一、redis 数据结构使用场景

原来看过 redisbook 这本书，对 redis 的基本功能都已经熟悉了，从上周开始看 redis 的源码。目前目标是吃透 redis 的数据结构。我们都知道，在 redis 中一共有 5 种数据结构，那每种数据结构的使用场景都是什么呢？

String——字符串
Hash——字典
List——列表
Set——集合
Sorted Set——有序集合

下面我们就来简单说明一下它们各自的使用场景：

1. String——字符串

String 数据结构是简单的 key-value 类型，value 不仅可以是 String，也可以是数字（当数字类型用 Long 可以表示的时候 encoding 就是整型，其他都存储在 sdshdr 当做字符串）。使用 Strings 类型，可以完全实现目前 Memcached 的功能，并且效率更高。还可以享受 Redis 的定时持久化（可以选择 RDB 模式或者 AOF 模式），操作日志及 Replication 等功能。除了提供与 Memcached 一样的 get、set、incr、decr 等操作外，Redis 还提供了下面一些操作：

复制代码代码如下:

```
1.LEN niushuai：O(1)获取字符串长度
2.APPEND niushuai redis：往字符串 append 内容，而且采用智能分配内存（每次 2 倍）
3.设置和获取字符串的某一段内容
4.设置及获取字符串的某一位（bit）
5.批量设置一系列字符串的内容
6.原子计数器
7.GETSET 命令的妙用，请于清空旧值的同时设置一个新值，配合原子计数器使用
```

2. Hash——字典

在 Memcached 中，我们经常将一些结构化的信息打包成 hashmap，在客户端序列化后存储为一个字符串的值（一般是 JSON 格式），比如用户的昵称、年龄、性别、积分等。这时候在需要修改其中某一项时，通常需要将字符串（JSON）取出来，然后进行反序列化，修改某一项的值，再序列化成字符串（JSON）存储回去。简单修改一个属性就干这么多事情，消耗必定是很大的，也不适用于一些可能并发操作的场合（比如两个并发的操作都需要修改积分）。而 Redis 的 Hash 结构可以使你像在数据库中 Update 一个属性一样只修改某一项属性值。

复制代码代码如下:

```
存储、读取、修改用户属性
```

3. List——列表

List 说白了就是链表（redis 使用双端链表实现的 List），相信学过数据结构知识的人都应该能理解其结构。使用 List 结构，我们可以轻松地实现最新消息排行等功能（比如新浪微博的 TimeLine ）。List 的另一个应用就是消息队列，可以利用 List 的 *PUSH 操作，将

任务存在 List 中，然后工作线程再用 POP 操作将任务取出进行执行。Redis 还提供了操作 List 中某一段元素的 API，你可以直接查询，删除 List 中某一段的元素。

复制代码代码如下:

1.微博 TimeLine

2.消息队列

4. Set——集合

Set 就是一个集合，集合的概念就是一堆不重复值的组合。利用 Redis 提供的 Set 数据结构，可以存储一些集合性的数据。比如在微博应用中，可以将一个用户所有的关注人存在一个集合中，将其所有粉丝存在一个集合。因为 Redis 非常人性化的为集合提供了求交集、并集、差集等操作，那么就可以非常方便的实现如共同关注、共同喜好、二度好友等功能，对上面的所有集合操作，你还可以使用不同的命令选择将结果返回给客户端还是存集到一个新的集合中。

1.共同好友、二度好友

2.利用唯一性，可以统计访问网站的所有独立 IP

3.好友推荐的时候，根据 tag 求交集，大于某个 threshold 就可以推荐

5. Sorted Set——有序集合

和 Sets 相比，Sorted Sets 是将 Set 中的元素增加了一个权重参数 score，使得集合中的元素能够按 score 进行有序排列，比如一个存储全班同学成绩的 Sorted Sets，其集合 value 可以是同学的学号，而 score 就可以是其考试得分，这样在数据插入集合的时候，就已经进行了天然的排序。另外还可以用 Sorted Sets 来做带权重的队列，比如普通消息的 score 为 1，重要消息的 score 为 2，然后工作线程可以选择按 score 的倒序来获取工作任务。让重要的任务优先执行。

1.带有权重的元素，比如一个游戏的用户得分排行榜

2.比较复杂的数据结构，一般用到的场景不算太多

二、redis 其他功能使用场景

1. 订阅-发布系统

Pub/Sub 从字面上理解就是发布（Publish）与订阅（Subscribe），在 Redis 中，你可以设定对某一个 key 值进行消息发布及消息订阅，当一个 key 值上进行了消息发布后，所有订阅它的客户端都会收到相应的消息。这一功能最明显的用法就是用作实时消息系统，比如普通的即时聊天，群聊等功能。

2. 事务——Transactions

谁说 NoSQL 都不支持事务，虽然 Redis 的 Transactions 提供的并不是严格的 ACID 的事务（比如一串用 EXEC 提交执行的命令，在执行中服务器宕机，那么会有一部分命令执行了，剩下的没执行），但是这个 Transactions 还是提供了基本的命令打包执行的功能（在服务器不出问题的情况下，可以保证一连串的命令是顺序在一起执行的，中间有会有其它客户端命令插进来执行）。Redis 还提供了一个 Watch 功能，你可以对一个 key 进行 Watch，然后再执行 Transactions，在这过程中，如果这个 Watched 的值进行了修改，那么这个 Transactions 会发现并拒绝执行。

1. 使用 redis 有哪些好处？

(1) 速度快，因为数据存在内存中，类似于 HashMap，HashMap 的优势就是查找和操作的时间复杂度都是 O(1)

(2) 支持丰富数据类型，支持 string，list，set，sorted set，hash

(3) 支持事务，操作都是原子性，所谓的原子性就是对数据的更改要么全部执行，要么全部不执行

(4) 丰富的特性：可用于缓存，消息，按 key 设置过期时间，过期后将会自动删除

2. redis 相比 memcached 有哪些优势？

(1) memcached 所有的值均是简单的字符串，redis 作为其替代者，支持更为丰富的数据类型

(2) redis 的速度比 memcached 快很多

(3) redis 可以持久化其数据

3. redis 常见性能问题和解决方案：

(1) Master 最好不要做任何持久化工作，如 RDB 内存快照和 AOF 日志文件

(2) 如果数据比较重要，某个 Slave 开启 AOF 备份数据，策略设置为每秒同步一次

(3) 为了主从复制的速度和连接的稳定性，Master 和 Slave 最好在同一个局域网内

(4) 尽量避免在压力很大的主库上增加从库

(5) 主从复制不要用图状结构，用单向链表结构更为稳定，即：Master <- Slave1 <- Slave2 <- Slave3...

这样的结构方便解决单点故障问题，实现 Slave 对 Master 的替换。如果 Master 挂了，可以立刻启用 Slave1 做 Master，其他不变。

4. mySQL 里有 2000w 数据，redis 中只存 20w 的数据，如何保证 redis 中的数据都是热点数据

相关知识：redis 内存数据集大小上升到一定大小的时候，就会施行数据淘汰策略。redis 提供 6 种数据淘汰策略：

voltile-lru：从已设置过期时间的数据集（server.db[i].expires）中挑选最近最少使用的数据淘汰

volatile-ttl：从已设置过期时间的数据集（server.db[i].expires）中挑选将要过期的数据淘汰

volatile-random：从已设置过期时间的数据集（server.db[i].expires）中任意选择数据淘汰

allkeys-lru：从数据集（server.db[i].dict）中挑选最近最少使用的数据淘汰

allkeys-random：从数据集（server.db[i].dict）中任意选择数据淘汰

no-enviction（驱逐）：禁止驱逐数据

5. Memcache 与 Redis 的区别都有哪些？

1)、存储方式

Memecache 把数据全部存在内存之中 ,断电后会挂掉 ,数据不能超过内存大小。

Redis 有部份存在硬盘上，这样能保证数据的持久性。

2)、数据支持类型

Memcache 对数据类型支持相对简单。

Redis 有复杂的数据类型。

3)、使用底层模型不同

它们之间底层实现方式 以及与客户端之间通信的应用协议不一样。

Redis 直接自己构建了 VM 机制 ，因为一般的系统调用系统函数的话，会浪费一定的时间去移动和请求。

4），value 大小

redis 最大可以达到 1GB，而 memcache 只有 1MB

6. Redis 常见的性能问题都有哪些？如何解决？

1).Master 写内存快照，save 命令调度 rdbSave 函数，会阻塞主线程的工作，当快照比较大时对性能影响是非常大的，会间断性暂停服务，所以 Master 最好不要写内存快照。

2).Master AOF 持久化，如果不重写 AOF 文件，这个持久化方式对性能的影响是最小的，但是 AOF 文件会不断增大，AOF 文件过大会影响 Master 重启的恢

复速度。Master 最好不要做任何持久化工作，包括内存快照和 AOF 日志文件，特别是不要启用内存快照做持久化,如果数据比较关键，某个 Slave 开启 AOF 备份数据，策略为每秒同步一次。

3).Master 调用 BGREWRITEAOF 重写 AOF 文件，AOF 在重写的时候会占大量的 CPU 和内存资源，导致服务 load 过高，出现短暂服务暂停现象。

4). Redis 主从复制的性能问题，为了主从复制的速度和连接的稳定性，Slave 和 Master 最好在同一个局域网内

7，redis 最适合的场景

Redis 最适合所有数据 in-momory 的场景，虽然 Redis 也提供持久化功能，但实际更多的是一个 disk-backed 的功能，跟传统意义上的持久化有比较大的差别，那么可能大家就会有疑问，似乎 Redis 更像一个加强版的 Memcached，那么何时使用 Memcached,何时使用 Redis 呢?

如果简单地比较 Redis 与 Memcached 的区别，大多数都会得到以下观点：

1 、Redis 不仅仅支持简单的 k/v 类型的数据，同时还提供 list，set，zset，hash 等数据结构的存储。

2 、Redis 支持数据的备份，即 master-slave 模式的数据备份。

3 、Redis 支持数据的持久化，可以将内存中的数据保持在磁盘中，重启的时候可以再次加载进行使用。

（1）、会话缓存（Session Cache）

最常用的一种使用 Redis 的情景是会话缓存（session cache）。用 Redis 缓存会话比其他存储（如 Memcached）的优势在于：Redis 提供持久化。当维护一个不是严格要求一致性的缓存时，如果用户的购物车信息全部丢失，大部分人都会不高兴的，现在，他们还会这样吗？

幸运的是，随着 Redis 这些年的改进，很容易找到怎么恰当的使用 Redis 来缓存会话的文档。甚至广为人知的商业平台 Magento 也提供 Redis 的插件。

（2）、全页缓存（FPC）

除基本的会话 token 之外，Redis 还提供很简便的 FPC 平台。回到一致性问题，即使重启了 Redis 实例，因为有磁盘的持久化，用户也不会看到页面加载速度的下降，这是一个极大改进，类似 PHP 本地 FPC。

再次以 Magento 为例，Magento 提供一个插件来使用 Redis 作为全页缓存后端。此外，对 WordPress 的用户来说，Pantheon 有一个非常好的插件 wp-redis，这个插件能帮助你以最快速度加载你曾浏览过的页面。

（3）、队列

Reids 在内存存储引擎领域的一大优点是提供 list 和 set 操作，这使得 Redis 能作为一个很好的消息队列平台来使用。Redis 作为队列使用的操作，就类似于本地程序语言（如 Python）对 list 的 push/pop 操作。

如果你快速的在 Google 中搜索“Redis queues”，你马上就能找到大量的开源项目，这些项目的目的就是利用 Redis 创建非常好的后端工具，以满足各种队列需求。例如，Celery 有一个后台就是使用 Redis 作为 broker，你可以从这里去查看。

（4），排行榜/计数器

Redis 在内存中对数字进行递增或递减的操作实现的非常好。集合（Set）和有序集合（Sorted Set）也使得我们在执行这些操作的时候变的非常简单，Redis 只是正好提供了这两种数据结构。所以，我们要从排序集合中获取到排名最靠前的 10 个用户–我们称之为“user_scores”，我们只需要像下面一样执行即可：

当然，这是假定你是根据你用户的分数做递增的排序。如果你想返回用户及用户的分数，你需要这样执行：

```
ZRANGE user_scores 0 10 WITHSCORES
```

Agora Games 就是一个很好的例子，用 Ruby 实现的，它的排行榜就是使用 Redis 来存储数据的，你可以在这里看到。

（5）、发布/订阅

最后（但肯定不是最不重要的）是 Redis 的发布/订阅功能。发布/订阅的使用场景确实非常多。我已看见人们在社交网络连接中使用，还可作为基于发布/订阅的脚本触发器，甚至用 Redis 的发布/订阅功能来建立聊天系统！（不，这是真的，你可以去核实）。

Redis 提供的所有特性中，我感觉这个是喜欢的人最少的一个，虽然它为用户提供如果此多功能。