

超级教程系列

# 微服务架构的分布式事务解决方案

分布式架构系统中，分布式事务是一个绕不过去的挑战！  
微服务架构的流行，让分布式事务问题日益突出！

补偿型

最大努力  
通知型

幂等性  
操作

异步  
确保型

定期  
校对

可查询  
操作

两阶段型

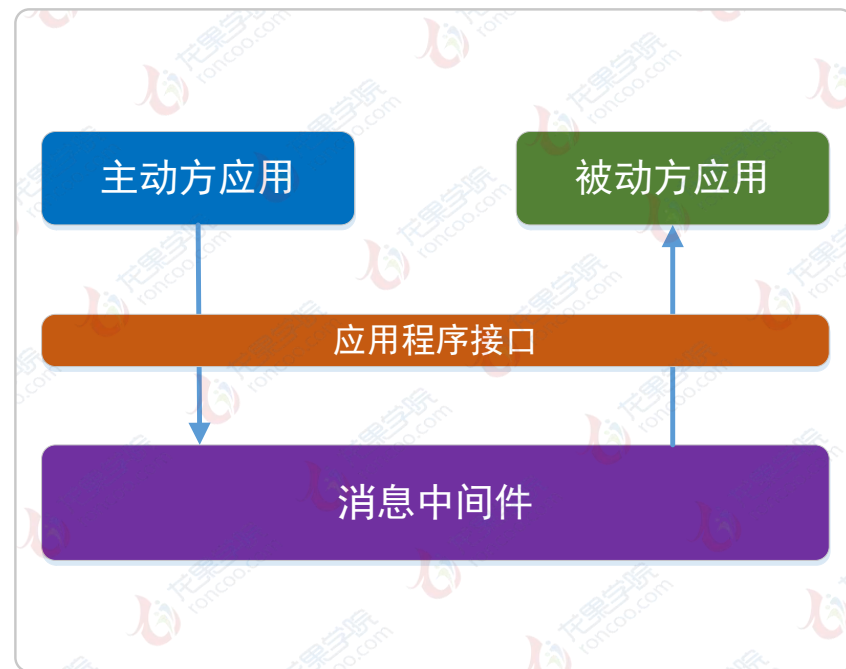
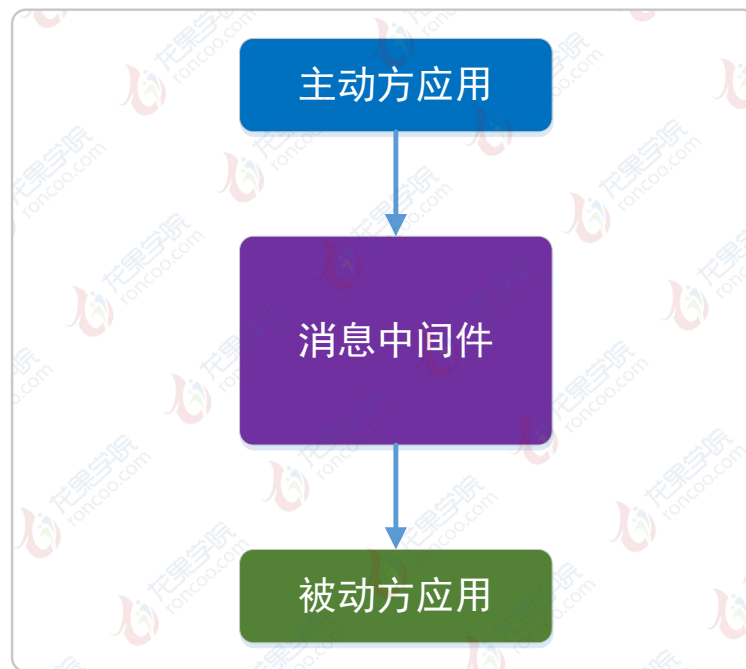


## 第04节

# 消息发送一致性（可靠消息的前提保障）

（基于可靠消息的最终一致性方案课程）

# 消息中间件的应用场景



消息中间件在分布式系统中的主要作用：**异步通讯、解耦、并发缓冲**

如图：通过引入消息中间件来解耦应用间（服务间）的直接调用，同时也会起到异步通讯和缓冲并发的作用

# 消息发送和投递的不可靠性



分布式部署环境下，需要通过网络进行通讯，就引入了数据传输的不确定性  
也就是CAP理论中的P（分区容错性的问题）



# 消息发送一致性

- **消息发送一致性**：是指产生消息的业务动作与消息发送的一致。  
( 也就是说，如果业务操作成功，那么由这个业务操作所产生的消息一定要成功投递出去，否则就丢消息 )



# 消息发送一致性如何保障？

## ➤ 处理方式1

/\*\* 支付订单处理 \*\*/

```
public void completeOrder() {
```

```
    // 订单处理（业务操作）
```

```
    orderBiz.process();
```

```
    // 发送会记原始凭证消息（发送消息）
```

```
    sendAccountingVoucherMsg ();
```

```
}
```

- 1、如果业务操作成功，执行消息发送前应用故障，消息发不出去，导致消息丢失（订单系统与会计系统的数据不一致）；
- 2、如果业务操作成功，应用正常，但消息系统故障或网络故障，也会导致消息发不出去（订单系统与会计系统的数据不一致）；

# 消息发送一致性如何保障？

## ➤ 处理方式2

```
/** 支付订单处理 **/  
public void completeOrder() {  
    // 发送会记原始凭证消息（发送消息）  
    sendAccountingVoucherMsg ();  
    // 订单处理（业务操作）  
    orderBiz.process();  
}
```

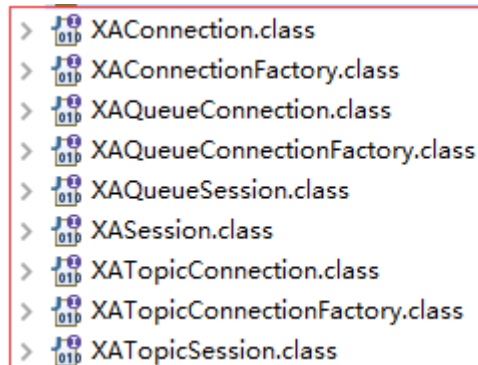
1、这种情况下，更不可控，消息发出去了，但业务可能会失败（订单系统与会计系统的数据不一致）；

**前面两种方式，都不能保证业务数据的一致性。**

**还有没有其他办法？**

# JMS标准中的XA协议方式是否可以保障发送一致性？

- JMS协议标准的API中，有很多以XA开头的接口，其实就是前面课程讲到的支持XA协议（基于两阶段提交协议）的全局事务型接口。
- JMS中的XA系列接口，可以提供分布式事务支持。
- 但引用了XA方式的分布式事务，又会带来很多的局限：
  - 要求业务操作的资源必须支持XA协议（并不是所有资源都支持XA）
  - 两阶段提交协议的成本
  - 持久化成本等DTP模型的局限性（全局锁定，成本高，性能低）

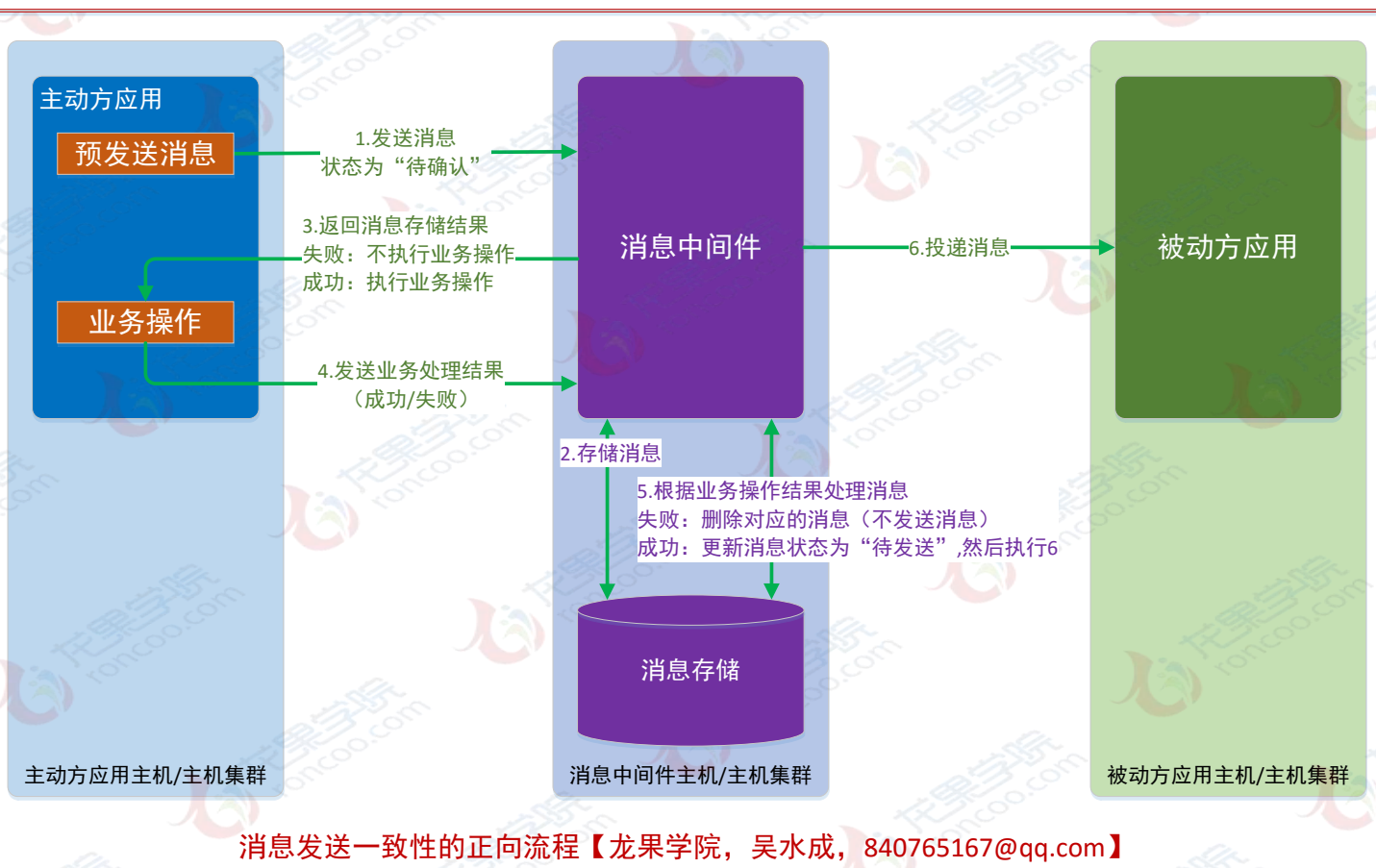
A screenshot of a Java IDE showing a list of JMS XA API classes. The classes are listed with their package icons and names: XAConnection.class, XAConnectionFactory.class, XAQueueConnection.class, XAQueueConnectionFactory.class, XAQueueSession.class, XASession.class, XATopicConnection.class, XATopicConnectionFactory.class, and XATopicSession.class. Each class name is preceded by a small icon representing the JMS package.

```
> XAConnection.class
> XAConnectionFactory.class
> XAQueueConnection.class
> XAQueueConnectionFactory.class
> XAQueueSession.class
> XASession.class
> XATopicConnection.class
> XATopicConnectionFactory.class
> XATopicSession.class
```

引入XA，违背了柔性事务的初衷！

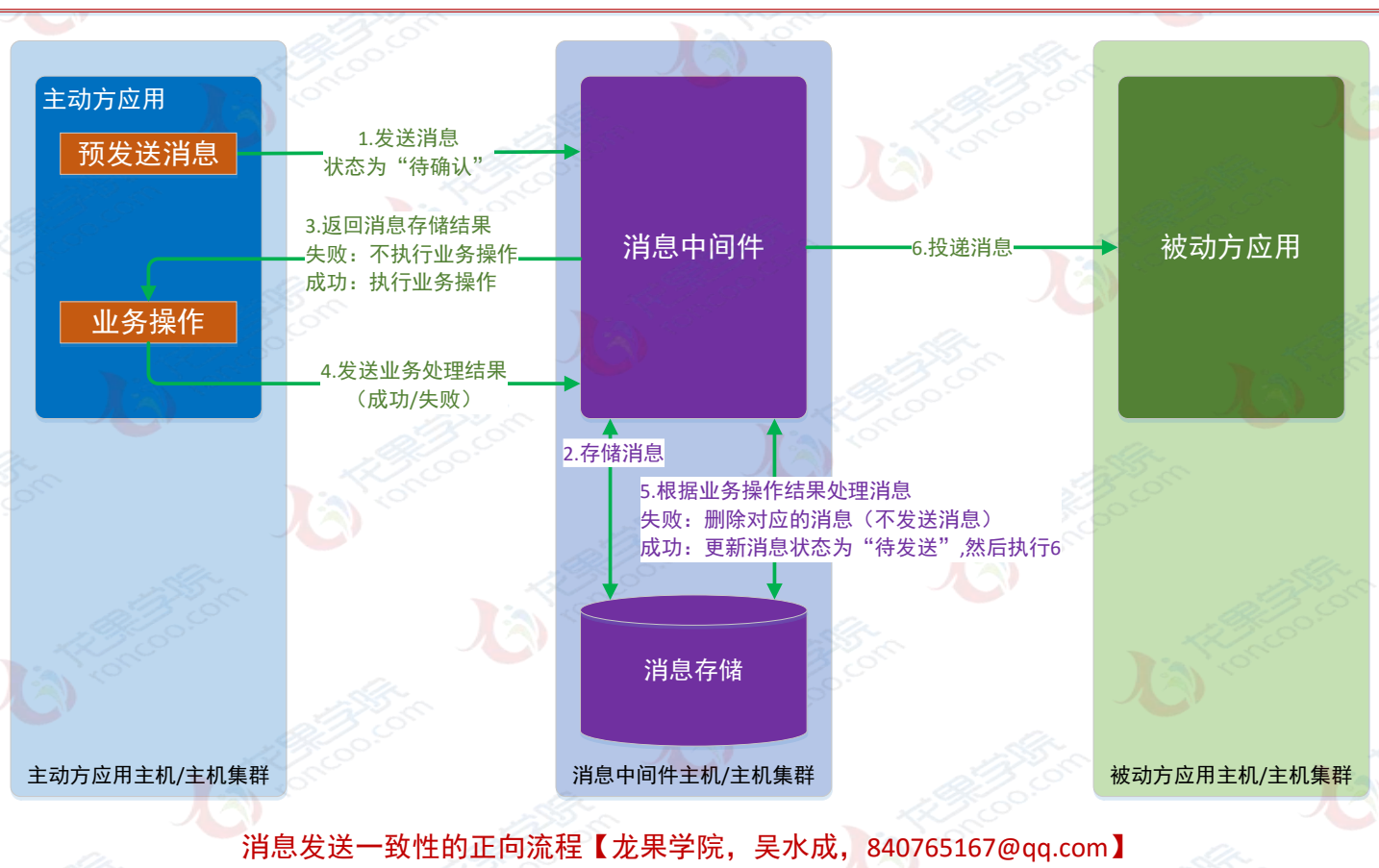


# 消息发送一致性：变通的做法



1. 主动方应用先把消息发给消息中间件，消息状态标记为“待确认”；
2. 消息中间件收到消息后，把消息持久化到消息存储中，但并不向被动方应用投递消息；
3. 消息中间件返回消息持久化结果（成功/失败），主动方应用根据返回结果进行判断如何进行业务操作处理：
  - a) 失败：放弃业务操作处理，结束（必要时向上层返回失败结果）；
  - b) 成功：执行业务操作处理；
4. 业务操作完成后，把业务操作结果（成功/失败）发送给消息中间件；
5. 消息中间件收到业务操作结果后，根据业务结果进行处理：
  - a) 失败：删除消息存储中的消息，结束；
  - b) 成功：更新消息存储中的消息状态为“待发送（可发送）”，紧接着执行消息投递；
6. 前面的正向流程都成功后，向被动方应用投递消息；

# 消息发送一致性：变通的做法



- 消息发送一致性方案的正向流程是可行的，但异常流程怎么处理呢？
- 消息发送到消息中间件中能得到保障了，但消息的准确消费（投递）又如何保障呢？
- 有没有支持这种发送一致性流程的现成消息中间件？

（往下的课程为大家逐一讲述）

# 谢谢

THANK YOU

技术支持：Along、Hugo、Peter



龙果学院官方微信公众号