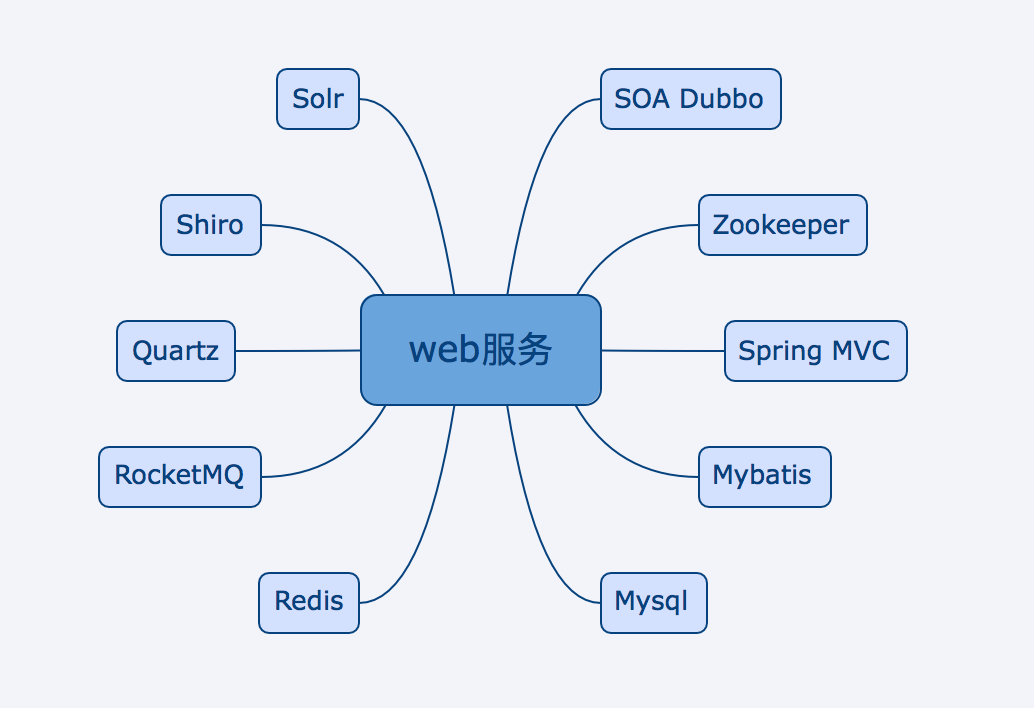
分布式SOA架构

1. **技术架构**

****

1. Dubbo: SOA 分布式系统的服务治理框架
2. Zookeeper: 注册中心。多几点配置
3. Mybatis: ORM框架
4. SpringMVC：轻量级web框架
5. Redis： 三级缓存，解决session共享问题等。
6. RocketMQ: 解决分布式事务问题等。
7. Quartz：定时控件
8. Shiro：安全框架
9. Solr：搜索引擎
10. Mysql：读写分离
11. … …
12. **系统架构演变**

**1、单一架构**

一个简单的电商网站说起，它可能包含如下的几个模块和功能，如首页、detail页、list页、下单页、支付页以及后台管理等页面和功能。单一的系统架构，使得在开发过程中，占用的资源越来越多，而且随着流量的增加使得维护起来越来越难以维护。



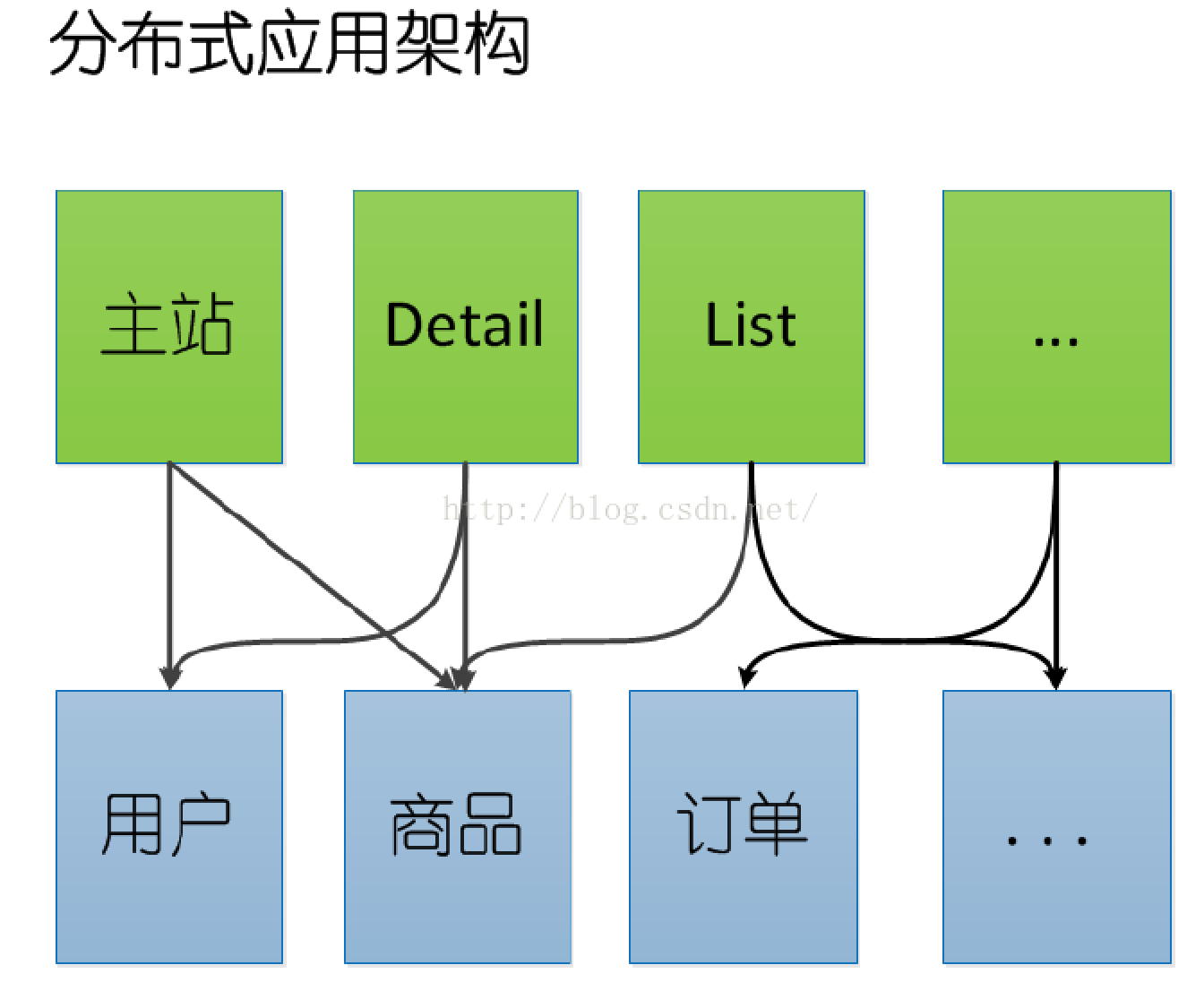
1. **垂直应用架构**

于是就产生了垂直应用架构，垂直应用架构解决了单一应用架构所面临的扩容问题，流量能够分散到各个子系统当中，且系统的体积可控，一定程度上降低了开发人员之间协同以及维护的成本，提升了开发效率。但是在垂直架构中相同逻辑代码需要不断的复制，不能复用。所以分布式系统就这样应运而生了。



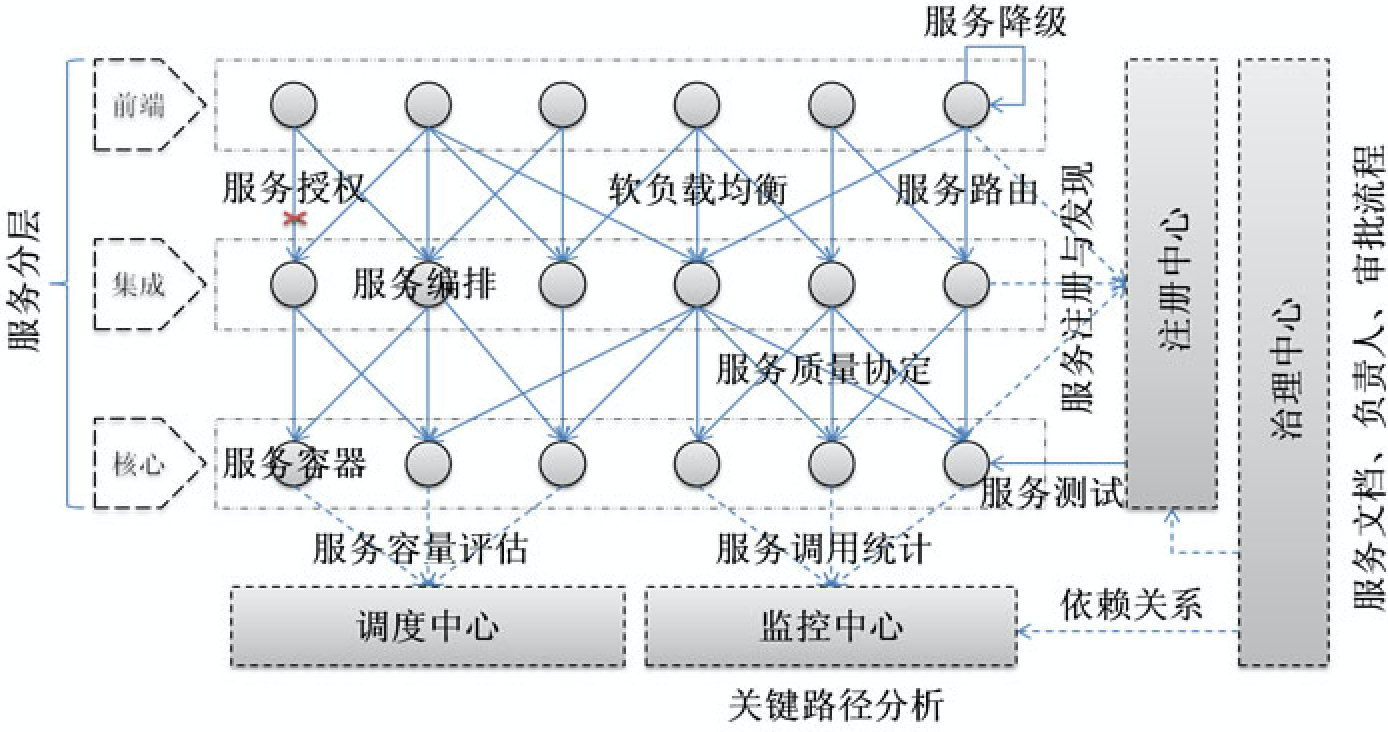
1. **分布式架构**

公共的逻辑业务提取出来形成服务，对外提供。这样对于维护和升级都只需要切分成一个一个的小系统去维护，也可以让前端业务系统与底层数据访问分离，团队分工更为明确。

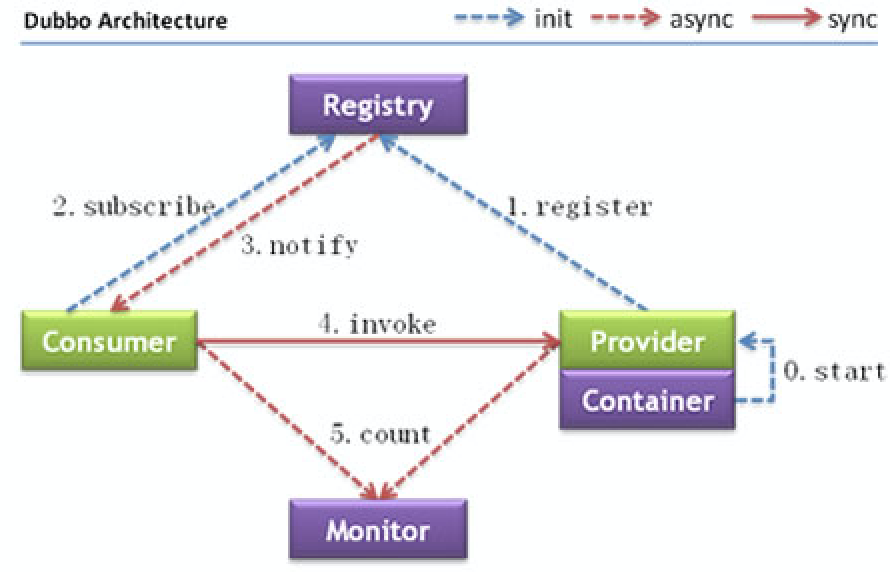


1. **SOA分布式系统的服务治理框架**

分布式系统所依赖的基础设施包括服务框架、消息中间件、数据访问中间件、配置中心、分布式缓存系统、持久化存储(关系数据库、nosql数据库)、搜索引擎、CDN网络、负载均衡系统、运维自动化系统、硬件虚拟化及镜像管理系统、分布式文件系统、日志收集系统、监控系统、离线计算、实时计算、数据仓库等等。随着服务化的进一步发展，服务越来越多，服务之间的调用和依赖关系也越来越复杂，诞生了面向服务的架构体系(SOA)，也因此衍生出了一系列相应的技术，如对服务提供、服务调用、连接处理、通信协议、序列化方式、服务发现、服务路由、日志输出等行为进行封装的服务框架。

**三、DUBBO**

使用Dubbo SOA分布式系统的服务治理框架，解决上述问题，分布式系统的架构中，按照分层的架构来架构，使得各个层级之间最大限度的松耦合，根据业务需要微服务节点随时可增加。



**(1) 连通性：**

* 注册中心负责服务地址的注册与查找，相当于目录服务，服务提供者和消费者只在启动时与注册中心交互，注册中心不转发请求，压力较小
* 监控中心负责统计各服务调用次数，调用时间等，统计先在内存汇总后每分钟一次发送到监控中心服务器，并以报表展示
* 服务提供者向注册中心注册其提供的服务，并汇报调用时间到监控中心，此时间不包含网络开销
* 服务消费者向注册中心获取服务提供者地址列表，并根据负载算法直接调用提供者，同时汇报调用时间到监控中心，此时间包含网络开销
* 注册中心，服务提供者，服务消费者三者之间均为长连接，监控中心除外
* 注册中心通过长连接感知服务提供者的存在，服务提供者宕机，注册中心将立即推送事件通知消费者
* 注册中心和监控中心全部宕机，不影响已运行的提供者和消费者，消费者在本地缓存了提供者列表
* 注册中心和监控中心都是可选的，服务消费者可以直连服务提供者

**(2) 健状性：**

* 监控中心宕掉不影响使用，只是丢失部分采样数据
* 数据库宕掉后，注册中心仍能通过缓存提供服务列表查询，但不能注册新服务
* 注册中心对等集群，任意一台宕掉后，将自动切换到另一台
* 注册中心全部宕掉后，服务提供者和服务消费者仍能通过本地缓存通讯
* 服务提供者无状态，任意一台宕掉后，不影响使用
* 服务提供者全部宕掉后，服务消费者应用将无法使用，并无限次重连等待服务提供者恢复

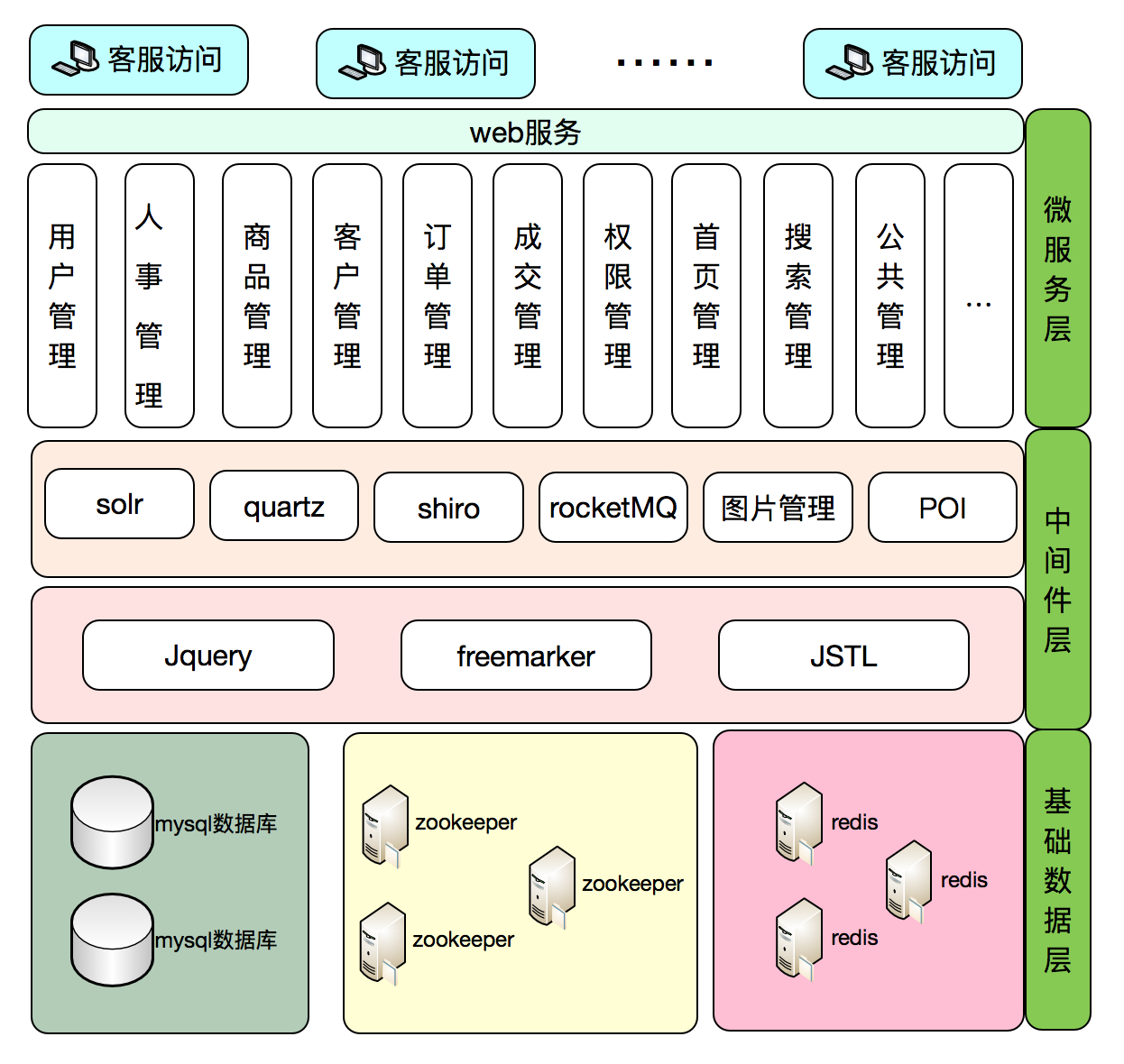
**(3) 伸缩性：**

* 注册中心为对等集群，可动态增加机器部署实例，所有客户端将自动发现新的注册中心
* 服务提供者无状态，可动态增加机器部署实例，注册中心将推送新的服务提供者信息给消费者

**(4) 升级性：**

* 当服务集群规模进一步扩大，带动IT治理结构进一步升级，需要实现动态部署，进行流动计算，现有分布式服务架构不会带来阻力

**四、业务系统架构图**

****

**五、系统架构图**

