# ActiveMQ--7.优缺点

**优点：**

**可以用JDBC**

虽然使用JDBC会降低ActiveMQ的性能，但是数据库一直都是开发人员最熟悉的存储介质。将消息存到数据库，看得见摸得着。而且公司有专门的DBA去对数据库进行调优，主从分离。

**监控完善**

拥有完善的监控，包括Web Console，JMX，Shell命令行，Jolokia的REST API。应有尽有，对于一个7\*24的产品，没有监控就不能上线。

**界面友善**

提供的Web Console可以满足大部分情况，还有很多第三方的组件可以使用，如hawtio。

**支持JMS**

支持JMS的统一接口，可以使用Sprinig的JmsTemplate提升开发效率。

**推送模式**

Broker推送消息到Consumer，消息实时性高。

**不丢消息**

支持基于XA的事务。

**有安全机制**

支持基于shiro，jaas等多种安全配置机制，可以对Queue/Topic进行认证和授权。

**Queue支持高可用和分片**

使用Exclusive Consumer可以实现Queue的高可用，使用Message Group可以实现Queue的高可用加消息分片。

**Broker支持数据复制**

可以使用基于数据库的主从复制，或者基于Level DB的复制实现数据的高可用。

**缺点：**

**Broker不支持消息分片**

没有像Kafka那样提供消息分片写入Broker的功能，不能实现动态扩展Broker。

**Topic不支持高可用和分片**

基于Exclusive Consumer和Message Group，只能在Queue上起作用。

**可以把Topic转成Queue，以支持高可用和分片，但是数据爆炸**

使用Virtual Topic可以将Topic转成Queue，但是每个订阅者会复制一份消息到Queue，使Broker的存储空间爆发式增长。

**基于LevelDb高可用，必须至少使用3台机器，造成资源浪费**

LevelDb使用Zookeeper作为调度中心，常用的配置是1台主机，2台备机，而且2台备机不提供服务。造成资源浪费。而且一旦有2台机器死机，虽然仍然有一台机器存活，但是仍然不能提供服务。

**适用场景：**

只有在需要强一致性事务时需要使用ActiveMQ。使用JDBC的存储形式，这样一旦出错，整个数据库回滚。其他的文件存储机制，不管是levedb还是kahadb，都是鸡肋，提高了性能，但是不能完全保证事务一致性。如果考虑性能问题，可以使用kafka，metaq。最佳实践是：使用jdbc存储，master-slave方式备份数据，static静态集群的方式高可用，同时满足服务和数据的高可用。如果需要，自行开发数据分片功能。