分布式系统第二次作业

MG1833019 冯哲

1 实验内容

这次实验的内容是实现 raft 的日志复制算法。在 raft 算法中，每个 server 都会存储一份日志，日志存储的是一系列命令，而 server 的状态机会按顺序执行这些日志中的命令。每份日志中以同样的顺序存储了同样的命令，而状态机以同样的顺序执行这些相同的命令。每台 server 的状态机都是确定的，它们以同样的顺序执行同样的命令，最终的状态和输出也必然是一样的。

在 raft 的设计中，日志只能从 leader 复制到其他节点。日志项包括 index，term，command 三个元素。其中 index 为日志索引，term 为任期，而 command 为具体的日志内容。通常的流程为：client 先发送请求给 leader，leader 接收请求后，把命令存进自己的 log 中，然后通过 AppendEntries RPC 向其他 server 同步自己的 log。此时日志尚处于未提交状态，只要过半数的 follower 都向 leader 报告已经复制了 log，那么 leader 就把该日志的状态变为已提交，然后报告 client 写入成功。对于已提交的日志，每个 server 会将其按顺序在自己的状态机上执行。这样，raft 就完成 log 的复制，并且保证了一致性。

在具体的系统中，我们首先需要实现 log 从 leader 到其他 follower 的同步机制，然后需要实现 persist，从而让系统具有对错误容忍的能力，最后还需要实现 AppendEntries RPC 因为不一致性失败的优化。

1.1 log 的复制

每当 leader 接收 client 的请求，就会增加新的 log，现在我们需要把 log 复制到其他 server 上。在 raft 中，这一过程是通过 AppendEntries RPC 实现的。在具体实现中，我们没有必要每次都额外判断 leader 是否已经增加了新 log，可以在发送 heartbeat 信号的同时，如果 leader 的日志比 server 要新，就直接把更新的日志发送过去，这样不仅可以提高效率，而且在实现上也更简洁明了。

这里要注意的是，因为网络连接中断等原因可能会导致不同 server 的日志出现不一致的情况，根据协议的设计，AppendEntries RPC 可能会因为日志的不一致性

而失败，这时 leader 需要减少 nextIndex 值并重试，直到两个 server 间日志满足一致性的要求，然后才能进行日志的覆盖写入。值得一提的是，为确保一致性，leader 不能直接提交之前 term 的日志，而是只能通过提交当前 term 的日志，然后之前 term 的日志因为 index 比提交的日志小从而一并被提交，这一点在论文的 Figure 8 中有具体的说明。我耗费了很长时间在这里的 debug 上，直到我注意到论文明确说明 commitIndex 的更新要求。

1.2 persist 的实现

相比之前，persist 的实现非常简单。根据要求，只要在每次修改 server 的 persist 状态后都用 persist 方法保存下来就可以了。这相当于我们从磁盘上对状态进行了同步，并且在 server 崩溃后可以用磁盘存储的信息恢复原先的状态。

1.3 AppendEntries RPC 的优化

最后，为了通过所有测试点，我们需要实现一个 AppendEntries RPC 的优化。在最原始的实现中，因为 follower 和 leader 因为日志不一致性而导致 AppendEntries RPC 失败时，leader 的 nextIndex 会减 1，然后重试。显然这样做的效率很低，尤其在 leader 和 follower 的日志差别很大的时候（比如某个 server 断开网络，过了很长时间后重连），这会使得调用 RPC 的次数非常多。

在论文中，作者已经提到了一个优化，就是当 AppendEntries RPC 因为不一致性失败，可以返回产生冲突的 entry 对应的 term 和该 term 的第一个 index，有了这个信息，leader 就可以直接跳过这个 term 所有的 entry，尝试下一个可能的 nextIndex 来复制日志。实现了这个优化，就可以通过所有的测试点了。

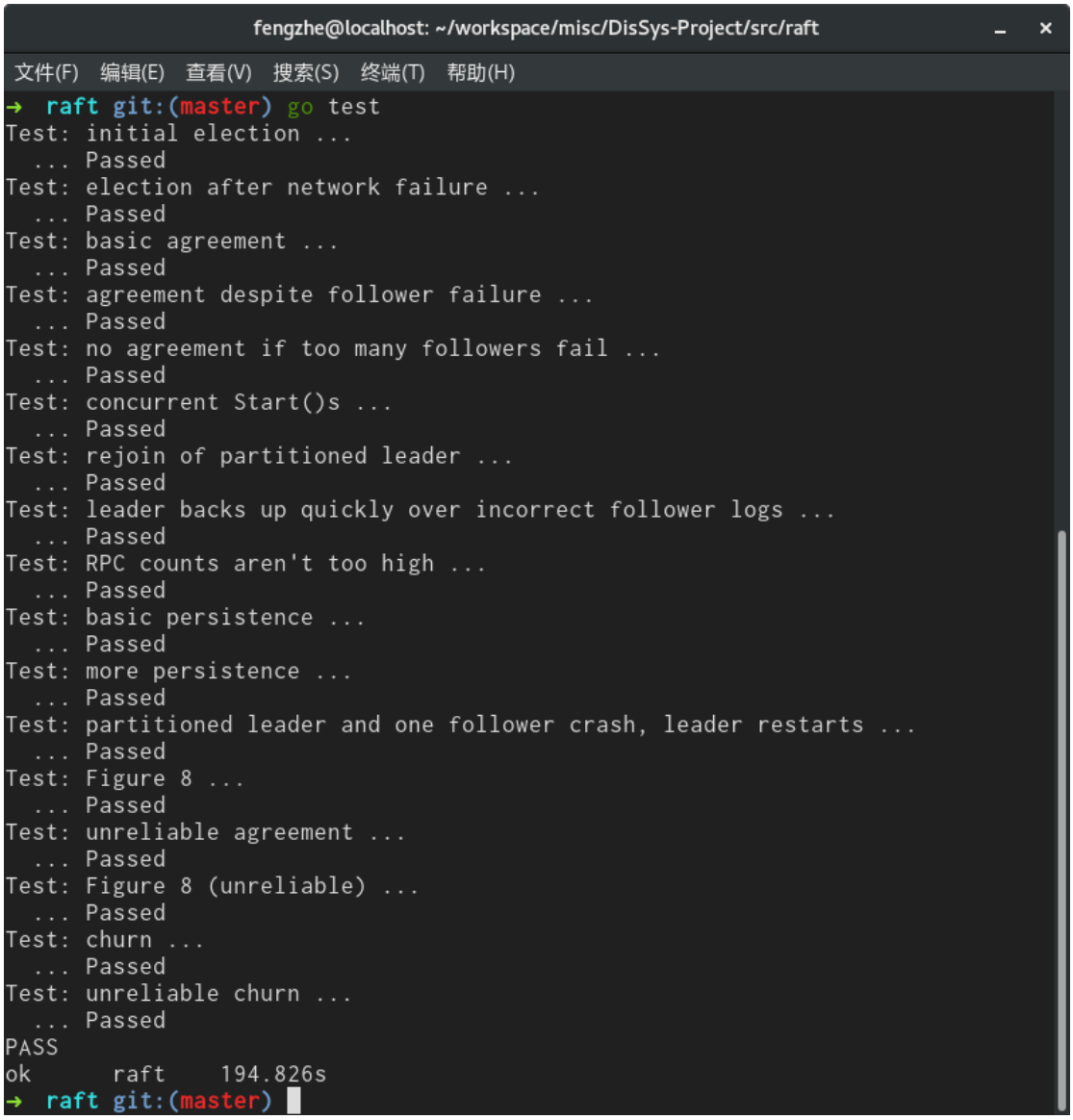
2 实验环境

操作系统：Fedora 28

编程语言：Golang 1.10

3 实验结果

我完成了选举机制的实现，并且通过了 raft 的所有测试点。



4 总结

通过这次实验，我深入了解了 Raft 算法的原理，不仅有总体框架上的理解，还有许多细节上的精妙设计，这对我学习分布式系统这门课程有很大的帮助。这次实验很有价值，为我以后对分布式系统方面的进一步学习打下了基础。