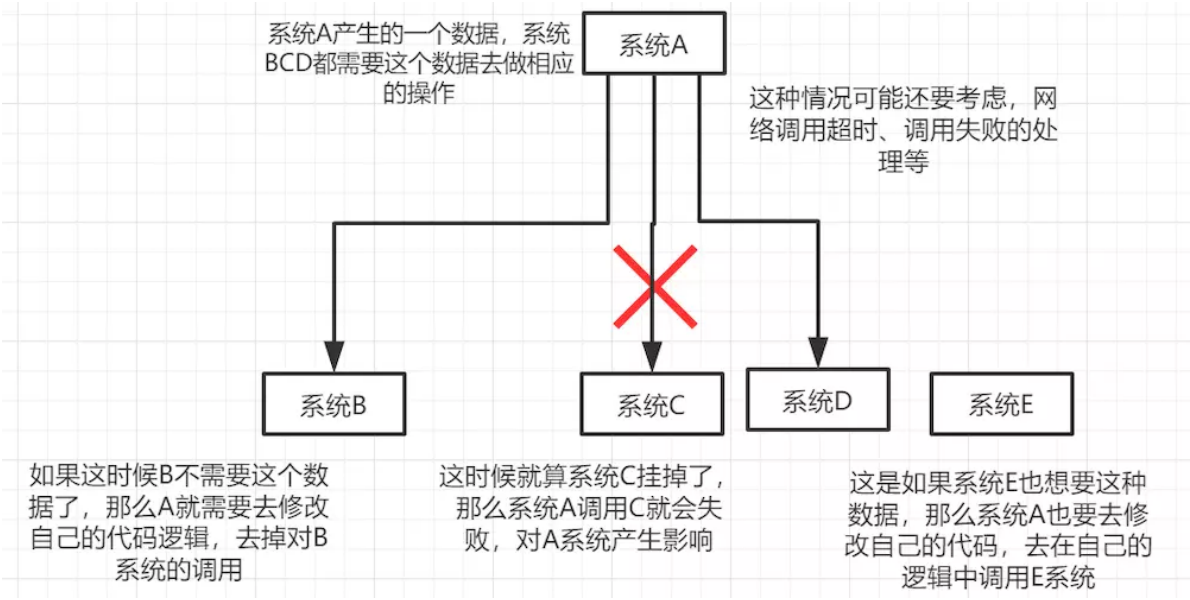
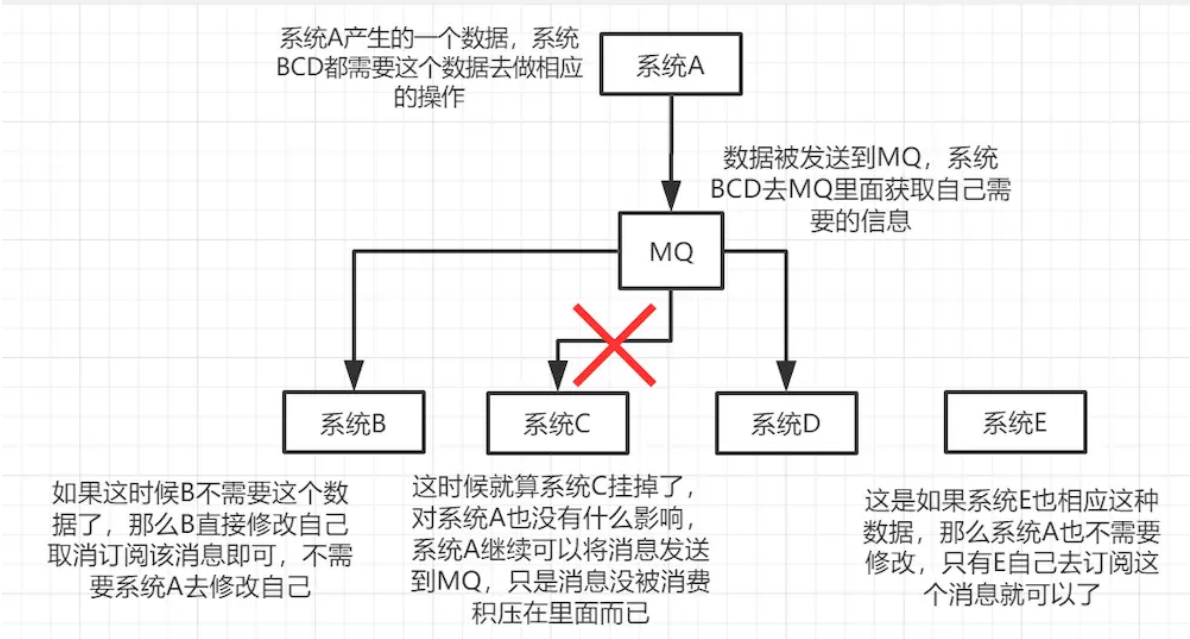
# 消息中间件

## 解耦

> 不使用消息中间件的情况

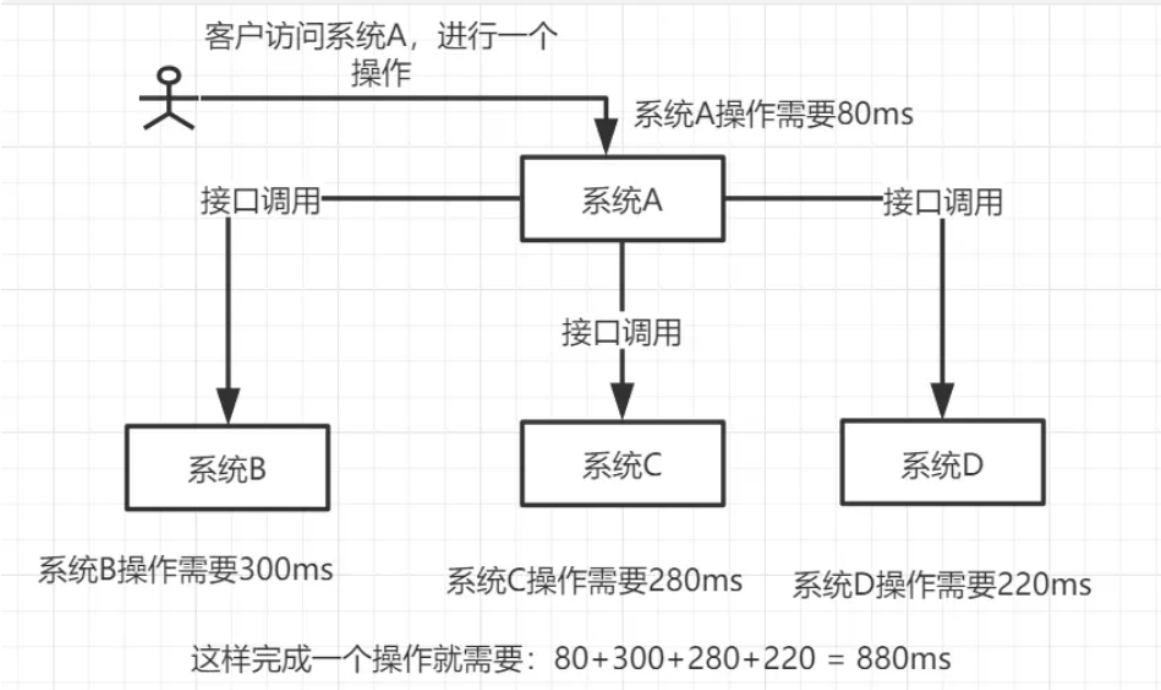



> 使用了消息中间件

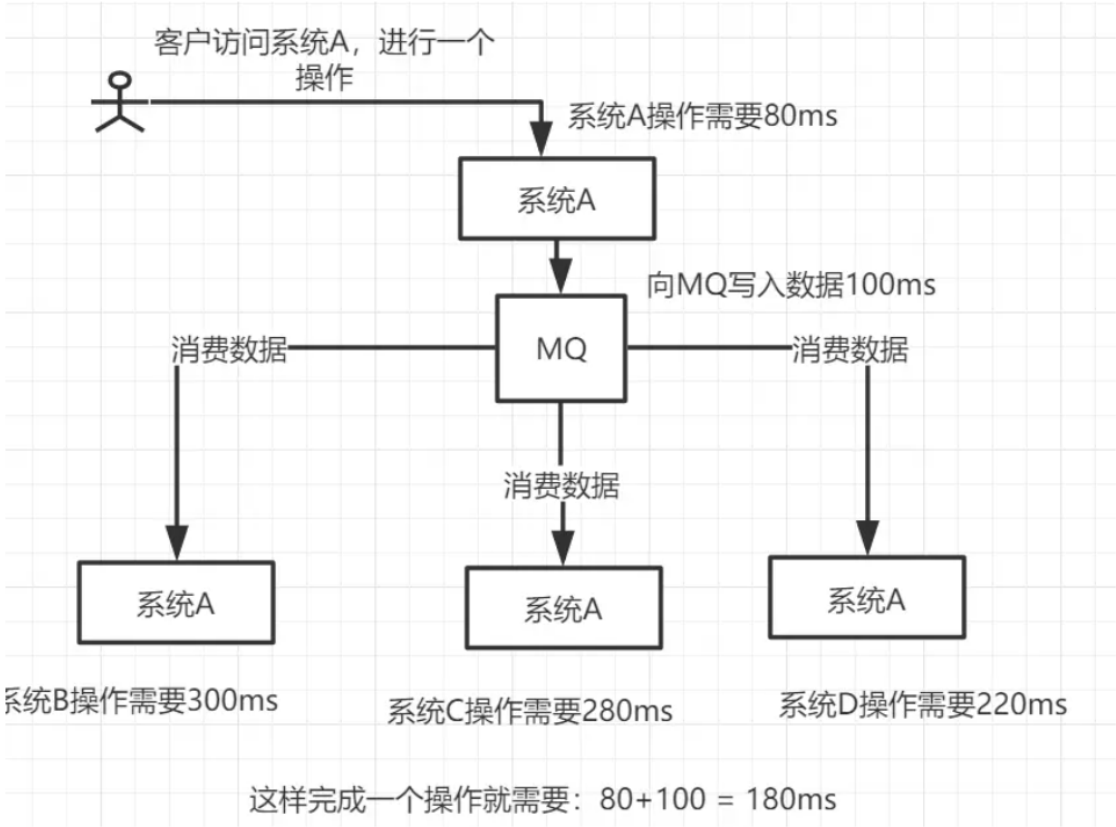


# 异步

> 不使用MQ的情况




> 使用MQ



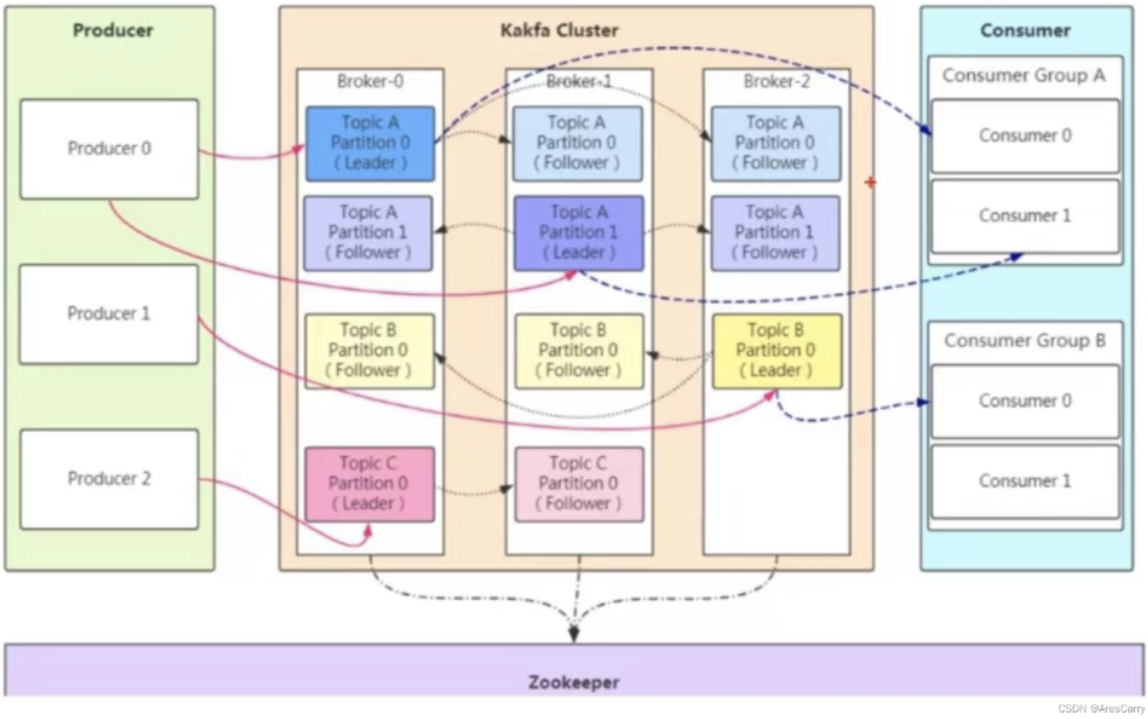

# 背景

- 消息中间件的天然特性 削封 异步 解耦，MCA的业务大部分以IO密集型系统为主，异步对性能的提升
- 之前的开发工作发现一些业务非常适合采用消息中间件 （关联关系，Push等等操作）

- 使用MQ-解决分布式事务问题 （消息中间件的使用-本地消息表）
- 慢SQL
- 业务耦合度高，维护很容易出现问题，如何使用消息中间件进行解耦

# kafka

> kafka是一个分布式、高吞吐量、高扩展性的消息队列系统,主要应用在日志收集系统和消息系统。




## kafka消息不丢失

> 生产者发送消息时消息丢失

- 使用带有回调函数的API
- 做好业务的重试工作
- 选择正确的 ack 参数配置 0,1,-1

```
@Retryable(value = KafkaException.class, maxAttempts = 3, backoff =
@Backoff(delay = 3600, multiplier = 1.5))
    public void sendMessage(String topic, Object o) {
        kafkaTemplate.send(topic, o).addCallback(success -> {
            eventMapper.insertEvent((MQEvent) o);
            logger.info("业务发起方成功发送消息->{} ", o.toString());
        }, failete -> {
            logger.error("业务方发送消息失败,准备重试->{} ,MsgId{} ", o.toString(),
((MQEvent) o).getMsgID().toString());
            throw new KafkaException("kafka业务发起方发送异常");
        });
    }
```

> 消费者消费数据消息丢失

- 手动提交offset

```java
public void consumeMessageFromSender(ConsumerRecord<?, ?> record, Acknowledgment ack) throws Exception{
    try {
        logger.info("业务接受放接收到了消息 msgID{}",ConvertObjectFromMQ(record).getMsgID());
        //业务处理
        reviceBusinessService.processBusiness(ConvertObjectFromMQ(record));
        //第一次 成功消费 参数重新组装改变状态码 并向发送方回调
        consumerReplyService.replyMessage(replyTopic, assembleAfterBusiness(ConvertObjectFromMQ(record)));
        logger.info("业务接受方成功发送回调函数 msgID{}",ConvertObjectFromMQ(record).getMsgID());
        ack.acknowledge();
        logger.info("业务方成功消费并且向发送方发送回调  MsgId{}",ConvertObjectFromMQ(record).getMsgID());
    } catch (Exception e) {
```

# 分布式事务

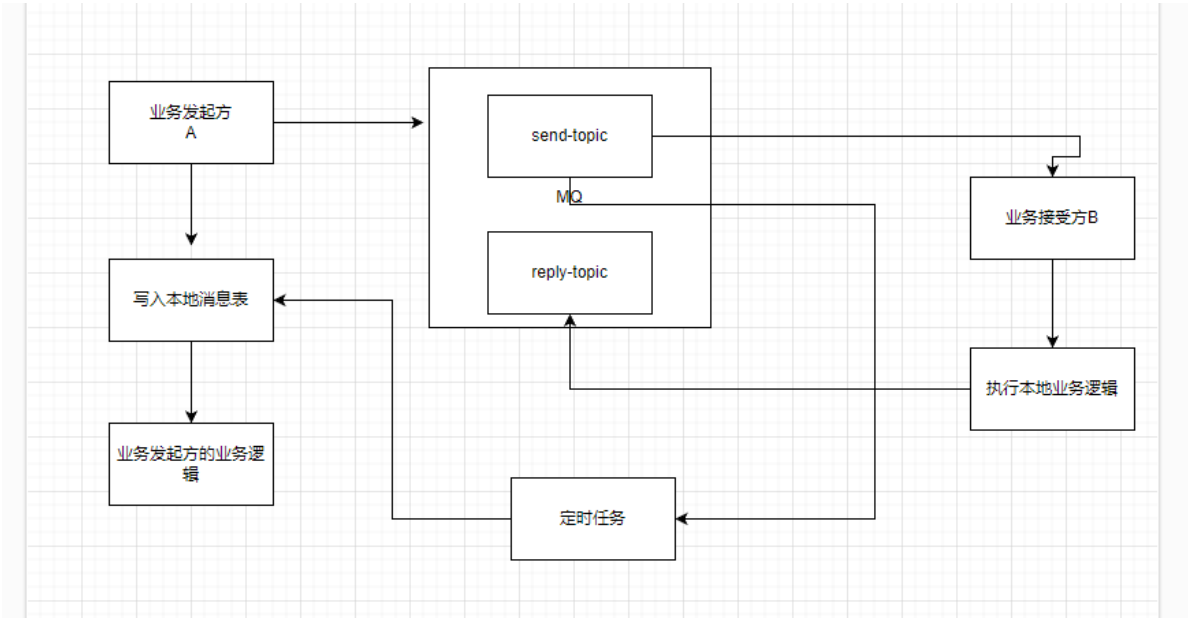
## 本地事务

```java
@Transactional
    public  void testTranscational(){
        serviceLocalBusiness();
        rpcCallOTherService();
    }
```

> 不是同一个JVM进程中的事务没办法进行控制

分布式事务

### 本地消息表




1. 业务方A发起事务执行本地业务 并写入本地消息表 （本地事务保证）
2. 发往send-topic
3. 业务接受方监听消息 执行本地业务
4. 业务接受方 执行完消息 发送消息往reply-topic 实现 **业务消息回调**
5. 定时任务扫表没有状态位出错的消息