小象科技 ChinaHadoop.cn

# Redis 从入门到精通

黄健宏（huangz）

小象科技 ChinaHadoop.cn

# HyperLogLog

使用常量空间估算大量元素的基数。

# 问题

记录网站每天获得的独立 IP 数量。

# 集合实现



使用集合来储存每个访客的 IP，通过集合性质(集合中的每个元素都各不相同)来得到多个独立 IP，然后通过调用 SCARD 命令来得出独立 IP 的数量。

举个例子，程序可以使用以下代码来记录 2014 年 8 月 15 日，每个网站访客的 IP：

```
ip = get_vistor_ip()
SADD '2014.8.15::unique::ip' ip
```

然后使用以下代码来获得当天的唯一 IP 数量：

```
SCARD '2014.8.15::unique::ip'
```

# 集合实现的问题



使用字符串来储存每个 IPv4 地址最多需要耗费 15 字节(格式为 'XXX.XXX.XXX.XXX'，比如 '202.189.128.186')。

下表给出了使用集合记录不同数量的独立 IP 时，需要耗费的内存数量：

| 独立 IP 数量 | 一天 | 一个月 | 一年 |
|---|---|---|---|
| 一百万 | 15 MB | 450 MB | 5.4 GB |
| 一千万 | 150 MB | 4.5 GB | 54 GB |
| 一亿 | 1.5 GB | 45 GB | 540 GB |

随着集合记录的 IP 越来越多，消耗的内存也会越来越多。

另外如果要储存 IPv6 地址的话，需要的内存还会更多一些。

# HyperLogLog

为了更好地解决像独立 IP 地址计算这种问题，
Redis 在 2.8.9 版本添加了 HyperLogLog 结构。

# HyperLogLog 介绍



HyperLogLog 可以接受多个元素作为输入，并给出输入元素的基数估算值：

- 基数：集合中不同元素的数量。比如 {'apple', 'banana', 'cherry', 'banana', 'apple'} 的基数就是 3 。
- 估算值：算法给出的基数并不是精确的，可能会比实际稍微多一些或者稍微少一些，但会控制在合理的范围之内。

HyperLogLog 的优点是，即使输入元素的数量或者体积非常非常大，计算基数所需的空间总是固定的、并且是很小的。

在 Redis 里面，每个 HyperLogLog 键只需要花费 12 KB 内存，就可以计算接近 2^64 个不同元素的基数。这和计算基数时，元素越多耗费内存就越多的集合形成鲜明对比。

但是，因为 HyperLogLog 只会根据输入元素来计算基数，而不会储存输入元素本身，所以 HyperLogLog 不能像集合那样，返回输入的各个元素。

# 将元素添加至 HyperLogLog



PFADD key element [element ...]

将任意数量的元素添加到指定的 HyperLogLog 里面。

这个命令可能会对 HyperLogLog 进行修改，以便反映新的基数估算值，如果 HyperLogLog 的基数估算值在命令执行之后出现了变化，那么命令返回 1 ，否则返回 0 。

命令的复杂度为 O(N)，N 为被添加元素的数量。

# 返回给定 HyperLogLog 的基数估算值



PFCOUNT key [key ...]

当只给定一个 HyperLogLog 时，命令返回给定 HyperLogLog 的基数估算值。

当给定多个 HyperLogLog 时，命令会先对给定的 HyperLogLog 进行并集计算，得出一个合并后的 HyperLogLog，然后返回这个合并 HyperLogLog 的基数估算值作为命令的结果（合并得出的 HyperLogLog 不会被储存，使用之后就会被删掉）。

当命令作用于单个 HyperLogLog 时，复杂度为 O(1)，并且具有非常低的平均常数时间。

当命令作用于多个 HyperLogLog 时，复杂度为 O(N)，并且常数时间也比处理单个 HyperLogLog 时要大得多。

# PFADD 和 PFCOUNT 的使用示例



```
redis> PFADD unique::ip::counter '192.168.0.1'
(integer) 1

redis> PFADD unique::ip::counter '127.0.0.1'
(integer) 1

redis> PFADD unique::ip::counter '255.255.255.255'
(integer) 1

redis> PFCOUNT unique::ip::counter
(integer) 3
```

# 合并多个 HyperLogLog



PFMERGE destkey sourcekey [sourcekey ...]

将多个 HyperLogLog 合并为一个 HyperLogLog，合并后的 HyperLogLog 的基数估算值是通过对所有给定 HyperLogLog 进行并集计算得出的。

命令的复杂度为 O(N)，其中 N 为被合并的 HyperLogLog 数量，不过这个命令的常数复杂度比较高。

# PFMERGE 的使用示例



```
redis> PFADD str1 "apple" "banana" "cherry"
(integer) 1
redis> PFCOUNT str1
(integer) 3

redis> PFADD str2 "apple" "cherry" "durian" "mongo"
(integer) 1
redis> PFCOUNT str2
(integer) 4

redis> PFMERGE str1&2 str1 str2
OK
redis> PFCOUNT str1&2
(integer) 5
```

# 唯一计数器的 API 及其实现



| API | 作用 | 实现原理 |
| --- | --- | --- |
| UniqueCounter(client, key) | 指定客户端以及键名。 | |
| UniqueCounter.include(element) | 将给定的元素计算在内。 | PFADD |
| UniqueCounter.result() | 返回计数器当前的值。 | PFCOUNT |

这个唯一计数器的实现代码可以在 unique_counter.py 看到。

# 唯一计数器的使用示例

小象科技 ChinaHadoop.cn

```
# 创建一个 IP 地址唯一计数器
>>> ip_counter = UniqueCounter(client, 'unique::ip::counter')

# 添加一些 IP
>>> ip_counter.include('192.168.0.1')
>>> ip_counter.include('8.8.8.8')
>>> ip_counter.include('255.255.255.255')

# 查看计数器当前的值
>>> ip_counter.result()
3
```

# HyperLogLog 实现独立 IP 计算功能



下表列出了使用 HyperLogLog 记录不同数量的独立 IP 时，需要耗费的内存数量：

| 独立 IP 数量 | 一天 | 一个月 | 一年 | 一年（使用集合） |
|---|---|---|---|---|
| 一百万 | 12 KB | 360 KB | 4.32 MB | 5.4 GB |
| 一千万 | 12 KB | 360 KB | 4.32 MB | 54 GB |
| 一亿 | 12 KB | 360 KB | 4.32 MB | 540 GB |

可以看到，要统计相同数量的独立 IP，HyperLogLog 所需的内存要比集合少得多。

# 复习

回顾一下本节所学的知识。

# 复习



HyperLogLog 接受多个元素作为输入，估算出输入元素的基数。

因为 HyperLogLog 只需要使用少量内存就可以对非常多的元素进行计数，对于那些只想知道输入元素的基数，但是并不需要知道具体输入的是哪些元素的程序来说，使用 HyperLogLog 而不是集合来计算基数，可以节约大量内存。

三个命令：PFADD、PFCOUNT、PFMERGE 。