

Redis 从入门到精通

黄健宏(huangz)



感谢！

我会努力地将自己所知道的 东西分享给大家，也希望大家能够享受这个课程。



课程目标

由浅入深、循序渐进地学习关于 Redis 的应用、管理和实现等方面的知识。

其中包括：

1. 学会使用 Redis 的单机和多机功能。
2. 学会使用 Redis 来构建实际的应用程序。
3. 学会使用 Redis 自带的工具以及第三方工具，来 维护和管理 Redis 。
4. 了解 Redis 的实现原理，以便更好、更高效地使用 Redis 。



Redis 的诞生

在学习 Redis 之前, 让我们先来了解 Redis 的诞生过程。



Redis 的创建者

Salvatore Sanfilippo (antirez), 男, 意大利人, 出生并居住在西西里 岛, 个人网站 <http://invece.org/> 。

早年为系统管理员, 关注计算机安全领域, 于 1999 年发明了 idle scan 扫描技术, 该技术现在在 nmap 扫描器上也有实现。

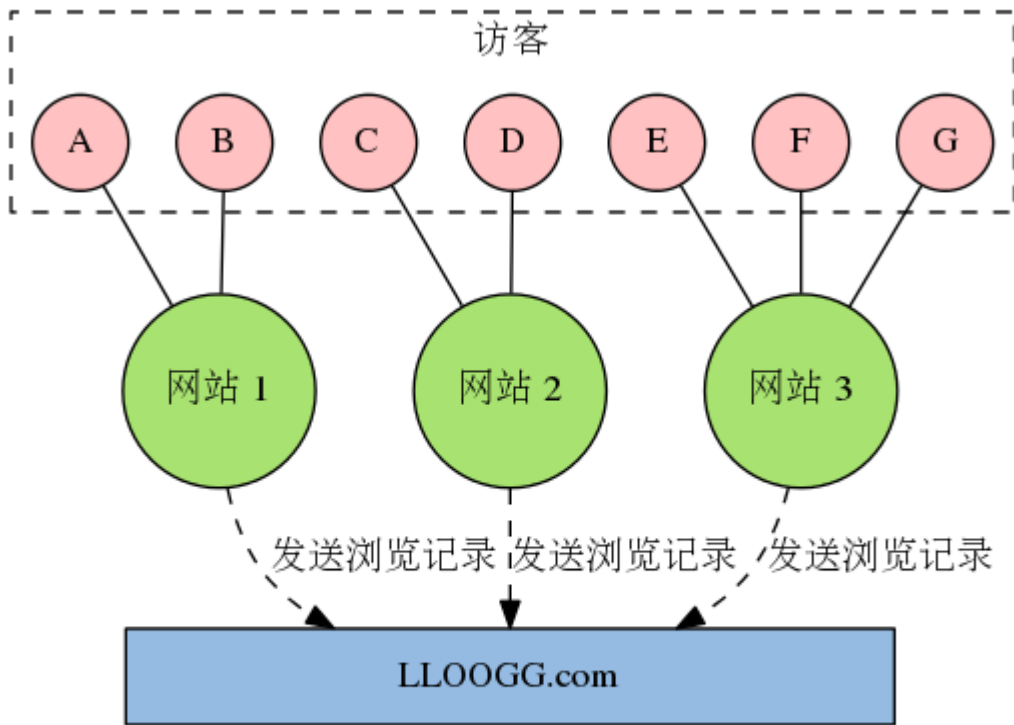
2004 年~2006 年期间在做嵌入式方面的工作, 并为此写了名为 Jim 的 Tcl 解释器、《Tcl Wise: Guide to the Tcl programming language》一书以及《Tcl the Misunderstood》文档。(Redis 的事件处理器就重写自 Jim 的事件循环, 而 Redis 的测试套件也使用 Tcl 语言来写的)。除此之外, 他在 2006 还写了 Hping —— 一个 TCP/IP 包分析器。

之后开始接触 web, 在 2007 年和另一个朋友共同创建了 LLOOGG.com, 并因为解决这个网站的负载问题而在 2009 年 2 月 26 日发明了 Redis。



一个访客信息追踪网站，网站可以通过 JavaScript 脚本，将访客的 IP 地址、所属国家、浏览器信息、被访问页面的地址等数据传送给 LLOOGG.com。

然后 LLOOGG.com 会将这些浏览数据通过 web 页面**实时地**展示给用户，并储存起最新的 5 至 10,000 条浏览记录以便进行查阅。



三个网站正在向 LLOOGG.com 发送它们的访客浏览记录

LLOOGG.com 用户界面



your web2.0 tail -f access.log



home - logout

demo [pro] | View realtime logs | Settings | Feedbacks

10



load history

open filters

<http://redis.io/commands/lpush> from <https://www.google.com/>

211.151.238.51 **China** 1366x768 **Chrome/36.0.1985.125** running on **Linux x86_64**
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/36.0.1985.125 Safari/537.36
19:32:53 Jul 27 - 0 urls previously visited in the same session/tab

<http://redis.io/documentation> from <http://redis.io/>

213.176.234.10 **Russian Federation** 1366x768 **Chrome/33.0.1750.152** running on **Linux x86_64**
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.152 Safari/537.36
19:32:31 Jul 27 - 2 urls previously visited in the same session/tab

<http://redis.io/> from direct url (no referer)

213.176.234.5 **Russian Federation** 1366x768 **Chrome/33.0.1750.152** running on **Linux x86_64**
Mozilla/5.0 (X11; Linux x86_64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/33.0.1750.152 Safari/537.36
19:32:25 Jul 27 - 1 urls previously visited in the same session/tab

<http://redis.io/> from <http://stackoverflow.com/questions/15198316/embedded-java-key-value-storage>

173.74.151.238 **United States** 768x1024 **Safari/9537.53** running on **Mac OS X**
Mozilla/5.0 (iPad; CPU OS 7_0_4 like Mac OS X) AppleWebKit/537.51.1 (KHTML, like Gecko) CriOS/35.0.1916.41
19:32:15 Jul 27 - 0 urls previously visited in the same session/tab

p.s. Google Analytics 直到 2011 年才有了实时功能, LLOOGG.com 的实时反馈想法在当时(2007 年)还是很有新意的。

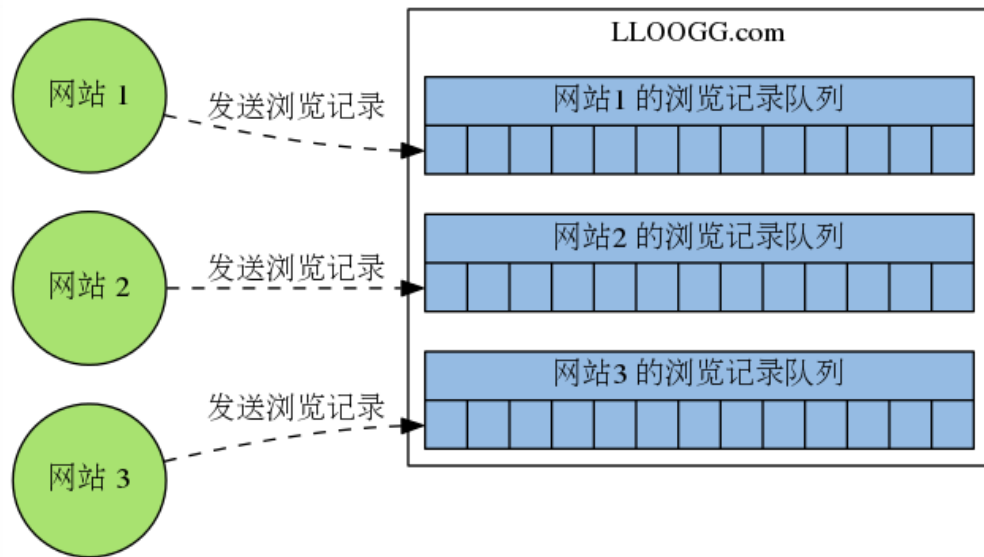


小象科技

让你的数据产生价值

LLOOGG.com 的运作方式(1)

为了记录每个被追踪网站的浏览信息，LLOOGG.com 需要为每个被追踪的网站创建一个列表(list)，每个列表需要根据用户的设置，储存最新的 5 至 10,000 条浏览记录。

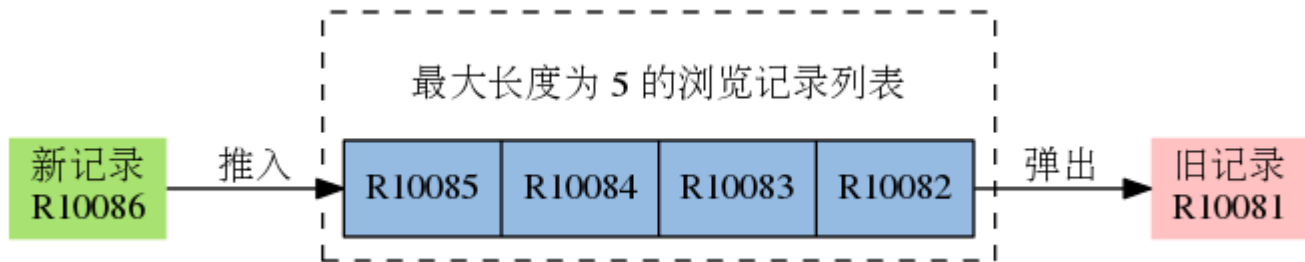
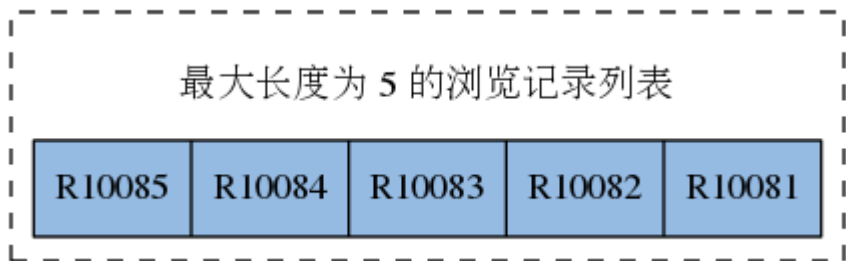


各个网站发送的浏览记录会分别进入相应的队列



LLOOGG.com 的运作方式(2)

每当某个被追踪的网站新增一条 浏览记录时, LLOOGG.com 就会将这条新的浏览记录推入 (push) 到与该网站相对应的列表里面, 当列表的长度超过用户指定的最大长度时, 程序每向列表推入一条新的记录, 就需要从列表中弹出 (pop) 一条最旧的记录。



LLOOGG.com 的负载问题



随着 LLOOGG.com 的用户越来越多，LLOOGG.com 要维护的列表数量也越来越多，要执行的推入和弹出操作也越来越多。

LLOOGG.com 当时使用 MySQL 数据库，而 MySQL 每次执行推入和弹出操作都要进行硬盘写入和读取，程序的性能严重受制于硬盘 I/O 。

最终，LLOOGG.com 所使用的 MySQL 再也无法在当时的 VPS 上处理新增的大量负载，因为 LLOOGG.com 当时还没有找到盈利模式，所以为了尽量节约开支，antirez 没有选择直接升级 LLOOGG.com 所使用的 VPS，而是打算另寻办法，在现有硬件的基础上，通过提升列表操作的性能来解决 负载问题。



Redis 的诞生

为了在不升级 VPS 的前提下, 解决 LLOOGG.com 的负载问题, antirez 决定自己写一个具有列表结构的内存数据库原型(prototype)。

这个数据库原型支持 $O(1)$ 复杂的推入和弹出操作, 并且将数据储存在内存而不是硬盘, 所以程序的性能不会受到硬盘 I/O 限制, 可以以极快的速度执行针对列表的推入和弹出操作。

经过实验, 这个原型的确可以在不升级 VPS 的前提下, 解决 LLOOGG.com 当时的负载问题。

于是 antirez 使用 C 语言重写了这个内存数据库, 并给它加上了持久化功能, Redis 就此诞生!



Redis 的演进

经过五年时间的演进, Redis 发生了以下变化.....

刚开始	现在
只支持列表结构	支持字符串、列表、散列等六种 结构, 以及丰富的附加功能
只能单机运行, 没有内置的方法可以方便地将数据库分布到多台机器上	支持多机运行(包括复制、自 动故障转移以及分布式数据库)
少有人知的开源 项目	广为人知并广泛使用在很多大型网站的 热门开源项目
接受捐款支持, antirez 自己无偿开发	Pivotal 公司出资支持开发, 并且有非常多的开发者通过 GitHub 和论坛为这个项目添砖加瓦



Redis 的特色

世界上有无数种数据库, 为什么要选择使用 Redis 呢?



独特的键值对模型

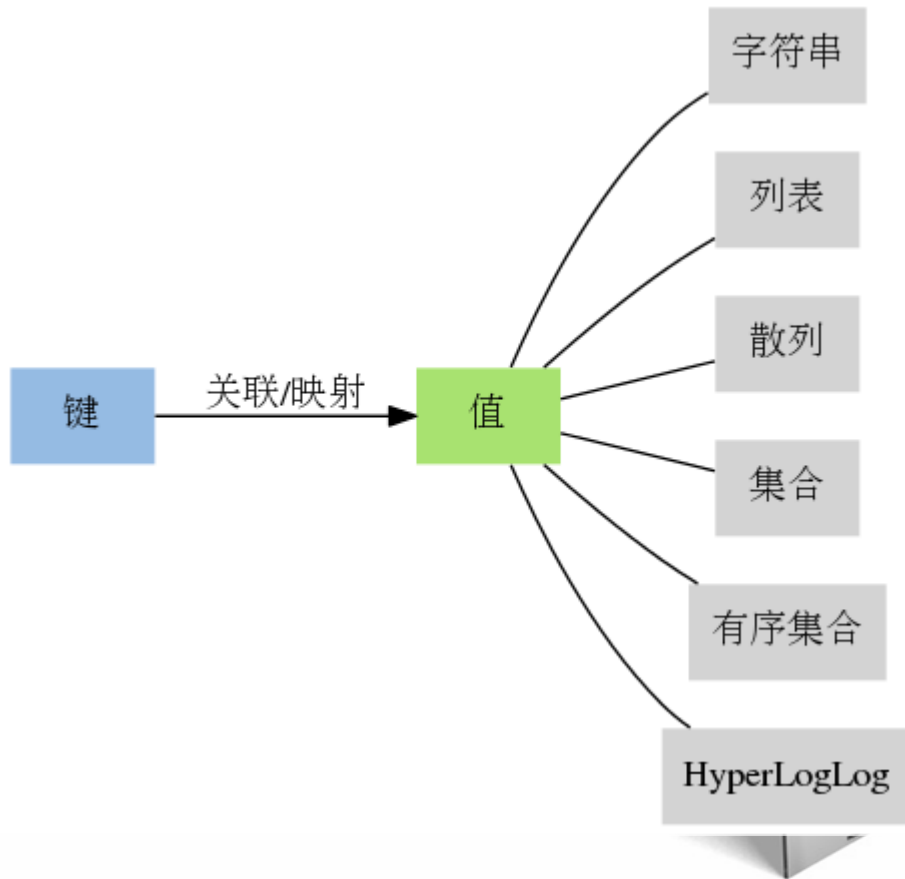
很多数据库只能处理一种数据结构：

- SQL 数据库 —— 表格
- Memcached —— 键值对数据库，键和值都是字符串
- 文档数据库 (CouchDB、MongoDB) —— 由 JSON/BSON 组成的文档 (document)

而一旦数据库提供的数据结构不适合去做某件事的话，程序写起来就会非常地麻烦和不自然。

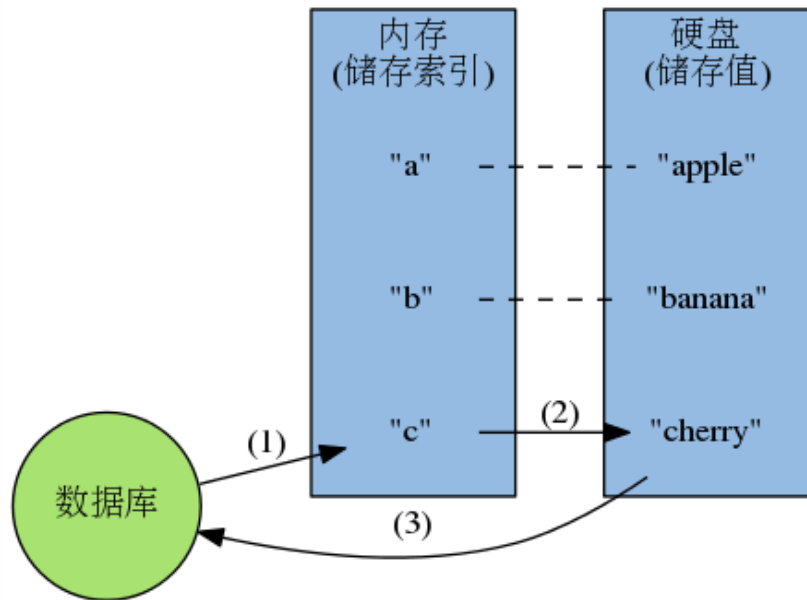
Redis 也是键值对数据库，但和 Memcached 不同的是，Redis 的值不仅可以是字符串，还可以其他五种数据结构中的任意一种。

通过选用不同的数据结构，用户可以使用 Redis 解决各式各样的问题。

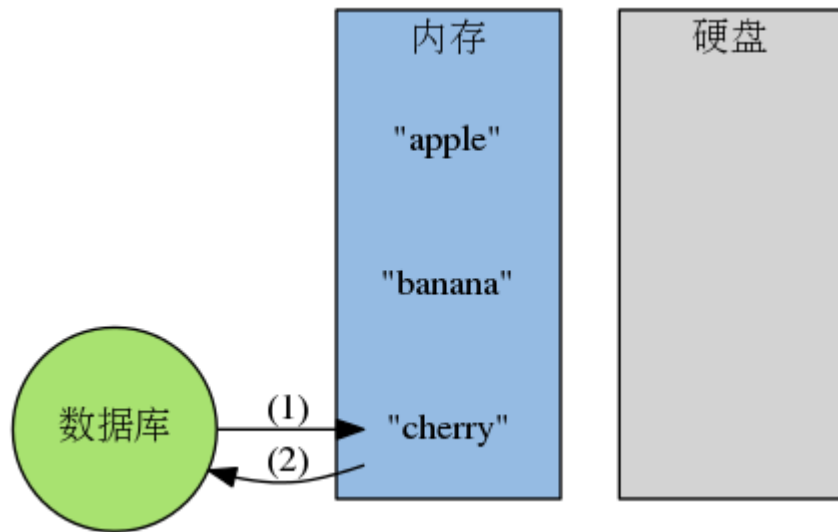


内存储存, 速度极快

Redis 将数据储存在内存里面, 读写数据的时候都不会受到硬盘 I/O 速度的限制, 所以速度极快。



硬盘数据库的工作模式。



内存数据库数据库的工作模式。



丰富的附加功能

持久化功能:将储存在内存里面的数据保存到硬盘里面,保障数据安全,方便进行数据备份和恢复。

发布与订阅功能:将消息同时分发给多个客户端,用于构建广播系统。

过期键功能:为键设置一个过期时间,让它在指定的时间之后自动被删除。

事务功能:原子地执行多个操作,并提供乐观锁功能,保证处理数据时的安全性。

脚本功能:在服务器端原子地执行多个操作,完成复杂的功能,并减少客户端与服务器之间的通信往返次数。

复制:为指定的 Redis 服务器创建一个或多个复制品,用于提升数据安全性,并分担读请求的负载。


Sentinel:监控 Redis 服务器的状态,并在服务器发生故障时,进行自动故障转移。

集群:创建分布式数据库,每个服务器分别执行一部分写操作和读操作。



完善的文档

Redis 具有完善、易读的文档, 加上 Redis 本身功能的简单性, 就算是新手也可以轻松上手。

 Commands Clients Documentation Community Download Issues Support License			
All Keys Strings Hashes Lists Sets Sorted Sets HyperLogLog Pub/Sub Transactions Scripting Connection Server			
APPEND key value Append a value to a key	GETBIT key offset Returns the bit value at offset in the string value stored at key	PEXPIRE key milliseconds-timestamp Set the expiration for a key as a UNIX timestamp specified in milliseconds	SMEMBERS key Get all the members in a set
AUTH password Authenticate to the server	GETRANGE key start end Get a substring of the string stored at a key	PFADD key element [element ...] Adds the specified elements to the specified HyperLogLog.	SMOVE source destination member Move a member from one set to another
BGREWRITEAOF Asynchronously rewrite the append-only file	GETSET key value Set the string value of a key and return its old value	PFCOUNT key [key ...] Return the approximated cardinality of the set(s) observed by the HyperLogLog at key(s).	SORT key [BY pattern] [LIMIT offset count] [GET pattern [...]] Sort the elements in a list, set or sorted set
BGSAVE Asynchronously save the dataset to disk	HDEL key field [field ...] Delete one or more hash fields	PFMERGE destkey sourcekey [sourcekey ...] Merge N different HyperLogLogs into a single one.	SPOP key Remove and return a random member from a set
BITCOUNT key start end [start end ...] Count set bits in a string	HEXISTS key field Determine if a hash field exists	PING Ping the server	SRANDMEMBER key [count] Get one or multiple random members from a set
BITOP operation destkey key [key ...] Perform bitwise operations between strings	HGET key field Get the value of a hash field	PSETEX key milliseconds value Set the value and expiration in milliseconds of a key	SREM key member [member ...] Remove one or more members from a set
BITPOS key bit [start] [end] Find first bit set or clear in a string	HGETALL key Get all the fields and values in a hash	PSUBSCRIBE pattern [pattern ...] Listen for messages published to channels matching the given patterns	STRLEN key Get the length of the value stored in a key



antirez 非常勤奋, 在每个版本都会不断地增加有用的新功能:

- 2.6 新增脚本功能, 并为很多命令添加了多参数支持(比如 SADD、ZADD、等等);
- 2.8 添加了数据库通知功能, HyperLogLog 数据结构以及 SCAN 命令, 实现了部分重同步;
- 3.0 将推出稳定版的 Redis 集群, 另外还有更多新功能陆续开发中.....

Bug 一旦出现就会很快被修复, 齐全测试套件以及稳扎稳打的开发策略, 使得 Redis 非常健壮可靠。

有问题时, 在 Redis 的论坛上发贴, 或者到 Redis 的 GitHub 页面发 issue, 又或者直接和作者 antirez 联系, 通常都可以很快得到回应。

Pivotal 公司雇用 antirez 全力开发 Redis, 无后顾之忧; 这间公司也提供专门的 Redis 开发和维护咨询服务。

阿里云、百度云、Amazon、RedisLab 等公司都提供了基于 Redis 的应用服务。



广泛的使用

Twitter 使用 Redis 来储存用户时间线(user timeline)。

StackOverflow 使用 Redis 来进行缓存和消息分发。

Pinterest 使用 Redis 来构建关注模型(follow model)和兴趣图谱(interest graph)。

Flickr 使用 Redis 来构建队列。

Github 使用 Redis 作为持久化的键值对数据库, 并使用 Resque 来实现消息队列。

新浪微博使用 Redis 来实现计数器、反向索引、排行榜、消息队列, 并储存用户关系。

知乎使用 Redis 来进行计数、缓存、消息分发和任务调度。

在课程的后续内容中, 我们也会使用 Redis 来实现这里提到的一些功能。



对 Redis 的简单介绍就到这

接下来，让我们来学习如何使用 Redis 提供的各种数据结构。



Pivotal 公司对 antirez 的采访, 讲述了 antirez 接触和学习编程的契机、他的工作背景, 以及他 发明 Redis 的原因: <http://blog.gopivotal.com/pivotal/pivotal-people/pivotal-people-salvatore-sanfilippo-inventor-of-redis>

《[Redis in Action](#)》一书的《[forward](#)》也谈到了 antirez 发明 Redis 的细节。

