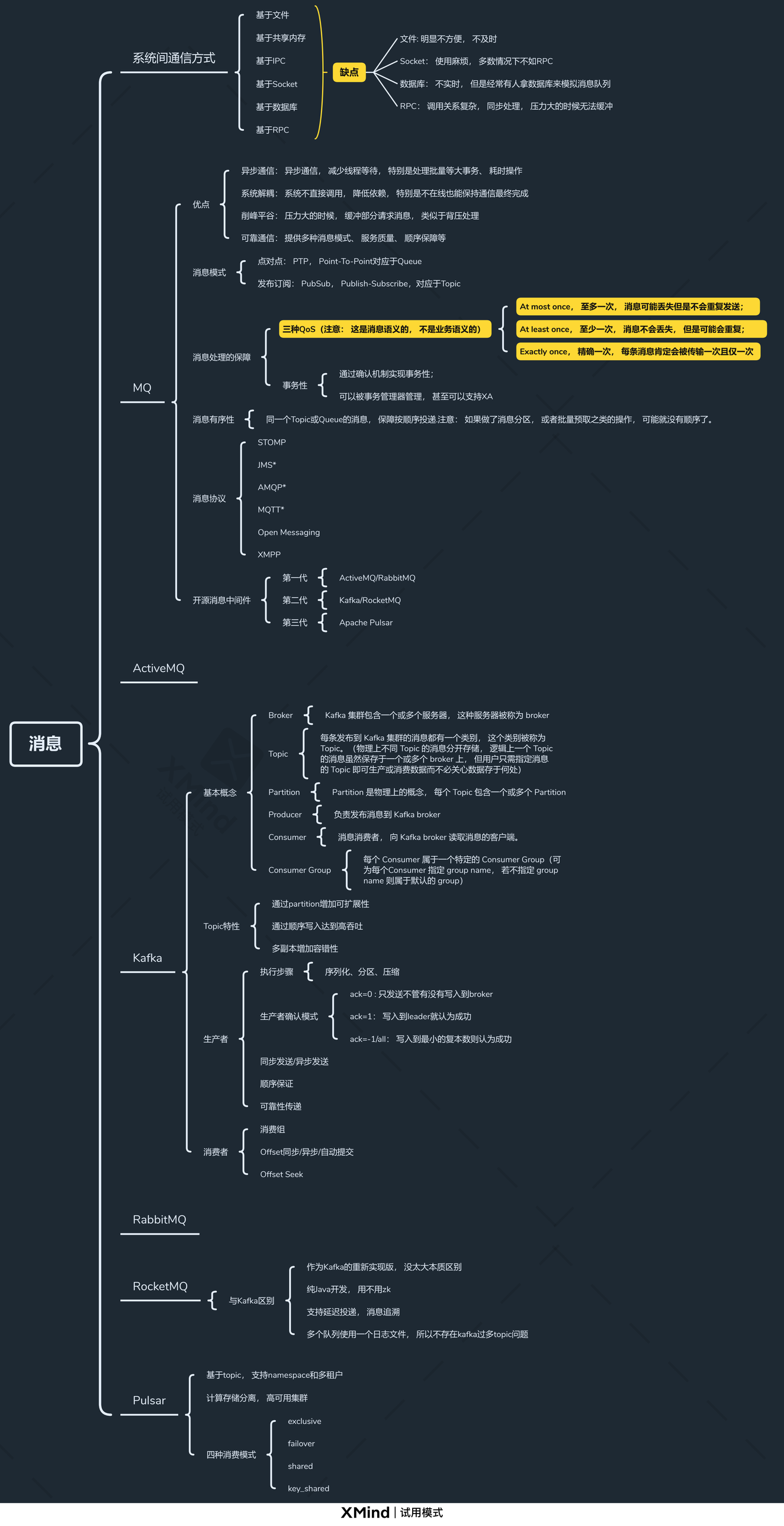
消息
系统间通信方式
基于文件
基于共享内存
基于IPC
基于Socket
基于数据库
基于RPC
缺点
文件: 明显不方便， 不及时
Socket： 使用麻烦， 多数情况下不如RPC
数据库： 不实时， 但是经常有人拿数据库来模拟消息队列
RPC： 调用关系复杂， 同步处理， 压力大的时候无法缓冲
MQ
优点
异步通信： 异步通信， 减少线程等待， 特别是处理批量等大事务、 耗时操作
系统解耦： 系统不直接调用， 降低依赖， 特别是不在线也能保持通信最终完成
削峰平谷： 压力大的时候， 缓冲部分请求消息， 类似于背压处理
可靠通信： 提供多种消息模式、 服务质量、 顺序保障等
消息模式
点对点： PTP， Point-To-Point对应于Queue
发布订阅： PubSub， Publish-Subscribe，对应于Topic
消息处理的保障
三种QoS（注意： 这是消息语义的， 不是业务语义的）
At most once， 至多一次， 消息可能丢失但是不会重复发送；
At least once， 至少一次， 消息不会丢失， 但是可能会重复；
Exactly once， 精确一次， 每条消息肯定会被传输一次且仅一次
事务性
通过确认机制实现事务性；
可以被事务管理器管理， 甚至可以支持XA
消息有序性
同一个Topic或Queue的消息， 保障按顺序投递.注意： 如果做了消息分区， 或者批量预取之类的操作， 可能就没有顺序了。
消息协议
STOMP
JMS*
AMQP*
MQTT*
Open Messaging
XMPP
开源消息中间件
第一代
ActiveMQ/RabbitMQ
第二代
Kafka/RocketMQ
第三代
Apache Pulsar
ActiveMQ
Kafka
基本概念
Broker
Kafka 集群包含一个或多个服务器， 这种服务器被称为 broker
Topic
每条发布到 Kafka 集群的消息都有一个类别， 这个类别被称为 Topic。（物理上不同 Topic 的消息分开存储， 逻辑上一个 Topic 的消息虽然保存于一个或多个 broker 上， 但用户只需指定消息的 Topic 即可生产或消费数据而不必关心数据存于何处）
Partition
Partition 是物理上的概念， 每个 Topic 包含一个或多个 Partition
Producer
负责发布消息到 Kafka broker
Consumer
消息消费者， 向 Kafka broker 读取消息的客户端。
Consumer Group
每个 Consumer 属于一个特定的 Consumer Group（可为每个Consumer 指定 group name， 若不指定 group name 则属于默认的 group）
Topic特性
通过partition增加可扩展性
通过顺序写入达到高吞吐
多副本增加容错性
生产者
执行步骤
序列化、分区、压缩
生产者确认模式
ack=0 : 只发送不管有没有写入到broker
ack=1： 写入到leader就认为成功
ack=-1/all： 写入到最小的复本数则认为成功
同步发送/异步发送
顺序保证
可靠性传递
消费者
消费组
Offset同步/异步/自动提交
Offset Seek
RabbitMQ
RocketMQ
与Kafka区别
作为Kafka的重新实现版， 没太大本质区别
纯Java开发， 用不用zk
支持延迟投递， 消息追溯
多个队列使用一个日志文件， 所以不存在kafka过多topic问题
Pulsar
基于topic， 支持namespace和多租户
计算存储分离， 高可用集群
四种消费模式
exclusive
failover
shared
key_shared