文 Redis高级用法

redis linux **Julylovin** 5 天前发布

Redis认识误区

- 用get/set方式使用Redis

```
作为一个key value存在，很多开发者自然的使用set/get方式来使用Redis
实际上这并不是最优化的使用方法。尤其在未启用VM情况下，Redis全部数据需要放入内存节约内存尤其重要。

假如一个key-value单元需要最小占用512字节，即使只存一个字节也占了512字节。
这时候就有一个设计模式，可以把key复用，几个key-value放入一个key中，value再作为一个set存入
这样同样512字节就会存放10-100倍的容量。

这就是为了节约内存，建议使用hashset而不是set/get的方式来使用Redis
```

- 单台Redis的存放数据必须比物理内存小

工具命令

```
#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
```

管理命令

```
# 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 制造一次服务器当机。
# 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
```

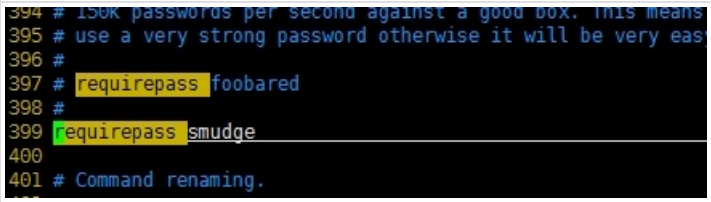
redis 安全性

- 添加密码

```
#编辑redis.conf
[root@localhost etc]# vim  /usr/local/redis/etc/redis.conf
```

```
#添加密码 smudge

#保存退出
```



- 重启redis

```
pkill redis

/usr/local/redis/bin/redis-server /usr/local/redis/etc/redis.conf
```

- 使用redis-cli命令 授权 -a smudge

```
 [root@localhost etc]# /usr/local/redis/bin/redis-cli -a smudge
```

```
127.0.0.1:6379>

    #如果不使用 -a 授权 可以单独授权
        127.0.0.1:6379> auth smudge
        OK
```

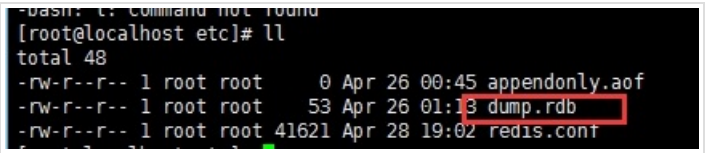
- php 代码中授权

```
$redis = new Redis();
$redis->connect('192.168.206.128', 6379);
$redis->auth('smudge');
$redis->set('key','TK');
```

redis 持久化

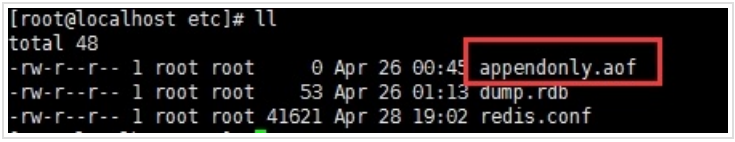
通常 Redis 将数据存储在内存中或虚拟内存中， 它是通过以下两种方式实现对数据的持久化

- 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
  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
  每隔一段时间写快照，如果redis意外宕机，最后一个写入快照后所有的更新，数据丢失



- 日志追加方式
  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
  弊端：持久化文件会变的越来越大

bgrewriteaof 命令 用于压缩持久化文件
配置redis.conf
vim /usr/local/redis/etc/redis.conf
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



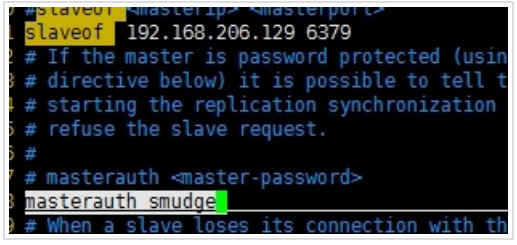
主从同步

```
# Redis 支持将数据同步到多台从库上
# 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
  master 可以有多个 slave。
# 主从复制不会阻塞 master，master可以同时进行复制和处理客户端请求
  但是slave初次同步会阻塞客户端的请求
# 主从复制可以用来提高系统的可伸缩性 可以使用slave 进行读操作
  比如 sort 操作可以使用 slave 来处理
# master 禁用数据持久化 在slave使用持久化
```

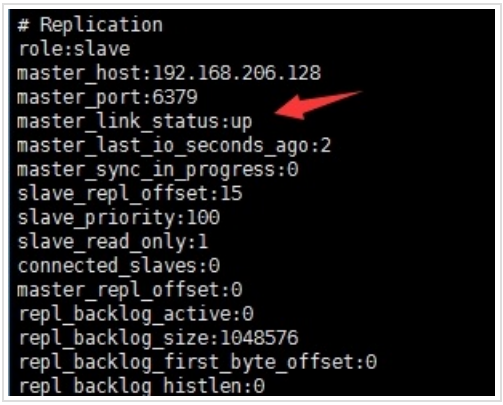
- 同步原理

```
^ 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
```

- 主从配置
  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
  masterauth smudge #主服务器有密码 所以要设置该项




```
#主服务器是 192.168.206.128 从服务器是 192.168.206.130
# /usr/local/redis/bin/redis-cli -a smudge #登录从服务器客户端
127.0.0.1:6379> info
```



值为up时，则主从配置成功

至此可以实验在主服务器存储 在从服务器获取

redis 事务性

- 基础事务性
  127.0.0.1:6379> mget name age #获取键值
  1) "TK"
  2) "28"
  127.0.0.1:6379> multi # 开启事务体
  OK
  127.0.0.1:6379> incr name # 命令插入队列
  QUEUED
  127.0.0.1:6379> incr age # 命令插入队列
  QUEUED
  127.0.0.1:6379> exec # 执行队列中的命令
  1) (error) ERR value is not an integer or out of range #name(string型)数值加1执行失败
  2) (integer) 29 #age数值加1执行成功
  #队列中命令有一个执行失败 并没有回滚，所以 redis不是严格的事务 异于RDBS的事务
  127.0.0.1:6379> discard # 清除事务
- Redis事务乐观锁
  多说几句,我们平时用的svn就是基于乐观锁原理, 如A和B同时改文件C
  A和Bupdate到C到本地的版本是1
  A修改完毕之后提交服务器,C的版本号为2
  B提交时候，服务器中C的版本号比B本地C版本号大,则提交失败
  B就要update,将C本地版本更新到2,再次提交
  redis有watch就是svn中的commit命令,每次修改时就要,核对版本号
  127.0.0.1:6379> watch age # 监视age键 记录当前版本号
  OK
  127.0.0.1:6379> incr age # 非事务更新age ,此时age键版本号已经更新
  (integer) 30
  127.0.0.1:6379> multi #开启事务
  OK
  127.0.0.1:6379> incr age #命令插入队列
  QUEUED
  127.0.0.1:6379> exec #执行事务
  (nil) # 执行事务失败,因为watch的版本号已经小于此时键age的版本号
- 总结
  执行exec时，如果监视的key从watch命令执行以来被修改过，则exec执行失败
  watch与multi exec搭配使用 watch仅对当前连接有效 ，断开连接，所有watch也就失效了
  exec unwatch discard 命令 执行后， 监视取消 ，不论之前监视多少个key 所有的watch全部失效了
  127.0.0.1:6379> watch age # 监视age
  OK
  127.0.0.1:6379> watch name # 监视name
  OK

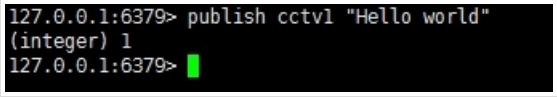
127.0.0.1:6379> incr age
(integer) 31
127.0.0.1:6379> multi
OK
127.0.0.1:6379> incr age
QUEUED
127.0.0.1:6379> exec # 此时所有的watch已经失效了
(nil)
127.0.0.1:6379> set name TK1
OK
127.0.0.1:6379> multi
OK
127.0.0.1:6379> set name TK2
QUEUED
127.0.0.1:6379> exec # 因为没有监视了 所以执行成功了
1) OK

- PHP代码

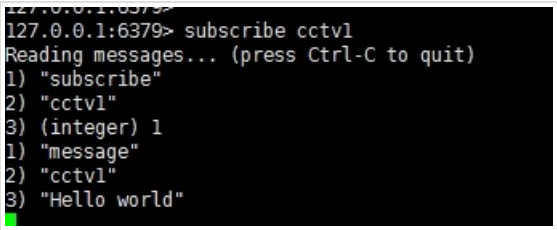
```php
<?php
    $redis->watch('key');
    # 监控键key 是否被其他客户端修改，如果KEY在调用watch()和exec()之间被修改，exec失败
    $ret = $redis->multi()
        ->incr('x')
        ->exec();
?>
```

发布订阅消息

- subscribe / publish 命令
  在Redis中，一旦一个client发出了SUBSCRIBE命令，它就处于监听的模式
  此时除了SUBSCRIBE， PSUBSCRIBE，UNSUBSCRIBE，PUNSUBSCRIBE这4条命令之外
  的所有其它命令都不能用。



publish 发布 publish tv1 "Hello world" 单通道发布



subscribe 订阅 subscribe tv1 tv2 tv3 订阅多个平台

- php代码
  #配置文件
  $redis = new Redis();
  $redis->pconnect('192.168.206.128', 6379);
  $redis->auth('smudge');
  #publish.php
  $redis->publish('chan-1', 'first');
  $redis->publish('chan-2', 'second');
  $redis->publish('chan-3', 'third');
  # subscribe.php
  function f($redis, $chan, $msg) { //频道订阅

```
    switch($chan) {
        case 'chan-1':
            echo 'chan-1:'.$msg."\n";
            break;

        case 'chan-2':
            echo 'chan-2:'.$msg."\n";
            break;

        case 'chan-3':
            echo 'chan-3:'.$msg."\n";
            break;
    }
```

}
$redis->subscribe(array('chan-1', 'chan-2', 'chan-3'),'f');

redis 虚拟内存

```
如果我们的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 server数据库的容量
而且也不会对性能造成太多影响。

# 开启VM功能
#vim redis.conf

vm-enabled yes                          #开启vm功能
vm-swap-file /tmp/redis.swap              #交换出来的value保存的文件路径
vm-max-memory 1000000                 #redis使用的最大内存上限
vm-page-size 32                         #每个页面的大小32个字节
vm-pages 134217728                     #最多使用多少页面
vm-max-threads 4                       #用于执行value对象换入换出的工作线程数量
```

redis 核心配置说明

```
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
```

5 天前发布

你可能感兴趣的文章

Linux（CentOS）下安装Redis 4 收藏，483 浏览

开发环境配置(linux 与 win7) 1 收藏，261 浏览

【安全公告】Redis Crackit 入侵事件通告 428 浏览

讨论区

请先 登录 后评论

本文隶属于专栏

Still Moving Under Gunfire

Still Moving Under Gunfire ，仍在枪林弹雨中前进 ，持续学习，永无止境，做顶尖大牛

Julylovin
作者

关注专栏

相关收藏夹

换一组

缓存
15 个条目 | 0 人关注

SessionID安全
4 个条目 | 0 人关注

程序设计
5 个条目 | 0 人关注

分享扩散：