项目申请书

Apache APISIX (incubating) - 支持 etcd v3 协议

个人信息

基础信息 **姓名：吴舒旸**
邮箱： wosoyoung@gmail.com
GitHub： Yiyiyimu
个人网站： http://www.shuyangwu.com

教育经历 **上海交通大学**
环境科学与工程，学士，预计 2020 年 7 月毕业
佐治亚理工学院
计算科学与工程，硕士，预计 2020 年 8 月入学

相关经历
- 两年 Python 和 C++ 开发经验
- 刚学习 Lua 并读完了《OpenResty最佳实践》

项目信息

项目名称
支持 etcd v3 协议

项目描述
开源软件供应链点亮计划2020，APISIX任务列表

背景介绍
Apache APISIX 是一个云原生、高性能、可扩展的微服务 API 网关。它基于 Nginx 和 etcd 实现，具备动态路由、插件热加载和 gRPC 代理和协议转换功能，特别适合微服务体系下的 API 管理。如今包括NASA、中国航信、360 等众多公司和组织把 APISIX 部署在生产环境中。

Etcd 是一个强一致的分布式KV存储，提供了可以为分布式系统访问的可靠方法存储数据。通过 raft 一致性算法，它巧妙地在网络分区期间解决了 leader 选举问题，这种方法可以有效处理 leader 节点在内的机器故障。因此 etcd 被用作 kubernetes 所有集群数据的后备存储并被部署在各种环境中。

对于APISIX，etcd被用作储存 http 请求数据以用来实现云原生微服务，目前支持 etcd v2 版本。考虑到 etcd v3 版本实现了速度更快的服务并且为用户提供了更清晰的抽象层，加上社区对于 v3 的需求，APISIX 有必要支持 etcd v3 版本。因为 etcd 官方的客户端只提供了 golang

的版本，因此社区写了基于 lua 和 OpenResty 的客户端 —— lua-resty-etcd。因为 lua-resty-etcd 已经提供了 APISIX 所需的基本 API，所以本项目将会基于这个客户端工作。

详细方案

本项目的主要目标为基于 APISIX 目前对于 etcd v2 的支持，进一步支持 etcd v3 功能。在 APISIX 中有两个主要控制与 etcd 连接的文件：apisix/core/etcd.lua 和 apisix/core/config_etcd.lua，本项目的主要工作也是对这两个文件进行升级。APISIX 现在基于的 etcd 客户端，lua-resy-etcd 目前支持 etcd v2 和 v3 的主要 API，因此本项目可以基于客户端中两个版本的区别，对 APISIX 中的 etcd 进行相应的改动升级。

尽管 etcd v2 和 v3 两个版本在架构上有了很大变化，但主要的 API 并没有进行大幅度的改动。而在 lua-resty-etcd 客户端中，API 改动被进一步减少，包括 new/get/set/delete 等在内的 API 除了输入输出格式以外几乎没有改动。因此在项目中，有两个主要功能需要实现：etcd v3 新提供的函数 txn，和 lua-resty-etcd v3 中新支持的函数 watch。

1. **Txn** 提供了包裹多个微事务的接口。etcd v2 中支持原子化的比较并交换（compare and swap, CAS）功能以支持分布式锁，而 v3 中的 txn 则支持同时读多个请求并将其作为一个原子化事务进行处理。也就是 txn 可以同时进行多个 CAP 操作，以达到更高层次的并发控制并取得更好的性能。
2. **Watch** 提供了一个基于事件用于异步监视键的更改的接口。 它通过不断地监视给定版本，以达到等待密钥更改并将密钥更新传回客户端的需求。 在 etcd v3 中，watch API 可以监听一个范围内的所有键值，而不仅是 v2 中只能监听单个键（及以它为前缀的键）。 同样在 etcd v2 中，EventHistory 的最大长度为1000，因此较早的更改可能会丢失， 而 v3 支持不受限制地查看所有可能的版本。 为实现此目的，watch 的输入/输出发生了较大变化，因此在 watch API 升级方面需要进行较多工作。

在以上功能得到实现之后，需要对 v3 的新功能进一步添加单元测试，以满足 CI 的要求。 在 APISIX 中，测试基于 test::nginx 构建，后者是基于 OpenResty 的库。

另外由于当前lua-resty-etcd 不支持某些 v3 新添加的 API，在时间允许的情况下，项目计划把支持这些 API 作为可选任务，以便将来在 APISIX 中进一步优化 etcd 的使用。

1. **Compaction** 提供了压缩历史修改版本的功能。前面提到 v3 中默认 EventHistory 不限长度，为了避免历史版本过多降低性能，通过此函数可以对历史版本进行压缩也即是删除，在压缩之后选定压缩版本之前的历史修改记录就无法通过 get/watch 所获取。参考 etcd API 可能的输入格式：
   `res, err = cli:compaction(revision:int64, physical:bool)`
   `revision` - 修改版本号；`physical` - 是否从后端数据库完全删除
2. **Lease** 提供了为键设置超时时间的功能。当一个键被设定 lease 后，它的生命周期就由存活时间（TTL）所控制。当一个 lease 的 TTL 到期后，所有与之绑定的键就会被一并删除。另外 v3 还提供了手动撤销 lease 的选项，撤销时也会一并删除绑定的键，以及通过不断更新 lease 的 TTL 以保持 lease 存活防止到期的功能。在目前的 lua-resty-etcd v3 中，在 set 的参数中提供了绑定 lease 的选项，但还没有移植创建 lease 的函数，因此需要补充。参考 etcd API 可能的输入格式：
   `res, err = cli:lease(TTL:int64, ID:int64)`
   `TTL` - 存活时间；`ID` - 手动设置或自动分配的编号

开发时间计划

时间	工作
7.1之前	• 通过查阅收集并尝试解决 APISIX 的 github 中与 etcd 相关的 issue，更好的熟悉社区和环境，并且整理可能的可选任务
7.1 - 7.21	• 与社区讨论确定这个项目必需和可选需要实现的功能 • 基于 APISIX 中已经支持 etcd v2 的接口，构建支持 v3 的接口
7.22 - 8.7	• 基于 Test::Nginx 构建单元测试以支持 CI • 完成相关文档撰写
8.8 - 8.14	• 撰写中期报告，准备中期评审
8.15	• **中期评审**
8.15 - 8.31	• 继续构建 APISIX 中支持 etcd v3 的单元测试 • 在 lua-resty-etcd 中实现 etcdctl v3 中其他API • 尝试解决其他 APISIX 和 lua-resty-etcd 与 etcd 相关的 issue
9.1 - 9.30	• 继续修 issue • 撰写项目报告 • 缓冲时间