1. 基于zookeeper构建的redis cache集群

本文目标是设计高可用，易伸缩的[Redis](http://www.dataguru.cn/article-8569-1.html?union_site=innerlink)Cache集群方案，需求： 

* 故障转移：某个Redis实例故障应当可以把负责的key转移到其他实例。故障实例保存的数据可能丢失，这是符合Cache应用场景需求的。
* 动态水平伸缩：应当可以在运行时动态增加Redis实例，以达到容量水平扩容。水平扩容可能造成部分Cache Key丢失。

1. 总体架构   
http://dl.iteye.com/upload/picture/pic/132436/8005f6ac-fef4-3087-99f9-765d0c997963.png 

* 每一个Redis实例都在Zookeeper当中注册一个EPHEMERAL SEQUENTIAL节点。
* Zookeeper就可以负责维护Redis实例是否可用的状态信息，支持动态增减Redis实例
* 每个Redis实例获得一个唯一ID方便一致性哈希[算法](http://www.dataguru.cn/article-5747-1.html?union_site=innerlink)的实现
* 每个Redis实例都需要有一个Keeper，代理维护其与Zookeeper之间的连接

2. Keeper   
负责在Zookeeper中注册维护Redis实例的信息。Keeper和Redis位于同一台机器，Keeper通过LOCAL HOST地址访问它维护的Redis实例。   
  
2.1. 启动   
Keeper启动过程的输入： 

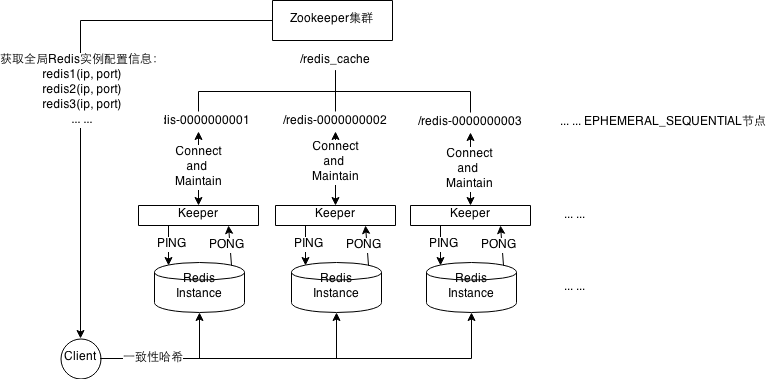
* Zookeeper IP
* Zookeeper 端口
* redis-server程序的路径
* Redis实例端口
* Client应当通过什么IP访问Redis实例

Keeper启动完成的工作： 

* 启动本机Redis实例
* 与Zookeeper建立Session
* 在Zookeeper中创建EPHEMERAL SEQUENTIAL节点，并保存Redis实例的IP、Port

2.2. Redis实例故障的检测与转移 

* Keeper每隔一段时间就向它的Redis实例发送一个PING命令
* 若Keeper连续几次都PING失败则删除Zookeeper中对应的节点
* 若Keeper与Redis实例一起发生故障，则在Zookeeper检测到Session关闭之后自动删除节点
* 若Keeper故障Redis实例未故障，则逻辑上也认为Redis实例故障，将其移除

故障转移耗时评估：   


* 假设配置Zookeeper集群tickTime为500ms
* 要求Zookeeper Session 超市时间为tickTime的：2~20倍，配置为3倍
* 配置Keeper Ping Redis实例的间隔为500ms，故障重试次数也未3次
* 则故障转移最长耗时评估为：1.5s

2.3 水平扩容的过程   
水平扩容十分简便，过程如下 

* 分配机器，在上面安装好Redis和Keeper程序
* 配置好Keeper启动需要的输入参数（见2.1）
* 启动Keeper

3.Client   
3.1 数据分片   
通过一致性哈希算法，在Client端实现数据分片。   
  
3.2 动态容错   
Client访问Redis实例的过程可以分为三步： 

* 1.在Zookeeper中查询Redis集群的动态配置信息
* 2.根据一致性哈希算法计算Key->Redis实例的映射
* 3.访问Redis实例进行数据操作

可能在上述的第1步之后，第3步之前发生Redis集群的动态变化，导致Redis实例访问失败。此时需要进行重试访问，超过一定次数之后，向上层返回错误。具体流程图如下所示：  
