

什么是可靠消息最终一致性

大道至简 悟在天成

Andy | 悟纤

可靠消息最终一致

可靠消息最终一致性方案是指当事务发起方执行完本地事务后并发出一条消息，事务参与方（消息消费者）一定能够接收消息并处理事务成功，此方案强调的是只要消息发送给事务参与方最终事务要达到一致。

可靠消息最终一致

此方案是利用消息中间件来完成，如下图：



事务发起方（消息生产方）将消息发送给消息中间件，事务参与方从消息中间件接收消息；事务发起方和消息中间件之间，事务参与方和消息中间件之间都是通过网络进行通信，由于网络的不确定会导致分布式事务。

可靠消息最终一致

可靠消息最终一致性要解决以下几个问题：

(1) **本地事务与消息发送的原子性问题**：事务发起方在本地事务执行成功之后消息必须发送出去，否则就丢弃消息。也就是说：本地消息和消息的发送要么一起成功，要么一起失败。

可靠消息最终一致

可靠消息最终一致性要解决以下几个问题：

```
begin transaction;  
    //1.发送消息  
    //2.数据库操作  
  
commit transaction;
```

这种方式也不能保证数据库操作与发送消息的一致性，因为可能消息发送成功，本地事务操作失败。

可靠消息最终一致

可靠消息最终一致性要解决以下几个问题：

(2) **事务参与方接收消息的可靠性**：事务参与方必须能够从消息队列中接收到消息，如果接收消息失败可以重复接收消息。

(3) **消息重复消费的问题**：由于网络的存在，如某一个消费节点超时但是消费成功了，此时消息中间件会重复投递消息，就导致了消息的重复消费。

本地消息表

大道至简 悟在天成

Andy | 悟纤

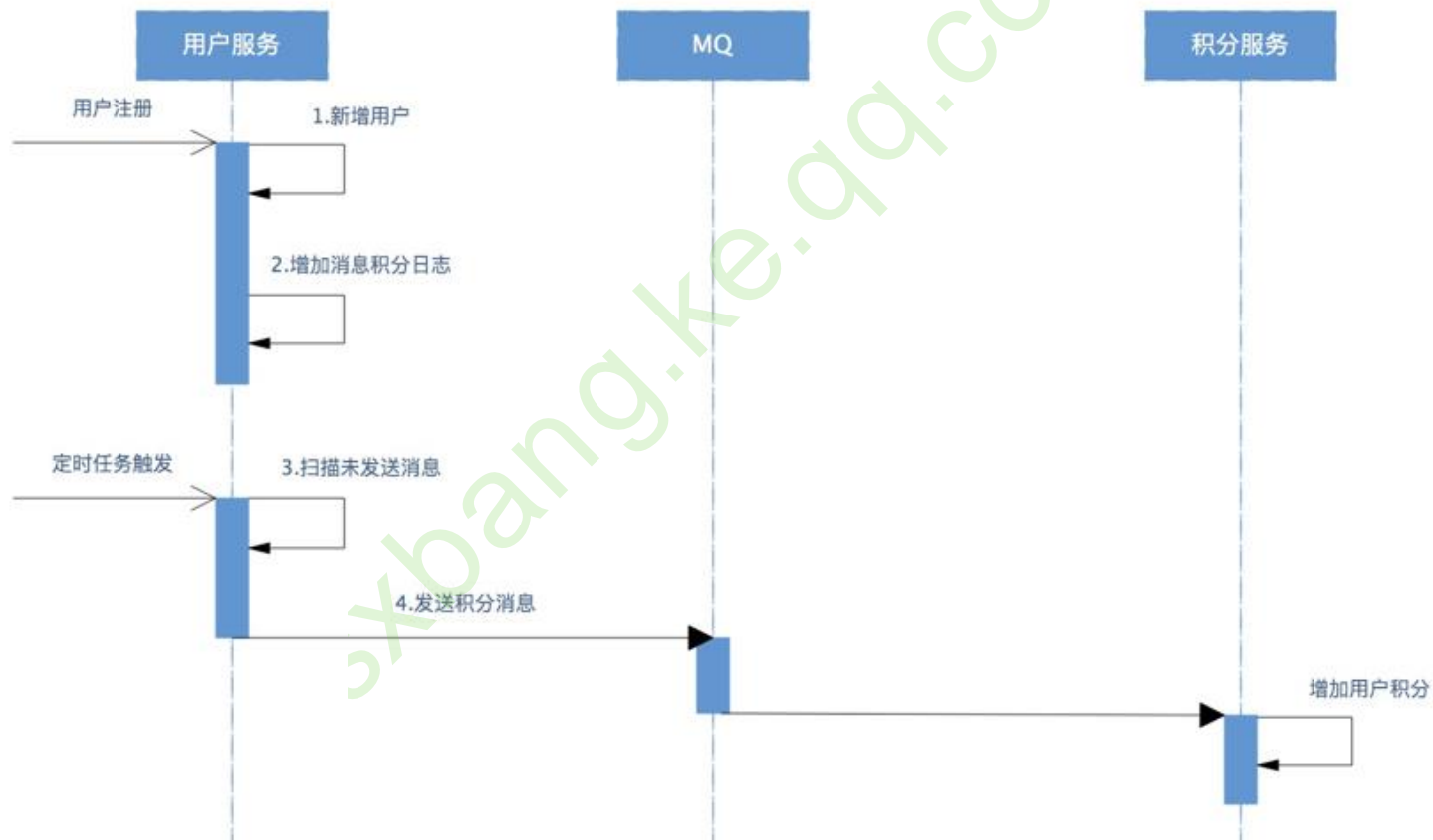
本地消息表

本地消息表方案是由ebay提出的，此方案的核心是通过本地事务保证数据业务操作和消息的一致性，然后通过定时任务将消息发送至消息中间件，待消息发送给消费方成功再将消息删除。

简单理解：消息本地化，定时任务定时发送消息中间件。

本地消息表

1.1 注册送积分分析



RocketMQ

大道至简 悟在天成

Andy | 悟纤

RocketMQ

RocketMQ是一个来自阿里巴巴的分布式消息中间件，于2012年开源，并在2017年成为Apache顶级项目。Apache RocketMQ 4.3之后的改版正式支持事务消息，为分布式事务实现提供了便利性支持。

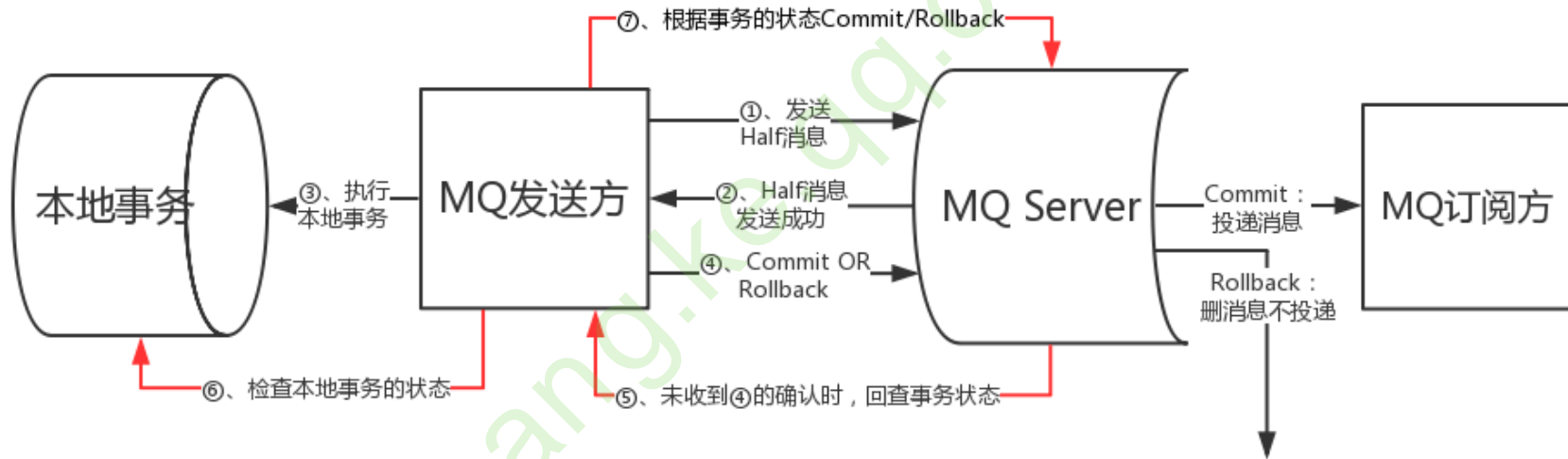
RocketMQ事务消息设计则主要是为了解决Producer端的消息发送与本地事务执行的原子性问题。

RocketMQ

RocketMQ的设计中broker与producer端的双向通信能力，使得broker天生可以作为一个事务协调器存在，而RocketMQ本身提供的存储机制为事务消息提供了持久化能力，RocketMQ的高性能机制以及可靠消息机制则为事务消息在系统发生异常时依然能够保证达成事务的最终一致性。

在RocketMQ 4.3后实现了完整的事务消息，实际上其实是对本地消息表的封装，将本地消息移动到了MQ内部，解决Producer端的消息发送与本地事务执行的原子性问题。

RocketMQ



使用RocketMQ实战

大道至简 悟在天成

Andy | 悟纤

一、RocketMQ的安装

1.1 下载RocketMQ

上官网下载RocketMQ (Binary包) :

http://rocketmq.apache.org/release_notes/release-notes-4.4.0/

一、RocketMQ的安装

1.2 启动/关闭

#启动RocketMQ的注册中心

nohup sh mqnamesrv &

#启动broker


nohup sh mqbroker -n localhost:9876 &

一、RocketMQ的安装

1.2 启动/关闭

启动成功之后使用jps命令查看：

```
linxiangxiandeMacBook-Pro:bin linxiangxian$ jps  
1760  
2443 BrokerStartup  
1756  
2444 Jps  
2429 NamesrvStartup  
847
```



一、RocketMQ的安装

关闭指令：

```
#关闭broker
```

```
sh mqshutdown broker
```

```
#关闭namesrv
```

```
sh mqshutdown namesrv
```

一、RocketMQ的安装

1.3 使用RocketMQ控制台

方式一：下载源码 - 编译 - 打包 - 运行：

<https://github.com/apache/rocketmq-externals/tree/master/rocketmq-console>

下载打包jar运行：

CSDN下载：<https://download.csdn.net/download/linxingliang/12003245>

百度云盘下载：<https://pan.baidu.com/s/1yGvD5kvxcg2VH4rwoBdK9g>

一、RocketMQ的安装

1.3 使用RocketMQ控制台

启动:

```
java -jar rocketmq-console-ng-201903.jar --server.port=8080 --rocketmq.config.namesrvAddr=127.0.0.1:9876
```

访问地址: <http://127.0.0.1:8080/>

RocketMQ配置

大道至简 悟在天成

Andy | 悟纤

二、RocketMQ配置

2.1 复制一份项目

将tx-transfer复制一份出来，取名为：
rocketmq-eventually-tx-transfer。

二、RocketMQ配置

2.2 在父项目声明spring-cloud-alibaba依赖

在lcn-tx-transfer父类pom.xml文件，添加版本：

```
<spring-cloud-alibaba.version>0.9.0.RELEASE</spring-cloud-alibaba.version>
```

声明spring-cloud-alibaba依赖：

```
<dependency>
  <groupId>org.springframework.cloud</groupId>
  <artifactId>spring-cloud-alibaba-dependencies</artifactId>
  <version>${spring-cloud-alibaba.version}</version>
  <type>pom</type>
  <scope>import</scope>
</dependency>
```

二、RocketMQ配置

2.3 在子项目引入RocketMQ依赖

在需要RocketMQ的子项目引入依赖：

```
<dependency>  
  <groupId>org.apache.rocketmq</groupId>  
  <artifactId>rocketmq-spring-boot-starter</artifactId>  
  
</dependency>
```


二、RocketMQ配置

2.4 在子项目配置生成组信息和连接地址

在applicationo.properties添加如下配置：

```
# 配置组名称和name server ,注意组名称不同服务要取名不一样。
```

```
rocketmq.producer.group = producer_bank1
```

```
rocketmq.name-server = 127.0.0.1:9876
```

扣款编码

大道至简 悟在天成

Andy | 悟纤

三、扣款编码

扣款编码主要思路：

- (1) 使用发送事务消息的方法：`rocketMQTemplate.sendMessageInTransaction` 进行发送事务消息。
- (2) 消息发送成功之后会回调到实现了 `RocketMQLocalTransactionListener`。
- (3) 在 `RocketMQLocalTransactionListener` 主要实现执行本地事务和校验本地事务是否执行了的方法。
- (4) 另外要确保方法的幂等性。

三、扣款编码

3.1 定义TxNo实体类

定义一个TxNo事务的实体类，用来保存每次执行的事务，确保方法的幂等性：

```
@Entity
public class TxNo {
    @Id

    private String txNo;
}
```

三、扣款编码

3.2 定义TxNo的持久类操作

```
public interface TxNoReposiroty extends JpaRepository<TxNo, String> {  
}
```

三、扣款编码

3.3 定义消息传递对象

消息在传递的时候，需要保存一些基本的信息，事务id, 转账信息：

```
public class AccountChangeEvent {  
    private long uid;  
    private BigDecimal money;  
    private long toUid;//转账给谁  
  
    private String txNo = UUID.randomUUID().toString();  
  
}
```

三、扣款编码

3.4 定义AccountService接口

定义接口，主要是发送消息和扣减金额接口：

```
public interface AccountService {  
    /**扣减金额*/  
    public void debit(AccountChangeEvent accountChangeEvent);  
  
    /**发送消息*/  
    public void sendMsgToUpdateAccountBalance(AccountChangeEvent accountChangeEvent);  
  
}
```

三、扣款编码

3.5 定义AccountService接口的实现

```
@Service
public class AccountServiceImpl implements AccountService {
    @Autowired
    private AccountRepository accountRepository;
    @Autowired
    private TxNoRepository txNoRepository;

    @Autowired
    private RocketMQTemplate rocketMQTemplate;

    /**
     * 发送消息
     */
    @Override
    public void sendMsgToUpdateAccountBalance(AccountChangeEvent accountChangeEvent) {

        /**
         * txProducerGroup: 事务生成组
         * destination: topicName
         * message: 信息
         * arg: 扩展参数
         */
        Account account = accountRepository.findById(accountChangeEvent.getUid());
        if(account == null) {
            throw new RuntimeException("Account not exist");
        } else if(account.getMoney().compareTo(accountChangeEvent.getMoney())<0) {
            throw new RuntimeException("Money not enough");
        }
    }
}
```


三、扣款编码

3.6 实现回调本地事务

```
@Component
@RocketMQTransactionListener(txProducerGroup="producer_group_bank1")
public class ProducerTransferListener implements RocketMQLocalTransactionListener{

    @Autowired
    private AccountService accountService;
    @Autowired
    private TxNoReposiroty txNoReposiroty;
    /**
     * 事务消息发送成功后的回调方法。
     * 执行事务方法。
     */
    @Override
    public RocketMQLocalTransactionState executeLocalTransaction(Message msg, Object arg) {
        System.out.println("ProducerTransferListener.executeLocalTransaction().V1,");
        String str = new String((byte[])msg.getPayload());//object是byte[]数组类型
        AccountChangeEvent accountChangeEvent = JSON.parseObject(str,AccountChangeEvent.class);
        System.out.println("uid="+accountChangeEvent.getUid()+" ,money="+accountChangeEvent.getMoney());
        accountService.debit(accountChangeEvent);
        return RocketMQLocalTransactionState.COMMIT;
    }

    /**
     * 事务检查
     */
    @Override
    public RocketMQLocalTransactionState checkLocalTransaction(Message msg) {
        System.out.println("ProducerTransferListener.checkLocalTransaction()");
        String str = new String((byte[])msg.getPayload());//object是byte[]数组类型
        AccountChangeEvent accountChangeEvent = JSON.parseObject(str,AccountChangeEvent.class);
```

三、扣款编码

3.7 修改一下BusinessService的transfer方法

这个方法要修改一下，也没有必要要添加事务了：

```
public boolean transfer(BigDecimal money) {  
  
    AccountChangeEvent accountChangeEvent = new AccountChangeEvent();  
    accountChangeEvent.setUid(1001L);  
    accountChangeEvent.setToUid(1002L);  
    accountChangeEvent.setMoney(money);  
    accountService.sendMsgToUpdateAccountBalance(accountChangeEvent);  
    return true;  
  
}
```

收款编码

大道至简 悟在天成

Andy | 悟纤

四、收款编码

收款编码主要思路：

- (1) 主要监听RocketMQListener接收消息。
- (2) 实现方法onMessage。
- (3) 确保增加钱的方法的幂等性。

四、收款编码

4.1 定义TxNo2实体类

这里主要是一个库了，所以定义另外一个实体类来确保幂等性，还有别的方式，就是添加分支ID参数，这样的话，就只需要一个表就能搞定了：

```
@Entity
public class TxNo2 {
    @Id

    private String txNo;
}
```

四、收款编码

4.2 定义TxNo的持久类操作

```
public interface TxNo2Reposiroty extends JpaRepository<TxNo, String> {  
}
```

四、收款编码

4.3 确保increase方法的幂等性

```
@Override
@Transactional
public void increase(AccountChangeEvent event) {
    if(txNo2Repository.existsById(event.getTxNo())) {
        System.out.println("执行过了，不重复执行");
        return;
    }
    Account account = accountRepository.findById(event.getToUid());
    if(account == null) {
        throw new RuntimeException("Account not exist");
    }

    account.setMoney(account.getMoney().add(event.getMoney()));
    accountRepository.save(account);

    txNo2Repository.save(new TxNo2(event.getTxNo()));
}
```

四、收款编码

4.4 监听消息

```
@Component
@RocketMQMessageListener(consumerGroup="consumer_group_transfer",topic="topic_transfer")
public class ConsumerTransferListener implements RocketMQListener<String>{

    @Autowired
    private AccountService accountService;

    @Override
    public void onMessage(String message) {
        System.out.println("接收到消息处理");
        AccountChangeEvent accountChangeEvent = JSON.parseObject(message,AccountChangeEvent.class);
        accountService.increase(accountChangeEvent);
    }
}
```


四、收款编码

4.5 验证分布式事务

访问地址进行测试：

<http://127.0.0.1:2222/transfer?money=1>