**REDIS大key优化方案**

**版本： 1.0**

**拟制:**

**修改记录**

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **文件编号** | **版本号** | **拟制人/修改人** | **拟制/修改日期** | **更改理由** | **主要更改内容** |
| 1 | 1.0 | 冉高攀 | 2018-1-23 |  | 拟制 |
|  |  |  |  |  |  |
| 注1：每次更改归档文档时必须填写该表 注2：一次归档时，“更改理由”和“主要更改内容”填无 | | | | | |

目录

[1 编写目的 4](#_Toc505101083)

[2 设计思路 4](#_Toc505101084)

[2.1 HASH 4](#_Toc505101085)

[2.2 LIST 4](#_Toc505101086)

[2.3 SET 5](#_Toc505101087)

[3 大KEY分析 5](#_Toc505101088)

[4 大key的删除 6](#_Toc505101089)

# 编写目的

避免redis设计中出现大key，已经出现大key后的处理思路。

# 设计思路

结合业务现状，尽量避免出现大key的出现。如果业务场景需要大key，可以拆分成多个中小key，因为redis大key不但为造成redis内存节点分布不均匀，读取大key的时候，还有影响redis整体性能，下面我就大概分不同种数据结构来解析怎么避免这种大key的出现。

## HASH

如果业务hash存在大量的数据进行处理，我提出下面几种方案：

1. 首先我们根据业务预估整个fieds 要多少条数据，总共需要消耗多少内存。
2. 假设整个hash有500M,那么我们就可以拆分为10个小的hash数据结构。
3. 整个hash的key我们可以动态生成比如这种格式：

Newkey=oldKey+hash(filed)%10

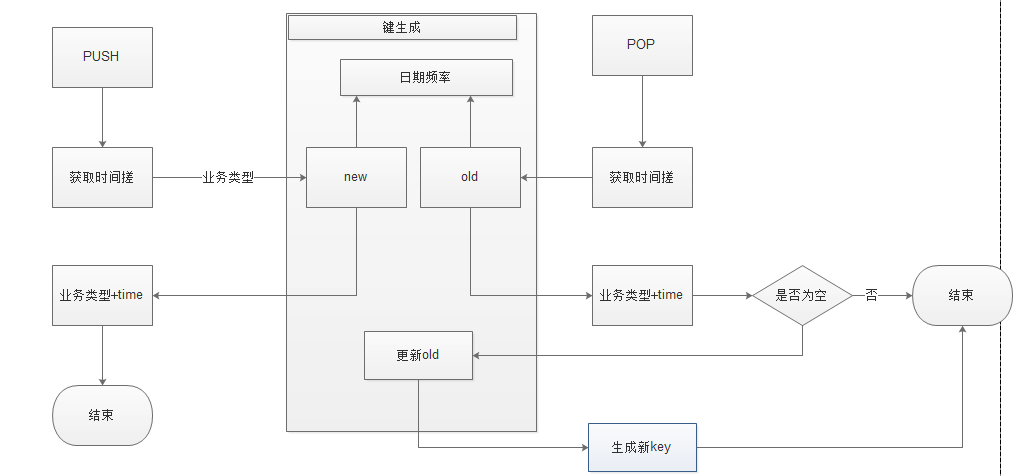
1. 那么我们读取或者写入数据就可以根据新key去操作。

LIST

因为list大多数情况可能是要来做队列，那么上面的那种算法应该是不适合做list的，因为要保证pop都是最先push到队列里面的数据，那么我可以可能要按照日期来做分区，所以我们首先还是要评估整个队列的大小，可以按照月，日，天去拆分数据。

比如 list\_业务\_20170130, 20170130通过封装的日期函数获取。

分两种场景 push 和pop：



Push:

1. 调用日期函数new，生成日期搓yyyymmdd。
2. 拼接key,list\_业务\_yyyymmdd完成push。

Pop：

1. 调用日期old函数，返回时间搓yyyymmdd。
2. 拼接新key=list\_业务+yyyymmdd
3. Pop（key）,如果数据为空，调用日期函数next(XXXXX),如果返回为空则结束，否则重新pop，同时更新set(list\_业务,XXXXX)。

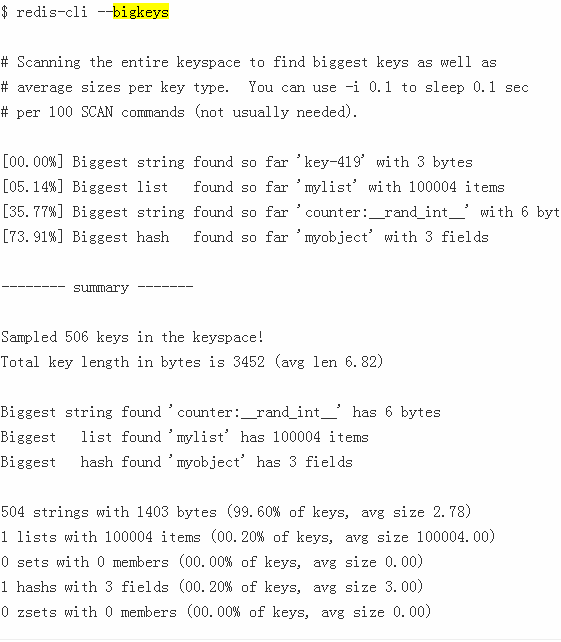
## SET

集合类型，主要考虑业务场景，放多个小的集合，通过redis支持的集合之间的操作，或者自己通过客户端代码实现。

# 大KEY分析

发现大key有两种方式：

1. redis-cli --bigkeys:



遇到有的key会报错，会中断。

1. 导入数据，通过mysql进行分析，利用redis-rdb-tools生成csv文件导入mysql分析，

步骤如下：

1. 安装redis-rdb-tools。
2. yum install python-pip && pip install rdbtools
3. redis 生成dump 文件，bgsave，bgsav会fork一个子进程，查看是否成功 redis-cli info|grep ,0表示成功
4. 生成cvs文件：

rdb -c memory dump.rdb > memory.csv

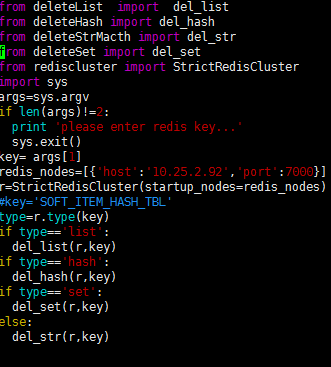
生成 CSV 格式的内存报告。包含的列有：数据库 ID，数据类型，key，内存使用量(byte)，编码。内存使用量包含 key、value 和其他值，csv文件可以导入sqllite，mysql等数据库分析。

# 大key的删除

直接删除大key风险：

DEL key  
Time complexity: O(N) where N is the number of keys that will be removed. When a key to remove holds a value other than a string, the individual complexity for this key is O(M) where M is the number of elements in the list, set, sorted set or hash. Removing a single key that holds a string value is O(1)。

自己用python封装了一个可以删除各种类型的大key,还支持模糊删除，主函数main.py如下：



其余代码见附件，confluence地址如下：

http://10.206.2.221:6090/pages/viewpage.action?pageId=9004862