**分布式**

# 分布式理论

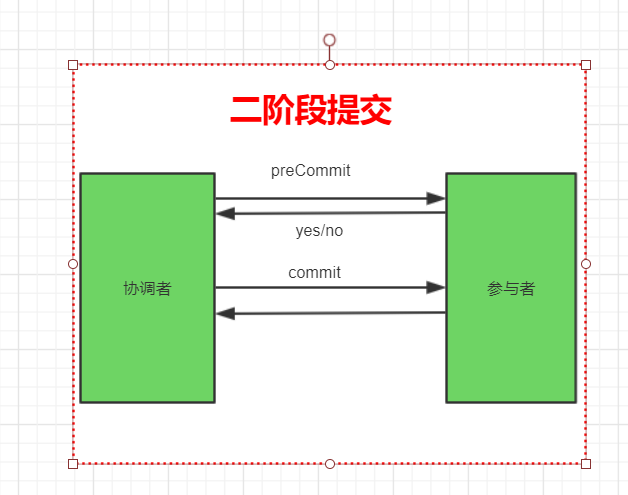
## cap理论

## Base理论

# 分布式事务

## 二阶段提交

### 流程



### 过程描述

(1) 请求阶段(表决)：

事务协调者通知每个参与者准备提交或取消事务，然后进入表决过程，参与者要么在本地执行事务，写本地的redo和undo日志，但不提交，到达一种"万事俱备，只欠东风"的状态。请求阶段，参与者将告知协调者自己的决策: 同意(事务参与者本地作业执行成功)或取消（本地作业执行故障）

(2) 提交阶段(执行):

在该阶段，写调整将基于第一个阶段的投票结果进行决策: 提交或取消

当且仅当所有的参与者同意提交事务，协调者才通知所有的参与者提交事务，否则协调者将通知所有的参与者取消事务

参与者在接收到协调者发来的消息后将执行响应的操作

### 缺点

1.同步阻塞问题。执行过程中，所有参与节点都是事务阻塞型的。

当参与者占有公共资源时，其他第三方节点访问公共资源不得不处于阻塞状态。

2.单点故障。由于协调者的重要性，一旦协调者发生故障。

参与者会一直阻塞下去。尤其在第二阶段，协调者发生故障，那么所有的参与者还都处于锁定事务资源的状态中，而无法继续完成事务操作。（如果是协调者挂掉，可以重新选举一个协调者，但是无法解决因为协调者宕机导致的参与者处于阻塞状态的问题）

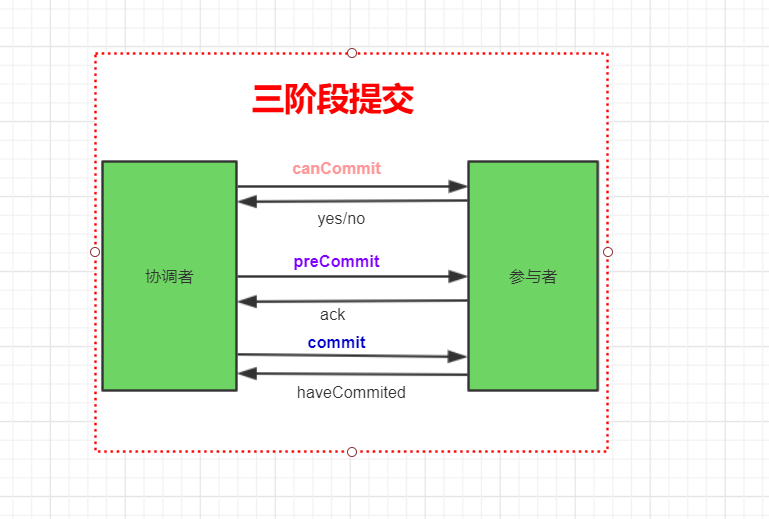
3.数据不一致。在二阶段提交的阶段二中，当协调者向参与者发送commit请求之后，发生了局部网络异常或者在发送commit请求过程中协调者发生了故障，这回导致只有一部分参与者接受到了commit请求。

而在这部分参与者接到commit请求之后就会执行commit操作。但是其他部分未接到commit请求的机器则无法执行事务提交。于是整个分布式系统便出现了数据不一致性的现象。

## 三阶段提交

减少了锁表的时间

### 流程图



### 过程描述

1.CanCommit阶段  
3PC的CanCommit阶段其实和2PC的准备阶段很像。  
协调者向参与者发送commit请求，参与者如果可以提交就返回Yes响应，否则返回No响应。

2.PreCommit阶段  
Coordinator根据Cohort的反应情况来决定是否可以继续事务的PreCommit操作。  
根据响应情况，有以下两种可能。  
A.假如Coordinator从所有的Cohort获得的反馈都是Yes响应，那么就会进行事务的预执行：  
发送预提交请求。Coordinator向Cohort发送PreCommit请求，并进入Prepared阶段。  
事务预提交。Cohort接收到PreCommit请求后，会执行事务操作，并将undo和redo信息记录到事务日志中。  
响应反馈。如果Cohort成功的执行了事务操作，则返回ACK响应，同时开始等待最终指令。

B.假如有任何一个Cohort向Coordinator发送了No响应，或者等待超时之后，Coordinator都没有接到Cohort的响应，那么就中断事务：  
发送中断请求。Coordinator向所有Cohort发送abort请求。  
中断事务。Cohort收到来自Coordinator的abort请求之后（或超时之后，仍未收到Cohort的请求），执行事务的中断。

3.DoCommit阶段

该阶段进行真正的事务提交，也可以分为以下两种情况:

执行提交

A.发送提交请求。Coordinator接收到Cohort发送的ACK响应，那么他将从预提交状态进入到提交状态。并向所有Cohort发送doCommit请求。  
B.事务提交。Cohort接收到doCommit请求之后，执行正式的事务提交。并在完成事务提交之后释放所有事务资源。  
C.响应反馈。事务提交完之后，向Coordinator发送ACK响应。  
D.完成事务。Coordinator接收到所有Cohort的ACK响应之后，完成事务。

中断事务

Coordinator没有接收到Cohort发送的ACK响应（可能是接受者发送的不是ACK响应，也可能响应超时），那么就会执行中断事务。

### 三阶段提交协议和两阶段提交协议的不同

对于协调者(Coordinator)和参与者(Cohort)都设置了超时机制（在2PC中，只有协调者拥有超时机制，即如果在一定时间内没有收到cohort的消息则默认失败）。

在2PC的准备阶段和提交阶段之间，插入预提交阶段，使3PC拥有CanCommit、PreCommit、DoCommit三个阶段。

PreCommit是一个缓冲，保证了在最后提交阶段之前各参与节点的状态是一致的。

### 三阶段提交协议的缺点

如果进入PreCommit后，Coordinator发出的是abort请求，假设只有一个Cohort收到并进行了abort操作，

而其他对于系统状态未知的Cohort会根据3PC选择继续Commit，此时系统状态发生不一致性。

## 刚性事务

满足ACID的事务

导致锁表的时间太长，降低性能

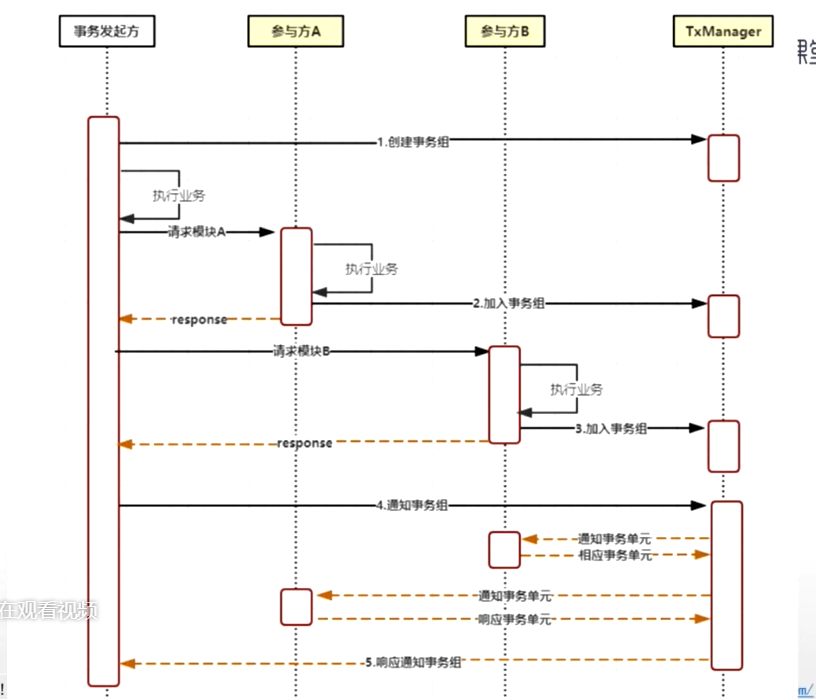
## TCC

# LCN分布式框架

是通过groupId来判断多个服务是否在处于同一个事务。

默认采用的3PC事务。

@TCCTransaction采用的是tcc事务



# Session共享及单点登录

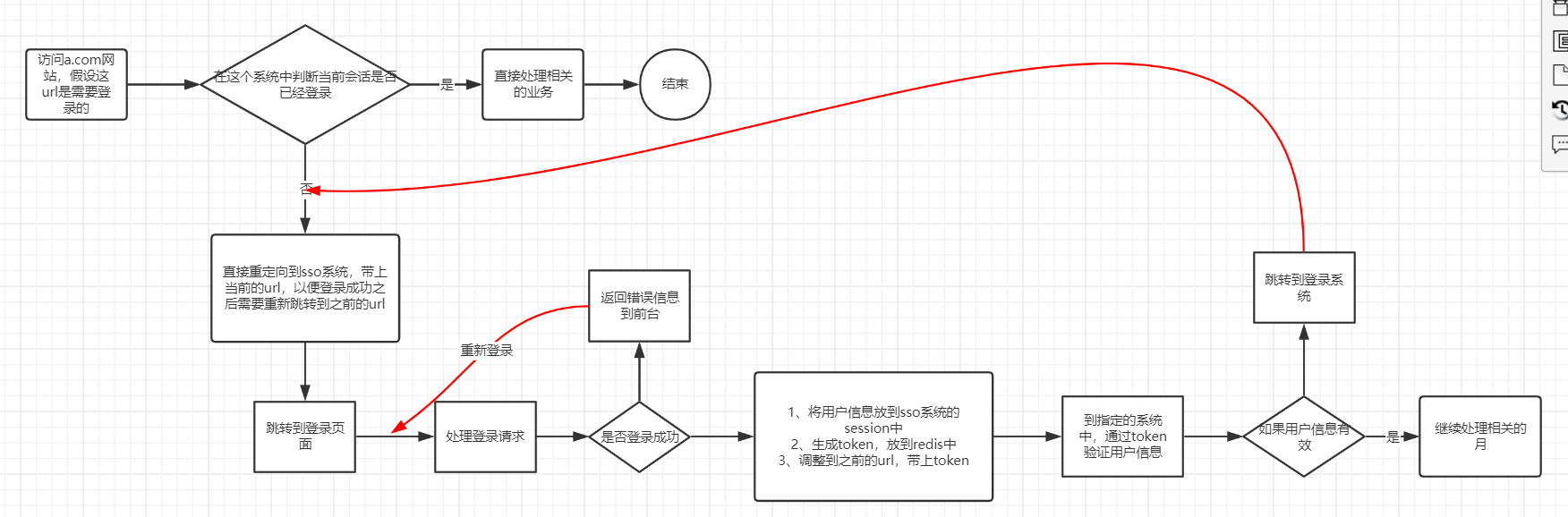
## session共享

是由客户端进行实现的，每次请求都带上相同的sessionid

单机环境

集群环境

## 单点登录



获取当前会话，根据从浏览器中携带的sessionId来获取session。



# 分布式调度

## Spring-task

是同步模式