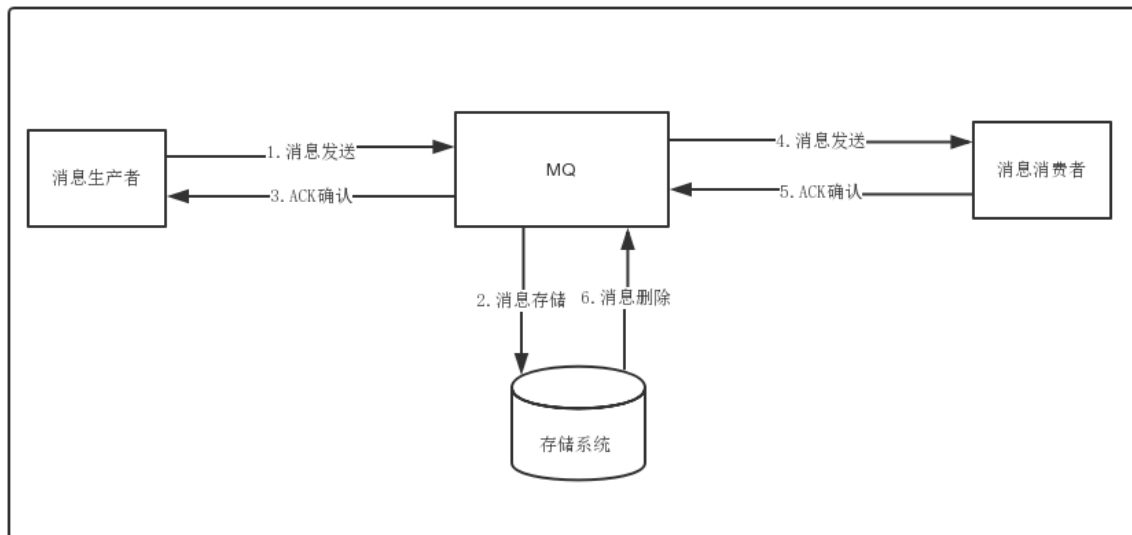


# RocketMQ消息存储概述

分布式队列因为有高可靠性的要求，所以数据要进行持久化存储。



1. 消息生成者发送消息
2. MQ收到消息，将消息进行持久化，在存储中新增一条记录
3. 返回ACK给生产者
4. MQ push 消息给对应的消费者，然后等待消费者返回ACK
5. 如果消息消费者在指定时间内成功返回ack，那么MQ认为消息消费成功，在存储中删除消息，即执行第6步；如果MQ在指定时间内没有收到ACK，则认为消息消费失败，会尝试重新push消息,重复执行4、5、6步骤
6. MQ删除消息

## 存储介质

- 关系型数据库DB

Apache下开源的另外一款MQ—ActiveMQ（默认采用的KahaDB做消息存储）可选用JDBC的方式来做消息持久化，通过简单的xml配置信息即可实现JDBC消息存储。由于，普通关系型数据库（如Mysql）在单表数据量达到千万级别的情况下，其IO读写性能往往会出现瓶颈。在可靠性方面，该种方案非常依赖DB，如果一旦DB出现故障，则MQ的消息就无法落盘存储会导致线上故障

- 文件系统

目前业界较为常用的几款产品（RocketMQ/Kafka/RabbitMQ）均采用的是消息刷盘至所部署虚拟机/物理机的文件系统来做持久化（刷盘一般可以分为异步刷盘和同步刷盘两种模

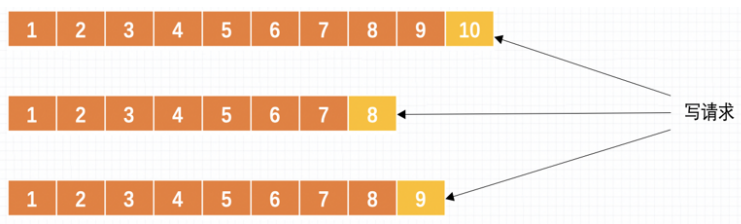
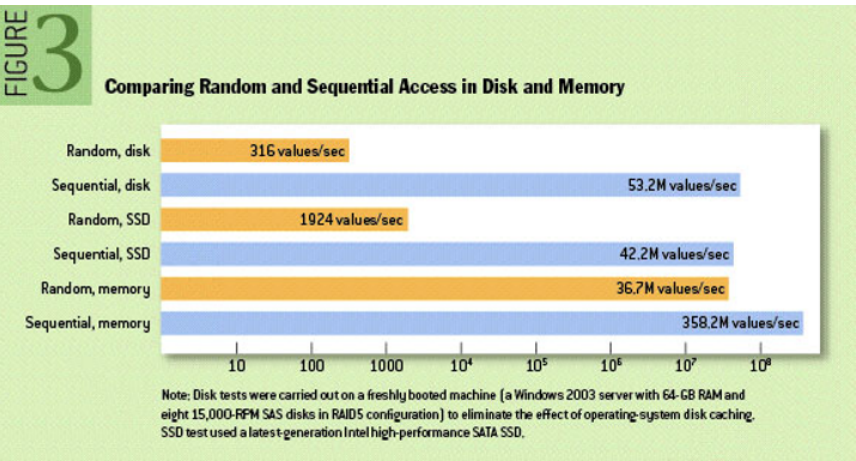
式)。消息刷盘为消息存储提供了一种高效率、高可靠性和高性能的数据持久化方式。除非部署MQ机器本身或是本地磁盘挂了，否则一般是不会出现无法持久化的故障问题。

- 文件系统 > 关系型数据库DB

## 消息的存储和发送

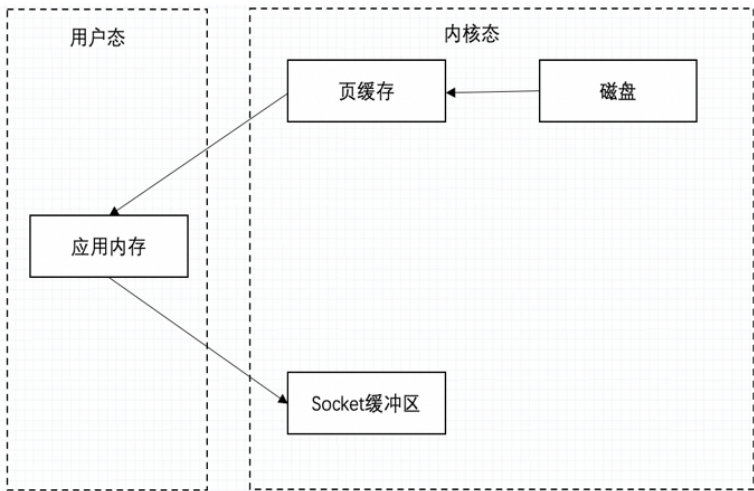
- 消息存储

RocketMQ的消息用顺序写, 随机读, 保证了消息存储的速度。



- 消息发送

Linux操作系统分为【用户态】和【内核态】，文件操作、网络操作需要涉及这两种形态的切换，免不了进行数据复制。



RocketMQ充分利用了上述特性，也就是所谓的“零拷贝”技术，提高消息存盘和网络发送的速度。

