美团Leaf-基于DB的Segment模式

实现原理:

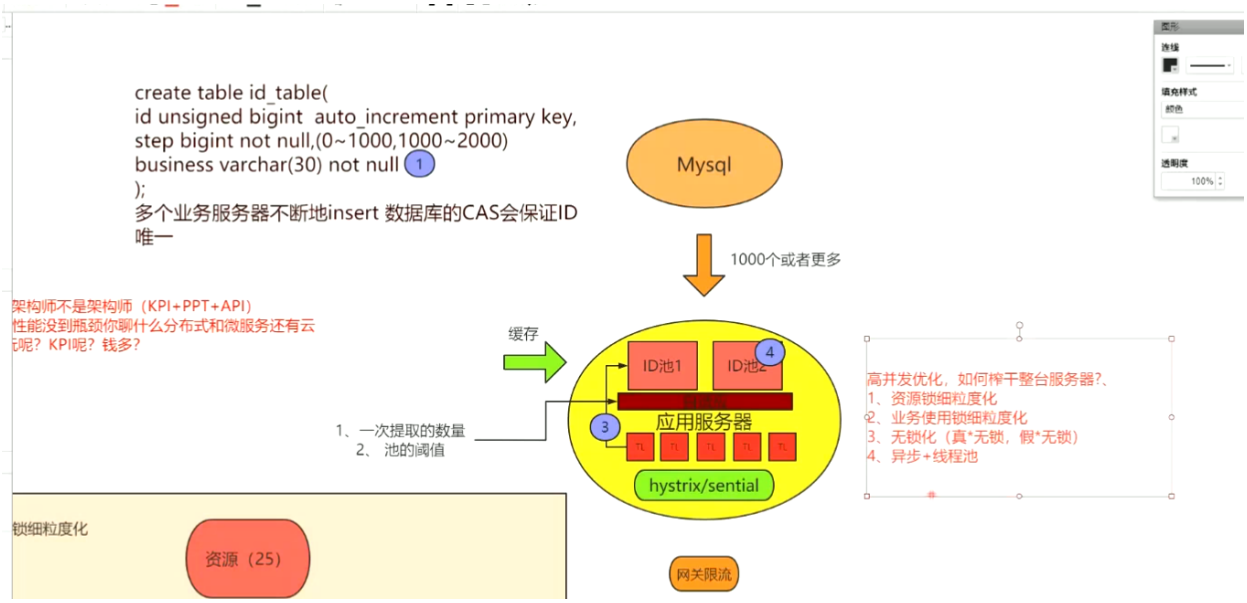
- 利用proxy server批量获取，每次获取一个segment(step决定大小)号段的值。用完之后再去数据库获取新的号段，可以大大的减轻数据库的压力。 各个业务不同的发号需求用biz_tag字段来区分，每个biz-tag的ID获取相互隔离，互不影响。如果以后有性能需求需要对数据库扩容，不需要上述描述的复杂的扩容操作，只需要对biz_tag分库分表就行
- 内存实现基于双Buffer实现性能提升

优点:

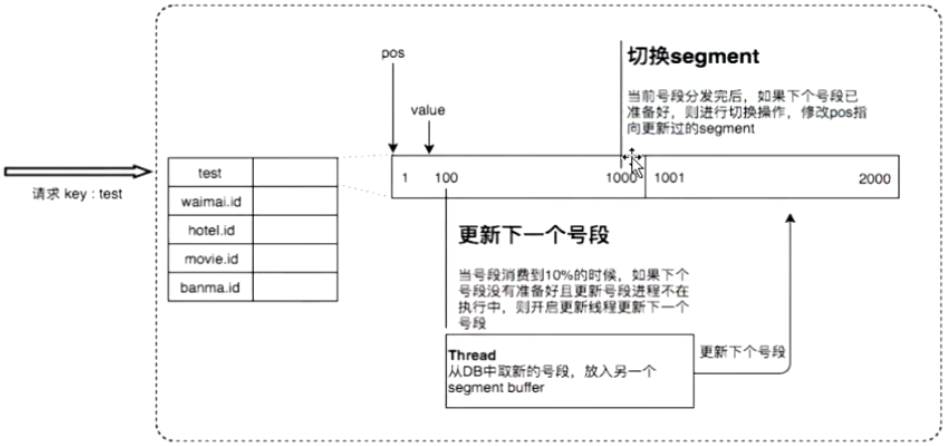
- Leaf服务可以很方便的线性扩展，性能完全能够支撑大多数业务场景
- ID号码是趋势递增的8byte的64位数字，满足上述数据库存储的主键要求
- 容灾性高：Leaf服务内部有号段缓存，即使DB宕机，短时间内Leaf仍能正常对外提供服务。
- 可以自定义max_id的大小，非常方便业务从原有的ID方式上迁移过来

缺点:

- ID号码不够随机，能够泄露发号数量的信息，不太安全。
- DB宕机会造成整个系统不可用



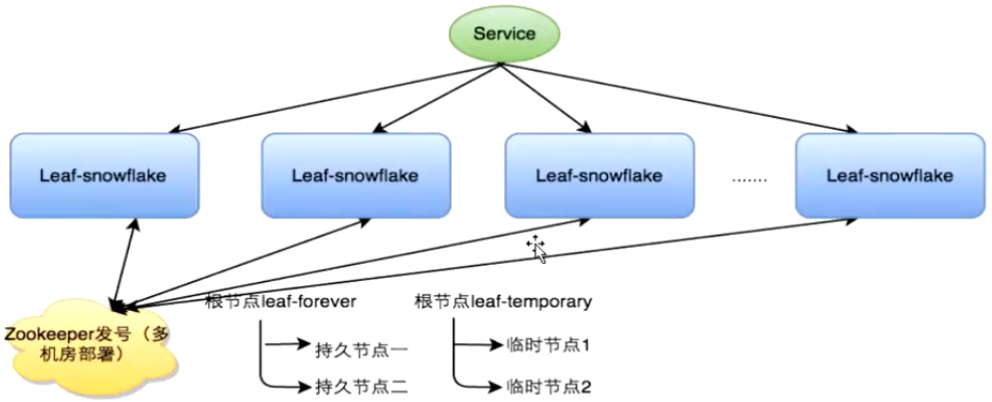
美团Leaf-双Buffer优化



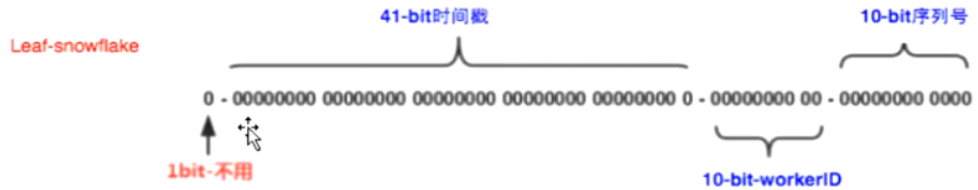
image

美团Leaf-基于ZK的SnowFlake算法

官方设计文档地址:https://tech.meituan.com/2017/04/21/mt-leaf.html

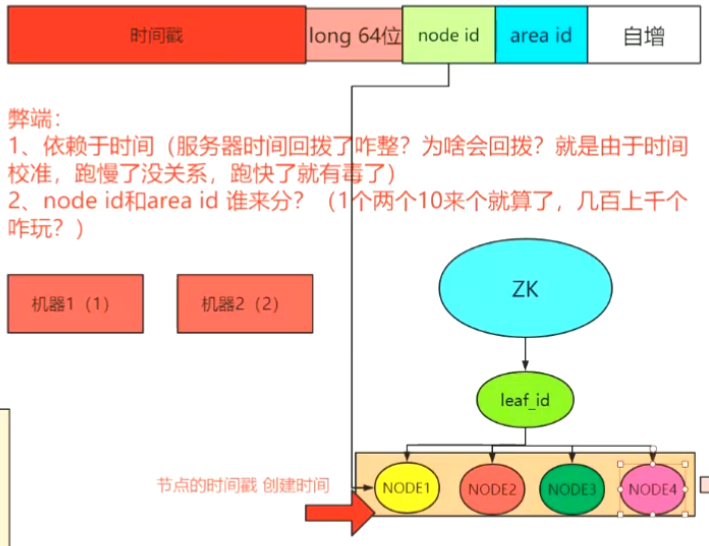


美团Leaf-SnowFlake的 ID格式



image

Leaf-snowflake方案完全沿用snowflake方案的bit位设计.
即是“1+41+10+12”的方式组装ID号。
对于workerID的分配，当服务集群数量较小的情况下，完全可以手动配置。
Leaf服务规模较大，动手配置成本太高。所以使用Zookeeper持久顺序节点的特性自动对snowflake节点配置wokerID



以多不以少

1、依赖于时间（服务器时间回拨了咋整？为啥会回拨？就是由于时间校准，跑慢了没关系，跑快了就有毒了）
节点有一个创建时间，加入node1下线了，断开了链接，由于是持久性节点，再一次上来的时候它可以拿到唯一的id也就是上一次分配的id保证了ID不会重复和新分配。可以根据节点的创建时间和当前重连的时间做对比，大于则报错。

2、node id和area id 谁来分？（1个两个10来个就算了，几百上千个咋玩？）、
通过ZK的顺序性来分配唯一的node id或者area id