两阶段三阶段提交协议

## Two Phase Commit

两阶段提交协议是为了保持各个节点的数据一致性而设计的。在这个分布式系统中，其中一个节点作为协调者（coordinator），负责统合各个节点的意见。其它节点作为参与者（participants）

#### 具体协议

该协议分为两个阶段，提交请求阶段和提交执行阶段。

1. 提交请求阶段
   1. 协调者向参与者发送询问是否可以执行操作
   2. 参与者回复是否同意执行操作
2. 提交执行阶段
   1. 协调者发送知否确认执行的通知
   2. 参与者回复是否成功执行的结果

图示

描述已自动生成

#### 缺陷

这个协议如果节点工作正常时可以正常执行，但是假设节点崩溃时会出现问题。现在假设有协调者节点为C，参与者节点N1、N2、N3.

在提交阶段时，如果只有C崩溃，而且N1、N2、N3都没有收到提交信息，可以安全地推断没有发生提交。但是C与N1同时崩溃的话，无法推断提交是否已经发生。因为N1既有可能已经提交，也有可能没有提交。所以整个分布式系统需要等待N1或者C恢复才能继续正常运行，导致系统阻塞。

#### 实现

我使用了http协议模拟这两个协议的过程。运行了程序之后，在浏览器地址框输入http://localhost:8000/two-phase会显示模拟主页面。

主页面包括一个协调者，多个参与者，通过点击模拟两阶段协议。

#### 实验

下图为实验环境，有一个协调者和四个参与者。

图形用户界面

描述已自动生成

#### 提交成功实验

将run这个变量修改为10

1. 首先在run右边的标题输入10，然后点击Prepare。这一步是模拟第一阶段协调者向参与者发送prepare请求。

图形用户界面, 应用程序

描述已自动生成

1. 现在所有的参与者都接收到了prepare请求，可以看见参与者的状态都为prepare。接下来点击所有参与者的Prepared按钮，这一步模拟参与者第一阶段的prepared响应。

图形用户界面

描述已自动生成

1. 现在所有的参与者都已经响应完毕。然后点击绿色的Commit按钮，模拟协调者第一阶段的Commit请求。

图形用户界面, 应用程序

描述已自动生成

1. Commit之后，所有参与者的值都被同步成10，提交成功。

#### 提交失败实验

将go这个变量修改为20

1. 首先在go标题的右边输入20，然后点击Prepare。这一步是模拟第一阶段协调者向参与者发送prepare请求。

图形用户界面, 应用程序

描述已自动生成

1. 现在所有的参与者都接收到了prepare请求，可以看见参与者的状态都为prepare。接下来点击一部分参与者的Prepared按钮，这一步模拟参与者第一阶段的prepared响应。不点击Prerpaed按钮表示参与者不同意变更。

图形用户界面, 应用程序

描述已自动生成

1. 现在进入下一步，点击绿色的Commit按钮，模拟协调者第一阶段的Commit请求。

图形用户界面, 应用程序

描述已自动生成

1. Commit之后，go的值并没有变化，提交失败。

## Three Phase Commit

三阶段提交协议希望能够解决两阶段提交协议的缺陷导致的系统阻塞。在Prepare阶段和Commit阶段之间加入PreCommit阶段。现在假设有协调者C，参与者N1、N2、N3。

现在，假设在提交阶段C崩溃，同时N1崩溃。如果N2或N3有一个处于PrePare ToCommit、状态，那么可视作所有的参与者都已经完成提交。如果N2或N3没有处于PrepareToCommit状态，那么可视作所有的参与者都否认提交。

#### 实验

三阶段提交协议只测试提交成功的情况。把score改变成60。

1. 首先协调者向所有参与者发送Prepare更改请求

图形用户界面, 应用程序

描述已自动生成

1. 所有的协调节点都必须同意准备更改，如果没有全部同意，更改在下一个阶段取消。

图形用户界面

描述已自动生成

1. 现在点击绿色的PreCommit按钮，进入第二阶段，协调者发送PreCommit请求。

图形用户界面

描述已自动生成

1. 这个阶段只需要有一个协调者同意提交，在第三阶段时就会更新值。否则不会更新值。
2. 点击绿色的Commit按钮，进入第三阶段。

图形用户界面, 应用程序

描述已自动生成

1. 修改成功