[**CentOS7下yum安装Redis**](https://www.cnblogs.com/diantong/p/11255019.html)

(1).Redis概述

　　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（键值型）数据库（非关系型数据库），并提供多种语言的API。

　　Redis是一个高性能的Key-Value数据库。它的出现很大程度补偿来Memcached这类Key-Value型存储的不足，在部分场合下可以对关系型数据库起到很好的补充作用。它提供来Java、C/C++、PHP、JavaScript、Perl、Object-C、Python、Ruby、Erlang等客户端，使用方便。

　　Redis支持主从同步，Redis能够借助于Sentinel（哨兵，Redis自带的）工具来监控主从节点，当主节点发生故障时，会自己提升另外一个从节点成为新的主节点。

　1）支持的数据类型

　　和Memcached类似，但它支持存储的Value类型相对更多，包括String（字符串）、List（列表）、Sets（集合）、Sorted Sets（有序集合）和Hash（哈希类型、关联数组）、Bitmaps（位图）和HyperLoglog。

　2）性能

　　100万较小的键存储字符串，大概消耗100M内存；

　　由于Redis是单线程，如果服务器主机上有多个CPU，只有一个能够使用，但并不意味着CPU会成为瓶颈，因为Redis是一个比较简单的K-V数据存储，CPU通常不会成为瓶颈的；

　　在常见的linux服务器上，500K（50万）的并发，只需要一秒钟处理，如果主机硬件较好的情况下，每秒钟可以达到上百万的并发.

　3）Redis与Memcache对比

　　Memcache只能使用内存来缓存对象。而Redis除了可以使用内存来缓存对像，还可以周期性的将数据保存到磁盘上，对数据进行永久存储。当服务器突然断电或死机后， redis基于磁盘中的数据进行恢复；

Redis是单线程服务器，只有一个线程来响应所有的请求。Memcache是多线程的；

　　Redis支持更多的数据类型。

(2).安装

　　yum安装redis时，建议使用Remi repository源。因为Remi源提供了目前最新版本的Redis，可以通该源使用YUM安装目前最新版本的Redis。另外还提供了PHP和MySQL的最新yum源，以及相关服务程序。

　1）Remi repository源依赖于epel源，因此需要先安装epel源

|  |  |
| --- | --- |
| 1 | [root@youxi1 ~]# yum -y install epel-release |

　2）安装Remi repository源

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@youxi1 ~]# yum -y install http://rpms.remirepo.net/enterprise/remi-release-7.rpm  [root@youxi1 ~]# ls /etc/yum.repos.d/　　//下载完成后会出现许多remi的yum源，这里要用到的是remi.repo这个源  CentOS-Base.repo CentOS-Sources.repo remi-glpi92.repo remi-php70.repo remi-safe.repo  CentOS-CR.repo CentOS-Vault.repo remi-glpi93.repo remi-php71.repo  CentOS-Debuginfo.repo epel.repo remi-glpi94.repo remi-php72.repo  CentOS-fasttrack.repo epel-testing.repo remi-modular.repo remi-php73.repo  CentOS-Media.repo remi-glpi91.repo remi-php54.repo remi.repo |

　3）使用指定的yum源安装Redis

|  |  |
| --- | --- |
| 1  2  3 | [root@youxi1 ~]# yum --enablerepo=remi install -y redis　　//--enablerepo指定yum源  [root@youxi1 ~]# redis-cli --version　　//安装完成后使用命令查看一下版本  redis-cli 5.0.5 |

　　注意：remi源安装完成后，默认为不启动，在需求使用remi repository源安装程序时，需求--enablerepo=remi选项指定使用remi repository源是可以被使用的，然后进行安装。

　4）启动Redis并设置开机自启

|  |  |
| --- | --- |
| 1  2  3 | [root@youxi1 ~]# systemctl start redis  [root@youxi1 ~]# systemctl enable redis  Created symlink from /etc/systemd/system/multi-user.target.wants/redis.service to /usr/lib/systemd/system/redis.service. |

　　注意：Redis的端口号是6379

(3).配置文件信息

　　Linux下，Redis的配置文件存放在/etc/目录下的redis.conf。以下列出来可能会用到的参数：

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23  24  25  26  27  28  29  30  31  32  33  34  35  36  37  38  39  40  41  42  43  44  45  46  47  48  49  50  51  52  53  54  55  56  57  58  59  60  61  62  63  64  65  66  67  68  69  70  71  72  73  74  75  76  77  78  79  80  81  82  83  84  85  86  87  88  89  90  91  92  93 | //第69行，Redis监听的地址  bind 127.0.0.1  //第88行，安全模式，是否禁止外网访问Redis，yes表示启用，只能通过本地访问  protected-mode yes  //第92行，监听的端口号  port 6379  /\*第101行，指定tcp-backlog长度。tcp-backlog是一个等待队列。  \* 当大量请求需要Redis处理时，需要等待的请求队列会通过backlog来缓存。  \* backlog的数量决定来可以缓存的队列数  \*/  tcp-backlog 511  //第109行，指定使用sock文件通信以及sock文件位置。如果服务端和客户端在同一台主机上，建议打开。sock方式通信可以直接在内存中交换，不经过TCP/IP协议栈进行封装、拆封  # unixsocket /tmp/redis.sock  //第110行，定义sock文件的权限  # unixsocketperm 700  //第113行，表示客户端连接成功后，空闲多长时间超时（非活跃，没有数据交互）。0表示不开启此功能  timeout 0  //第130行，维持长链接的时间，单位秒  tcp-keepalive 300  //第136行，是否在后台运行守护进程。如果使用redis服务脚本启动，即使为no，也会运行一个守护进程。一般设置yes  daemonize no  //第147行，是由upstart还是systemd接管redis进程。默认无监督互动，不需要修改。  supervised no  //第158行，pid文件地址  pidfile /var/run/redis\_6379.pid  //第166行，日志级别  loglevel notice  //第174行，日志文件位置  logfile /var/log/redis/redis.log  //第186行，默认由多少个数据库。但是在分布式中，只能有一个  databases 16    /\*第218到220行，快照存储策略，存到磁盘的持久化策略  \*第一个数字是单位时间，单位为秒；第二个数字是键值发生变化的次数  \* 例如第一个，900秒内至少发生1次键值变化，则做一次快照（持久化）  \* 第二个就是，300秒内至少发生10次键值变化，则做一次快照（持久化）  \* 所有条件是并列关系，根据不同的键值变化选择使用规则  \*/  save 900 1  save 300 10  save 60 10000  //第235行，在进行快照备份时，一旦发生错误是否停止，默认yes即可  stop-writes-on-bgsave-error yes  //第241行，指定RDB文件是否压缩。yes表示压缩，会消耗CPU资源  rdbcompression yes  /\*第250行，是否对RDB文件做校验码检测。  \*此项定义在redis启动时加载RDB文件是否对文件检查校验码，在redis生成RDB文件是会生成校验信息，在redis再次启动或装载RDB文件时，是否检测校验信息。  \*如果检测的情况下会消耗时间，会导致redis启动时慢，但是能够判断RDB文件是否产生错误。  \*/  rdbchecksum yes  //第253行，RDB文件名称   dbfilename dump.rdb  //第263行，RDB文件存放的路径  dir /var/lib/redis    //第286行，定义Master服务器的IP和端口，主从复制的配置信息  # replicaof <masterip> <masterport>  //第293行，定义Master服务器的密码，主从复制的配置信息  # masterauth <master-password>  //第308行，当从端在主从复制过程中与主端断开连接，yes表示继续提供服务，即使数据可能不是最新的；no表示对请求返回错误信息  replica-serve-stale-data yes  //第324行，从端只读  replica-read-only yes  //第355行，默认不使用diskless（无磁盘）同步方式  repl-diskless-sync no  //第367行，diskless（无磁盘）方式进行数据传递之前会有一个时间的延迟，以便从端能够进行到待传送的目标队列中，默认5秒  repl-diskless-sync-delay 5  //第373行，从端向主端发送ping的时间间隔，默认10秒  # repl-ping-replica-period 10  //第385行，设置超时时间  # repl-timeout 60  /\*第400行，是否启用TCP\_NODELAY。  \* 如果启用则会使用少量的TCP包和带宽去进行数据传输到从端，速度较慢；  \* 如果不启用则使用较多的带宽进行数据传输，速度较快。  \*/  repl-disable-tcp-nodelay no  /\*第413行，设置backlog的大小。backlog是一个缓冲区，在从端失联时存放要同步到从端的数据。  \* 因此当从端重连时，一般是不需要完全同步的。backlog越大，从端可以失联的时间就越长（相对来说）  \*/  # repl-backlog-size 1mb  //第426行，一段时间后，从端还没有连上master，那么backlog（缓冲区）的内存将被释放。0表示永不释放，默认3600秒  # repl-backlog-ttl 3600  //第441行，从端的优先级设置，数字越小优先级越高。主端故障会根据优先级高的从端来进行恢复。如果设置的是0，那么该从端永远不会被选中  replica-priority 100  //第457到458行，当主端的可用从端小于3个或网路延迟岛屿10秒时，主端拒绝接收用户的写请求。  # min-replicas-to-write 3  # min-replicas-max-lag 10    //第507行，指定认证密码，默认不启动  # requirepass foobared    //第539行，同时连接redis的最大数量  # maxclients 10000 |

　　特别注意：第69行的bind，第88行的protected-mode，第136行的daemonize，第539行的maxclients，还有主从复制的所有配置参数。

(4).redis的简单使用

　　最最简单的使用：redis-cli -h [IP地址] -p [端口号]。如果是连接本地，-h和-p都可以省略。

　1）Reids字符串操作

　　键的命名规则：可以使用ASCII字符；键的长度不要过长，键的长度越长则消耗的空间越多；在同一个库中（名称空间），键的名称不得重复，如果复制键的名称，实际上是修改键中的值；在不同的库中（名称空间），键的多种名称可以重复；键可以实现自动过期。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14 | [root@youxi1 ~]# redis-cli  127.0.0.1:6379> set system centos　　//创建键值  OK  127.0.0.1:6379> get system　　//获取键值  "centos"  127.0.0.1:6379> exit  [root@youxi1 ~]# systemctl restart redis　　//重启  [root@youxi1 ~]# redis-cli  127.0.0.1:6379> get system　　//可以看到键值是存在的  "centos"  127.0.0.1:6379> set name "youxi" EX 10　　//如这般设置，则表示该键值值存在10秒  OK  127.0.0.1:6379> get name　　//10秒过后再次查看，这就是一个临时键值  (nil) |

　2）如果开启认证功能

　　首先修改配置文件，开启认证功能

|  |  |
| --- | --- |
| 1  2  3 | [root@youxi1 ~]# vim /etc/redis.conf  requirepass 123456　　//在第507行，去除注释，设置认证密码  [root@youxi1 ~]# systemctl restart redis |

　　此时再次使用redis

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7 | [root@youxi1 ~]# redis-cli  127.0.0.1:6379> get system　　//没有认证时，会报错  (error) NOAUTH Authentication required.  127.0.0.1:6379> auth 123456　　//认证  OK  127.0.0.1:6379> get system　　//认证后，再次获取键值，成功  "centos" |

(5).配置持久化

　　Redis工作时所有数据集都是存储于内存中的。如果Redis崩溃或断电会导致所有数据丢失，所以Redis提供了持久化功能来保证数据的可靠性。Redis持久化有两种实现方法：RDB和AOF。

　　RDB: 存储为二进制格式的数据文件，是默认启动的持久化机制；按事先定制的策略，周期性地将数据保存至磁盘。

AOF：Append Only File类似于MySQL的二进制日志，记录每一次redis的写操作命令，以顺序IO方式附加在指定文件的尾部，是使用追加方式实现的，这也叫做一种附加日志类型的持久化机制。由于每一次的操作都记录，则会随着时间长而增大文件的容量，并且有些记录的命令是多余的。但是redis进程能够自动的去扫描这个对应的AOF文件，把其中一些冗余的操作给合并一个，以实现将来一次性把数据恢复。

　　RDB的配置其实上面已经说明过了，我这里再复制一下，如下。总共就6个配置点。

|  |  |
| --- | --- |
| 1  2  3  4  5  6  7  8  9  10  11  12  13  14  15  16  17  18  19  20  21  22  23 | /\*  \* 第218到220行，快照存储策略，存到磁盘的持久化策略  \*第一个数字是单位时间，单位为秒；第二个数字是键值发生变化的次数  \* 例如第一个，900秒内至少发生1次键值变化，则做一次快照（持久化）  \* 第二个就是，300秒内至少发生10次键值变化，则做一次快照（持久化）  \* 所有条件是并列关系，根据不同的键值变化选择使用规则  \*/  save 900 1  save 300 10  save 60 10000  //第235行，在进行快照备份时，一旦发生错误是否停止，默认yes即可  stop-writes-on-bgsave-error yes  //第241行，指定RDB文件是否压缩。yes表示压缩，会消耗CPU资源  rdbcompression yes  /\*第250行，是否对RDB文件做校验码检测。  \*此项定义在redis启动时加载RDB文件是否对文件检查校验码，在redis生成RDB文件是会生成校验信息，在redis再次启动或装载RDB文件时，是否检测校验信息。  \*如果检测的情况下会消耗时间，会导致redis启动时慢，但是能够判断RDB文件是否产生错误。  \*/  rdbchecksum yes  //第253行，RDB文件名称   dbfilename dump.rdb  //第263行，RDB文件存放的路径  dir /var/lib/redis |

　　AOF的配置从第679行开始，如果需要可以自己看一下。