

Assignment3. Raft Log Consensus

MF1733071, 严德美, 1312480794@qq.com

2017 年 12 月 18 日

1 简述分析与设计

本次任务是实现一个Raft [1]算法中的一个Raft Log replication。在Assignment2的基础上, 已经能够成功进行选举, 需要添加一些关于log的数据, 比如LogEntry, log []LogEntry.

1.1 日志复制过程中所有的server必须满足一些规则, 以下是一些规则:

1.1.1 接受者需要实现(即RequestVote函数):

- (1). 如果 $term < currentTerm$ 返回 false
- (2). 如果在prevLogIndex处的日志的任期号与prevLogTerm不匹配时, 返回 false
- (3). 如果一条已经存在的日志与新的冲突 (index 相同但是任期号 term 不同), 则删除已经存在的日志和它之后所有的日志
- (4). 添加任何在已有的日志中不存在的条目
- (5). 如果leaderCommit > commitIndex, 将commitIndex设置为leaderCommit和最新日志条目索引号中较小的一个
- (6). 如果term < currentTerm返回 false
- (7). 如果votedFor为空或者与candidateId相同, 并且候选人的日志和自己的日志一样新, 则给该候选人投票

1.1.2 所有 server:

- (1). 如果commitIndex > lastApplied, lastApplied自增, 将log[lastApplied]应用到状态机.
- (2). 如果 RPC 的请求或者响应中包含一个 term T 大于 currentTerm, 则currentTerm赋值为 T, 并切换状态为Follower.

1.1.3 Followers:

- (1). 响应来自候选人和领导人的 RPC
- (2). 如果在选举超时之前没有收到来自当前领导人的AppendEntries RPC或者没有收到候选人的投票请求, 则自己转换状态为Candidate

1.1.4 Candidates:

- (1). 转变为选举人之后开始选举:
 - (a). `currentTerm`自增
 - (b). 给自己投票
 - (c). 重置选举计时器
 - (d). 向其他服务器发送RequestVote RPC请求投票
- (2). 选举限制:
 - (a). 想要获得大多数选票，必须包含有全部的已经提交的日志条目
- (3). 如果收到了来自大多数服务器的投票（大于 $\text{len}(\text{peers})/2$ ）：成为Leader
- (4). 如果收到了来自新领导人的AppendEntries RPC（heartbeat）：转换状态为Follower
- (5). 如果选举超时：开始新一轮的选举

1.1.5 Leader:

- (1). 一旦成为Leader：向其他所有服务器发送空的AppendEntries RPC（heartbeat）；在空闲时间重复发送以防止选举超时
- (2). 如果收到来自客户端的请求：向本地日志增加条目，在该条目应用到状态机后响应客户端
- (3). 对于一个追随者来说，如果上一次收到的日志索引大于将要收到的日志索引（`nextIndex`）：通过AppendEntries RPC将 `nextIndex` 之后的所有日志条目发送出去
 - (a). 如果发送成功：将该追随者的 `nextIndex`和`matchIndex`更新
 - (b). 如果由于日志不一致导致AppendEntries RPC失败：`nextIndex`递减并且重新发送
- (4). 如果存在一个满足 $N > \text{commitIndex}$ 和 $\text{matchIndex}[i] \geq N$ 并且 $\text{log}[N].\text{term} == \text{currentTerm}$ 的 N ，则将`commitIndex`赋值为 N

1.1.6 Raft一致性问题

- (1). 领导人选取（Leader election）：在一个领导人宕机之后必须要选取一个新的领导人
- (2). 日志复制（Log replication）：领导人必须从客户端接收日志然后复制到集群中的其他服务器，并且强制要求其他服务器的日志保持和自己相同
- (3). 安全性（Safety）：如果一个服务器已经将给定索引位置的日志条目应用到状态机中，则所有其他服务器不会在该索引位置应用不同的条目

1.2 结合 Assignment2,对一些数据结构进行说明:

名称	描述
currentTerm	服务器最后知道的任期号（从0开始递增）
votedFor	在当前任期内收到选票的候选人 id（如果没有就为 -1）
state	当前状态
log[]	LogEntry数组；每个元素包含状态机的要执行命令和LogEntry Term，和log的索引
voteAcquired	每个candidate获得的票数
commitIndex	已知的被提交的最大日志条目的索引值（从0开始递增）
lastApplied	被状态机执行的最大日志条目的索引值（从0开始递增）
nextIndex[]	记录发给每一个server下一个日志条目的索引（初始化为领导人最新日志的索引值+1）
matchIndex[]	记录每个server已经复制到该服务器的日志的最高索引值（从0开始递增）

表 1: Raft struct

名称	描述
Index	LogEntry在log中的索引
Term	LogEntry的Term
Command	LogEntry里面包含的命令

表 2: LogEntry struct

参数	描述
Term	候选人的任期号
CandidateId	请求投票的候选人 id
LastLogIndex	候选人最新日志条目的索引值
LastLogTerm	候选人最新日志条目对应的任期号

表 3: RequestVote RPC arguments structure

返回值	描述
Term	回复者目前的任期号，用于候选人更新自己
IsVote	如果候选人收到选票为 true

表 4: RequestVote RPC reply structure

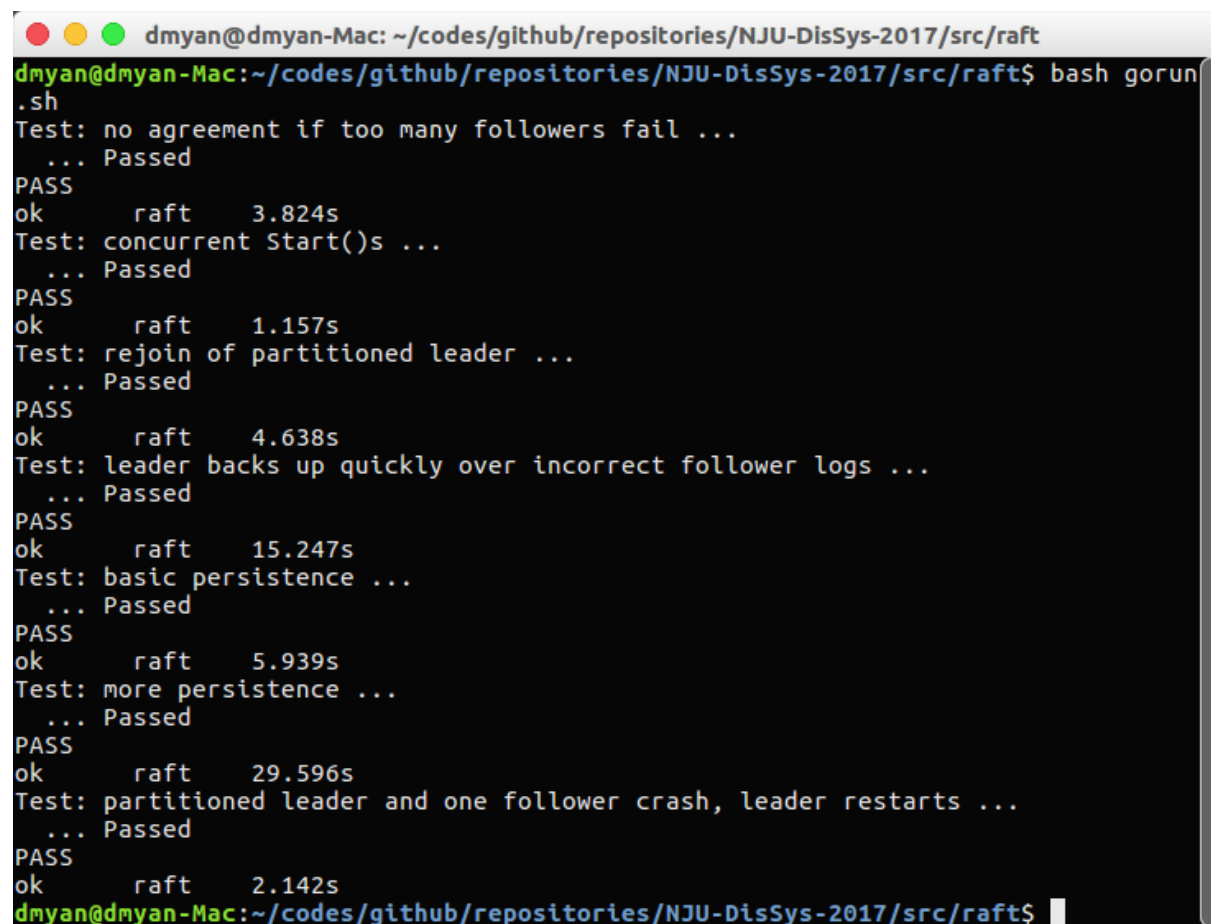
参数	描述
Term	领导人的任期号
LeaderId	领导人的 id，为了其他服务器能重定向到客户端
PrevLogIndex	最新日志之前的日志的索引值
PrevLogTerm	最新日志之前的日志的领导人任期号
Entries[]	将要存储的日志条目（表示 heartbeat 时空，有时会为了效率发送超过一条）
LeaderCommit	领导人提交的日志条目索引值

表 5: AppendEntries RPC arguments structure

返回值	描述
Term	返回的任期号，用于领导人更新自己的任期号
Success	如果其它服务器包含能够匹配上 prevLogIndex 和 prevLogTerm 的日志时为真
NextIndex	如果匹配失败，回复者期望下一次匹配的index

表 6: AppendEntries RPC reply structure

2 实现演示

A terminal window on a Mac showing the execution of a Raft consensus implementation. The user runs a script named 'gorun.sh'. The script performs several tests, each with a 'PASS' result. The tests include: 'no agreement if too many followers fail ...', 'concurrent Start()s ...', 'rejoin of partitioned leader ...', 'leader backs up quickly over incorrect follower logs ...', 'basic persistence ...', 'more persistence ...', and 'partitioned leader and one follower crash, leader restarts ...'. Each test is preceded by 'ok raft' and a duration in seconds. The terminal output is as follows:

```
dmyan@dmyan-Mac: ~/codes/github/repositories/NJU-DisSys-2017/src/raft
dmyan@dmyan-Mac:~/codes/github/repositories/NJU-DisSys-2017/src/raft$ bash gorun.sh
Test: no agreement if too many followers fail ...
... Passed
PASS
ok      raft      3.824s
Test: concurrent Start()s ...
... Passed
PASS
ok      raft      1.157s
Test: rejoin of partitioned leader ...
... Passed
PASS
ok      raft      4.638s
Test: leader backs up quickly over incorrect follower logs ...
... Passed
PASS
ok      raft      15.247s
Test: basic persistence ...
... Passed
PASS
ok      raft      5.939s
Test: more persistence ...
... Passed
PASS
ok      raft      29.596s
Test: partitioned leader and one follower crash, leader restarts ...
... Passed
PASS
ok      raft      2.142s
dmyan@dmyan-Mac:~/codes/github/repositories/NJU-DisSys-2017/src/raft$
```

图 1: Raft Log Consensus

3 总结

通过本次实验，对分布式系统一致性的问题有了更多的理解，实现中才发现有很多细节需要考虑和处理，实现Log replication加深了对论文中作者描述的算法理解，也对go语言更加熟悉了。

参考文献

- [1] Diego Ongaro and John K. Ousterhout. In search of an understandable consensus algorithm. In *USENIX Annual Technical Conference*, 2014.