两台计算机进行通信 需要IP地址、协议、端口号

IP就知道发往那台计算机、协议是数据传输方式、端口号发往那个地址

IP+端口构成socket套接字

针对网络通信的不同层次，Java提供了不同的API，其提供的网络功能有四大类：

InetAddress:用于标识网络上的硬件资源，主要是IP地址

URL：统一资源定位符，通过URL可以直接读取或写入网络上的数据

Sockets：使用TCP协议实现的网络通信Socket相关的类

Datagram:使用UDP协议，将数据保存在用户数据报中，通过网络进行通信。

常见并发类型有进程和线程两种，后面的帖子会讲两种并发类型下的应用

分布式就是通过计算机网络将后端工作分布到多台主机上，多个主机一起协同完成工作。

1.分布式系统一定是由多个节点组成的系统。

其中，节点指的是计算机服务器，而且这些节点一般不是孤立的，而是互通的。

2.这些连通的节点上部署了我们的节点，并且相互的操作会有协同。

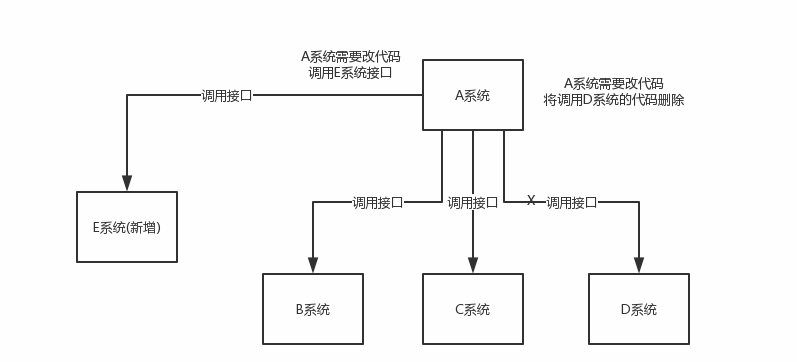
分布式系统对于用户而言，他们面对的就是一个服务器，提供用户需要的服务而已，而实际上这些服务是通过背后的众多服务器组成的一个分布式系统，因此分布式系统看起来像是一个超级计算机一样。

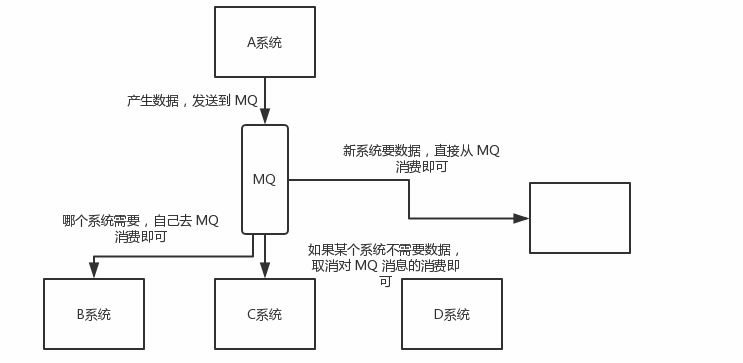
例如淘宝，平时大家都会使用，它本身就是一个分布式系统，我们通过浏览器访问淘宝网站时，这个请求的背后就是一个庞大的分布式系统在为我们提供服务，整个系统中有的负责请求处理，有的负责存储，有的负责计算，最终他们相互协调把最后的结果返回并呈现给用户。

它们虽然都说是"多个进程同时运行"，但是它们的"同时"不是一个概念。并行的"同时"是同一时刻可以多个进程在运行(处于running)，并发的"同时"是经过上下文快速切换，使得看上去多个进程同时都在运行的现象，是一种OS欺骗用户的现象。

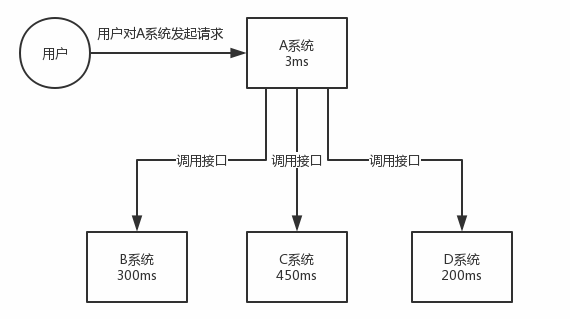
MQ Message Queue消息队列

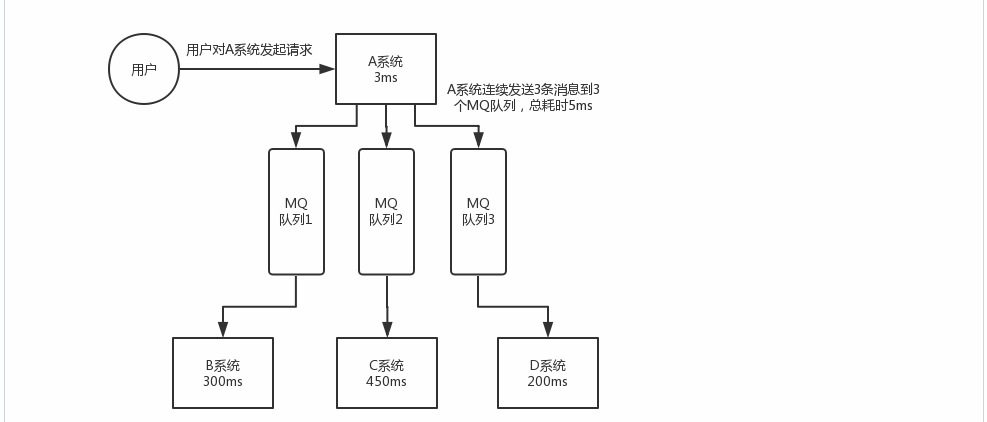
MQ 解耦 异步 削峰





在上面系统中A系统与其余系统直接耦合，通过消息队列进行连接后，可以解耦合。

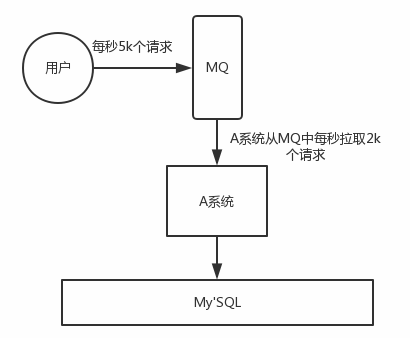
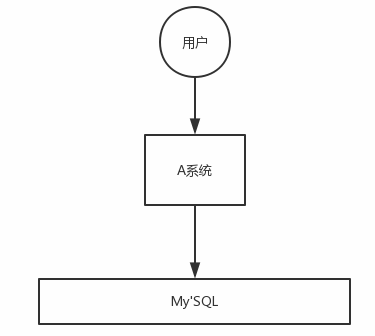




在这里是不是就可以理解为A系统是生产者主要用来生产消息，BCD系统主要用来消费消息

通过消息队列，可以减少系统时延，在普通模式中，用户需要等待四个系统进行写库操作，通过MQ以后，用户可以将消息发送至MQ队列？这里有个问题：消息写入MQ中，BCD系统什么时候进行写库操作啊？写库时不也需要经历系统时延吗?

削峰



假如A系统每秒处理请求最多2K个，当高峰期，可能每秒通过5K请求，就有可能造成系统崩溃，通过消息队列，A系统每秒从MQ拉取2K请求，不会造成系统崩溃，当高峰期度过，A系统仍然可以每秒从MQ拉取2K，可以处理积压的请求

缺点：

系统复杂性、数据一致性、系统可用性。

引入消息队列以后消息队列挂掉整个系统会挂掉；系统复杂性升级。如果BCD中，有某一个库数据没有更新，但此时A系统已经进行请求返回、会造成数据不一致。



银行系统要求数据可靠性！其次才是数据时延

分布式架构，并发！

基于此一般使用RocketMQ进行架构设计

topic为消息主题

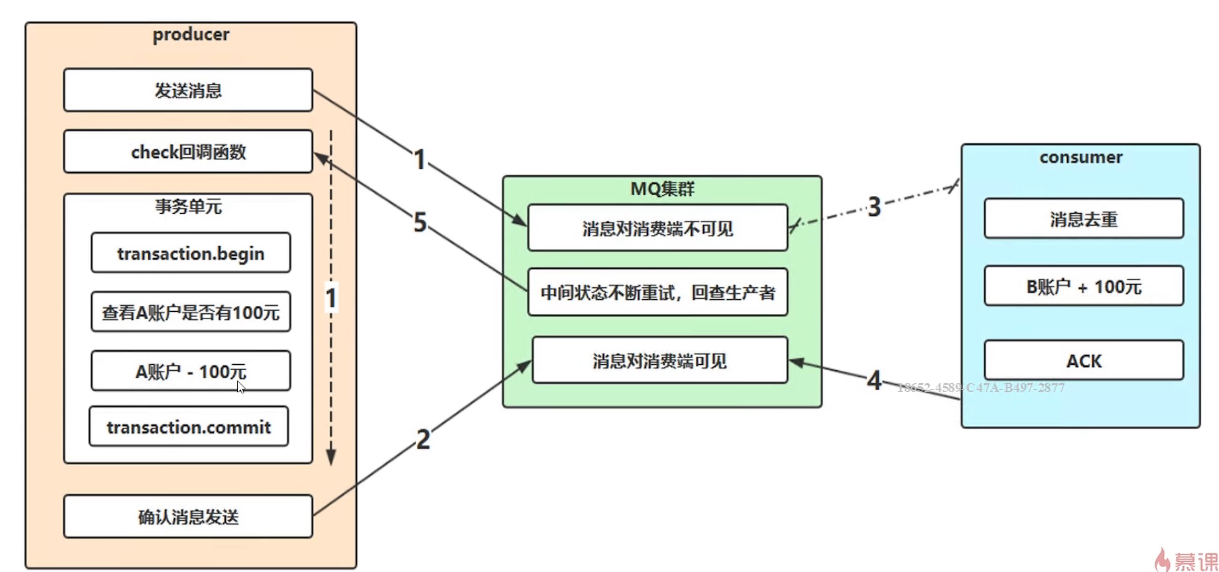
幂等性，通俗点说，就一个数据，或者一个请求，给你重复来多次，你得确保对应的数据是不会改变的，**不能出错**。

NameServer Broker?

Broker

Nameserver

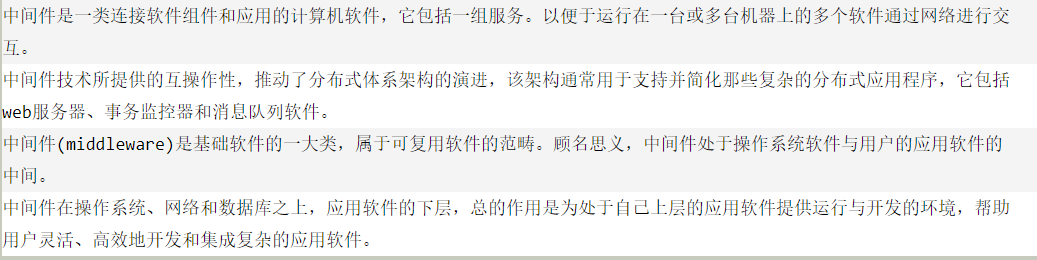
生产者消费者模型

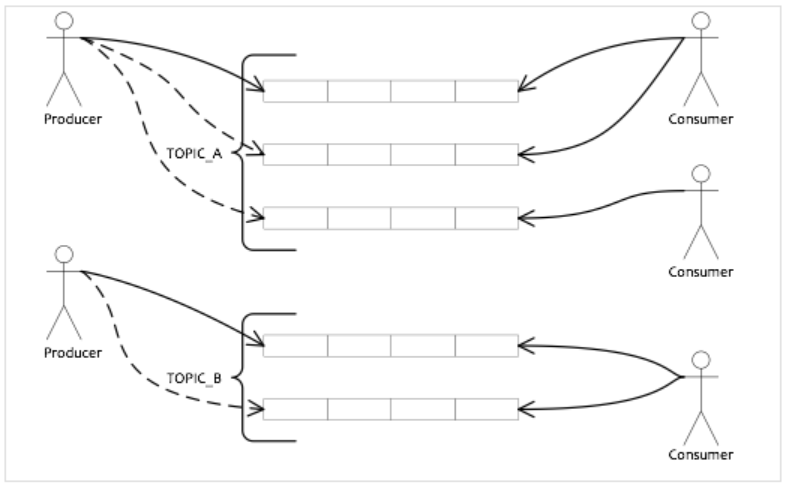


没懂？ 生产者是产生消息？

消息有序指的是一类消息消费时，能按照发送的顺序来消费。例如：一个订单产生了3条消息，分别是订单创建，订单付款，订单完成。消费时，要按照这个顺序消费才能有意义。但是同时订单之间是可以并行消费的。

RocketMQ可以严格的保证消息有序。





简单说来，RocketMQ具有以下特点：

　　①是一个队列模型的消息中间件，具有高性能、高可靠、高实时、分布式特点。

　②Producer、Consumer、队列都可以分布式。

　③Producer向一些队列轮流发送消息，队列集合称为Topic，Consumer如果做广播消费，则一个consumer实例消费这个Topic对应的所有队列，如果做集群消费，则多个Consumer实例平均消费这个topic对应的队列集合。

　④能够保证严格的消息顺序。

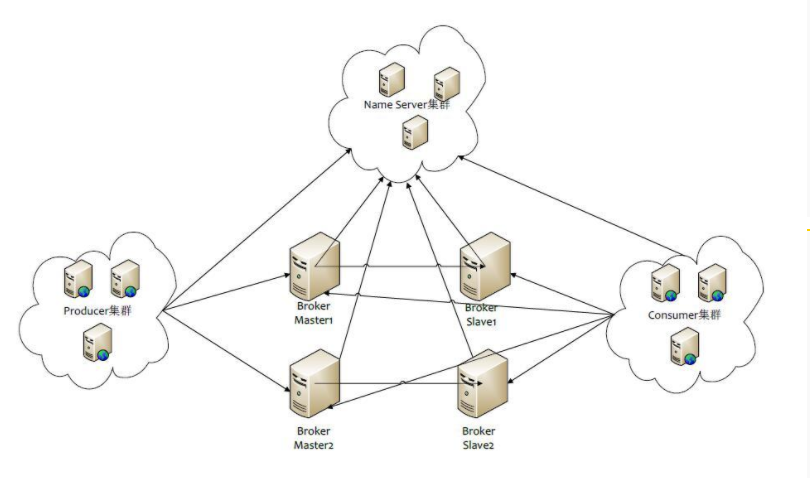
　⑤提供丰富的消息拉取模式。

　⑥高效的订阅者水平扩展能力。

　⑦实时的消息订阅机制。

　⑧亿级消息堆积能力。

　⑨较少的依赖。



如上图所示，整体可以分成4个角色，分别是：Producer，Consumer，Broker以及NameServer；

1.NameServer

可以理解为是消息队列的协调者，Broker向它注册路由信息，同时Client向其获取路由信息，如果使用过Zookeeper，就比较容易理解了，但是功能比Zookeeper弱；

NameServer本身是没有状态的，并且多个NameServer直接并没有通信，可以横向扩展多台，Broker会和每一台NameServer建立长连接；

2.Broker

Broker是RocketMQ的核心，提供了消息的接收，存储，拉取等功能，一般都需要保证Broker的高可用，所以会配置Broker Slave，当Master挂掉之后，Consumer然后可以消费Slave；

Broker分为Master和Slave，一个Master可以对应多个Slave，Master与Slave的对应关系通过指定相同的BrokerName，不同的BrokerId来定义，BrokerId为0表示Master，非0表示Slave；

3.Producer

消息队列的生产者，需要与NameServer建立连接，从NameServer获取Topic路由信息，并向提供Topic服务的Broker Master建立连接；Producer无状态，看集群部署；

4.Consumer

消息队列的消费者，同样与NameServer建立连接，从NameServer获取Topic路由信息，并向提供Topic服务的Broker Master，Slave建立连接；

5.Topic和Message Queue

在介绍完以上4个角色以后，还需要重点介绍一下上面提到的Topic和Message Queue；字面意思就是主题，用来区分不同类型的消息，发送和接收消息前都需要先创建Topic，针对Topic来发送和接收消息，为了提高性能和吞吐量，引入了Message Queue，一个Topic可以设置一个或多个Message Queue，有点类似kafka的分区(Partition)，这样消息就可以并行往各个Message Queue发送消息，消费者也可以并行的从多个Message Queue读取消息；