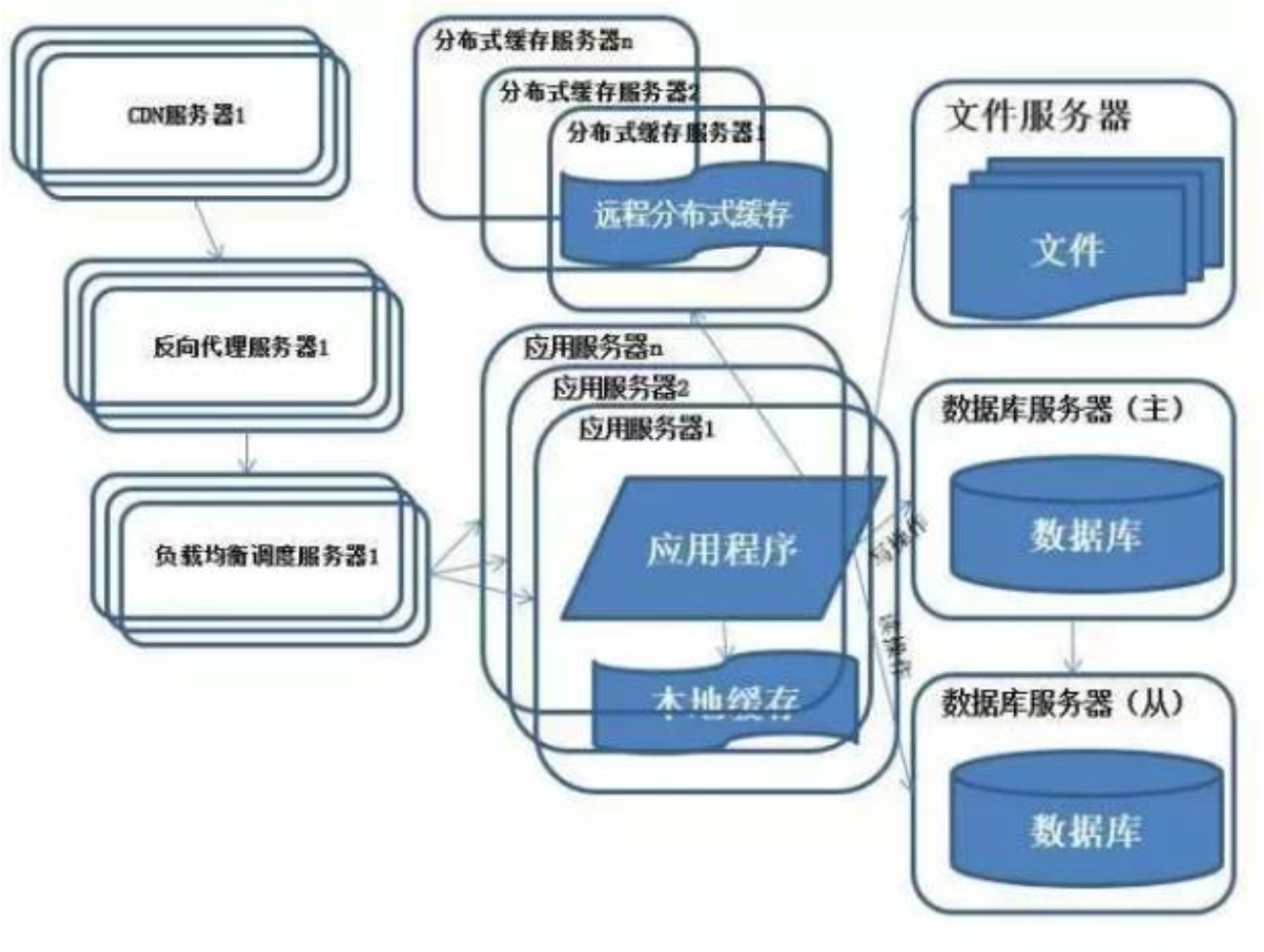


享受了一段时间的系统访问量高速增长的幸福后，发现系统又开始变慢了，这次又是什么状况呢，经过查找，发现数据库写入、更新的这些操作的部分数据库连接的资源竞争非常激烈，导致了系统变慢。

特征：多台服务器通过负载均衡同时向外部提供服务，解决单台服务器处理能力和存储空间上限的问题。

描述：使用集群是系统解决高并发、海量数据问题的常用手段。通过向集群中追加资源，使得服务器的负载压力不在成为整个系统的瓶颈。

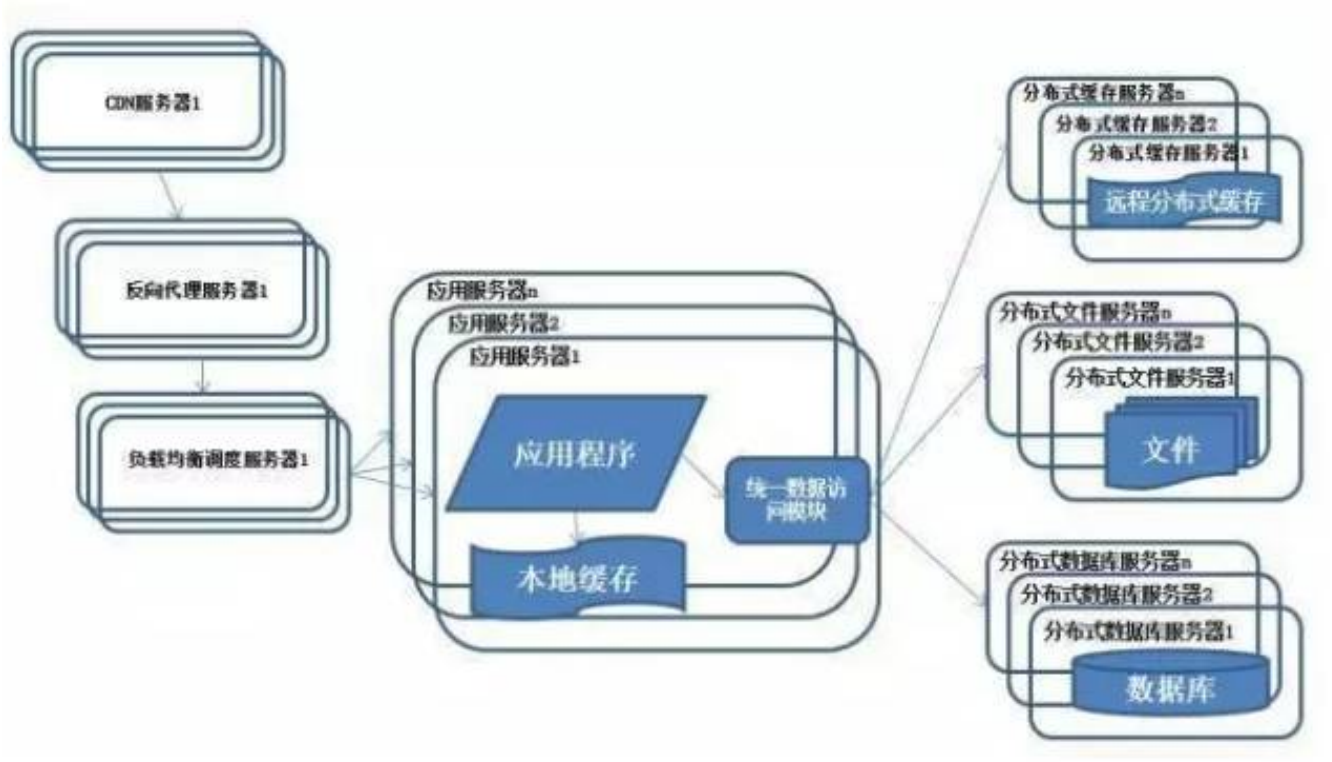
反向代理和CDN加速



特征：采用CDN和反向代理加快系统的 访问速度。

描述：为了应付复杂的网络环境和不同地区用户的访问，通过CDN和反向代理加快用户访问的速度，同时减轻后端服务器的负载压力。CDN与反向代理的基本原理都是缓存。

分布式文件系统和分布式数据库

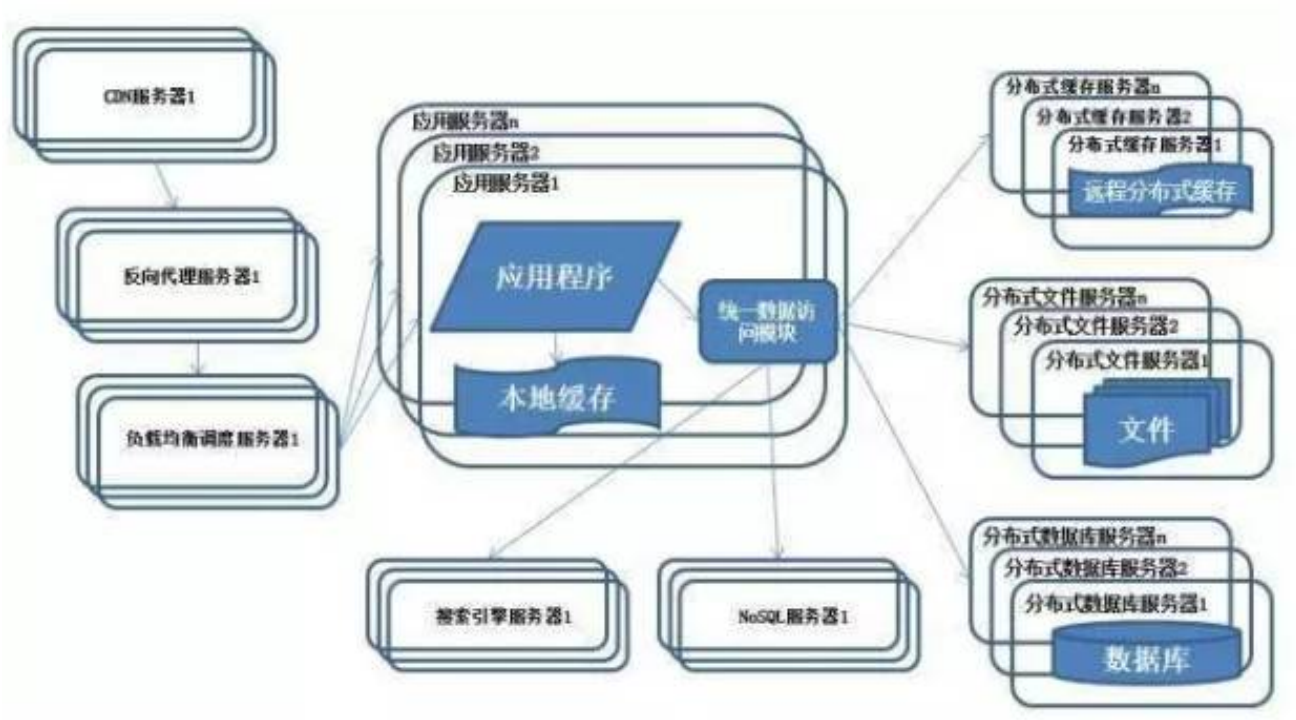


随着系统的不断运行，数据量开始大幅度增长，这个时候发现分库后查询仍然会有些慢，于是按照分库的思想开始做分表的工作。

特征：数据库采用分布式数据库，文件系统采用分布式文件系统。

描述：任何强大的单一服务器都满足不了大型系统持续增长的业务需求，数据库读写分离随着业务的发展最终也将无法满足需求，需要使用分布式数据库及分布式文件系统来支撑。分布式数据库是系统数据库拆分的最后方法，只有在单表数据规模非常庞大的时候才使用，更常用的数据库拆分手段是业务分库，将不同的业务数据库部署在不同的物理服务器上。

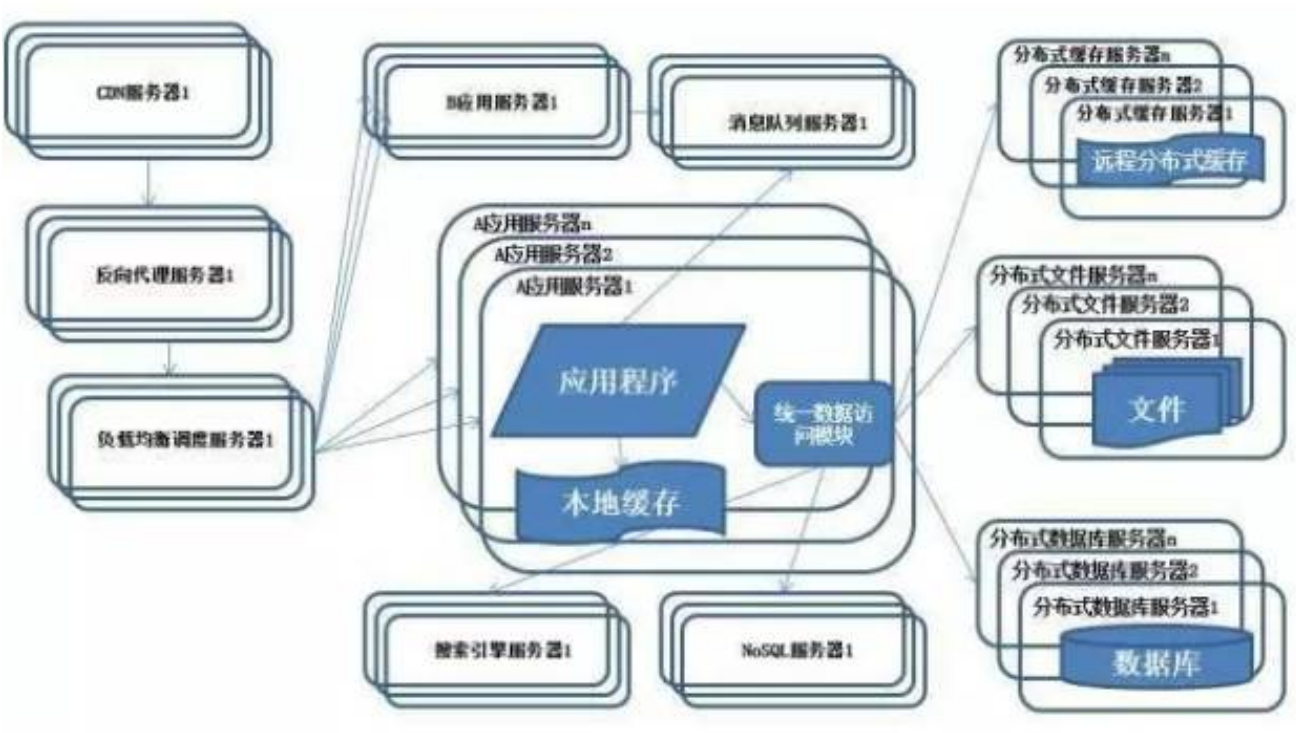
使用NoSQL和搜索引擎



特征：系统引入NoSQL数据库及搜索引擎。

描述：随着业务越来越复杂，对数据存储和检索的需求也越来越复杂，系统需要采用一些非关系型数据库如NoSQL和分数据库查询技术如搜索引擎。应用服务器通过统一数据访问模块访问各种数据，减轻应用程序管理诸多数据源的麻烦。

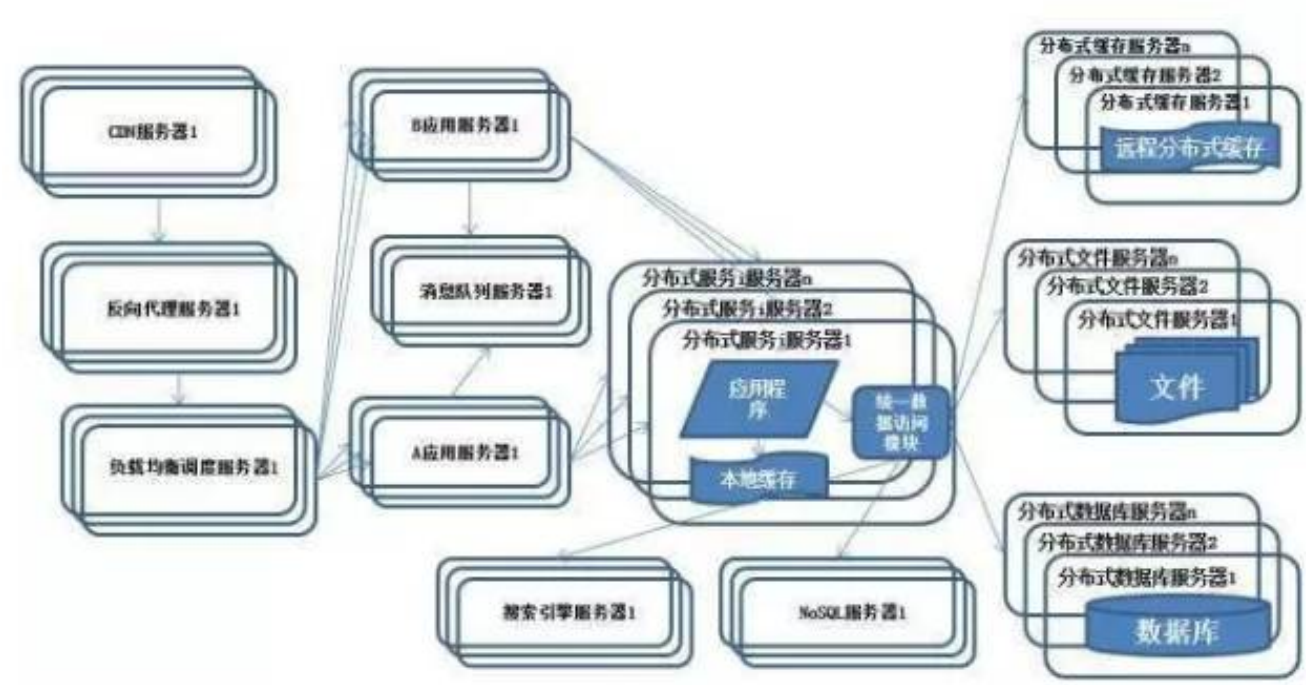
业务拆分



特征：系统上按照业务进行拆分改造，应用服务器按照业务区分进行分别部署。

描述：为了应对日益复杂的业务场景，通常使用分而治之的手段将整个系统业务分成不同的产品线，应用之间通过超链接建立关系，也可以通过消息队列进行数据分发，当然更多的还是通过访问同一个数据存储系统来构成一个关联的完整系统。纵向拆分：将一个大应用拆分为多个小应用，如果新业务较为独立，那么就直接将其设计部署为一个独立的Web应用系统纵向拆分相对较为简单，通过梳理业务，将较少相关的业务剥离即可。横向拆分：将复用的业务拆分出来，独立部署为分布式服务，新增业务只需要调用这些分布式服务横向拆分需要识别可复用的业务，设计服务接口，规范服务依赖关系。

分布式服务



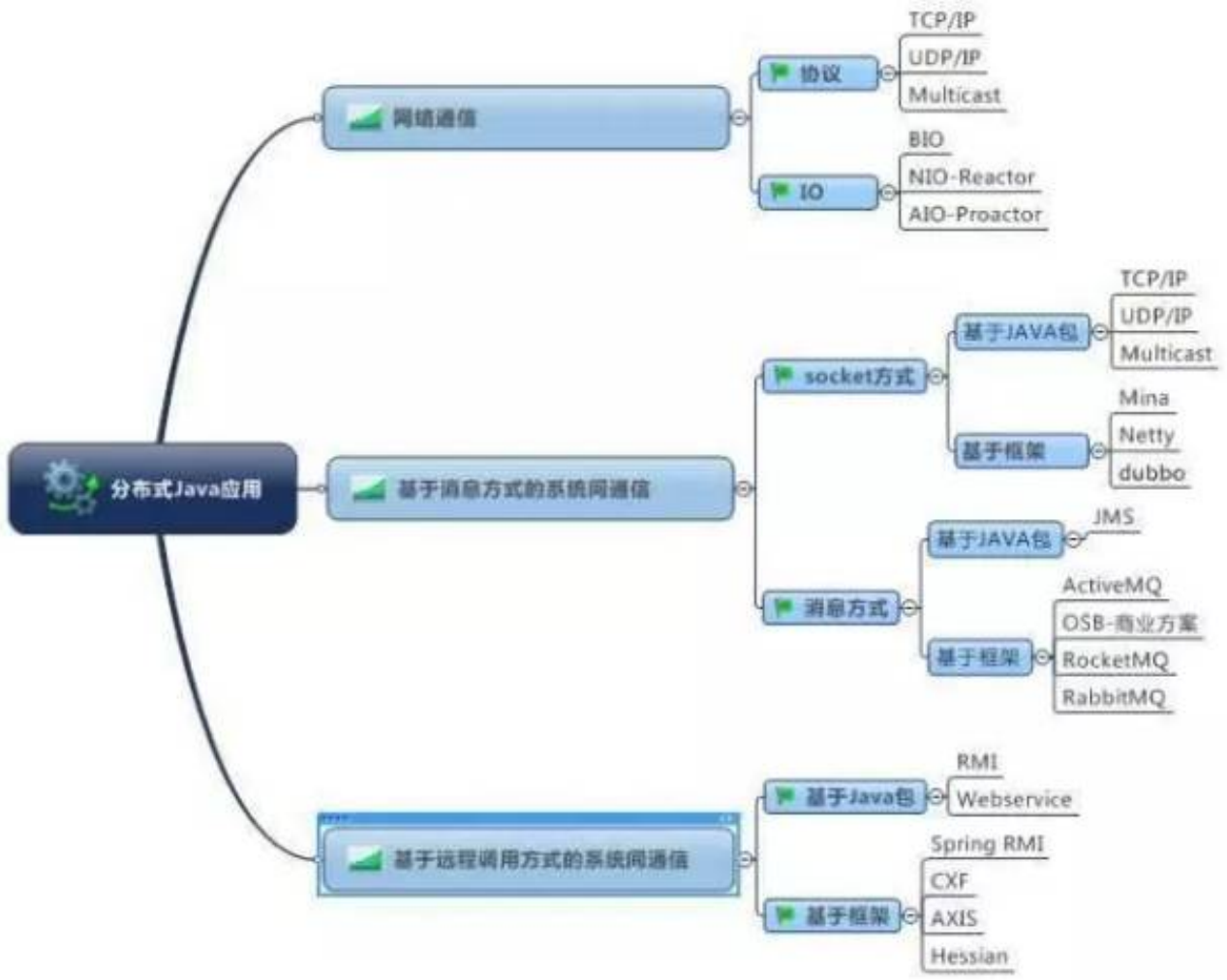
特征：公共的应用模块被提取出来，部署在分布式服务器上供应用服务器调用。

描述：随着业务越拆越小，应用系统整体复杂程度呈指数级上升，由于所有应用要和所有数据库系统连接，最终导致数据库连接资源不足，拒绝服务。

分布式服务应用会面临哪些问题?

(1) 当服务越来越多时，服务URL配置管理变得非常困难，F5硬件负载均衡器的单点压力也越来越大。(2) 当进一步发展，服务间依赖关系变得错踪复杂，甚至分不清哪个应用要在哪个应用之前启动，架构师都不能完整的描述应用的架构关系。(3) 接着，服务的调用量越来越大，服务的容量问题就暴露出来，这个服务需要多少机器支撑？什么时候该加机器？(4) 服务多了，沟通成本也开始上升，调某个服务失败该找谁？服务的参数都有什么约定？ (5) 一个服务有多个业务消费者，如何确保服务质量？(6) 随着服务的不停升级，总有些意想不到的事发生，比如cache写错了导致内存溢出，故障不可避免，每次核心服务一挂，影响一大片，人心慌慌，如何控制故障的影响面？服务是否可以功能降级？或者资源劣化?

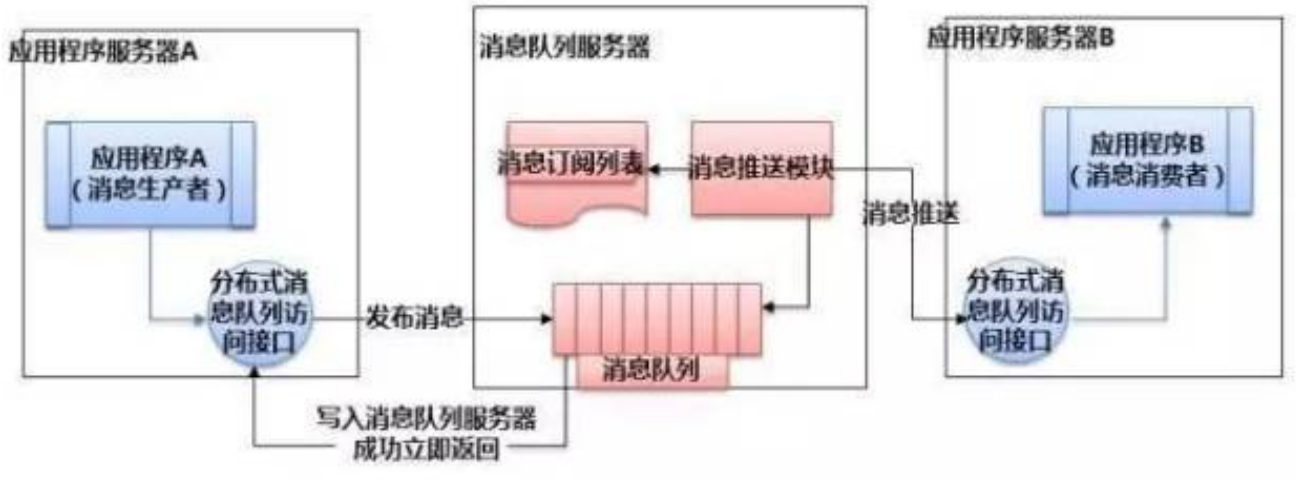
Java分布式应用技术基础



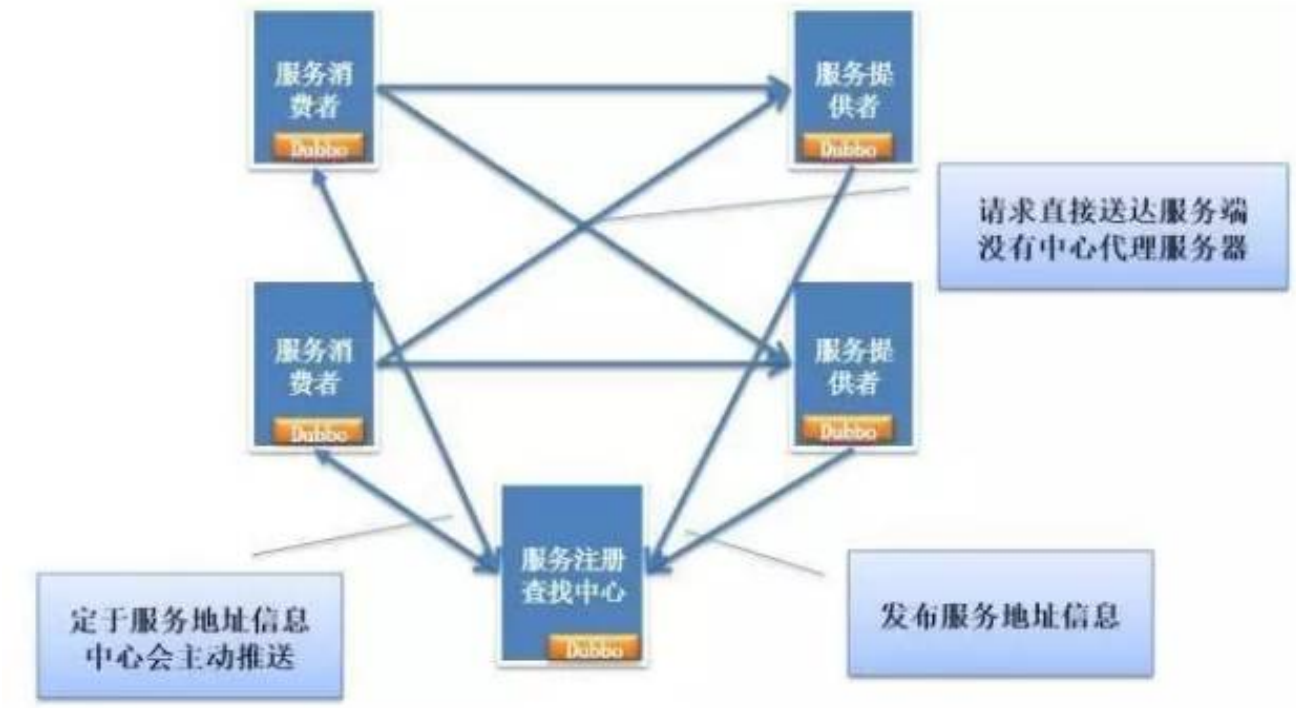
分布式服务下的关键技术：消息队列架构

消息队列通过消息对象分解系统耦合性，不同子系统处理同一个消息。

分布式服务下的关键技术：消息队列原理

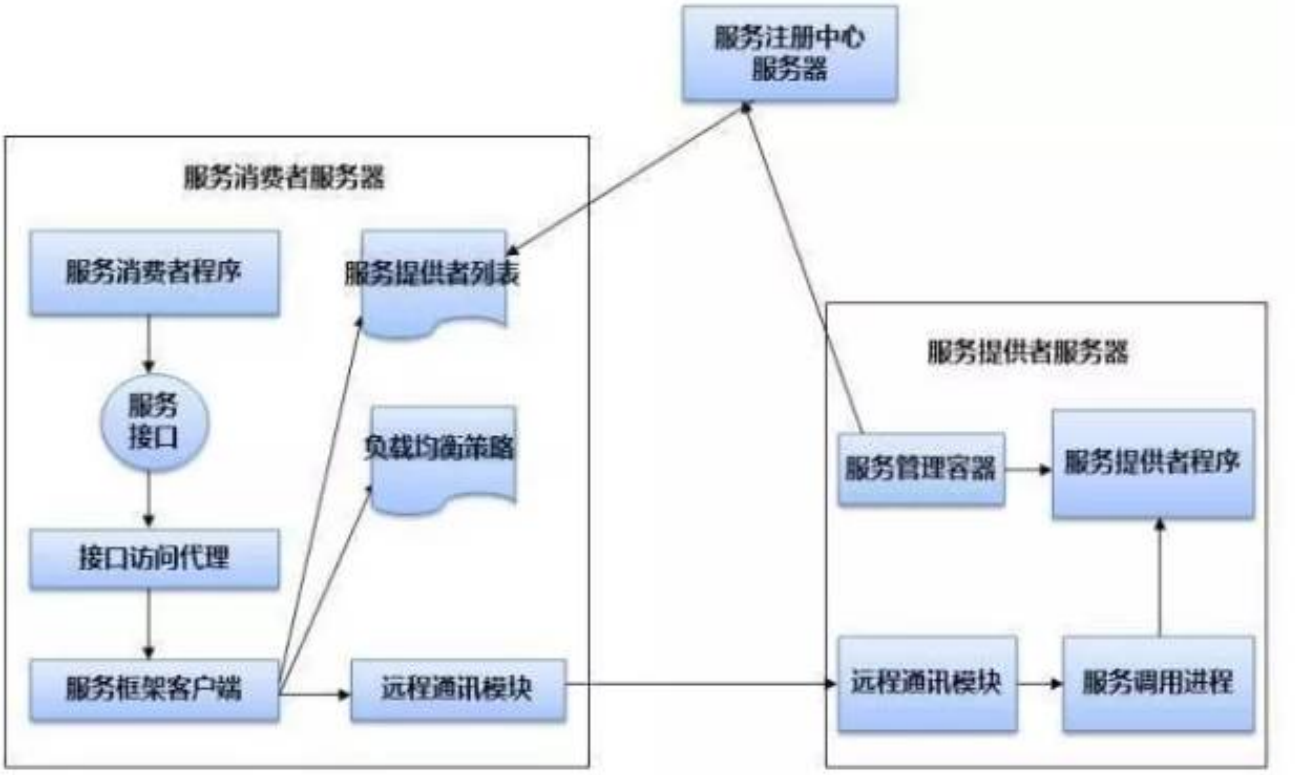


分布式服务下的关键技术：服务框架架构

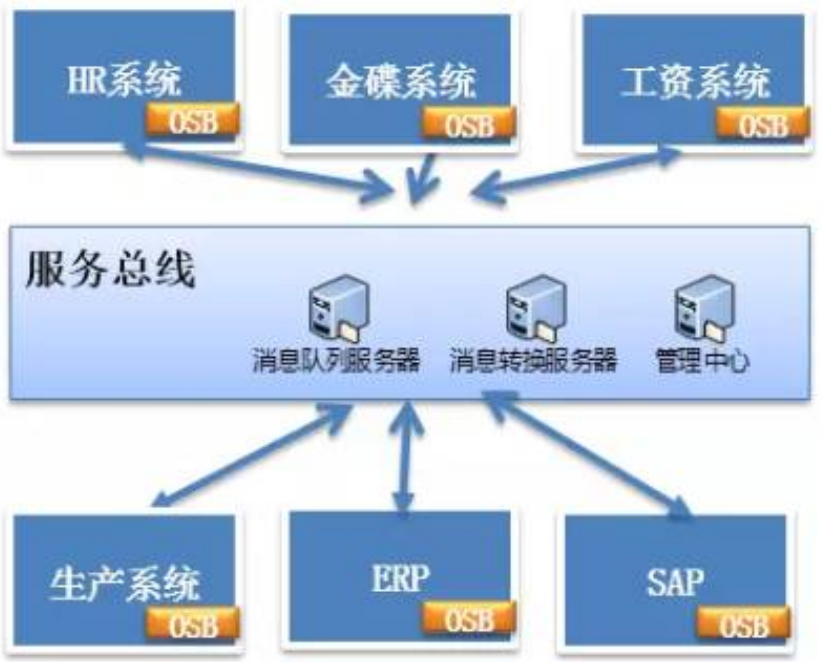


服务框架通过接口分解系统耦合性，不同子系统通过相同的接口描述进行服务启用服务框架是一个点对点模型服务框架面向同构系统适合：移动应用、互联网应用、外部系统。

分布式服务下的关键技术：服务框架原理

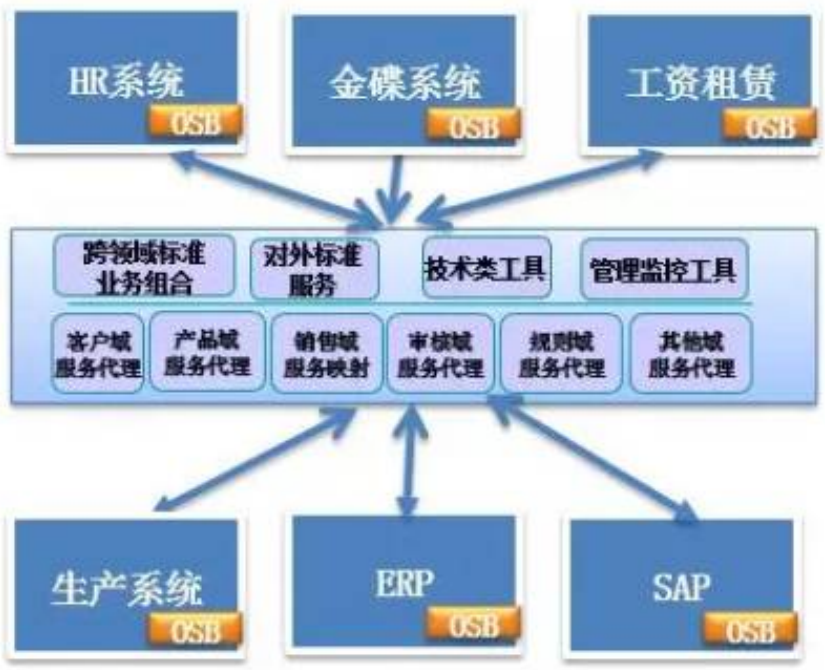


分布式服务下的关键技术：服务总线架构



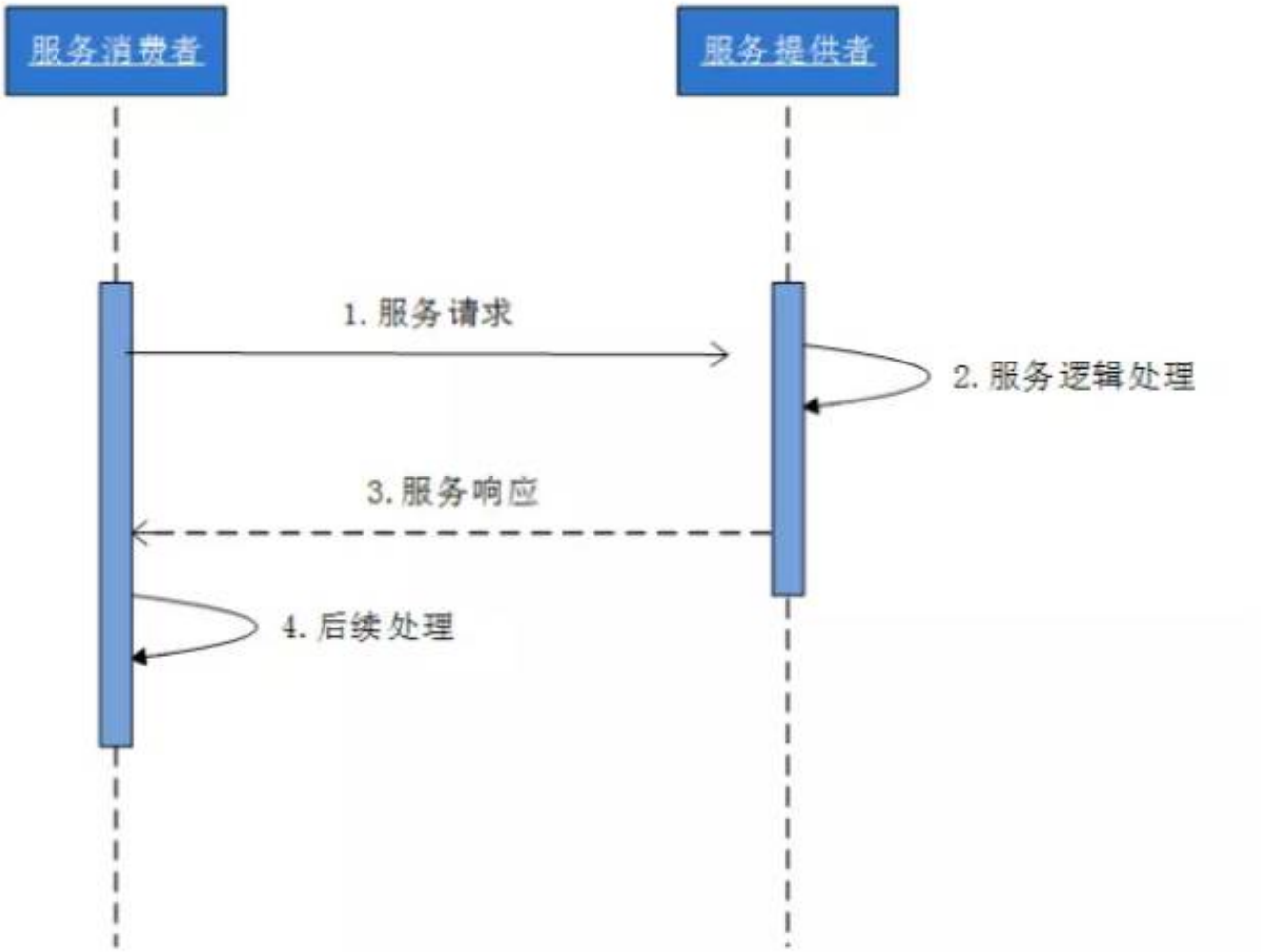
服务总线同服务框架一样，均是通过接口分解系统耦合性，不同子系统通过相同的接口描述进行服务启用服务总线是一个总线式的模型服务总线面向同构、异构系统适合：内部系统。

分布式服务下的关键技术：服务总线原理

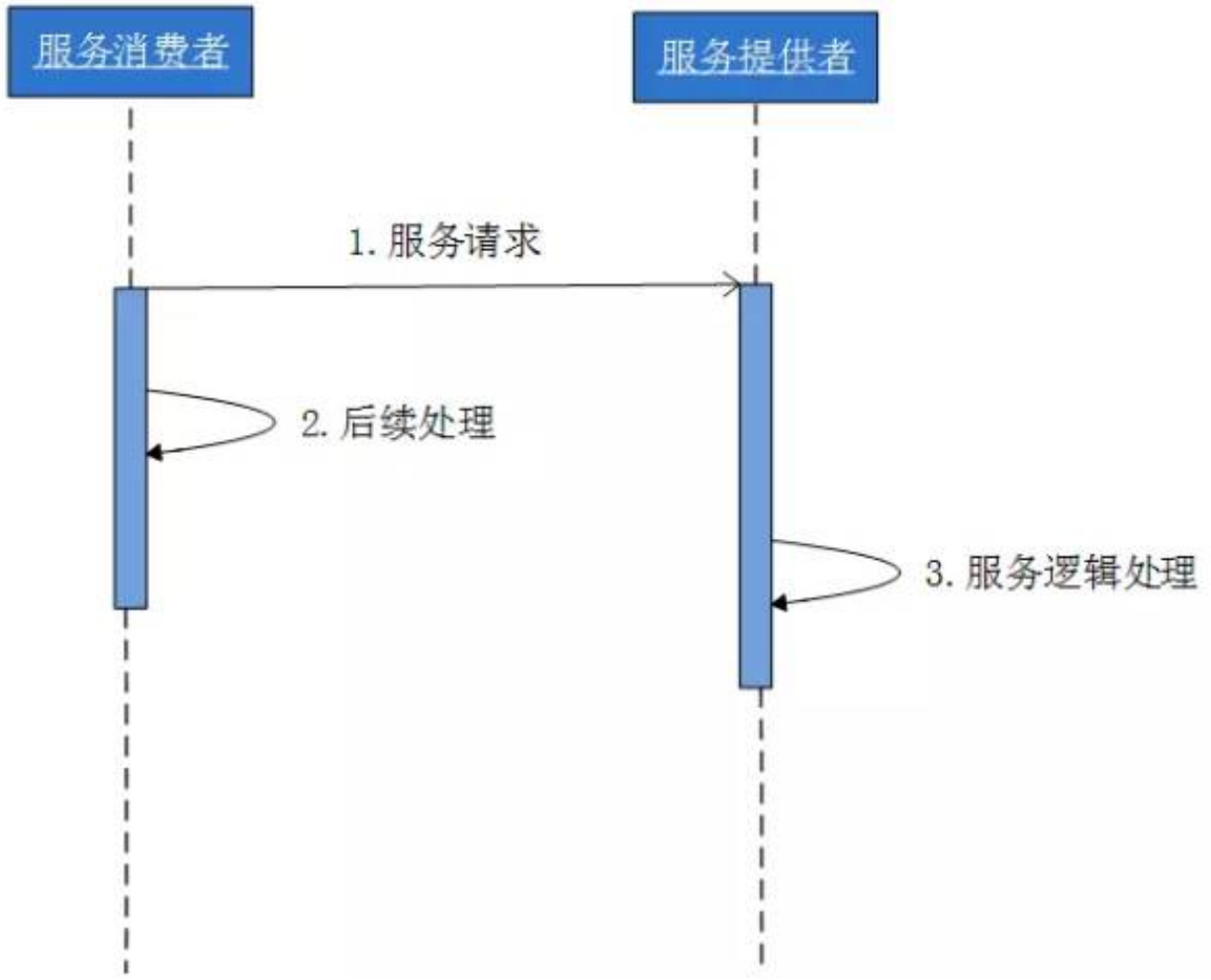


分布式架构下系统间交互的5种通信模式request/response模式（同步模式）：客户端发起请求一直阻塞到服务端返回请求为止。Callback（异步模式）：客户端发送一个RPC请求给服务器，服务端处理后再发送一个消息给消息发送端提供的callback端点，此类情况非常合适以下场景：A组件发送RPC请求给B，B处理完成后，需要通知A组件做后续处理。Future模式：客户端发送完请求后，继续做自己的事情，返回一个包含消息结果的Future对象。客户端需要使用返回结果时，使用Future对象的.get(),如果此时没有结果返回的话，会一直阻塞到有结果返回为止。Oneway模式：客户端调用完继续执行，不管接收端是否成功。Reliable模式：为保证通信可靠，将借助于消息中心来实现消息的可靠送达，请求将做持久化存储，在接收方在线时做送达，并由消息中心保证异常重试。

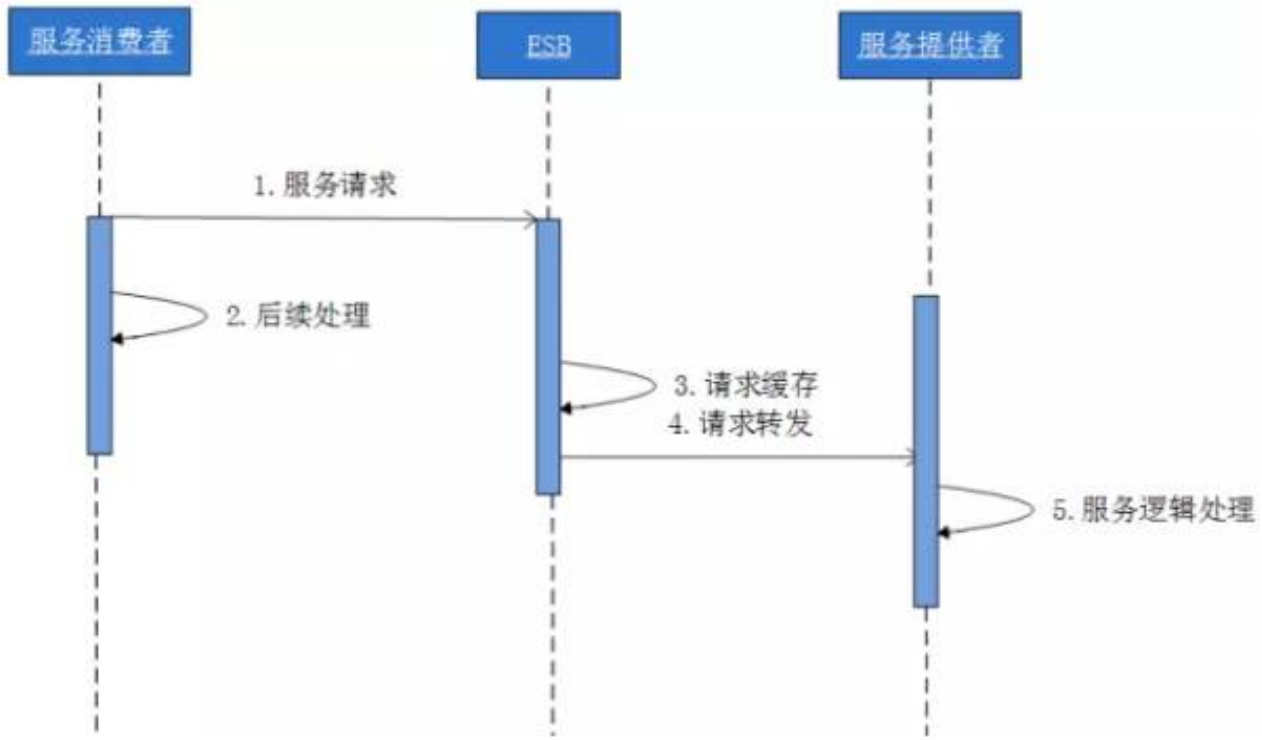
五种通信模式的实现方式-同步点对点服务模式



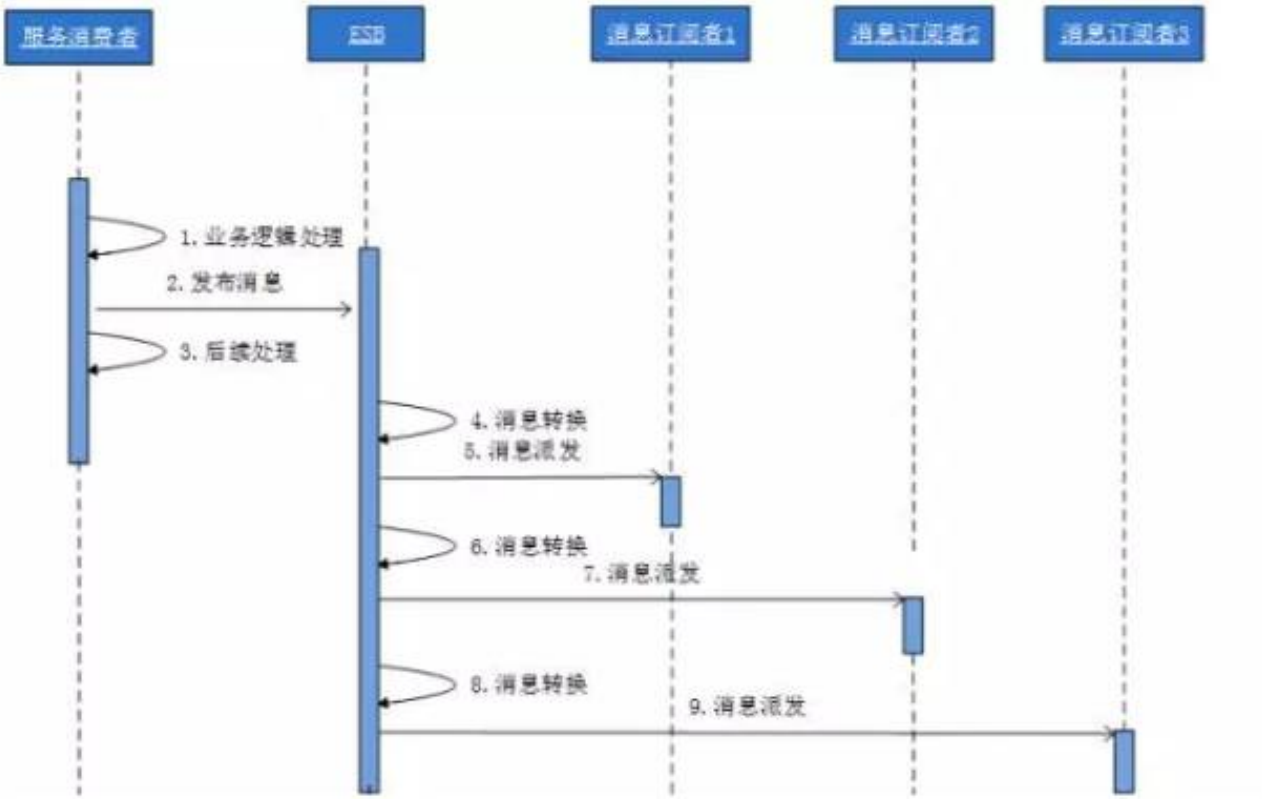
五种通信模式的实现方式-异步点对点消息模式1



五种通信模式的实现方式-异步点对点消息模式2



五种通信模式的实现方式-异步广播消息模式

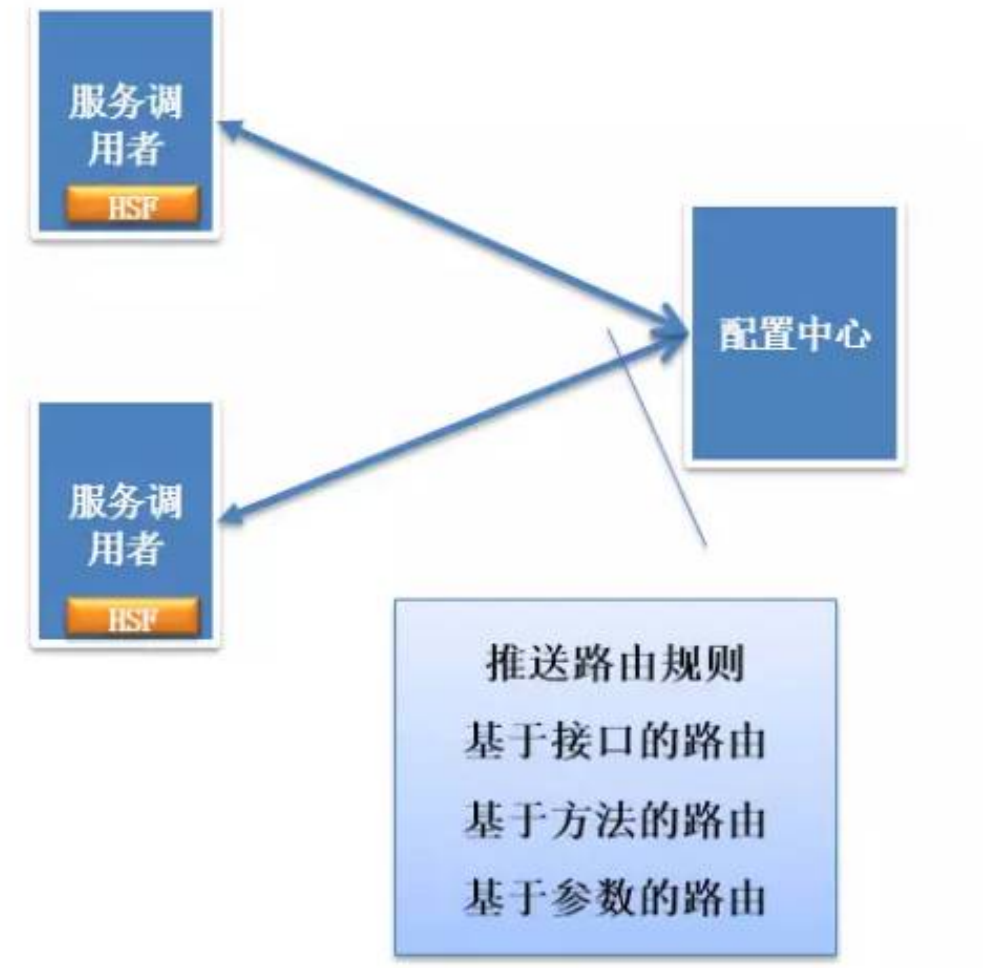


分布式架构下的服务治理

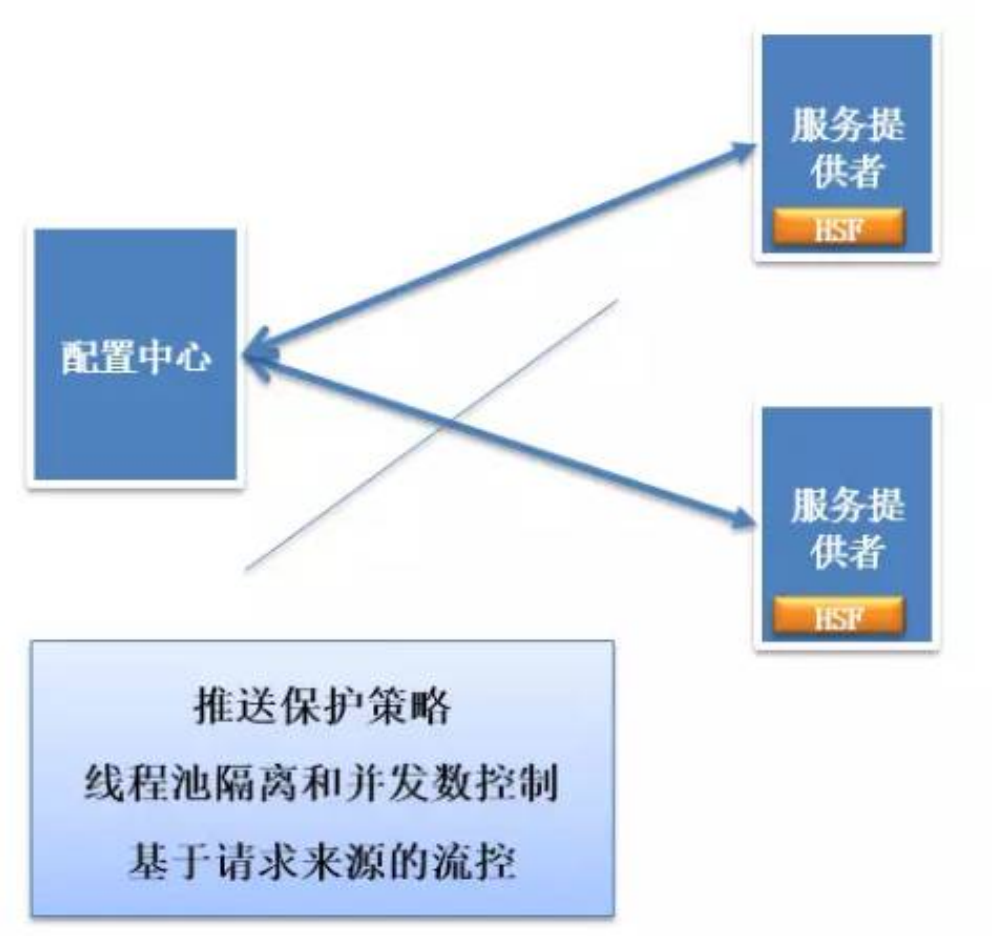
服务治理是服务框架/服务总线的核心功能。所谓服务治理，是指服务的提供方和消费方达成一致的约定，保证服务的高质量。服务治理功能可以解决将某些特定流量引入某一批机器，以及限制某些非法消费者的恶意访问，并在提供者处理量达到一定程度是，拒绝接受新的访问。

基于服务框架Dubbo的服务治理-服务管理可以知道你的系统，对外提供了多少服务，可以对服务进行升级、降级、停用、权重调整等操作可以知道你提供的服务，谁在使用，因业务需求，可以对该消费者实施屏蔽、停用等操作基于服务框架Dubbo的服务治理-服务监控可以统计服务的每秒请求数、平均响应时间、调用量、峰值时间等，作为服务集群规划、性能调优的参考指标。

基于服务框架Dubbo的服务治理-服务路由



基于服务框架Dubbo的服务治理-服务保护



基于服务总线OSB的服务治理-功能介绍

序号	支持功能	优势说明
1	将网状结构变为总线结构	1.架构清晰、灵活方便管理 2.专业团队负责服务接入，减少服务提供方沟通
2	多协议支持	1.解决php，java系统的信息交互 2.解决webservice、mq、rmi信息交互
3	服务编排（与流程编排相似）	1.站在公司角度对服务进行组合，统一服务的接入标准
4	服务路由	基于报文内容的判断进行服务执行的自由选择
5	消息转换	将差异化交给总线处理，系统只关注业务无需考虑标准的不同
6	异步同步	对于执行时间长的服务需要同步变异步
7	代理服务	只需要知道服务的地址就可以将服务集成到总线，客户端只需修改ip和端口
8	消息事物保全	通过mq保证消息在多个服务传递中不丢失（研究中）
11	支持热部署、服务的启动、停止	修改服务定义文件后即时生效，但更新jar从控制台中还没有找到方法
10	管理监控和事后日志追踪	

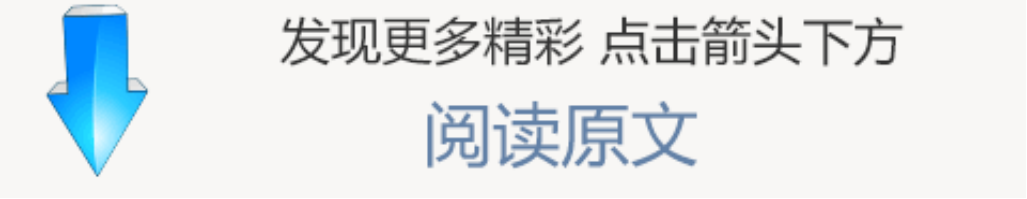
基于服务总线OSB的服务治理

说明：以上内容来自网络，作者佚名，仅供学习参考，版权归原作者所有。

温馨提示：

请搜索“ICT_Architect”或“扫一扫”二维码关注公众号，点击**原文链接**获取技术**电子书**详情。

求知若渴，虚心若愚



阅读原文