Paxos 算法实验

姓名：李研　　班级：硕 1902 班　　学号：M201973122

Paxos 算法实验

姓名：李研　　班级：硕 1902 班　　学号：M201973122

1. 实验目的

本次实训考察学生对 Paxos 算法思想的理解，并完成相关程序设计。

2. 实验内容

(1) 根据 Paxos 算法流程完成相关核心成员函数设计。
(2) 需要了解的基本概念：
- 角色
  - 提议者（proposer）：提案的发起者。
  - 批准者（acceptor）：对提案进行批准。
- 提案（propose）
  - Proposer 可以提出提案，最终要达成一致的 value 就在提案里；
  - Acceptor 根据提案的编号来选择是否接受（accept）提案；
  - 如果超过半数的 Acceptor 接收了一个提案，那么这个提案就被接受（accepted）了，提案里的 value 也就被选定了。
- 最终如何达成一致：一个提案被接受后，如果还有 Proposer 在继续提出提案，会修改提案的 value 为已被接受的 value，然后继续下一轮提案。

(3) Paxos 算法更详细的流程和思想需要学生自行查阅资料学习。

3. 实验方法

本次实验模拟了 Paxos 算法达成一致的过程，设计了 5 个 Proposer 进行提案，11 个 Acceptor 对提案进行投票。Proposer 的初始提案被设定为[编号 i+1，提议 i+1]，即 Proposer0 号的初始提案为[编号 1，提议 1]，依此类推。无论 Propose 阶段还是 Accept 阶段，每个 Proposer 都需要取得半数以上，即至少 6 个 Acceptor 的同意，否则就要增大自己的提案编号重新发起拉票请求，在实验中提案编号每次按 Proposer 的数量增加，即每次增加 5。

3.1 Propose 阶段

3.1.1 Acceptor

Propose 阶段 Acceptor 通过比较编号决定投票还是拒绝：

① 当 m_maxSerialNum > serialNum，即当前提案的编号小于记录的最大编号时，拒绝该提案或者不回应。在代码中使用 return false 显式地通知给 Proposer 拒绝信息，也可以选择

不做回应，Proposer 通过超时推算出自己的提案没有通过，从而进如新一轮提案。

② 提案通过（m_maxSerialNum <= serialNum），会执行两个动作：

1) 更新记录的最大编号 m_maxSerialNum = serialNum，从而 Acceptor 作出承诺保证后面不会再投票给编号小于 serialNum 提案；

2) 如果存在同意过的其他提案，则推荐最后一次同意的提案内容 m_lastAcceptValue，从而使 Proposer 在 Accept 阶段使用 lastAcceptValue 达成一致。

```
bool Acceptor::Propose(unsigned int serialNum, PROPOSAL &lastAcceptValue)
{
    if ( 0 == serialNum ) return false;
    //提议不通过
    if ( m_maxSerialNum > serialNum ) return false;
    //接受提议
    //请完善下面逻辑

    /**********Begin**********/
    m_maxSerialNum = serialNum;
    lastAcceptValue = m_lastAcceptValue;
    /**********End**********/

    return true;
}
```

拒绝

同意

3.1.2 Proposer

Propose 阶段 Proposer 行为如下：

① 如果半数以上的 Acceptor 拒绝或者没有回应，则将编号增大（m_value.serialNum += m_proposerCount），进行新一轮提案；

② 如果收到投票，且存在返回的推荐提案，则在 Accept 阶段会改用推荐的提案内容；

③ 如果累计收到超过半数 Acceptor 的投票，则 Propose 阶段结束，开始 Accept 阶段。

```
bool Proposer::Proposed(bool ok, PROPOSAL &lastAcceptValue)
{
    if ( m_proposeFinished ) return true;//可能是一阶段迟到的回应，直接忽略消息

    if ( !ok )
    {
        m_refuseCount++;
        //已有半数拒绝，不需要等待其它acceptor投票了，重新开始Propose阶段
        //使用StartPropose(m_value)重置状态

        //请完善下面逻辑
        /**********Begin**********/
        if (m_refuseCount > m_acceptorCount / 2) {
            m_value.serialNum += m_proposerCount;
            StartPropose(m_value);
            return false;
        }
        /**********End**********/

        //拒绝数不到一半
        return true;
    }
```

半数以上Acceptor拒绝，重新提案

```
    m_okCount++;
    /*
        没有必要检查分支：serialNum为null
        因为serialNum>m_maxAcceptedSerialNum，与serialNum非0互为必要条件
    */
    //如果已经有提议被接受，修改成已被接受的提议
    //请完善下面逻辑
    /**********Begin**********/
    if (lastAcceptValue.serialNum > m_maxAcceptedSerialNum) {
        m_maxAcceptedSerialNum = lastAcceptValue.serialNum;
        m_value.value = lastAcceptValue.value;
    }
    /**********End**********/

    //如果自己的提议被接受
    if ( m_okCount > m_acceptorCount / 2 )
    {
        m_okCount = 0;
        m_proposeFinished = true;
    }
    return true;
}
```

修改为Acceptor推荐提案

半数以上同意，Propose阶段结束，准备进入Accept阶段

3.1.3 交互过程

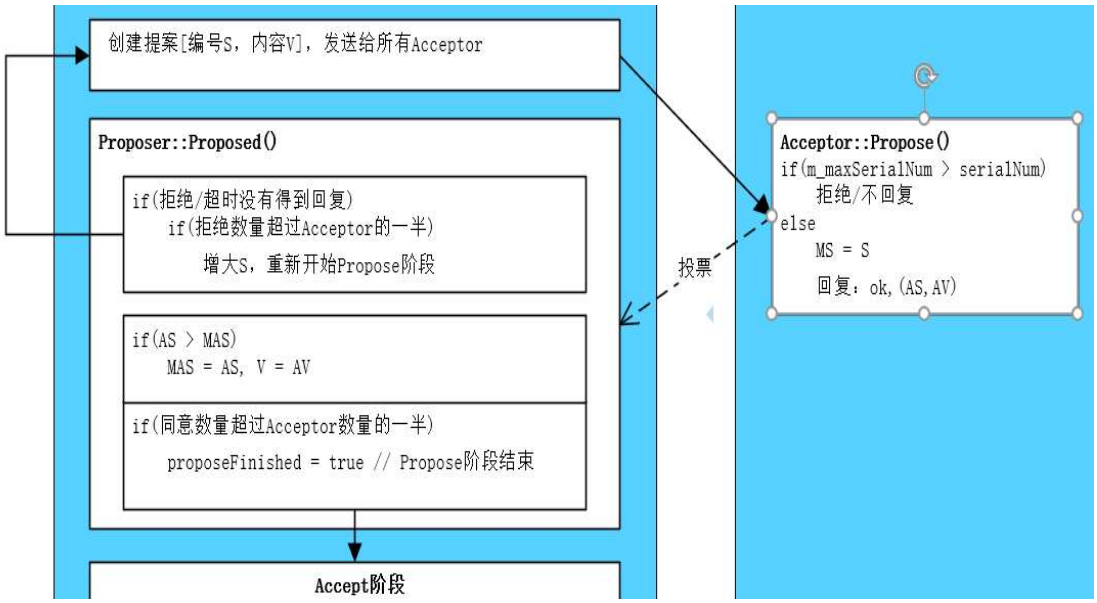
Propose 阶段即“拉票”阶段，主要包括三个动作，两次通信：

① Proposer 创建一个提案【编号 S，内容 V】并发送给所有 Acceptor；

② Acceptor 收到提案后，通过比较编号的大小决定是否投票；

③ Acceptor 将投票结果反馈给 Proposer，Proposer 根据投票结果选择重新提案还是进入 Accept 阶段。

Proposer 和 Acceptor 交互图示如下：



Propose 阶段交互过程代码如下：

```
while ( true )
{
    value = proposer.GetProposal();//拿到提议
    log.Info("Info", "Proposer%d号开始(Propose阶段):提议=[编号:%d, 提议:%d]\n",
        (long)id, value.serialNum, value.value);
    count = 0;
    int i = 0;
    for (i = 0; i < 11; i++ )
    {
    /*
        * 发送消息到第i个acceptor
        * 经过一定时间达到acceptor, sleep(随机时间)模拟
        * acceptor处理消息, mAcceptors[i].Propose()
        * 回应proposer
        * 经过一定时间proposer收到回应, sleep(随机时间)模拟
        * proposer处理回应mProposer.proposed(ok, lastValue)
    */
        mdk::m_sleep(rand()%500);//经过随机时间，消息到达了mAcceptors[i]
        //处理消息
        l[i].Lock();
        bool ok = a[i].Propose(value.serialNum, lastValue);
        l[i].Unlock();
        mdk::m_sleep(rand()%500);//经过随机时间,消息到达Proposer
        //处理Propose回应
        if ( !proposer.Proposed(ok, lastValue) ) //重新开始Propose阶段
        {
            mdk::m_sleep(1000);//为了降低活锁，多等一会让别的proposer有机会完成自己的2阶段批准
            break;
        }
        paxos::PROPOSAL curValue = proposer.GetProposal();//拿到提议
        if ( curValue.value != value.value )//acceptor本次回应可能推荐了一个提议
        {
            log.Info("Info", "Proposer%d号修改了提议:提议=[编号:%d, 提议:%d]\n",
                (long)id, curValue.serialNum, curValue.value);
            break;
        }
        acceptorId[count++] = i;//记录愿意投票的acceptor
        if ( proposer.StartAccept() ) break;
    }
    //检查有没有达到Accept开始条件，如果没有表示要重新开始Propose阶段
    if ( !proposer.StartAccept() ) continue;
```

① ② ③

交互代码中有三处关键点：

① Acceptor 调用 Propose()方法对提案进行投票；

② Proposer 调用 Proposed()方法根据投票结果决定是否重新提案；

③ 当有半数以上 Acceptor 同意时，进入 Accept 阶段，否则重新开始 Propose 阶段。

3.2 Accept 阶段

3.2.1 Acceptor

Accept 阶段 Acceptor 行为如下：

① 如果 m_maxSerialNum > value.serialNum，则说明 Acceptor 又同意了其他提案，对该次的 accept 请求回复拒绝或者不回复，Proposer 收到反馈会重新进入 Propose 阶段；

② 批准提案通过。

```
bool Acceptor::Accept(PROPOSAL &value)
{
    if ( 0 == value.serialNum ) return false;
    //Acceptor又重新答应了其他提议
   //请完善下面逻辑
    /**********Begin**********/
    if (m_maxSerialNum > value.serialNum) return false;
    /**********End**********/


    //批准提议通过
    //请完善下面逻辑
    /**********Begin**********/
    m_lastAcceptValue = value;
    /**********End**********/

    return true;
}
```

拒绝

批准提案通过

3.2.2 Proposer

Accept 阶段 Proposer 行为如下：

① 与 Propose 阶段相同，如果半数以上的 Acceptor 拒绝或者没有回应，则将编号增大（m_value.serialNum += m_proposerCount），进行新一轮提案；

② 如果累计收到超过半数 Acceptor 的批准，则 Accept 阶段结束，达成一致。

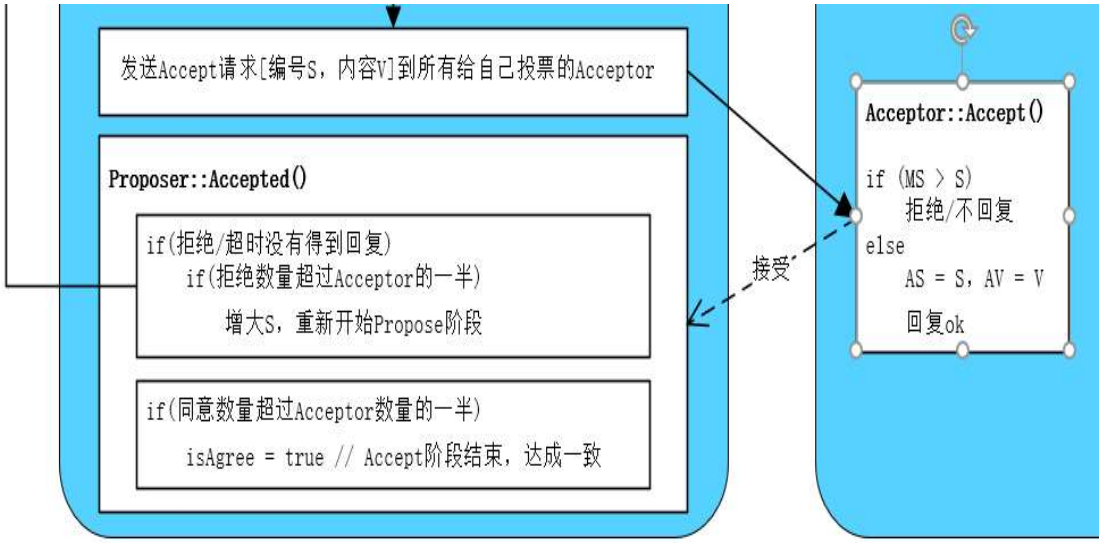
```
bool Proposer::Accepted(bool ok)
{
    if ( !m_proposeFinished ) return true;//可能是上次第二阶段迟到的回应，直接忽略消息

    if ( !ok )
    {
        m_refuseCount++;
        //已有半数拒绝，不需要等待其它acceptor投票了，重新开始Propose阶段
        //使用StartPropose(m_value)重置状态
        //请完善下面逻辑
        /**********Begin**********/
        if (m_refuseCount > m_acceptorCount / 2) {
            m_value.serialNum +=  m_proposerCount;
            StartPropose(m_value);
            return false;
        }
        /**********End**********/

        return true;
    }

    m_okCount++;
    if ( m_okCount > m_acceptorCount / 2 ) m_isAgree = true;

    return true;
}
```

半数以上拒绝，重新提案

半数以上批准，达成一致

3.2.3 交互过程

Accept 阶段 Proposer 和 Acceptor 交互图示如下：



Accept 阶段交互过程代码如下：

```
//开始Accept阶段
//发送Accept消息到所有愿意投票的acceptor
value = proposer.GetProposal();
log.Info("Info", "Proposer%d号开始(Accept阶段):提议=[编号:%d，提议:%d]\n",
    (long)id, value.serialNum, value.value);
for (i = 0; i < count; i++ )
{
    //发送accept消息到acceptor
    //减少accept阶段等待时间，加快收敛
    mdk::m_sleep(rand()%200);//经过随机时间,accept消息到达acceptor
    //处理accept消息
    l[acceptorId[i]].Lock();
    bool ok = a[acceptorId[i]].Accept(value); ①
    l[acceptorId[i]].Unlock();
    mdk::m_sleep(rand()%200);//经过随机时间,accept回应到达proposer
    //处理accept回应
    if ( !proposer.Accepted(ok) ) //重新开始Propose阶段 ②
    {
        mdk::m_sleep(1000);//为了降低活锁，多等一会让别的proposer有机会完成自己的2阶段批准
        break;
    }
    if ( proposer.IsAgree() )//成功批准了提议 ③
```

交互代码中有三处关键点：

① Acceptor 调用 Accept()方法处理 accept 请求，决定是否批准提案；

② Proposer 调用 Acceptd()方法根据 Acceptor 返回的结果决定是否重新提案；

③ 当有半数以上 Acceptor 批准提案时，Accept 阶段结束，达成一致。

4. 实验结果

4.1 结果截图

```
2020-01-09 14:20:32 Tid:28604 [Info] 5个Proposer, 11个Acceptor准备进行Paxos
每个Proposer独立线程，Acceptor不需要线程
Proposer编号从0-10,编号为i的Proposer初始提议编号和提议值是（i+1, i+1）
Proposer每次重新提议会将提议编号增加5
Proposer被批准后结束线程,其它线程继续投票最终，全部批准相同的值，达成一致。

2020-01-09 14:20:32 Tid:28604 [Info] Paxos开始

2020-01-09 14:20:37 Tid:28609 [Info] Proposer4号的提议被批准,用时4657MS:最终提议 = [编号:5，提议:5]

2020-01-09 14:20:49 Tid:28607 [Info] Proposer2号的提议被批准,用时16829MS:最终提议 = [编号:13，提议:5]

2020-01-09 14:20:54 Tid:28606 [Info] Proposer1号的提议被批准,用时21764MS:最终提议 = [编号:17，提议:5]

2020-01-09 14:20:58 Tid:28605 [Info] Proposer0号的提议被批准,用时26198MS:最终提议 = [编号:21，提议:5]

2020-01-09 14:21:04 Tid:28608 [Info] Proposer3号的提议被批准,用时32206MS:最终提议 = [编号:24，提议:5]

2020-01-09 14:21:04 Tid:28608 [Info] Paxos完成，用时32213MS，最终通过提议值为：5
```

图 4.1 Paxos 达成一致过程

Paxos > bin > log > Proposer2 > 2020-01-09.log

```
1   2020-01-09 14:20:32 Tid:28607 [Info] Proposer2号开始(Propose阶段):提议=[编号:3，提议:3]
2
3   2020-01-09 14:20:36 Tid:28607 [Info] Proposer2号开始(Accept阶段):提议=[编号:3，提议:3]
4
5   2020-01-09 14:20:38 Tid:28607 [Info] Proposer2号开始(Propose阶段):提议=[编号:8，提议:3]
6
7   2020-01-09 14:20:39 Tid:28607 [Info] Proposer2号修改了提议:提议=[编号:8，提议:5]
8
9   2020-01-09 14:20:39 Tid:28607 [Info] Proposer2号开始(Propose阶段):提议=[编号:8，提议:5]
10
11  2020-01-09 14:20:43 Tid:28607 [Info] Proposer2号开始(Accept阶段):提议=[编号:8，提议:5]
12
13  2020-01-09 14:20:45 Tid:28607 [Info] Proposer2号开始(Propose阶段):提议=[编号:13，提议:5]
14
15  2020-01-09 14:20:48 Tid:28607 [Info] Proposer2号开始(Accept阶段):提议=[编号:13，提议:5]
16
17  2020-01-09 14:20:49 Tid:28607 [Info] Proposer2号的提议被批准,用时16829MS:最终提议 = [编号:13，提议:5]
```

图 4.2 Proposer2 号提案-投票-修改-批准过程

4.2 结果分析

以 Proposer2 号为例，初始提案为[编号:3，提议:3]，得到了半数以上 Acceptor 投票进入 Accept 阶段，但在 Accept 阶段由于编号较小被拒绝，于是增大提案编号（3→8）重新提案；在第二次 Propose 阶段收到投票和推荐提案，则修改提案内容（3→5）重新提案……直到编号增大到 13 时，提案得到半数以上 Acceptor 批准。

同理，结合图 4.1 可知，各个 Proposer 编号增长过程如下：

- Propose0 号：1→6→11→16→21
- Propose1 号：2→7→12→17
- Propose3 号：4→9→19→24

说明当提案编号增长到大于上一次批准的提案编号时，提案才能被批准。

5. 总结与收获

Paxos 是一种基于消息传递的一致性算法，这个算法被认为是类似算法中最有效的。Paxos 算法目前在 Google 的 Chubby、MegaStore、Spanner 等系统中得到了应用，Hadoop 中的 ZooKeeper 也使用了 Paxos 算法的思想。

通过实验理解和掌握了 Paxos 的整个流程，总结出 Paxos 达成一致的过程图示如下：

Paxos过程