**一：为何选用RabbitMQ？**

**ActiveMQ:** 老牌的消息中间件，国内很多公司过去运用的还是非常广泛的，功能很强大。但是问题在于没法确认ActiveMQ可以支撑互联网公司的高并发、高负载以及高吞吐的复杂场景，在国内互联网公司落地较少。而且使用较多的是一些传统企业，用ActiveMQ做异步调用和系统解耦。

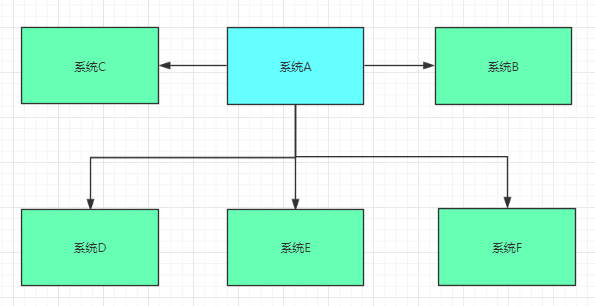
RocketMQ：阿里开源的，经过阿里的生产环境的超高并发、高吞吐的考验，性能卓越，同时还支持分布式事务等特殊场景。基于Java语言开发的，适合深入阅读源码，有需要可以站在源码层面解决线上生产问题，包括源码的二次开发和改造。但是对JAVA以外的语言极其不友好，不稳定，最典型的部署在IIS上会时不时宕机。

Kafka：Kafka提供的消息中间件的功能明显较少一些，相对上述几款MQ中间件要少很多。优势在于专为超高吞吐量的实时日志采集、实时数据同步、实时数据计算等场景来设计。因此Kafka在大数据领域中配合实时计算技术（比如Spark Streaming、Storm、Flink）使用的较多。但是在传统的MQ中间件使用场景中较少采用。

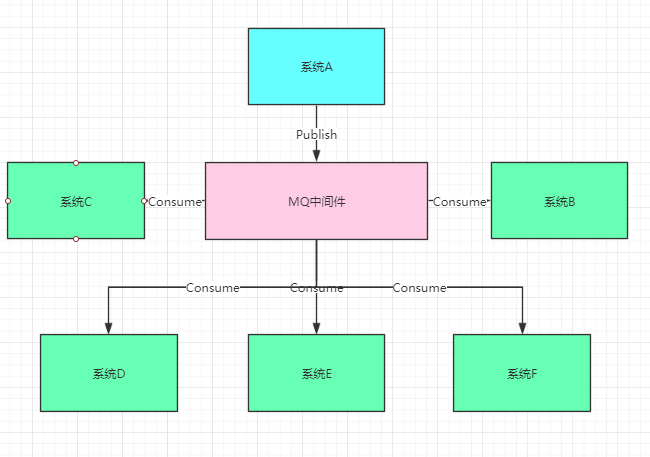
RabbitMQ: 经过调研，国内各大互联网公司落地大规模RabbitMQ集群支撑自身业务的case较多，国内各种中小型互联网公司使用RabbitMQ的实践也比较多。RabbitMQ的开源社区很活跃，较高频率的迭代版本，来修复发现的bug以及进行各种优化，综合考虑过，采取RabbitMQ。缺陷：自身是基于erlang语言开发的，所以导致较为难以分析里面的源码，也较难进行深层次的源码定制和改造，毕竟需要较为扎实的erlang语言功底才可以。

**二：为何系统架构中需要引入消息中间件？消息中间件的使用场景**

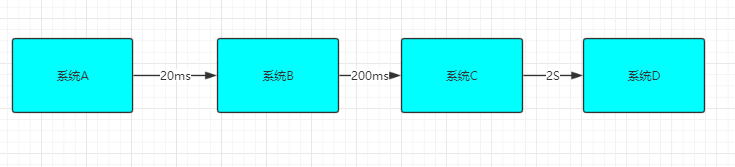
1. **系统解耦**



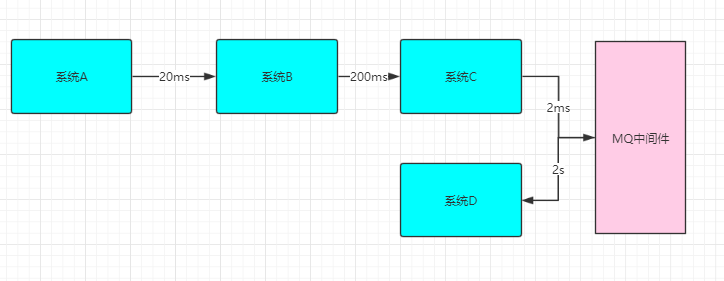
**演变为**



1. **异步调用**



生活中叫外卖的例子。



1. **流量削峰**

假设有一个系统，平时正常的时候每秒可能就1000个请求，系统部署在8核16G的机器的上，正常处理都是ok的，每秒1000请求是可以轻松抗住的。

但是双11当天，在高峰期一下子来了每秒钟10000请求，瞬时出现了流量高峰，**此时你的选择是要搞10台机器，抗住每秒1万个请求的瞬时高峰吗？**

**双11过后就没用了，造成资源极度浪费。**

但是如果你就部署一台机器，那会导致瞬时高峰时，一下子压垮你的系统，因为绝对无法抗住每秒10000的请求高峰。一旦到了瞬时高峰期，一下涌入每秒几千的请求，就可以积压在MQ里面，然后那一台机器慢慢的处理和消费。等高峰期过了，再消费一段时间，MQ里积压的数据就消费完毕了。

