## Redis为什么这么快

1、完全基于内存，绝大部分请求是纯粹的内存操作，非常快速。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、数据结构简单，对数据操作也简单，Redis中的数据结构是专门进行设计的；

3、采用单线程，避免了不必要的上下文切换和竞争条件，也不存在多进程或者多线程导致的切换而消耗 CPU，不用去考虑各种锁的问题，不存在加锁释放锁操作，没有因为可能出现死锁而导致的性能消耗；

4、使用多路I/O复用模型，非阻塞IO；

5、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；

以上几点都比较好理解，下边我们针对多路 I/O 复用模型进行简单的探讨：

（1）多路 I/O 复用模型

多路I/O复用模型是利用 select、poll、epoll 可以同时监察多个流的 I/O 事件的能力，在空闲的时候，会把当前线程阻塞掉，当有一个或多个流有 I/O 事件时，就从阻塞态中唤醒，于是程序就会轮询一遍所有的流（epoll 是只轮询那些真正发出了事件的流），并且只依次顺序的处理就绪的流，这种做法就避免了大量的无用操作。

**这里“多路”指的是多个网络连接，“复用”指的是复用同一个线程。**采用多路 I/O 复用技术可以让单个线程高效的处理多个连接请求（尽量减少网络 IO 的时间消耗），且 Redis 在内存中操作数据的速度非常快，也就是说内存内的操作不会成为影响Redis性能的瓶颈，主要由以上几点造就了 Redis 具有很高的吞吐量。

我们知道Redis是用”单线程-多路复用IO模型”来实现高性能的内存数据服务的，这种机制避免了使用锁，但是同时这种机制在进行sunion之类的比较耗时的命令时会使redis的并发下降。因为是单一线程，所以同一时刻只有一个操作在进行，所以，耗时的命令会导致并发的下降，不只是读并发，写并发也会下降。而单一线程也只能用到一个CPU核心，所以可以在同一个多核的服务器中，可以启动多个实例，组成master-master或者master-slave的形式，耗时的读命令可以完全在slave进行。

## SET

[SET key value](http://www.runoob.com/redis/strings-set.html)   
设置指定 key 的值

[GET key](http://www.runoob.com/redis/strings-get.html)   
获取指定 key 的值

[GETRANGE key start end](http://www.runoob.com/redis/strings-getrange.html)   
返回 key 中字符串值的子字符, GETRANGE mykey 0 -1

[MGET key1 [key2..]](http://www.runoob.com/redis/strings-mget.html)  
获取所有(一个或多个)给定 key 的值

[SETEX key seconds value](http://www.runoob.com/redis/strings-setex.html) ttl  
将值 value 关联到 key ，并将 key 的过期时间设为 seconds (以秒为单位)。

[SETNX key value](http://www.runoob.com/redis/strings-setnx.html)  
只有在 key 不存在时设置 key 的值。

[STRLEN key](http://www.runoob.com/redis/strings-strlen.html)  
返回 key 所储存的字符串值的长度，不存在的 key 长度为 0。

[MSET key value [key value ...]](http://www.runoob.com/redis/strings-mset.html)  
同时设置一个或多个 key-value 对。

[MSETNX key value [key value ...]](http://www.runoob.com/redis/strings-msetnx.html)   
同时设置一个或多个 key-value 对，当且仅当所有给定 key 都不存在。

当所有 key 都成功设置，返回 1 。 如果所有给定 key 都设置失败(至少有一个 key 已经存在)，那么返回 0 。MSET 是原子性操作。

[PSETEX key milliseconds value](http://www.runoob.com/redis/strings-psetex.html) pttl  
这个命令和 SETEX 命令相似，但它以毫秒为单位设置 key 的生存时间，而不是像 SETEX 命令那样，以秒为单位。

[INCR key](http://www.runoob.com/redis/strings-incr.html)  
将 key 中储存的数字值增一。如果 key 不存在，那么 key 的值会先被初始化为 0 ，然后再执行 INCR 操作。可以带符号。

[INCRBY key increment](http://www.runoob.com/redis/strings-incrby.html)  
将 key 所储存的值加上给定的增量值（increment） 。

[INCRBYFLOAT key increment](http://www.runoob.com/redis/strings-incrbyfloat.html)  
将 key 所储存的值加上给定的浮点增量值（increment） 。

[DECR key](http://www.runoob.com/redis/strings-decr.html)  
将 key 中储存的数字值减一。

[DECRBY key decrement](http://www.runoob.com/redis/strings-decrby.html)  
key 所储存的值减去给定的减量值（decrement） 。

[APPEND key value](http://www.runoob.com/redis/strings-append.html)  
如果 key 已经存在并且是一个字符串， APPEND 命令将指定的 value 追加到该 key 原来值（value）的末尾。

## HSAH

是一个string类型的field和value的映射表，hash特别适合用于存储对象。

这里的field叫‘域’或者‘字段’。

[HGET key field](http://www.runoob.com/redis/hashes-hget.html)   
获取存储在哈希表中指定字段的值。

[HMGET key field1 [field2]](http://www.runoob.com/redis/hashes-hmget.html)   
获取所有给定字段的值

[HSET key field value](http://www.runoob.com/redis/hashes-hset.html)   
将哈希表 key 中的字段 field 的值设为 value 。

[HMSET key field1 value1 [field2 value2 ]](http://www.runoob.com/redis/hashes-hmset.html)   
同时将多个 field-value (域-值)对设置到哈希表 key 中。

[HSETNX key field value](http://www.runoob.com/redis/hashes-hsetnx.html)   
只有在字段 field 不存在时，设置哈希表字段的值。

[HEXISTS key field](http://www.runoob.com/redis/hashes-hexists.html)   
查看哈希表 key 中，指定的字段是否存在。如果哈希表含有给定字段，返回 1 。 如果哈希表不含有给定字段，或 key 不存在，返回 0 。

[HDEL key field1 [field2]](http://www.runoob.com/redis/hashes-hdel.html)   
删除一个或多个哈希表字段，删除哈希表 key 中的一个或多个指定字段，不存在的字段将被忽略。

[HGETALL key](http://www.runoob.com/redis/hashes-hgetall.html)   
获取在哈希表中指定 key 的所有域和值。在返回值里，紧跟每个字段名(field name)之后是字段的值(value)，所以返回值的长度是哈希表大小的两倍。

[HKEYS key](http://www.runoob.com/redis/hashes-hkeys.html)   
根据key，获取哈希表中的所有域（field）。

[HLEN key](http://www.runoob.com/redis/hashes-hlen.html)   
根据key，获取哈希表中字段（field）的数量

[HINCRBY key field increment](http://www.runoob.com/redis/hashes-hincrby.html)   
为哈希表 key 中的指定字段的整数值加上增量 increment 。如果key和field都不存在，则创建并赋值。

[HINCRBYFLOAT key field increment](http://www.runoob.com/redis/hashes-hincrbyfloat.html)   
为哈希表 key 中的指定字段的浮点数值加上增量 increment 。

[HVALS key](http://www.runoob.com/redis/hashes-hvals.html)   
根据key获取哈希表中所有field对应的value

## LIST

简单的字符串列表，按照插入顺序排序。你可以添加一个元素到列表的头部（左边）或者尾部（右边）

L是头部，也是第一个

R是尾部，也是最后一个

[LRANGE key start stop](http://www.runoob.com/redis/lists-lrange.html)   
返回列表中指定区间内的元素，区间以偏移量 START 和 END 指定。 其中 0 表示列表的第一个元素， 1 表示列表的第二个元素，以此类推。 你也可以使用负数下标，以 -1 表示列表的最后一个元素， -2 表示列表的倒数第二个元素，以此类推。

[LREM key count value](http://www.runoob.com/redis/lists-lrem.html)   
移除列表元素

[LPUSH key value1 [value2]](http://www.runoob.com/redis/lists-lpush.html)   
将一个或多个值插入到列表头部。 如果 key 不存在，一个空列表会被创建并执行 LPUSH 操作。 当 key 存在但不是列表类型时，返回一个错误。

lpush letter aa bb cc dd ee ff

返回的顺序为“ff,ee,dd,cc,bb,aa”

[RPUSH key value1 [value2]](http://www.runoob.com/redis/lists-rpush.html)   
将一个或多个值插入到列表的尾部(最右边)。如果列表不存在，一个空列表会被创建并执行 RPUSH 操作。 当列表存在但不是列表类型时，返回一个错误。

[LPOP key](http://www.runoob.com/redis/lists-lpop.html)   
移除并返回列表的第一个元素。

[RPOP key](http://www.runoob.com/redis/lists-rpop.html)   
移除列表的最后一个元素，返回值为移除的元素。

[LPUSHX key value](http://www.runoob.com/redis/lists-lpushx.html)   
将一个值插入到已存在的列表头部，列表不存在时操作无效。

[RPUSHX key value](http://www.runoob.com/redis/lists-rpushx.html)   
将一个值插入到已存在的列表尾部(最右边)。如果列表不存在，操作无效。

[LLEN key](http://www.runoob.com/redis/lists-llen.html)   
用于返回列表的长度。 如果列表 key 不存在，则 key 被解释为一个空列表，返回 0 。 如果 key 不是列表类型，返回一个错误。

[LINDEX key index](http://www.runoob.com/redis/lists-lindex.html)   
通过索引获取列表中的元素

[LSET key index value](http://www.runoob.com/redis/lists-lset.html)   
通过索引设置列表元素的值，覆盖原来的值，当索引参数超出范围，或对一个空列表进行 LSET 时，返回一个错误。

[LREM key count value](http://www.runoob.com/redis/lists-lrem.html)   
根据参数 COUNT 的值，移除列表中与参数 VALUE 相等的元素。

COUNT 的值可以是以下几种：

count > 0 : 从表头开始向表尾搜索，移除与 VALUE 相等的元素，数量为 COUNT 。

count < 0 : 从表尾开始向表头搜索，移除与 VALUE 相等的元素，数量为 COUNT 的绝对值。

count = 0 : 移除表中所有与 VALUE 相等的值。

[LINSERT key BEFORE|AFTER pivot value](http://www.runoob.com/redis/lists-linsert.html)   
在列表的元素前或者后插入元素，在列表的元素前或者后插入元素。当指定元素不存在于列表中时，不执行任何操作。如果 key 不是列表类型，返回一个错误。

[BLPOP key1 [key2 ] timeout](http://www.runoob.com/redis/lists-blpop.html)   
移出并获取列表的第一个元素， 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。单位是秒（s）

[BRPOP key1 [key2 ] timeout](http://www.runoob.com/redis/lists-brpop.html)   
移出并获取列表的最后一个元素， 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。单位是秒（s）

[BRPOPLPUSH source destination timeout](http://www.runoob.com/redis/lists-brpoplpush.html)   
从列表中弹出最后一个值，将弹出的元素插入到另外一个列表中并返回它； 如果列表没有元素会阻塞列表直到等待超时或发现可弹出元素为止。

[RPOPLPUSH source destination](http://www.runoob.com/redis/lists-rpoplpush.html)   
移除列表的最后一个元素，并将该元素添加到另一个列表头部，并返回该值。

## SET

String 类型的无序集合。集合成员是唯一的，这就意味着集合中不能出现重复的数据。

集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是 O(1)。

集合中最大的成员数为 232 - 1 (4294967295, 每个集合可存储40多亿个成员)。

[SADD key member1 [member2]](http://www.runoob.com/redis/sets-sadd.html)   
向集合添加一个或多个成员，已经存在于集合的成员元素将被忽略。当集合 key 不是集合类型时，返回一个错误。

[SCARD key](http://www.runoob.com/redis/sets-scard.html)   
获取集合的元素总数。

[SDIFF key1 [key2]](http://www.runoob.com/redis/sets-sdiff.html)   
返回给定集合之间的差集。不存在的集合 key 将视为空集。差集的结果来自前面的 key1,而不是后面的 key2。

[SDIFFSTORE destination key1 [key2]](http://www.runoob.com/redis/sets-sdiffstore.html)   
返回给定所有集合的差集并存储在 destination 中

[SINTER key1 [key2]](http://www.runoob.com/redis/sets-sinter.html)   
返回给定所有给定集合的交集。 不存在的集合 key 被视为空集。 当给定集合当中有一个空集时，结果也为空集

[SINTERSTORE destination key1 [key2]](http://www.runoob.com/redis/sets-sinterstore.html)   
返回给定所有集合的交集并存储在 destination 中，如果指定的集合已经存在，则将其覆盖，是覆盖所有元素，不是覆盖相同元素。

[SISMEMBER key member](http://www.runoob.com/redis/sets-sismember.html)   
判断 member 元素是否是集合 key 的成员。如果成员元素是集合的成员，返回 1 。 如果成员元素不是集合的成员，或 key 不存在，返回 0 。

[SMEMBERS key](http://www.runoob.com/redis/sets-smembers.html)   
返回集合中的所有成员

[SMOVE source destination member](http://www.runoob.com/redis/sets-smove.html)   
将 member 元素从 source 集合移动到 destination 集合。

SMOVE 是原子性操作。如果 source 集合不存在或不包含指定的 member 元素，则 SMOVE 命令不执行任何操作，仅返回 0 。否则， member 元素从 source 集合中被移除，并添加到 destination 集合中去。

当 destination 集合已经包含 member 元素时， SMOVE 命令只是简单地将 source 集合中的 member 元素删除。

当 source 或 destination 不是集合类型时，返回一个错误。

[SPOP key](http://www.runoob.com/redis/sets-spop.html)   
移除并返回集合中的一个随机元素。该命令类似 [Srandmember](http://www.runoob.com/redis/sets-srandmember.html) 命令，但 SPOP 将随机元素从集合中移除并返回，而 Srandmember 则仅仅返回随机元素，而不对集合进行任何改动。

[SRANDMEMBER key [count]](http://www.runoob.com/redis/sets-srandmember.html)   
返回集合中一个或多个随机数。

从 Redis 2.6 版本开始， Srandmember 命令接受可选的 count 参数：

如果 count 为正数，且小于集合基数，那么命令返回一个包含 count 个元素的数组，数组中的元素各不相同。如果 count 大于等于集合基数，那么返回整个集合。

如果 count 为负数，那么命令返回一个数组，数组中的元素可能会重复出现多次，而数组的长度为 count 的绝对值。

[SREM key member1 [member2]](http://www.runoob.com/redis/sets-srem.html)   
移除集合中一个或多个成员。当 key 不是集合类型，返回一个错误。

[SUNION key1 [key2]](http://www.runoob.com/redis/sets-sunion.html)   
返回所有给定集合的并集

[SUNIONSTORE destination key1 [key2]](http://www.runoob.com/redis/sets-sunionstore.html)   
所有给定集合的并集存储在 destination 集合中。如果 destination 已经存在，则将其覆盖。

[SSCAN key cursor [MATCH pattern] [COUNT count]](http://www.runoob.com/redis/sets-sscan.html)   
迭代集合中的元素

## ZSET

有序集合和集合一样也是string类型元素的集合,且不允许重复的成员。

不同的是每个元素都会关联一个double类型的分数。redis正是通过分数来为集合中的成员进行从小到大的排序。

有序集合的成员是唯一的,但分数(score)却可以重复。

集合是通过哈希表实现的，所以添加，删除，查找的复杂度都是O(1)。 集合中最大的成员数为 232 - 1 (4294967295, 每个集合可存储40多亿个成员)。

[ZADD key score1 member1 [score2 member2]](http://www.runoob.com/redis/sorted-sets-zadd.html)   
向有序集合添加一个或多个成员，或者更新已存在成员的分数。分数值可以是整数值或双精度浮点数。

[ZCOUNT key min max](http://www.runoob.com/redis/sorted-sets-zcount.html)   
计算有序集合中指定分数区间的成员数。

[ZCARD key](http://www.runoob.com/redis/sorted-sets-zcard.html)   
获取有序集合的成员数

[ZINCRBY key increment member](http://www.runoob.com/redis/sorted-sets-zincrby.html)   
有序集合中对指定成员的分数加上增量 increment。当 key 不存在，或分数不是 key 的成员时， ZINCRBY key increment member 等同于 ZADD key increment member 。分数值可以是整数值或双精度浮点数。

[ZRANGE key start stop [WITHSCORES]](http://www.runoob.com/redis/sorted-sets-zrange.html)   
返回有序集合中指定索引区间的成员，分数从低到高，索引从0 开始

[ZREVRANGE key start stop [WITHSCORES]](http://www.runoob.com/redis/sorted-sets-zrevrange.html)   
返回有序集合中指定索引区间的成员，分数从高到底，索引从0 开始

[ZRANGEBYSCORE key min max [WITHSCORES] [LIMIT]](http://www.runoob.com/redis/sorted-sets-zrangebyscore.html)   
返回有序集合中指定分数区间内的成员，分数从低到高

[ZREVRANGEBYSCORE key max min [WITHSCORES]](http://www.runoob.com/redis/sorted-sets-zrevrangebyscore.html)   
返回有序集合中指定分数区间内的成员，分数从高到低

[ZRANK key member](http://www.runoob.com/redis/sorted-sets-zrank.html)   
返回有序集合中指定成员的排名。从小到大排第几名。第一名是0。

[ZREVRANK key member](http://www.runoob.com/redis/sorted-sets-zrevrank.html)   
返回有序集合中指定成员的排名。从大到小排第几名。第一名是0。

[ZSCORE key member](http://www.runoob.com/redis/sorted-sets-zscore.html)   
返回有序集合中指定成员的分数值。

[ZREM key member [member ...]](http://www.runoob.com/redis/sorted-sets-zrem.html)   
移除有序集合中的一个或多个成员

[ZREMRANGEBYRANK key start stop](http://www.runoob.com/redis/sorted-sets-zremrangebyrank.html)   
移除有序集合中给定的排名区间的所有成员

[ZREMRANGEBYSCORE key min max](http://www.runoob.com/redis/sorted-sets-zremrangebyscore.html)   
移除有序集合中给定的分数区间的所有成员

[ZINTERSTORE destination numkeys key [key ...]](http://www.runoob.com/redis/sorted-sets-zinterstore.html)   
所谓交集是member相同，所谓合并是score相加，所谓numkeys是集合个数

[ZUNIONSTORE destination numkeys key [key ...]](http://www.runoob.com/redis/sorted-sets-zunionstore.html)   
计算给定的一个或多个有序集的并集，并存储在新的 key 中

[ZSCAN key cursor [MATCH pattern] [COUNT count]](http://www.runoob.com/redis/sorted-sets-zscan.html)   
迭代有序集合中的元素（包括元素成员和元素分值）

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **String** | **Hash** | **List** | **Set** | **Zset** |
| [set](http://redisdoc.com/string/set.html) | [hset](http://redisdoc.com/hash/hset.html) | [lpush](http://redisdoc.com/list/lpush.html) | [sadd](http://redisdoc.com/set/sadd.html) | [zadd](http://redisdoc.com/sorted_set/zadd.html) |
| [setnx](http://redisdoc.com/string/setnx.html) | [hsetnx](http://redisdoc.com/hash/hsetnx.html) | [lpushx](http://redisdoc.com/list/lpushx.html) | [sismember](http://redisdoc.com/set/sismember.html) | [zscore](http://redisdoc.com/sorted_set/zscore.html) |
| [setex](http://redisdoc.com/string/setex.html) | [hget](http://redisdoc.com/hash/hget.html) | [rpush](http://redisdoc.com/list/rpush.html) | [spop](http://redisdoc.com/set/spop.html) | [zincrby](http://redisdoc.com/sorted_set/zincrby.html) |
| [psetex](http://redisdoc.com/string/psetex.html) | [hexists](http://redisdoc.com/hash/hexists.html) | [rpushx](http://redisdoc.com/list/rpushx.html) | [srandmember](http://redisdoc.com/set/srandmember.html) | [zcard](http://redisdoc.com/sorted_set/zcard.html) |
| [get](http://redisdoc.com/string/get.html) | [hdel](http://redisdoc.com/hash/hdel.html) | [lpop](http://redisdoc.com/list/lpop.html) | [srem](http://redisdoc.com/set/srem.html) | [zcount](http://redisdoc.com/sorted_set/zcount.html) |
| [getset](http://redisdoc.com/string/getset.html) | [hlen](http://redisdoc.com/hash/hlen.html) | [rpop](http://redisdoc.com/list/rpop.html) | [smove](http://redisdoc.com/set/smove.html) | [zrange](http://redisdoc.com/sorted_set/zrange.html) |
| [strlen](http://redisdoc.com/string/strlen.html) | [hstrlen](http://redisdoc.com/hash/hstrlen.html) | [rpoplpush](http://redisdoc.com/list/rpoplpush.html) | [scard](http://redisdoc.com/set/scard.html) | [zrevrange](http://redisdoc.com/sorted_set/zrevrange.html) |
| [append](http://redisdoc.com/string/append.html) | [hincrby](http://redisdoc.com/hash/hincrby.html) | [lrem](http://redisdoc.com/list/lrem.html) | [smembers](http://redisdoc.com/set/smembers.html) | [zrangebyscore](http://redisdoc.com/sorted_set/zrangebyscore.html) |
| [setrange](http://redisdoc.com/string/setrange.html) | [hincrbyfloat](http://redisdoc.com/hash/hincrbyfloat.html) | [llen](http://redisdoc.com/list/llen.html) | [sscan](http://redisdoc.com/set/sscan.html) | [zrevrangebyscore](http://redisdoc.com/sorted_set/zrevrangebyscore.html) |
| [getrange](http://redisdoc.com/string/getrange.html) | [hmset](http://redisdoc.com/hash/hmset.html) | [lindex](http://redisdoc.com/list/lindex.html) | [sinter](http://redisdoc.com/set/sinter.html) | [zrank](http://redisdoc.com/sorted_set/zrank.html) |
| [incr](http://redisdoc.com/string/incr.html) | [hmget](http://redisdoc.com/hash/hmget.html) | [linsert](http://redisdoc.com/list/linsert.html) | [sinterstore](http://redisdoc.com/set/sinterstore.html) | [zrevrank](http://redisdoc.com/sorted_set/zrevrank.html) |
| [incrby](http://redisdoc.com/string/incrby.html) | [hkeys](http://redisdoc.com/hash/hkeys.html) | [lset](http://redisdoc.com/list/lset.html) | [sunion](http://redisdoc.com/set/sunion.html) | [zrem](http://redisdoc.com/sorted_set/zrem.html) |
| [incrbyfloat](http://redisdoc.com/string/incrbyfloat.html) | [hvals](http://redisdoc.com/hash/hvals.html) | [lrange](http://redisdoc.com/list/lrange.html) | [sunionstore](http://redisdoc.com/set/sunionstore.html) | [zremrangebyrank](http://redisdoc.com/sorted_set/zremrangebyrank.html) |
| [decr](http://redisdoc.com/string/decr.html) | [hgetall](http://redisdoc.com/hash/hgetall.html) | [ltrim](http://redisdoc.com/list/ltrim.html) | [sdiff](http://redisdoc.com/set/sdiff.html) | [zremrangebyscore](http://redisdoc.com/sorted_set/zremrangebyscore.html) |
| [decrby](http://redisdoc.com/string/decrby.html) | [hscan](http://redisdoc.com/hash/hscan.html) | [blpop](http://redisdoc.com/list/blpop.html) | [sdiffstore](http://redisdoc.com/set/sdiffstore.html) | [zscan](http://redisdoc.com/sorted_set/zscan.html) |
| [mset](http://redisdoc.com/string/mset.html) |  | [brpop](http://redisdoc.com/list/brpop.html) |  | [zunionstore](http://redisdoc.com/sorted_set/zunionstore.html) |
| [msetnx](http://redisdoc.com/string/msetnx.html) |  | [brpoplpush](http://redisdoc.com/list/brpoplpush.html) |  | [zinterstore](http://redisdoc.com/sorted_set/zinterstore.html) |
| [mget](http://redisdoc.com/string/mget.html) |  |  |  |  |