第六次作业

学号：1300013022

姓名：武守北

一、阅读Paxos算法的材料并用自己的话简单叙述：

Paxos算法是一种基于消息传递通信模型的分布式系统，使得各节点就某个值达成一致的问题的算法，其既可以工作在单机的多个进程上面，也可以工作在网络上面的多个主机上面。Paxos协议假定各个节点之间的通信采用异步的方式，且基于非拜占庭模型，也就是允许消息的延迟、丢失或者重复，但是不会出现内容损坏、篡改的情况，在实践中通过添加额外的校验信息很容易保证收到的消息是完整的。

在paxos算法中，分为4种角色：

Proposer ：提议者

Acceptor：决策者

Client：产生议题者

Learner：最终决策学习者

Proposer就像Client的使者，由Proposer使者拿着Client的议题去向Acceptor提议，让Acceptor来决策。

在paxos算法中所有的行为为：

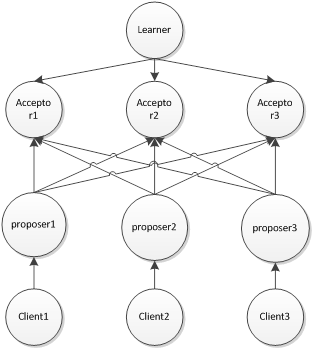
1、Proposer提出议题；

2、Acceptor初步接受或者Acceptor初步不接受；

3、如果上一步Acceptor初步接受则Proposer再次向Acceptor确认是否最终接受；

4、Acceptor 最终接受或者Acceptor 最终不接受；

算法流程图如下：



现在以一个简单的例子来解释一下Paxos算法的流程（两阶段提交）：

有2个Client（老板、老板之间是竞争关系）和3个Acceptor（政府官员）：

现在需要对一项议题来进行paxos过程，议题是“A项目我要中标！”，这里的“我”指每个带着他的秘书Proposer的Client老板，故事中的比特币是编号，议题是value。

一、第一阶段：

Proposer听老板的话，带着议题和现金去找Acceptor政府官员。

作为政府官员，当然想谁给的钱多就把项目给谁。

Proposer-1小姐带着现金同时找到了Acceptor-1~Acceptor-3官员，1与2号官员分别收取了10比特币，找到第3号官员时，没想到遭到了3号官员的鄙视，3号官员告诉她，Proposer-2给了11比特币。不过没关系，Proposer-1已经得到了1,2两个官员的认可，形成了多数派（如果没有形成多数派，Proposer-1会去银行提款在来找官员们给每人20比特币，这个过程一直重复每次+10比特币，直到多数派的形成），满意的找老板复命去了，但是此时Proposer-2保镖找到了1,2号官员，分别给了他们11比特币，1,2号官员的态度立刻转变，都说Proposer-2的老板懂事，这下子Proposer-2放心了，搞定了3个官员，找老板复命去了，当然这个过程是第一阶段提交，只是官员们初步接受贿赂而已。

这个过程保证了在某一时刻，某一个proposer的议题会形成一个多数派进行初步支持。

===============第一阶段结束================

二、第二阶段：

现在进入第二阶段提交，现在proposer-1小姐使用分身术(多线程并发)分了3个自己分别去找3位官员，最先找到了1号官员签合同，遭到了1号官员的鄙视，1号官员告诉他proposer-2先生给了他11比特币，因为上一条规则的性质proposer-1小姐知道proposer-2第一阶段在她之后又形成了多数派(至少有2位官员的赃款被更新了);此时她赶紧去提款准备重新贿赂这3个官员(重新进入第一阶段)，每人20比特币。刚给1号官员20比特币， 1号官员很高兴初步接受了议题。在他还没来得及见到2,3号官员的时候，proposer-2先生也使用分身术分别找3位官员(注意这里是proposer-2的第二阶段)，被第1号官员拒绝了告诉他收到了20比特币，第2,3号官员顺利签了合同，这时2，3号官员记录client-2老板用了11比特币中标，因为形成了多数派，所以最终接受了Client2老板中标这个议题，对于proposer-2先生已经出色的完成了工作；

这时proposer-1小姐找到了2号官员，官员告诉她合同已经签了，将合同给她看，proposer-1小姐是一个没有什么职业操守的聪明人，觉得跟Client1老板混没什么前途，所以将自己的议题修改为“Client2老板中标”，并且给了2号官员20比特币，这样形成了一个多数派。顺利的再次进入第二阶段。由于此时没有人竞争了，顺利的找3位官员签合同，3位官员看到议题与上次一次的合同是一致的，所以最终接受了，形成了多数派，proposer-1小姐跳槽到Client2老板的公司去了。

Paxos过程结束了，这样，一致性得到了保证，算法运行到最后所有的proposer都投“client2中标”所有的acceptor都接受这个议题，也就是说在最初的第二阶段，议题是先入为主的，谁先占了先机，后面的proposer在第一阶段就会学习到这个议题而修改自己本身的议题，因为这样没职业操守，才能让一致性得到保证，这就是paxos算法的一个过程。

二、模拟Raft协议工作的一个场景并叙述处理过程：

场景：Leader选举：

Raft算法是Paxos的一个替代品。 场景：Leader Election

1、一开始所有的节点都在等待监听Leader；

|  |  |  |  |
| --- | --- | --- | --- |
| NODE | A | B | C |
| TERM | 0 | 0 | 0 |

2、第一个计时结束的节点，C，要求别的节点给它投票；

|  |  |  |  |
| --- | --- | --- | --- |
| NODE | A | B | C |
| TERM | 0 | 0 | 1 |
| VOTE COUNT |  |  | 1 |

3、其它节点给C投票；

|  |  |  |  |
| --- | --- | --- | --- |
| NODE | A | B | C |
| TERM | 1 | 1 | 1 |
| VOTE | FOR C | FOR C | COUNT: 1 |

4、C获得了大多数选票，成为leader，定期与其它节点联络62；

|  |  |  |  |
| --- | --- | --- | --- |
| NODE | A | B | C |
| TERM | 1 | 1 | 1 |
| LEADER | C | C |  |

5、其它节点接收到联络时，刷新自己的计时器，并向C发送心跳信息；

|  |  |  |  |
| --- | --- | --- | --- |
| NODE | A | B | C |
| TERM | 1 | 1 | 1 |
| LEADER | C | C |  |

6、如果C突然宕机，其它节点会失去联络，定时器不会被重置

|  |  |  |
| --- | --- | --- |
| NODE | A | B |
| TERM | 1 | 1 |
| VOTE COUNT | C | C |
| NODE | A | B |
| TERM | 1 | 1 |
| VOTE COUNT | C | C |

7、B的定时器先到，成为candidate，要求其它节点为他投票

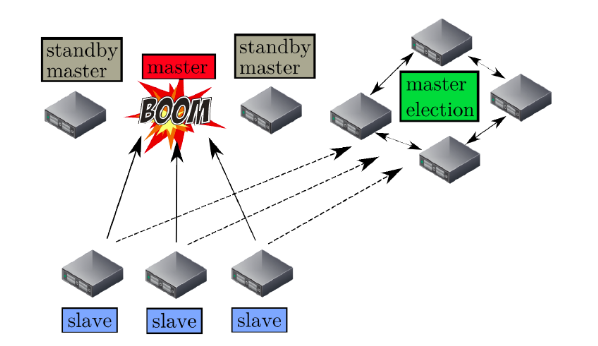
|  |  |  |
| --- | --- | --- |
| NODE | A | B |
| TERM | 1 | 2 |
|  | LEADER C | VOTR COUNT: 1 |

8、B成功成为了leader，并向其它节点发送联络

|  |  |  |
| --- | --- | --- |
| NODE | A | B |
| TERM | 2 | 2 |
| LEADER | B |  |

9、如果同时有多个节点成为candidate，那么可能不会选举出一个leader，因为没有一个节点获得大多数选票。直到有一个节点成为leader，心跳联络重新开始

三、简述Mesos的容错机制并验证：



mesos的容错机制体现在四个方面：

master出错、slave出错、executor出错、framework崩溃

1、master出错 ：Mesos使用热备份（hot-standby）设计来实现Master节点集合。一个Master节点与多个备用（standby）节点运行在同一集群中，并由开源软件Zookeeper来监控。Zookeeper会监控Master集群中所有的节点，并在Master节点发生故障时管理新Master的选举。Mesos的状态信息实际上驻留在Framework调度器和Slave节点集合之中。当一个新的Master当选后，Zookeeper会通知Framework和选举后的Slave节点集合，以便使其在新的Master上注册。新的Master可以根据Framework和Slave节点集合发送过来的信息，重建内部状态。

2、slave Mesos实现了Slave的恢复功能，当Slave和master失去连接时，可以让执行器/任务继续运行。当任务执行时，Slave会将任务的监测点元数据存入本地磁盘。当Master重新连接Slave，启动slaver进程后，因为此时没有可以响应的消息，所以重新启动的Slave进程会使用检查点数据来恢复状态，并重新与执行器/任务连接。当slave多次无响应，重连接失败，master会删除这个slave节点。

3、executor出错 当Slave节点上的进程失败时，mesos会通知framework，让framework决定下一步的处理。

4、framework崩溃 Framework调度器的容错是通过Framework将调度器注册2份或者更多份到Master来实现。当一个调度器发生故障时，Master会通知另一个调度来接管。这个需要调度器自己实现。

验证：

1、每台电脑下载一个zookeeper：

wget http://www-eu.apache.org/dist/zookeeper/zookeeper-3.4.9/zookeeper-3.4.9.tar.gz

tar -xvf zookeeper-3.4.9.tar.gz

mv zookeeper-3.4.9.tar.gz zookeepe

2、分别配置zookeeper、创建工作目录并启动：

cd zookeeper

cp conf/zoo\_sample.cfg conf/zoo.cfg

vim conf/zoo.cfg

dataDir=/var/lib/zookeeper

server.1=172.16.6.103:2888:3888

server.2=172.16.6.225:2888:3888

server.3=172.16.6.95:2888:3888

mkdir /var/lib/zookeeper

echo "1" > /var/lib/zookeeper/myid

mkdir /var/lib/zookeeper

echo "2" > /var/lib/zookeeper/myid

mkdir /var/lib/zookeeper

echo "3" > /var/lib/zookeeper/myid

bin/zkServer.sh start

ZooKeeper JMX enabled by default

Using config: /home/pkusei/zookeeper/bin/../conf/zoo.cfg

Starting zookeeper ... STARTED

3、两台是follower，一台是leader：

bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /home/pkusei/zookeeper/bin/../conf/zoo.cfg

Mode: follower

bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /home/pkusei/zookeeper/bin/../conf/zoo.cfg

Mode: leader

bin/zkServer.sh status

ZooKeeper JMX enabled by default

Using config: /home/pkusei/zookeeper/bin/../conf/zoo.cfg

Mode: follower

4、分别启动mesos：

mesos master --zk=zk://172.16.6.249:2181,172.16.6.153:2181,172.16.6.2:2181/mesos --quorum=2 --ip=172.16.6.2 --hostname=mas1 --work\_dir=/var/lib/mesos --log\_dir=/var/log/mesos

mesos master --zk=zk://172.16.6.249:2181,172.16.6.153:2181,172.16.6.2:2181/mesos --quorum=2 --ip=172.16.6.153 --hostname=mas2 --work\_dir=/var/lib/mesos --log\_dir=/var/log/mesos

mesos master --zk=zk://172.16.6.249:2181,172.16.6.153:2181,172.16.6.2:2181/mesos --quorum=2 --ip=172.16.6.249 --hostname=mas3 --work\_dir=/var/lib/mesos --log\_dir=/var/log/mesos

5、Kill掉Leader，log一下：

I0529 04:20:10.016351 60160 network.hpp:432] ZooKeeper group memberships changed

I0529 04:20:10.021173 60160 network.hpp:480] ZooKeeper group PIDs: { }

I0529 04:20:11.354900 60159 detector.cpp:152] Detected a new leader: (id='2')

I0529 04:20:11.355798 60159 group.cpp:697] Trying to get '/mesos/json.info\_0000000002' in ZooKeeper

I0529 04:20:11.358307 60159 zookeeper.cpp:259] A new leading master (UPID=master@172.16.6.153:5050) is detected

I0529 04:20:11.358961 60159 master.cpp:2017] Elected as the leading master!

I0529 04:20:11.359378 60159 master.cpp:1560] Recovering from registrar

I0529 04:20:11.362989 60163 log.cpp:553] Attempting to start the writer

可以看到，zookeeper从备选的master中重新选举了一个leader。

验证完毕。

四、综合作业：

编写一个mesos framework，使用calico容器网络自动搭建一个docker容器集群（docker容器数量不少于三个），并组成etcd集群，在etcd选举出的master节点上部署jupyter notebook，使得可以从外部访问该集群。同时满足以下条件：

这些docker容器共用一套分布式存储

这些docker容器可以互相免密码ssh登录，且在host表中的名字从XXX-0一直到XXX-n（XXX是自己取的名字，n是容器数量-1），其中XXX-0是etcd的master节点

当其中一个容器被杀死时，集群仍满足上一个条件

当etcd master节点被杀死时，jupyter notebook会转移到新的master节点提供服务，集群仍满足上一个条件。