# 1 Redis有哪些数据类型

| **类型** | **简介** | **特性** | **场景** |
| --- | --- | --- | --- |
| String(字符串) | 二进制安全 | 可以包含任何数据,比如jpg图片或者序列化的对象,一个键最大能存储512M | --- |
| Hash(字典) | 键值对集合,即编程语言中的Map类型 | 适合存储对象,并且可以像数据库中update一个属性一样只修改某一项属性值(Memcached中需要取出整个字符串反序列化成对象修改完再序列化存回去) | 存储、读取、修改用户属性 |
| List(列表) | 链表(双向链表) | 增删快,提供了操作某一段元素的API | 1,最新消息排行等功能(比如朋友圈的时间线) 2,消息队列 |
| Set(集合) | 哈希表实现,元素不重复 | 1、添加、删除,查找的复杂度都是O(1) 2、为集合提供了求交集、并集、差集等操作 | 1、共同好友 2、利用唯一性,统计访问网站的所有独立ip 3、好友推荐时,根据tag求交集,大于某个阈值就可以推荐 |
| Sorted Set(有序集合) | 将Set中的元素增加一个权重参数score,元素按score有序排列 | 数据插入集合时,已经进行天然排序 | 1、排行榜 2、带权重的消息 |

**对于二进制安全的理解**

它并不像C语言那样，使用’\0’作为判定一个字符串的结尾，所以如果你保存的字符串内存在’\0’，c语言自会识别前面的数据，后面的就会被忽略掉，所以说是不安全的。而redis是使用了独立的len，这样可以保证即使存储的数据中有’\0’这样的字符，它也是可以支持读取的。而且前面介绍到Redis的string可以支持各种类型（图片、视频、静态文件、css文件等）

# 2 Redis数据持久化

## 2.1 快照（snapshots）

默认配置

*save 900 1*

*save 300 10*

*save 60 10000*

上面参数表示：

* 900 秒内至少改变了一个 key，则在第900秒时执行一次快照持久化
* 300 秒内至少改变了十个 key，则在第300秒时执行一次快照持久化
* 60 秒内至少改变了 10000 个 key，则在第60秒时执行一次快照持久化

## 2.2 APPEND ONLY MODE（AOF）

## 2.3两种持久化比较

快照更新频率比较低，数据安全性没有aof高，当服务器宕机时，比较容易丢失数据。

Aof数据更新频率比较高，数据安全性也就会比较高，但是对服务器压力也相应增大。

# 3 Redis事务

# 4 Redis分布式方案

## 4.1 Redis Cluster

## 4.2 一致性hash

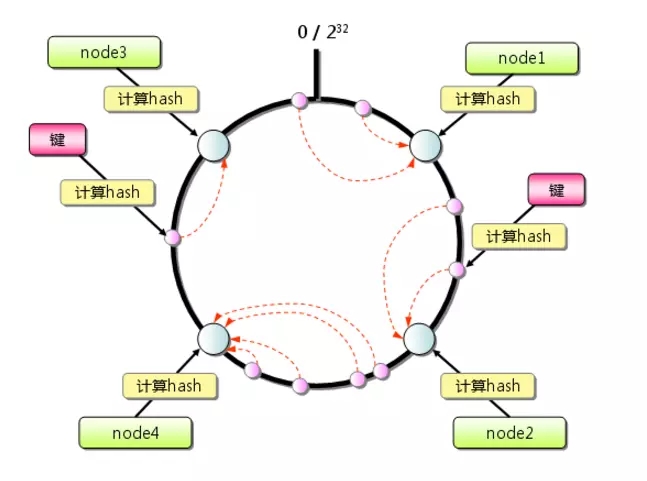
原理：

一个0-2^32的环形节点槽

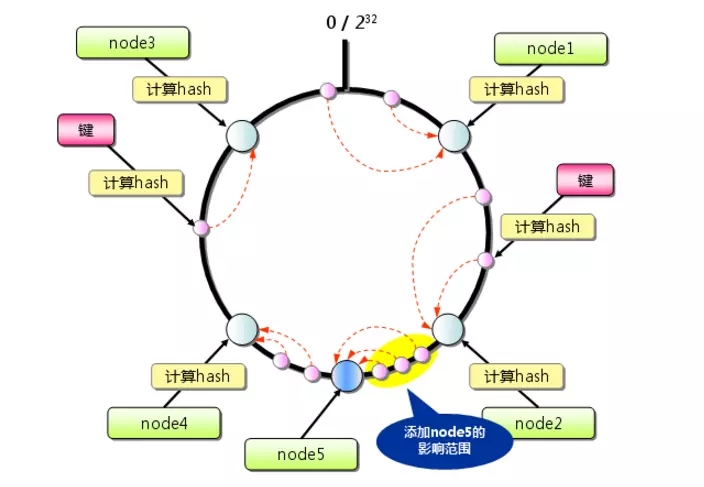
根据服务器ip或者是物理地址进行hash，然后配置到环上。

采用同样的hash算法计算key的hash值，并映射到环上

从数据映射的顺时针查找服务器节点，存储在第一个服务器节点上。



新增服务器情况下

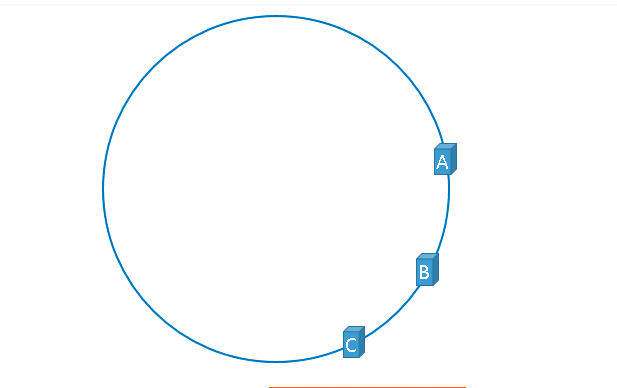


如图，只有node2-node5之间的数据需要重新分配，其他数据不变

收缩节点的时候也是类似。

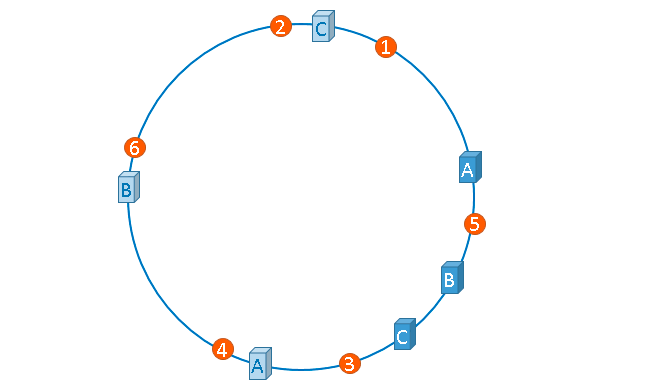
哈希倾斜问题：

这种情况下，大部分的数据将会存储到节点A中，造成服务器A的压力变大。



解决方案：使用虚拟节点

如图 服务器A、B、C都进行两次hash，然后映射到环上，数据1，3都是分配到服务器A上。如果使用三次hash，数据会分配更均匀一些。



**实现思路：**

只要维护好一个服务器hash队列，key的hash不用维护，只是用key的hash来查找服务器的位置。由于服务器hash是有序的，所以比较快的查找可以用二分查找算法。顺序查找对性能影响也不大。

**思考：**

一致性hash的环为啥取0——2^32-1次方。

Hash值是int类型，大部分语言int类型的范围是-2^31——2^31-1次方，无符号int类型就是0——2^32-1，所以取环的大小为0——2^32-1，当然有符号的int类型需要进行一下转换。