链滴

redis 安装

作者：tmedivh

原文链接：https://ld246.com/article/1486710021741

来源网站：链滴

**官方网站地址: http://redis.io
下载地址: http://redis.io/download
简介: redis 是一个 key-value 存储系统,和 Memcached 类似，它支持存储的 value 类型相对更多，括 string(字符串)、list(链表)、 set(集合)和 zset(有序集 合)。这些数据类型都支持 push/pop、add/emove 及取交集并集和差集及更丰富的操作，而且这些操作都是原子性的。在此基础上，redis 支持种不同方式的排序。与 memcached 一样，为了保证效率，数据都是缓存在内存中。区别的是 redis 周期性的把更新的数据写入磁盘或者把修改 操作写入追加的记录文件，并且在此基础上实现了 master slave(主从)同步.
安装:
下载安装包之后
tar -zxvf redis-2.4.14.tar.gz
cd redis-2.4.14
make
注：安装过程中可能遇到的问题
第一个问题:
make: Warning: File `Makefile' has modification time 5.4e+06 s in the future
系统时间调整错了，调过来就好了
第二个问题:
make[2]: Entering directory `/redis/redis-2.4.7/deps/hiredis'
cc -c -std=c99 -pedantic -O3 -fPIC -Wall -W -Wstrict-prototypes -Wwrite-strings -g -ggdb net.c
make[2]: cc: Command not found
没安装 gcc，
yum install gcc-c++
第三个问题:
make 的时候显示
make[1]: Entering directory `/redis/redis-2.4.7/src'
which: no tclsh8.5 in (/usr/kerberos/sbin:/usr/kerberos/bin:/usr/local/sbin:/usr/local/bin:/sbin:bin:/usr/sbin:/usr/bin:/root/bin)
You need 'tclsh8.5' in order to run the Redis test
没安装 tcl
按照官网 http:/www.linuxfromscratch.org/blfs/view/cvs/general/tcl.html 上的安装
解压后进入根目录
cd unix
./configure --prefix=/usr --enable-threads --mandir=/usr/share/man
make
sed -e "s@^(TCL_SRC_DIR='').*@\1/usr/include'@" -e "/TCL_B/s@='(-L)?.*unix@='\1/usr/lb@" -i tclConfig.sh
make test
make install
make install-private-headers
ln -v -sf tclsh8.5 /usr/bin/tclsh
chmod -v 755 /usr/lib/libtcl8.5.so
安装完成之后
make
make install
程序会自动执行:
mkdir -p /usr/local/bin
cp -pf redis-server /usr/local/bin
cp -pf redis-benchmark /usr/local/bin

cp -pf redis-cli /usr/local/bin
cp -pf redis-check-dump /usr/local/bin
cp -pf redis-check-aof /usr/local/bin
配置:
此时如果直接启动 redis 的话会如下错误:
Warning: no config file specified, using the default config. In order to specify a config file use redis-server /path/to/redis.conf'
<h2 id="WARNING-overcommit-memory-is-set-to-0--Background-save-may-fail-under-low memory-condition--To-fix-this-issue-add--vm-overcommit-memory---1--to--etc-sysctl-conf-nd-then-reboot-or-run-the-command--sysctl-vm-overcommit-memory-1--for-this-to-take-effect-">WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.</h2>
第一个警告是没创建配置文件,使用了默认的配置,关于配置在后面我们会详细讲解
第二个警告是系统配置参数问题:
这里说一下这个配置的含义：
/proc/sys/vm/overcommit_memory
该文件指定了内核针对内存分配的策略，其值可以是 0、1、2。
0，表示内核将检查是否有足够的可用内存供应用进程使用；如果有足够的可用内存，内存申请允许否则，内存申请失败，并把错误返回给应用进程。
1，表示内核允许分配所有的物理内存，而不管当前的内存状态如何。
2，表示内核允许分配超过所有物理内存和交换空间总和的内存
这里推荐设置成 1:
运行命令:
sysctl vm.overcommit_memory=1
测试 redis 安装情况:
我只在一台虚拟机上安装了 redis，所以这台虚拟机既是服务器，又是客户端
测试:
使用 SSH 工具开一个会话，redis-server，让其作为服务器运行
[13769] 03 Jun 18:30:47 * Server started, Redis version 2.4.14
[13769] 03 Jun 18:30:47 * The server is now ready to accept connections on port 6379
[13769] 03 Jun 18:30:48 - 0 clients connected (0 slaves), 717496 bytes in use
[13769] 03 Jun 18:30:53 - 0 clients connected (0 slaves), 717496 bytes in use
[13769] 03 Jun 18:30:58 - 0 clients connected (0 slaves), 717496 bytes in use
这样 redis 服务端就成功启动了...现在没有一个客户链接
打开另一个 SSH 会话:
输入 redis-cli ping
响应 PONG 表示客户端找到了服务端
响应 Could not connect to Redis at 127.0.0.1:6379: Connection refused 表示服务端未开启
确定服务端开启后
输入 redis-cli
响应 redis 127.0.0.1:6379> 表示客户端成功开启
此时服务端也有了相应的变化
[13802] 03 Jun 18:37:38 - Accepted 127.0.0.1:55828
[13802] 03 Jun 18:37:41 - 1 clients connected (0 slaves), 726024 bytes in use
[13802] 03 Jun 18:37:46 - 1 clients connected (0 slaves), 726024 bytes in use
[13802] 03 Jun 18:37:51 - 1 clients connected (0 slaves), 726024 bytes in use
redis 服务端接受了客户端的一个链接
接下来测试 set key 和 get key 都正确
客户端输入:
redis 127.0.0.1:6379> ping
PONG
redis 127.0.0.1:6379> set key test
OK

redis 127.0.0.1:6379> get key
"test"
服务端响应:
[13802] 03 Jun 18:42:42 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[13802] 03 Jun 18:42:42 - 1 clients connected (0 slaves), 726216 bytes in use
[13802] 03 Jun 18:42:47 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[13802] 03 Jun 18:42:47 - 1 clients connected (0 slaves), 726216 bytes in use
[13802] 03 Jun 18:42:52 - DB 0: 1 keys (0 volatile) in 4 slots HT.
[13802] 03 Jun 18:42:52 - 1 clients connected (0 slaves), 726216 bytes in use
redis 搭建测试通过!!!
redis.conf 配置:
参考配置
编辑 /etc/redis.conf
daemonize yes
pidfile /webser/logs/redis.pid
port 6379
bind 127.0.0.1
timeout 300
loglevel verbose
logfile /webser/logs/redis.log
databases 16
save 900 1
save 300 10
save 60 10000
rdbcompression yes
dbfilename dump.rdb
dir /webser/redis
slave-serve-stale-data yes
appendonly no
appendfsync everysec
no-appendfsync-on-rewrite no
vm-enabled no
vm-swap-file /tmp/redis.swap
vm-max-memory 0
vm-page-size 32
vm-pages 134217728
vm-max-threads 4
hash-max-zipmap-entries 512
hash-max-zipmap-value 64
list-max-ziplist-entries 512
list-max-ziplist-value 64
set-max-intset-entries 512
activerehashing yes
以下结合自己翻译，以及在网上找的，比较全的参数说明：
1,是否以后台进程运行，默认为 no
daemonize no
2,如以后台进程运行，则需指定一个 pid，默认为/var/run/redis.pid
pidfile /var/run/redis.pid
3,监听端口，默认为 6379
port 6379
4,绑定主机 IP，默认值为 127.0.0.1（注释）
bind 127.0.0.1
5,超时时间，默认为 300（秒）
timeout 300

6,日志记录等级，有 4 个可选值，debug，verbose（默认值），notice，warning
loglevel verbose
7,日志记录方式，默认值为 stdout
logfile stdout
8,可用数据库数，默认值为 16，默认数据库为 0
databases 16
9,指出在多长时间内，有多少次更新操作，就将数据同步到数据文件。这个可以多个条件配合，比如认配置文件中的设置，就设置了三个条件。
900 秒（15 分钟）内至少有 1 个 key 被改变
save 900 1
300 秒（5 分钟）内至少有 10 个 key 被改变
save 300 10
10,存储至本地数据库时是否压缩数据，默认为 yes
rdbcompression yes
11,本地数据库文件名，默认值为 dump.rdb
dbfilename /root/redis_db/dump.rdb
12,本地数据库存放路径，默认值为 ./
dir /root/redis_db/
13,当本机为从服务时，设置主服务的 IP 及端口（注释）
slaveof
14,当本机为从服务时，设置主服务的连接密码（注释）
masterauth
15,连接密码（注释）
requirepass foobared
16,最大客户端连接数，默认不限制（注释）
maxclients 128
17,设置最大内存，达到最大内存设置后，Redis 会先尝试清除已到期或即将到期的 Key，当此方法处后，任到达最大内存设置，将无法再进行写入操作。（注释）
maxmemory
18,是否在每次更新操作后进行日志记录，如果不开启，可能会在断电时导致一段时间内的数据丢失因为 redis 本身同步数据文件是按上面 save 条件来同步的，所以有的数据会在一段时间内只存在于存中。默认值为 no
appendonly yes
19,更新日志文件名，默认值为 appendonly.aof（注释）
appendfilename /root/redis_db/appendonly.aof
20,更新日志条件，共有 3 个可选值。no 表示等操作系统进行数据缓存同步到磁盘，always 表示每更新操作后手动调用 fsync()将数据写到磁盘，everysec 表示每秒同步一次（默认值）。
appendfsync everysec
21,是否使用虚拟内存，默认值为 no
vm-enabled yes
22,虚拟内存文件路径，默认值为/tmp/redis.swap，不可多个 Redis 实例共享
vm-swap-file /tmp/redis.swap
23, 将所有大于 vm-max-memory 的数据存入虚拟内存,无论 vm-max-memory 设置多小,所有索引据都是内存存储的 (Redis 的索引数据就是 keys),也就是说,当 vm-max-memory 设置为 0 的时候,其是所有 value 都存在于磁盘。默认值为 0。
vm-max-memory 0
24,虚拟内存文件以块存储，每块 32bytes
vm-page-size 32
25,虚拟内在文件的最大数
vm-pages 134217728
26,可以设置访问 swap 文件的线程数,设置最好不要超过机器的核数,如果设置为 0,那么所有对 swap 件的操作都是串行的.可能会造成比较长时间的延迟,但是对数据完整性有很好的保证.
vm-max-threads 4
27,把小的输出缓存放在一起，以便能够在一个 TCP packet 中为客户端发送多个响应，具体原理和真

效果我不是很清楚。所以根据注释，你不是很确定的时候就设置成 yes
glueoutputbuf yes
28,在 redis 2.0 中引入了 hash 数据结构。当 hash 中包含超过指定元素个数并且最大的元素没有超临界时，hash 将以一种特殊的编码方式（大大减少内存使用）来存储，这里可以设置这两个临界值
hash-max-zipmap-entries 64
29,hash 中一个元素的最大值
hash-max-zipmap-value 512
30, 开启之后，redis 将在每 100 毫秒时使用 1 毫秒的 CPU 时间来对 redis 的 hash 表进行重新 has，可以降低内存的使用。当你的使 用场景中，有非常严格的实时性需要，不能够接受 Redis 时不时对请求有 2 毫秒的延迟的话，把这项配置为 no。如果没有这么严格的实时性要求，可以设置 为 yes 以便能够尽可能快的释放内存
activerehashing yes
修改好配置后启动时注意指定配置文件
redis-server /etc/redis.conf
关闭服务:
redis-cli shutdown
注: .3.1 如果端口变化可以指定端口:
redis-cli -p 6380 shutdown
这时内存中的数据会自动写入硬盘，文件地址是在 redis.conf 里配置的:
dbfilename dump.rdb
dir /webser/redis
这是持久化存储的关键,务必保证该文件/文件夹有写入权限
强制保存内存数据到硬盘:
因为 redis 是异步写入磁盘的，如果要让内存中的数据马上写入硬盘可以执行如下命令：
redis-cli save 或者 redis-cli -p 6380 save（指定端口）
同步机制
redis 实现的同步机制相对简单，缺少同步机制常见的 check point 和校验机制。
在 运行时，如果 master -> slave 同步请求转发被丢弃, slave 将无法恢复该请求的相关信息，直到 slave 重启时从 master 全量加载数据时才能修复。因此，建议使用 redis 尽量利用其 key/value 和 value 支持多种类型的特性，存储一些相对不重要的数据
附加信息:
启动 redis-server 后可使用 redis-benchmark 检查当前机器的处理性能.
另外执行 redis-benchmark 命令时也会将内存数据写入硬盘.
redis 针对很多语言都有客户端 http://redis.io/clients
这对 PHP 扩展推荐 phpredis
解压后进入文件夹
/webser/php53/bin/phpize
./configure --with-php-config=/webser/php53/bin/php-config
make && make install
php.ini 中添加
extension = "redis.so"
重启服务器
查看 phpinfo,安装完成后开始使用了\
首先确认 redis server 正常运行
运行下面代码
$redis = new Redis();
$redis->connect（‘127.0.0.1’,6379);
$redis->set（‘test’,'hello world!’);
echo $redis->get（‘test’);
?>
输出 hello world!

在 redis-cli 运行
get test
同样会输出 hello world!
就是这么轻松简单...**