链滴

redis 的动态字符串 (sds) 原理

作者：PerrorOne

原文链接：https://ld246.com/article/1554533516433

来源网站：链滴

<blockquote>
本文基于《redis 的设计与实现》编写完成。
</blockquote>
<h3 id="什么是SDS-">什么是 SDS？</h3>
SDS 是 redis 构建的一种抽象类型，主要用于储存 redis 的默认字符串表示、AOF 模块中的 AF 缓冲区、客户端状态输入缓冲区。
<h3 id="SDS有什么优点-">SDS 有什么优点？</h3>
redis 为什么不使用 C 的字符串？我们要先来看看 C 字符串中的缺点：
1. C 字符串不记录自身长度信息，为了获取字符串长度必须遍历整个字符串，时间复杂度为 O(n)
2. 由于 C 字符串不记录自身长度，稍有不小心就会造成缓冲区溢出。
3. 对于 redis 这种缓存类型数据库，对于缓存的 Value 是有可能经常的更改的。但是 C 字符串每的增长或是缩小都需要一次内存的重分配操作。
4. redis 数据库中缓存的内容不是特定的，有可能会是图片、音频等等文件的二进制数据，但是 C 字符串中的字符必须符合某种编码，且字符串中不能包含空格，这些限制也导致了 redis 不能使用 C 符串来作为自身字符串的实现。
而 SDS 则将这些缺点都一一杜绝：
1. SDS 记录了自身的长度信息，使得获取字符串长度的时间复杂度为 O(1)。
2. SDS 使用了预分配空间以及惰性空间释放的算法，解决了频繁分配内存的操作。
3. SDS 由于保存了自身的长度，也导致了 SDS 不会像 C 一样按照'\0'确定字符串的结尾。
<h3 id="SDS是如何实现的-">SDS 是如何实现的？</h3>
redis 使用名为 sdshdr 的结构体表示 SDS 值：
<pre><code class="highlight-chroma">struct sdshdr{
    int len; // 记录buf数组中已使用字节的数量
    int free; // 记录buf中未使用字节数量
    char buf[]; // 存字符串的数组
}
</code></pre>
SDS 遵循这 C 字符串以空字符结尾的惯例，以便能够重用 C 字符串中的函数，这个结尾的空字串并不会增加 len 字段的值，例如现在需要使用 SDS 保存'golang'这个字符串，那么 SDS 对应的结体是：

由于 SDS 记录了自身的长度，所以 redis 中获取字符串的长度只需要返回 len 字段的字段，其时复杂度为 O(1)，而对于 C 字符串经常发生的缓冲区溢出，SDS 的空间分配策略完全杜绝了这种可能：当 redis 需要对 SDS 字符串修改时，首先会检查 SDS 的空间是否满足修改所需的空间，如果不满则将 SDS 空间扩容至满足该修改空间所需的容量，SDS 空间扩容的算法如下：
1. 如果 SDS 修改后的长度小于 1MB，那么程序将会分配 len 字段值同样大小的未使用空间，这是为什么我上面给出的例图中 len 的值是 6，free 值也是 6 的原因。对于上面那个例子，SDS 的总用大小为：6(len) + 6(free) + 1('\0') = 13 字节的容量。
2. 如果 SDS 修改后的长度大于 1MB，那么 SDS 将会为 SDS 分配 1MB 的未使用空间。如果一个DS len 值修改后的容量为 2MB，那么 redis 将会为 SDS 分配 1MB 的未使用空间。则此时 SDS 的占用大小为：2MB(len) + 1MB(len) + 1byte('\0') =3073 byte 容量。
这种扩容算法，减少了频繁的向系统申请内存的操作。SDS 的空间释放并不是实时的，而是惰性释放redis 认为如果一个 SDS 的容量到达过 N 大小，则极有可能在其缩小后也有可能到达 N，且惰性释也减少了下次内存分配的可能，假如现在有一个'golang'字符串存储在 redis 中， 现在我们需要将'goang'修改为'go'，那么 redis 将会对 SDS 结构体将会是这样的：

<img src="https://ld246.com/images/img-loading.svg" alt="%TF4}HTEC-93)3%JV}W-VRW.pn
" data-src="https://b3logfile.com/file/2019/04/TF4HTEC933JVWVRW-b32611d6.png?imageVi
w2/2/interlace/1/format/jpg">
那么 SDS 提供了清理内存的 API，我们可以在有需要时，调用该 API 以便真正的释放内存。
<h3 id="总结">总结</h3>
1. redis 只会使用 C 字符串作为字面量使用，大多数情况都使用 SDS 来表示字符串。
2. 能够使用常数级复杂度获取字符串的长度。
3. 杜绝了缓冲区溢出。
4. 减少了字符串所需内存重新分配的次数。以及对于二进制数据安全，且兼容部分 C 字符串函数