Redis 中文入门手册 1.0

中国海事服务网(www.cnss.com.cn)

zhangli 收集整理

目录

1）Redis 简介

Redis 是一个开源的使用 ANSI C 语言编写、支持网络、可基于内存亦可持久化的日志型、Key-Value 数据库。

2）数据类型

2.1. Redis 的 Key

Redis 的 key 是字符串类型，但是 key 中不能包括边界字符，由于 key 不是 binary safe 的字符串，所以像"my key"和"mykey\n"这样包含空格和换行的 key 是不允许的。

2.1.1. key 相关指令介绍

exits key 检测指定 key 是否存在，返回 1 表示存在，0 不存在
del key1 key2 ..... keyN 删除给定 key，返回删除 key 的数目，0 表示给定 key 都不存在
type key 返回给定 key 值的类型。返回 none 表示 key 不存在，string 字符类型，list 链表类型 set 无序集合类型.....
keys pattern 返回匹配指定模式的所有 key
randomkey 返回从当前数据库中随机选择的一个 key，如果当前数据库是空的，返回空串
rename oldkey newkey 重命名一个 key，如果 newkey 存在，将会被覆盖，返回 1 表示成功，0 失败。可能是 oldkey 不存在或者和 newkey 相同。
renamenx oldkey newkey 同上，但是如果 newkey 存在返回失败。
expire key seconds 为 key 指定过期时间，单位是秒。返回 1 成功，0 表示 key 已经设置过过期时间或者不存在。
ttl key 返回设置过过期时间 key 的剩余过期秒数。-1 表示 key 不存在或者未设置过期时间。
select db-index 通过索引选择数据库，默认连接的数据库是 0，默认数据库数是 16 个。返回 1 表示成功，0 失败。
move key db-index 将 key 从当前数据库移动到指定数据库。返回 1 表示成功。0 表示 key 不存在或者已经在指定数据库中。

2.2. Redis 的 vaule

redis 提供五种数据类型：string, hash, list, set 及 sorted set。

2.2.1. string 类型

string 是最基本的类型，而且 string 类型是二进制安全的。意思是 redis 的 string 可以包含任何数据。比如 jpg 图片或者序列化的对象。从内部实现来看其实 string 可以看作 byte

数组，最大上限是 1G 字节。

string 类型数据操作指令简介

set key value 设置 key 对应 string 类型的值，返回 1 表示成功，0 失败。
setnx key value 如果 key 不存在，设置 key 对应 string 类型的值。如果 key 已经存在，返回 0。
get key 获取 key 对应的 string 值,如果 key 不存在返回 nil
getset key value 先获取 key 的值，再设置 key 的值。如果 key 不存在返回 nil。
mget key1 key2 ..... keyN 一次获取多个 key 的值，如果对应 key 不存在，则对应返回 nil。
mset key1 value1 ..... keyN valueN 一次设置多个 key 的值，成功返回 1 表示所有的值都设置了，失败返回 0 表示没有任何值被设置。
msetnx key1 value1 ..... keyN valueN 一次设置多个 key 的值，但是不会覆盖已经存在的 key
incr key 对 key 的值做++操作，并返回新的值。注意 incr 一个不是 int 的 value 会返回错误，incr 一个不存在的 key，则设置 key 值为 1。
decr key 对 key 的值做--操作，decr 一个不存在 key，则设置 key 值为-1。
incrby key integer 对 key 加上指定值 ，key 不存在时候会设置 key，并认为原来的 value 是 0。
decrby key integer 对 key 减去指定值。decrby 完全是为了可读性,我们完全可以通过 incrby 一个负值来实现同样效果，反之一样。

2.2.2. hash 类型

hash 是一个 string 类型的 field 和 value 的映射表。添加,删除操作都是 O(1)(平均)。hash 特别适合用于存储对象。相对于将对象的每个字段存成单个 string 类型。将一个对象存储在 hash 类型中会占用更少的内存，并且可以更方便的存取整个对象。省内存的原因是新建一个 hash 对象时开始是用 zipmap（又称为 small hash）来存储的。这个 zipmap 其实并不是 hash table，但是 zipmap 相比正常的 hash 实现可以节省不少 hash 本身需要的一些元数据存储开销。尽管 zipmap 的添加，删除，查找都是 O(n)，但是由于一般对象的 field 数量都不太多。所以使用 zipmap 也是很快的,也就是说添加删除平均还是 O(1)。如果 field 或者 value 的大小超出一定限制后,redis 会在内部自动将 zipmap 替换成正常的 hash 实现. 这个限制可以在配置文件中指定。

```
hash-max-zipmap-entries 64        #配置字段最多 64 个
hash-max-zipmap-value 512         #配置 value 最大为 512 字节
```

hash 类型数据操作指令简介

hset key field value 设置 hash field 为指定值，如果 key 不存在，则创建
hget key field 获取指定的 hash field。
hmget key filed1....fieldN 获取全部指定的 hash filed。
hmset key filed1 value1 ..... filedN valueN 同时设置 hash 的多个 field。
hincrby key field integer 将指定的 hash filed 加上指定值。成功返回 hash filed 变更后的值。
hexists key field 检测指定 field 是否存在。
hdel key field 删除指定的 hash field。
hlen key 返回指定 hash 的 field 数量。

hkeys key 返回 hash 的所有 field。
hvals key 返回 hash 的所有 value。
hgetall 返回 hash 的所有 filed 和 value

2.2.3. list 类型

list 是一个链表结构，可以理解为一个每个子元素都是 string 类型的双向链表。主要功能是 push、pop、获取一个范围的所有值等。操作中 key 理解为链表的名字。

List 类型数据操作指令简介
lpush key string 在 key 对应 list 的头部添加字符串元素，返回 1 表示成功，0 表示 key 存在且不是 list 类型。
rpush key string 在 key 对应 list 的尾部添加字符串元素。
llen key 返回 key 对应 list 的长度，如果 key 不存在返回 0，如果 key 对应类型不是 list 返回错误。
lrange key start end 返回指定区间内的元素，下标从 0 开始，负值表示从后面计算，-1 表示倒数第一个元素 ，key 不存在返回空列表。
ltrim key start end 截取 list 指定区间内元素，成功返回 1，key 不存在返回错误。
lset key index value 设置 list 中指定下标的元素值，成功返回 1，key 或者下标不存在返回错误。
lrem key count value 从 List 的头部(count 正数)或尾部(count 负数)删除一定数量(count)匹配 value 的元素，返回删除的元素数量。count 为 0 时候删除全部。
lpop key 从 list 的头部删除并返回删除元素。如果 key 对应 list 不存在或者是空返回 nil，如果 key 对应值不是 list 返回错误。
rpop key 从 list 的尾部删除并返回删除元素。
blpop key1 ..... keyN timeout 从左到右扫描，返回对第一个非空 list 进行 lpop 操作并返回，比如 blpop list1 list2 list3 0 ，如果 list 不存在 list2，list3 都是非空则对 list2 做 lpop 并返回从 list2 中删除的元素。如果所有的 list 都是空或不存在，则会阻塞 timeout 秒，timeout 为 0 表示一直阻塞。当阻塞时，如果有 client 对 key1...keyN 中的任意 key 进行 push 操作，则第一在这个 key 上被阻塞的 client 会立即返回。如果超时发生，则返回 nil。有点像 unix 的 select 或者 poll。
brpop 同 blpop，一个是从头部删除一个是从尾部删除。

2.2.4. set 类型

set 是无序集合，最大可以包含(2 的 32 次方-1)个元素。set 的是通过 hash table 实现的，所以添加，删除，查找的复杂度都是 O(1)。hash table 会随着添加或者删除自动的调整大小。需要注意的是调整 hash table 大小时候需要同步（获取写锁）会阻塞其他读写操作。可能不久后就会改用跳表（skip list）来实现。跳表已经在 sorted sets 中使用了。关于 set 集合类型除了基本的添加删除操作，其它有用的操作还包含集合的取并集(union)，交集(intersection)，差集(difference)。通过这些操作可以很容易的实现 SNS 中的好友推荐和 blog 的 tag 功能。

set 类型数据操作指令简介

sadd key member 添加一个 string 元素到 key 对应 set 集合中，成功返回 1,如果元素以及在集合中则返回 0，key 对应的 set 不存在则返回错误。

srem key member 从 key 对应 set 中移除指定元素，成功返回 1，如果 member 在集合中不存在或者 key 不存在返回 0，如果 key 对应的不是 set 类型的值返回错误。

spop key 删除并返回 key 对应 set 中随机的一个元素,如果 set 是空或者 key 不存在返回 nil。

srandmember key 同 spop，随机取 set 中的一个元素，但是不删除元素。

smove srckey dstkey member 从 srckey 对应 set 中移除 member 并添加到 dstkey 对应 set 中，整个操作是原子的。成功返回 1,如果 member 在 srckey 中不存在返回 0，如果 key 不是 set 类型返回错误。

scard key 返回 set 的元素个数，如果 set 是空或者 key 不存在返回 0。

sismember key member 判断 member 是否在 set 中，存在返回 1，0 表示不存在或者 key 不存在。

sinter key1 key2 …… keyN 返回所有给定 key 的交集。

sinterstore dstkey key1 ..... keyN 返回所有给定 key 的交集，并保存交集存到 dstkey 下。

sunion key1 key2 ..... keyN 返回所有给定 key 的并集。

sunionstore dstkey key1 ..... keyN 返回所有给定 key 的并集，并保存并集到 dstkey 下。

sdiff key1 key2 ..... keyN 返回所有给定 key 的差集。

sdiffstore dstkey key1 ..... keyN 返回所有给定 key 的差集，并保存差集到 dstkey 下。

smembers key 返回 key 对应 set 的所有元素，结果是无序的。

2.2.5. sorted set 类型

sorted set 是有序集合，它在 set 的基础上增加了一个顺序属性，这一属性在添加修改元素的时候可以指定，每次指定后，会自动重新按新的值调整顺序。可以理解了有两列的 mysql 表，一列存 value，一列存顺序。操作中 key 理解为 sorted set 的名字。

Sorted Set 类型数据操作指令简介

add key score member 添加元素到集合，元素在集合中存在则更新对应 score。

zrem key member 删除指定元素，1 表示成功，如果元素不存在返回 0。

zincrby key incr member 增加对应 member 的 score 值，然后移动元素并保持 skip list 保持有序。返回更新后的 score 值。

zrank key member 返回指定元素在集合中的排名（下标），集合中元素是按 score 从小到大排序的。

zrevrank key member 同上,但是集合中元素是按 score 从大到小排序。

zrange key start end 类似 lrange 操作从集合中去指定区间的元素。返回的是有序结果

zrevrange key start end 同上，返回结果是按 score 逆序的。

zrangebyscore key min max 返回集合中 score 在给定区间的元素。

zcount key min max 返回集合中 score 在给定区间的数量。

zcard key 返回集合中元素个数。

zscore key element 返回给定元素对应的 score。

zremrangebyrank key min max 删除集合中排名在给定区间的元素。
zremrangebyscore key min max 删除集合中 score 在给定区间的元素

3）持久化

通常 Redis 将数据存储在内存中或虚拟内存中，它是通过以下两种方式实现对数据的持久化。

3.1. 快照方式：（默认持久化方式）

这种方式就是将内存中数据以快照的方式写入到二进制文件中，默认的文件名为 dump.rdb。

客户端也可以使用 save 或者 bgsave 命令通知 redis 做一次快照持久化。save 操作是在主线程中保存快照的，由于 redis 是用一个主线程来处理所有客户端的请求，这种方式会阻塞所有客户端请求。所以不推荐使用。另一点需要注意的是，每次快照持久化都是将内存数据完整写入到磁盘一次，并不是增量的只同步增量数据。如果数据量大的话，写操作会比较多，必然会引起大量的磁盘 IO 操作，可能会严重影响性能。

注意：由于快照方式是在一定间隔时间做一次的，所以如果 redis 意外当机的话，就会丢失最后一次快照后的所有数据修改。

3.2. 日志追加方式：

这种方式 redis 会将每一个收到的写命令都通过 write 函数追加到文件中（默认 appendonly.aof）。当 redis 重启时会通过重新执行文件中保存的写命令来在内存中重建整个数据库的内容。当然由于操作系统会在内核中缓存 write 做的修改，所以可能不是立即写到磁盘上。这样的持久化还是有可能会丢失部分修改。不过我们可以通过配置文件告诉 redis 我们想要通过 fsync 函数强制操作系统写入到磁盘的时机。有三种方式如下（默认是：每秒 fsync 一次）

```
appendonly yes              //启用日志追加持久化方式
#appendfsync always         //每次收到写命令就立即强制写入磁盘，最慢的，但是保证完全的持久化，不推荐使用
appendfsync everysec        //每秒钟强制写入磁盘一次，在性能和持久化方面做了很好的折中，推荐
#appendfsync no             //完全依赖操作系统，性能最好，持久化没保证
```

日志追加方式同时带来了另一个问题。持久化文件会变的越来越大。例如我们调用 incr test 命令 100 次，文件中必须保存全部 100 条命令，其实有 99 条都是多余的。因为要恢复数据库状态其实文件中保存一条 set test 100 就够了。为了压缩这种持久化方式的日志文件。redis 提供了 **bgrewriteaof** 命令。收到此命令 redis 将使用与快照类似的方式将内存中的数据以命令的方式保存到临时文件中，最后替换原来的持久化日志文件。

4）虚拟内存（适用于 value 比 key 大的情况）

4.1. Redis 虚拟内存简介

首先说明下 redis 的虚拟内存与操作系统虚拟内存不是一码事，但是思路和目的都是相同的。就是暂时把不经常访问的数据从内存交换到磁盘中，从而腾出宝贵的内存空间。对于 redis 这样的内存数据库，内存总是不够用的。除了可以将数据分割到多个 redis 服务器以外。另外的能够提高数据库容量的办法就是使用虚拟内存技术把那些不经常访问的数据交换到磁盘上。如果我们存储的数据总是有少部分数据被经常访问，大部分数据很少被访问，对于网站来说确实总是只有少量用户经常活跃。当少量数据被经常访问时，使用虚拟内存不但能提高单台 redis 数据库服务器的容量，而且也不会对性能造成太多影响。

redis 没有使用操作系统提供的虚拟内存机制而是自己在用户态实现了自己的虚拟内存机制。

主要的理由有以下两点：

1. 操作系统的虚拟内存是以 4k/页为最小单位进行交换的。而 redis 的大多数对象都远小于 4k，所以一个操作系统页上可能有多个 redis 对象。另外 redis 的集合对象类型如 list,set 可能存在于多个操作系统页上。最终可能造成只有 10%的 key 被经常访问，但是所有操作系统页都会被操作系统认为是活跃的，这样只有内存真正耗尽时操作系统才会进行页的交换。
2. 相比操作系统的交换方式。redis 可以将被交换到磁盘的对象进行压缩,保存到磁盘的对象可以去除指针和对象元数据信息。一般压缩后的对象会比内存中的对象小 10 倍。这样 redis 的虚拟内存会比操作系统的虚拟内存少做很多 IO 操作。

4.2. Redis 虚拟内存相关配置

```
vm-enabled yes                          #开启虚拟内存功能
vm-swap-file /tmp/redis.swap            #交换出来 value 保存的文件路径/tmp/redis.swap
vm-max-memory 268435456                 #redis 使用的最大内存上限（256MB），超过上限后 redis 开始交换 value 到磁盘 swap 文件中。建议设置为系统空闲内存的 60%-80%
vm-page-size 32                         #每个 redis 页的大小 32 个字节
vm-pages 134217728                      #最多在文件中使用多少个页,交换文件的大小 = （vm-page-size * vm-pages）4GB
vm-max-threads 8                        #用于执行 value 对象换入换出的工作线程数量。0 表示不使用工作线程（详情后面介绍）
```

redis 的虚拟内存在设计上为了保证 key 的查询速度，只会将 value 交换到 swap 文件中。**如果是由于太多 key 很小的 value 造成的内存问题，那么 redis 的虚拟内存并不能解决问题。**和操作系统一样 redis 也是按页来交换对象的。redis 规定同一个页只能保存一个对象。但是一个对象可以保存在多个页中。在 redis 使用的内存没超过 vm-max-memory 之前是不会交换任何 value 的。当超过最大内存限制后，redis 会选择把较老的对象交换到 swap 文件中去。如果两个对象一样老会优先交换比较大的对象，精确的交换计算公式 swappability = age*log(size_in_memory)。 对于 vm-page-size 的设置应该根据自己应用将页的大小设置为可以容纳大多数对象的尺寸。太大了会浪费磁盘空间，太小了会造成交换

文件出现过多碎片。对于交换文件中的每个页，redis 会在内存中用一个 1bit 值来对应记录页的空闲状态。所以像上面配置中页数量(vm-pages 134217728)会占用 16MB 内存用来记录页的空闲状态。vm-max-threads 表示用做交换任务的工作线程数量。如果大于 0 推荐设为服务器的 cpu 的核心数。如果是 0 则交换过程在主线程进行。

4.3. redis 虚拟内存工作方式简介

4.3.1. 当 vm-max-threads 设为 0 时（阻塞方式）

换出

主线程定期检查发现内存超出最大上限后，会直接以阻塞的方式,将选中的对象保存到 swap 文件中，并释放对象占用的内存空间，此过程会一直重复直到下面条件满足

1.内存使用降到最大限制以下

2.swap 文件满了。

3.几乎全部的对象都被交换到磁盘了

换入

当有客户端请求已经被换出的 value 时，主线程会以阻塞的方式从 swap 文件中加载对应的 value 对象，加载时此时会阻塞所有客户端。然后处理该客户端的请求

4.3.2. 当 vm-max-threads 大于 0 时(工作线程方式)

换出

当主线程检测到使用内存超过最大上限,会将选中要交换的对象信息放到一个队列中交给工作线程后台处理，主线程会继续处理客户端请求。

换入

如果有客户端请求的 key 已经被换出了，主线程会先阻塞发出命令的客户端，然后将加载对象的信息放到一个队列中，让工作线程去加载。加载完毕后工作线程通知主线程。主线程再执行客户端的命令。这种方式只阻塞请求的 value 是已经被换出 key 的客户端。

总的来说阻塞方式的性能会好一些，因为不需要线程同步、创建线程和恢复被阻塞的客户端等开销。但是也相应的牺牲了响应性。工作线程方式主线程不会阻塞在磁盘 IO 上，所以响应性更好。如果我们的应用不太经常发生换入换出，而且也不太在意有点延迟的话推荐使用阻塞方式。

关于 redis 虚拟内存更详细介绍可以参考下面链接
http://redis.io/topics/internals-vm

5) 主从同步

5.1. Redis 主从复制简介

Redis 支持将数据同步到多台从库上，这种特性对提高**读取**性能非常有益。

1) master 可以有多个 slave。
2) 除了多个 slave 连到相同的 master 外，slave 也可以连接其它 slave 形成图状结构。
3) 主从复制不会阻塞 master。也就是说当一个或多个 slave 与 master 进行初次同步数据时，master 可以继续处理客户端发来的请求。相反 slave 在初次同步数据时则会阻塞不能处理客户端的请求。
4) 主从复制可以用来提高系统的可伸缩性，我们可以用多个 slave 专门用于客户端的读请求，比如 sort 操作可以使用 slave 来处理。也可以用来做简单的数据冗余。
5) 可以在 master 禁用数据持久化，只需要注释掉 master 配置文件中的所有 save 配置，然后只在 slave 上配置数据持久化。

5.2. Redis 主从复制的过程介绍

当设置好 slave 服务器后，slave 会建立和 master 的连接，然后发送 sync 命令。无论是第一次同步建立的连接还是连接断开后的重新连接，master 都会启动一个后台进程，将数据库快照保存到文件中，同时 master 主进程会开始收集新的写命令并缓存起来。后台进程完成写文件后，master 就发送文件给 slave，slave 将文件保存到磁盘上，然后加载到内存恢复数据库快照到 slave 上。接着 master 就会把缓存的命令转发给 slave。而且后续 master 收到的写命令都会通过开始建立的连接发送给 slave。从 master 到 slave 的同步数据的命令和从 客户端发送的命令使用相同的协议格式。当 master 和 slave 的连接断开时 slave 可以自动重新建立连接。如果 master 同时收到多个 slave 发来的同步连接命令，只会启动一个进程来写数据库镜像，然后发送给所有 slave。

配置 slave 服务器很简单，只需要在配置文件中加入如下配置
slaveof 192.168.1.1 6379　　#指定 master 的 ip 和端口

附录 A：redis 的安装与配置

1.1. 安装

1.1.1. 编译安装

```
$ wget http://redis.googlecode.com/files/redis-2.2.7.tar.gz
$ tar xzf redis-2.2.7.tar.gz
$ cp -r redis-2.2.7 /usr/local/redis
$ cd /usr/local/redis
$ make
$ make install                          # 编译好的文件将被复制到/usr/local/bin 下
```

#redis-server：Redis 服务器的 daemon 启动程序
#redis-cli：Redis 命令行操作工具。当然，你也可以用 telnet 根据其纯文本协议来操作
#redis-benchmark：Redis 性能测试工具，测试 Redis 在你的系统及你的配置下的读写性能
$redis-benchmark -n 100000 –c 50　#模拟同时由 50 个客户端发送 100000 个 SETs/GETs 查

询
#redis-check-aof：更新日志检查
#redis-check-dump：本地数据库检查

1.1.2. 配置

修改配置文件，并将其复制到 etc 目录下
vi redis.conf
$cp redis.conf /etc/redis.conf

配置文件基本说明
daemonize:　　#是否以后台守护进程方式运行
pidfile:　　#pid 文件位置
port:　　#监听的端口号
timeout:　　#请求超时时间
loglevel:　　#log 信息级别，总共支持四个级别：debug、verbose、notice、warning，默认为 verbose
logfile:　　#默认为标准输出（stdout），如果配置为守护进程方式运行，而这里又配置为日志记录方式为标准输出，则日志将会发送给/dev/null
databases:　　#开启数据库的数量。使用“SELECT 库 ID”方式切换操作各个数据库
save * *:　　#保存快照的频率，第一个*表示多长时间，第二个*表示执行多少次写操作。在一定时间内执行一定数量的写操作时，自动保存快照。可设置多个条件。
rdbcompression: #保存快照是否使用压缩
dbfilename:　　#数据快照文件名（只是文件名，不包括目录）。默认值为 dump.rdb
dir:　　#数据快照的保存目录（这个是目录）
requirepass:　　#设置 Redis 连接密码，如果配置了连接密码，客户端在连接 Redis 时需要通过 AUTH <password>命令提供密码，默认关闭。

1.1.3. 启动 redis

$redis-server /etc/redis.conf

1.1.4. 关闭 redis

$ redis-cli shutdown
#关闭指定端口的 redis-server
$redis-cli -p 6379 shutdown

1.1.5. 更新安装 redis

其它同安装，只是最后 make install 之前需要把正在运行的老版本 redis 关闭。

1.1.6. redis 系统管理相关指令简介

DBSIZE 返回当前数据库 key 的数量。
INFO 返回当前 redis 服务器状态和一些统计信息。
MONITOR 实时监听并返回redis服务器接收到的所有请求信息。
SHUTDOWN 把数据同步保存到磁盘上，并关闭redis服务。
CONFIG GET parameter 获取一个 redis 配置参数信息。（个别参数可能无法获取）
CONFIG SET parameter value 设置一个 redis 配置参数信息。（个别参数可能无法获取）
CONFIG RESETSTAT 重置 INFO 命令的统计信息。（重置包括：Keyspace 命中数、Keyspace 错误数、 处理命令数，接收连接数、过期 key 数）
DEBUG OBJECT key 获取一个 key 的调试信息。
DEBUG SEGFAULT 制造一次 redis 服务崩溃。
FLUSHDB 删除当前数据库中所有 key,此方法不会失败。小心慎用
FLUSHALL 删除全部数据库中所有 key，此方法不会失败。小心慎用

附录 B：安装 phpredis 模块

https://github.com/nicolasff/phpredis

下载 phpredis 最新版本
解压
> cd phpredis
> /usr/local/php5/bin/phpize #这个 phpize 是安装 php 模块的
> ./configure –with-php-config=/usr/local/php5/bin/php-config
> make
> cp modules/redis.so /usr/local/php5/etc/redis.so
接下来在 php.ini 中添加 extension=redis.so.
重启 apache

php 测试代码

```
<?php
$redis = new Redis();
$redis->connect('127.0.0.1',6379);
$redis->set('test','hello world!');
echo $redis->get('test');
?>
```

如果输出“hello world!”就说明 phpredis 模块已经安装成功并能正常运行了。

phpredis 方法说明：
https://github.com/nicolasff/phpredis/blob/master/README.markdown#readme

参考资料与知识扩展

Redis 指令大全：
http://redis.io/commands

Redis 指令在线模拟练习：
http://try.redis-db.com/

交流反馈

如果文档有任何问题和错误，可以通过 **ccitt@tom.com** 与我联系交流。