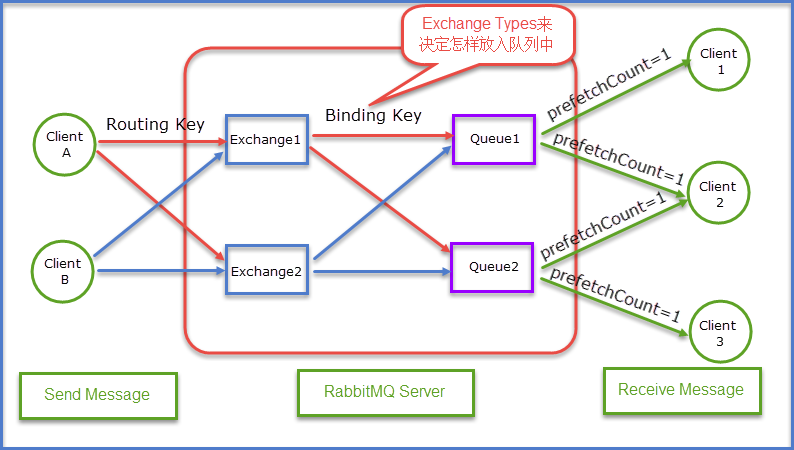
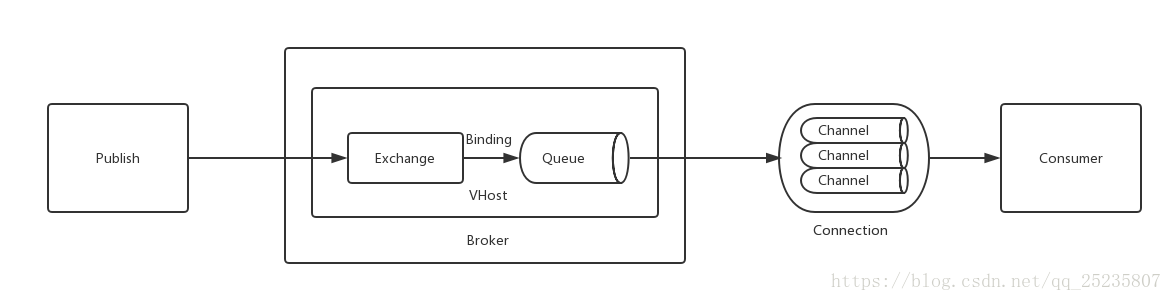
1. AMQP
2. AMQP，即Advanced Message Queuing Protocol，高级消息队列协议，是应用层协议的一个开放标准，为面向消息的中间件设计。消息中间件主要用于组件之间的解耦，消息的发送者无需知道消息使用者的存在，反之亦然。
3. 消息中间件是一种由消息传送机制或消息队列模式组成的中间件技术，利用高效可靠的消息传递机制进行平台无关的数据交流，并基于数据通信来进行分布式系统的集成。常见的异步消中间件有kafka（主要处理日志）> RabbitMQ > ActiveMQ > ZeroMQ [吞吐量和TPS]
4. RabbitMQ



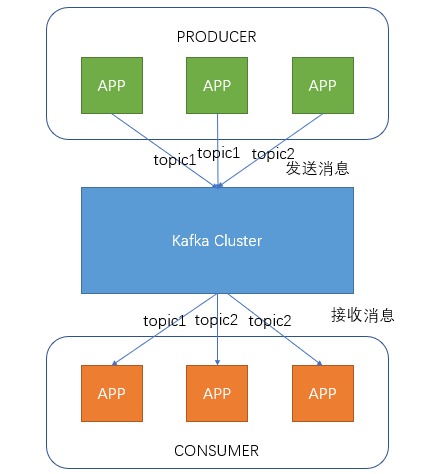
Exchange的匹配规则：

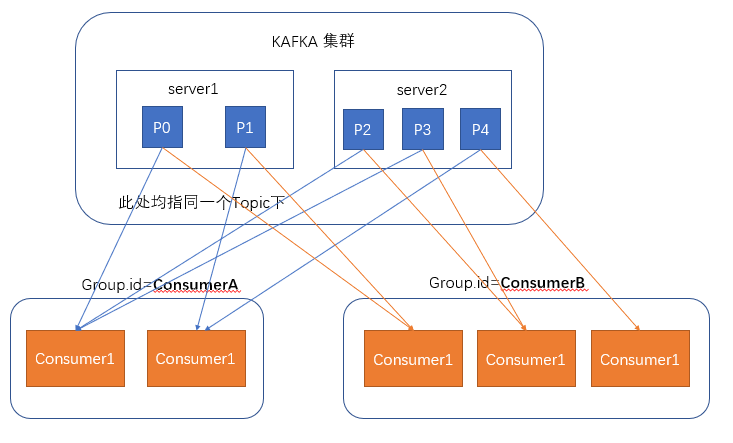


RPC匹配规则：MQ中的一种同步消息机制，客户端请求服务器，设定其返回的queue和唯一的标识，当服务器处理完后按照客户端的要求返回，而客户端则一直等到服务器返回后查看消息来判断下一步需要进行的操作。

1. Kafka

Kafka目前主要作为一个分布式的发布订阅式的消息系统使用





在消费者消费消息时，kafka使用offset来记录当前消费的位置

　　　　在kafka的设计中，可以有多个不同的group来同时消费同一个topic下的消息，如图，我们有两个不同的group同时消费，他们的的消费的记录位置offset各不项目，不互相干扰。

　　　　对于一个group而言，消费者的数量不应该多余分区的数量，因为在一个group中，每个分区至多只能绑定到一个消费者上，即一个消费者可以消费多个分区，一个分区只能给一个消费者消费。因此，若一个group中的消费者数量大于分区数量的话，多余的消费者将不会收到任何消息。

Kafka比其他的MQ都要快的原因：

1. 将消息变为了传输文件
2. 在生产者中数据是顺序IO写入磁盘
3. 使用逻辑内存来减少IO的缓慢带来的影响
4. 消费者使用了Zero-copy的策略

Zero-copy策略：

1.mmap()函数将文件直接映射到用户程序的内存中，映射成功时返回指向目标区域的指针。这段内存空间可以用作进程间的共享内存空间，内核也可以直接操作这段空间。

在映射文件之后，暂时不会拷贝任何数据到内存中，只有当访问这段内存时，发现没有数据，于是产生缺页访问，使用DMA操作将数据拷贝到这段空间中。可以直接将这段空间的数据拷贝到socket buffer中。

2.sendfile()函数借助文件描述符来实现数据拷贝：直接将文件描述in\_fd的数据拷贝给文件描述符out\_fd，其中in\_fd是数据提供方，out\_fd是数据接收方。文件描述符的操作都是在内核进行的，不会经过用户空间，所以数据不用拷贝到app buffer

3. splice()函数可以在两个文件描述符之间**移动**数据，且其中一个描述符必须是管道描述符。由于不需要在kernel buffer和app buffer之间拷贝数据，所以实现了零复制

4. tee()函数在两个管道描述符之间复制数据。由于从in\_fd复制给另一个管道out\_fd时，不认为数据是来自于in\_fd的，所以复制数据后，in\_fd仍可使用splice()函数进行数据移动。由于没有经过用户空间，所以实现了零复制

5. 当父进程fork生成子进程时，会复制它的所有内存页。这至少会导致两个问题：消耗大量内存；复制操作消耗时间。特别是fork后使用exec加载新程序时，由于会初始化内存空间，所以复制操作几乎是多余的。使用copy-on-write技术，使得在fork子进程时不复制内存页，而是共享内存页(也就是说，子进程也指向父进程的物理空间)，只有在该子进程需要修改某一块数据，才会将这一块数据拷贝到自己的app buffer中并进行修改，那么这一块数据就属于该子进程的私有数据，可随意访问、修改、复制。