redis 基础配置文档

redis稳定版本

http://download.redis.io/releases/redis-2.8.6.tar.gz

想要下载其他版本的，把版本号换一下应该就可以直接下载了。

redis安装步骤

```
上传至服务器
tar解压
make
make install
mkdir log/    #作为日志目录以及pid路径，自定义
mkdir etc/   #作为配置文件的存储路径，自定义
mkdir data/   #作为rdp文件的存储路径，自定义
```

注意：无需./configure ，直接make + make install 。

执行完后，当前目录即为redis程序目录。

```
echo vm.overcommit_memory = 1  >> /etc/sysctl.conf
sysctl -p
```

---

redis配置

蓝色是一般配置中需要修改的。

```
daemonize yes
```

```
port 6379
```

```
timeout 300
```

```
loglevel verbose
```

```
logfile /usr/local/redis/var/redis.log
```

```
databases 16
```

```
save 900 1
```

```
save 300 10
```

```
save 60 10000
```

```
stop-writes-on-bgsave-error no
```

```
rdbcompression yes
```

```
dbfilename dump.rdb
```

```
dir /usr/local/redis/data/
```

```
maxmemory 8gb
```

```
appendonly no
```

```
appendfsync everysec
```

redis配置说明

daemon：是否将redis作为后台进程运行，默认为no，一般改为yes

pidfile：指定redis pid文件的全路径

port：redis端口，默认为6379，可以自行修改

timout：指定socket连接空闲时间(秒).如果connection空闲超时,将会关闭连接(TCP socket选项)；如果为0,表示永不超时.

loglevel：生产环境使用notice即可

logfile：日志文件路径，notice下不会有set get等操作记录，只会有一些重要的信息，如启动以及bgsave等

databases：一个redis拥有很多个数据库，每一个数据库都是各自独立的，在数据库1中存在一个key，在数据库中可能没有，也可能拥有不一样的值；这样的做法对于一个redis给不同的程序使用带来了方便，可以很大程序避免各个程序的key值之间的冲突；默认为16（选择时为0-15）；在redis-cli中，通过select n即可完成不同数据库之间的切换

save：save <时间（秒）> <次数>，即满在规定时间内数据修更改达到规定次数的时候，就将数据同步到数据文件中，可以定义多个规则

rdbcompression：在将数据同步到数据文件中时，是否将数据文件进行压缩，默认为yes

dir：数据文件的存储路径，绝对路径、相对路径均可；恢复数据也放在这里

appendonly：aof持久化，默认为no，不开启　#后面单独开个文档做介绍

appendfsync：aof持久化间隔，默认为everysec

stop-writes-on-bgsave-error no：当执行bgsave失败的时候，不阻止对redis的写操作； 默认为yes，即当bgsave失败的时候，所有写操作会失败（这可能不是我们想见到的）

关于maxmemory以及过期keys清除的问题

1.当不开启maxmemory的时候，redis就认为不需要对内存做任何限制；当机器内存用尽的时候，就会使用虚拟内存或者磁盘，当然此时的性能是会下降的。 —— 而且这种方式总有一天也是会爆炸的。

2.开启了maxmemory的时候，当redis发现已经使用的内存达到了设置的值时，就会根据设置的【过期清除策略】（默认为volatile-lru）进行删除keys。

然而，如果程序在使用redis的时候没有对key进行过期时间的设置，那么带来的最终后果必定为【oom】。

在不修改程序并且不改变架构的前提下，这没的办法解决。

只能祈祷在现有的业务逻辑上，keys使用的memory增长缓慢。

不管怎么样，我们能做的就是设置maxmemory，过期清除策略使用默认即可。