## 项目背景

### 1.1系统架构

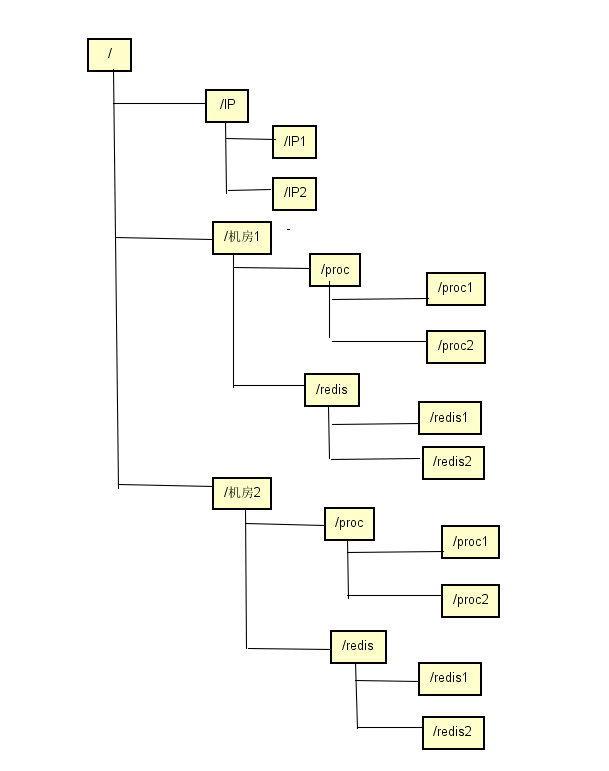


Zookeeper集群由三个zookeeper组成，集群跨机房部署，亚太部署两个，亦庄部署一个。

## 2.系统的优化目标

### 2.1 进程自动读取同机房信息

#### 2.1.1配置信息：



配置定义，配置信息以node,value的格式存储，定义如下：

1./IP:维护IP和机房的对应关系，包括以下节点：

Ip 部署服务的机器，value为groupid

2./proc:存储进程的配置信息，包括以下节点：

Name:进程名

IP:服务部署的ip

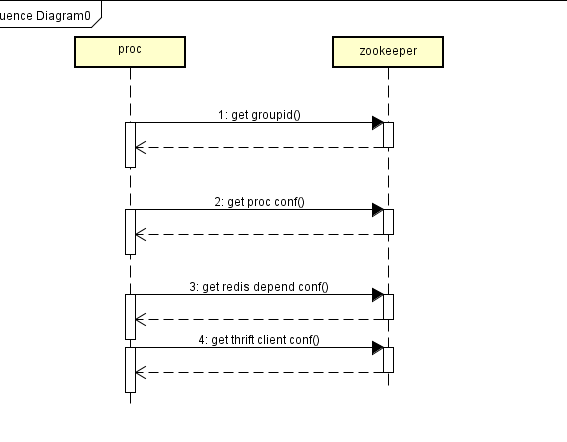
Prot:端口

Redis:进程读取的redis信息，依赖的redisname以子节点的形式在/redis父节点下

Thriftclient:进程依赖的thrift进程，依赖的thrift进程，以子节点的形式存在/thriftclient父节点下

4./redis:redis相关配置，包括以下节点:name,ip,port

#### 2.1.2进程启动读取配置流程

：

1.进程获取所在机器的IP，获取IP所在机房

2.读取进程相关配置

3.如果是依赖的redis配置，在相应的机房配置下，找到对应的reidis name配置

4.如果是依赖的thrift进程配置，在相应的机房配置下，找到对应的thrift proc配置

### 2.2 thrift client manager优化

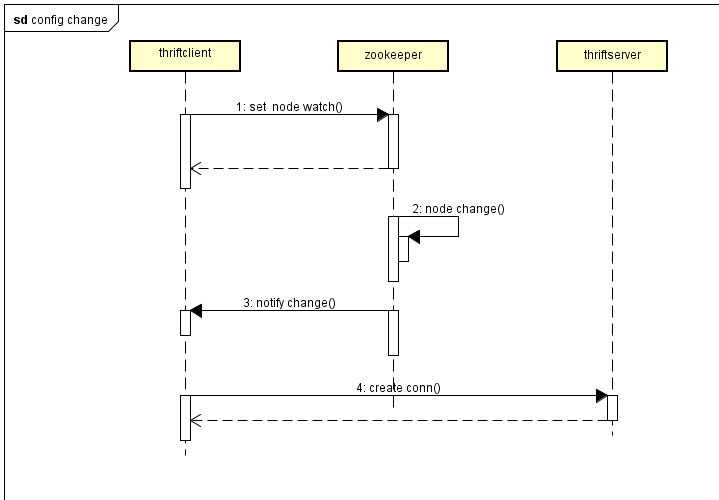
1.在传输层处理连接异常

2.热加载：配置修改，重置连接

3.FailOver：服务器出现故障,自动将该服务器摘除,故障恢复后再自动将该服务器加入

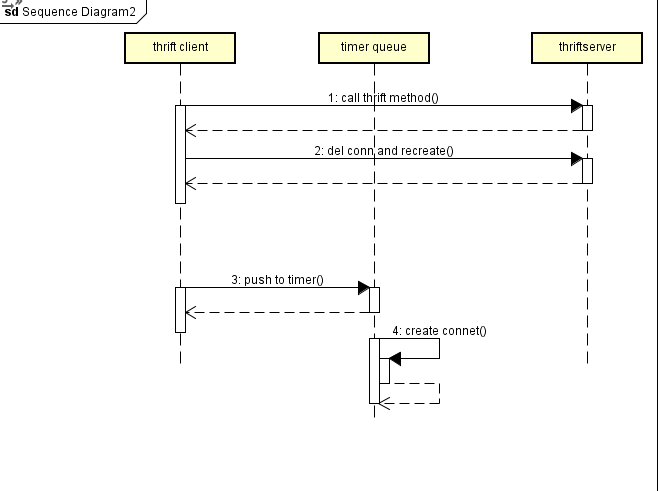
4.负载均衡：在多个同等服务直接轮询

#### 2.2.1配置热加载流程



1. thrift client启动时，在依赖的thrift进程的配置的node下创建 watch
2. 修改thrift进程配置
3. 下发变更通知
4. Thrift client 重新创建连接

#### 2.2.2 failover流程



1.调用thrift server相关接口

2.如果失败，从连接池删除该连接，thrift client 重建连接

3.如果创建链接失败，把错误连接放入定时器队列

4.定时器定时重连相关服务，成功则把连接加入连接池

### 2.3 thrift服务注册和发现

实现方式：

Thrift Server在 Zookeeper 上创建一个 EPHEMERAL 类型的目录节点，然后每个 Server 在它们创建目录节点的父目录节点上调用[getChildren](http://hadoop.apache.org/zookeeper/docs/r3.2.2/api/org/apache/zookeeper/ZooKeeper.html#getChildren%28java.lang.String,%20boolean%29)([String](http://java.sun.com/javase/6/docs/api/java/lang/String.html?is-external=true) path, boolean watch) 方法并设置 watch 为 true，由于是 EPHEMERAL 目录节点，当创建它的 Server 死去，这个目录节点也随之被删除，所以 Children 将会变化，这时 [getChildren](http://hadoop.apache.org/zookeeper/docs/r3.2.2/api/org/apache/zookeeper/ZooKeeper.html#getChildren%28java.lang.String,%20boolean%29)上的 Watch 将会被调用，所以其它 Server 就知道已经有某台 Server 死去了。

## 3故障发现和恢复

1. zookeeper单点故障:zookeeper部署集群，单节点故障不会影响系统可用性
2. zookeeper 和进程连接断开：连接断开有可能丢失watcher，进程需要全量更新配置

.

## 4监控

1. 进程所在机器CPU/MEM/IO/Load的监控
2. 单个接口的调用时间的监控：防止某些接口调用超时，导致线程池线程耗光
3. Zookeeper单机连接数，单机watcher数监控
4. zookeeper日志所在目录剩余空间告警

## 5系统缺陷与改进

由于网络不稳定，server可能跟zookeeper断开，因此必须引入第三方失效检测，判断server不可用还是网络异常。