Ceph monitor分析

## monitor简介

monitor作为ceph集群核心服务，负责自维护ceph集群健康状态，rados集群元数据mdsmap、monmap、osdmap、log、auth、health等，基于改进paxos算法，保证集群节点间的数据状态在同一时刻一致性。总的来说Monitor负责收集集群信息，更新集群信息，发布集群信息。



图 1 monito架构图

可以看出monitor的内部其实是一个分布式kv数据库。从下往上分别是MonitorDBStore、Paxos和PaxosService。PaxosService负责保证每次都只会有一个提案进入paxos流程。Paxos模块具体实现了multi-Paxos算法。MonitorDBStore是对底层DB的抽象封装，将DB的基本操作事务封装成统一接口，当前DB默认使用rocksdb。

## monitor启动过程

monitor服务启动过程主要包括：加载基本配置、检查和加载db、初始化网络模块、mon注册网络监听、monitor bootstrap启动。

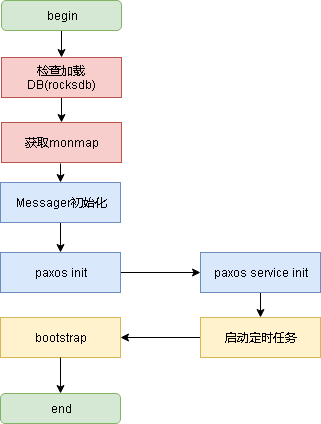


图 2 monitor启动步骤

下面介绍monitor的核心流程，选举和数据读写流程。

## monitor 核心流程

### Paxos算法介绍

Paxos算法是一种常用于分布式系统的共识一致性算法，它主要解决的问题是分布式系统中某个值(决议)如何达成一致。Paxos可以实现数据副本一致性，分布式锁，名字管理等。Paxos包括原始Paxos(basic paxos)和变种优化的multi-paxos等，其中multi-paxos更适合工程实践。

#### 1）Basic Paxos：

Basic Paxos包含的角色：

* 1. Proposer提出提案，可以是一个或多个proposer。
  2. Acceptor决策是否接受来自Proposer的提案。
  3. Learner最终提案的学习者。

Basic Paxos算法流程包含三个阶段：

1. prepare阶段

proposer向所有acceptor广播prepare的请求，请求中带有一个全局唯一且自增的proposal num（pn），acceptor接收消息后不再接收pn<=当前pn的prepare消息或propose消息。Acceptort接收之后向proposer回复promise消息。



图 3 prepare阶段a



图 4 prepare阶段b

1. accept阶段

proposer收到过半数且最大pn的提案[pn,value]，然后将提案发送给acceptor。Accept收到propose请求，大于等于本地pn，则将[pn,value]保存到本地，并且返回proposer已接收。



图 5 accept阶段a



图 6 accept阶段b

1. learn阶段

Proposer收到多数Acceptors的Accept后，决议形成，将通过的提案发送给所有Learners。



图 7 learn阶段

#### 2）Multi paxos：

原始的Paxos算法（Basic Paxos）只能对一个值形成决议，决议的形成至少需要两次网络来回，在高并发情况下可能需要更多的网络来回，极端情况下甚至可能形成活锁。如果想连续确定多个值，Basic Paxos搞不定了。因此Basic Paxos几乎只是用来做理论研究，并不直接应用在实际工程中。

实际应用中几乎都需要连续确定多个值，而且希望能有更高的效率。Multi-Paxos正是为解决此问题而提出。Multi-Paxos基于Basic Paxos做了两点改进：

1、针对每一个要确定的值，运行一次Paxos算法实例（Instance），形成决议。

2、在所有Proposers中选举一个Leader，由Leader唯一地提交Proposal给Acceptors进行表决。这样没有Proposer竞争，解决了活锁问题。在系统中仅有一个Leader进行Value提交的情况下，Prepare阶段就可以跳过，从而将两阶段变为一阶段，提高效率。

Ceph monitor采用的是Multi paxos算法。

### monitor选举过程

monitor::bootstrap()是选举入口，整个过程也由一系列状态变化而成。每个Monitor启动后，根据配置文件中的主机ip列表，发现其他monitor并且获取其他节点最新日志版本号，根据版本号大小判断是否需要从其他节点拉取db数据做同步，然后选出leader和peon，再做recovery，最后根据收到的信号shutdown。



图 8 选举状态机

首先看看probe过程：

本节点发送probe请求monmap里记录的其他节点。对端节点收到请求后，将本端first commit和last commit版本号返回。

接收到probe reply后比较版本号差距，如果本节点last commit版本号小于对端first version则需要从对端节点做全量数据同步。如果本节点的last commit和对端的last version之间只差paxos\_max\_join\_drift，则进行差异数据同步。

### monitor读写过程