1. Redis 安装（172.16.129.90/root/123456）
2. wget http://download.redis.io/releases/redis-2.8.19.tar.gz
3. $ tar xzf redis-2.8.19.tar.gz
4. $ cd redis-2.8.19
5. $ make
6. 启动命令：src/redis-server
7. Utils/redis\_init\_script通过初始化脚本启动redis，使Redis能随系统自动运行
8. src/redis-server redis.conf 按照配置文件中的设置启动redis：

daemonize: 默认情况下，redis不是在后台运行的，如果需要在后台运行，把该项的值更改为yes。

logfile /var/log/redis/redis.log 日志文件，默认为/dev/null

loglevel debug 日志级别

1. 进入命令界面：src/redis-cli
2. 关闭redis服务：src/redis-cli shutdown
3. Redis Command ：<http://redis.io/commands>
4. Redis 支持的数据类型包括字符串类型，散列表类型，列表类型，集合类型，有序集合类型

Redis 字典顺序：0<9<A<Z<a<z

1. 字符串： set key value / get key value
2. 散列表类型（对象存储）： Hset key field value/hget key field

Hmset key field value [field value …]/hmget key field [field …]

/Hgetall key

1. 列表类型（字符串列表）： lpush key value[value…]

Rpush key value[value…]

Lrange key start stop(0至-1显示全部)

1. 集合类型（字符串集合）：sadd key member [member…]

Smembers key

1. 事务（multi/exec/watch）, 过期时间（expire/ttl）,排序（sort/by/get/store），消息通知，队列实现（brpop key [key …]timeout，blpop key timeout）,发布/订阅模式（publish/subscribe/unsubscribe/psubscribe/punsubscribe）
2. Redis 内存数据结构：http://www.searchtb.com/2011/05/redis-storage.html

**typedef** **struct** redisObject {

    unsigned type:4;  // 类型

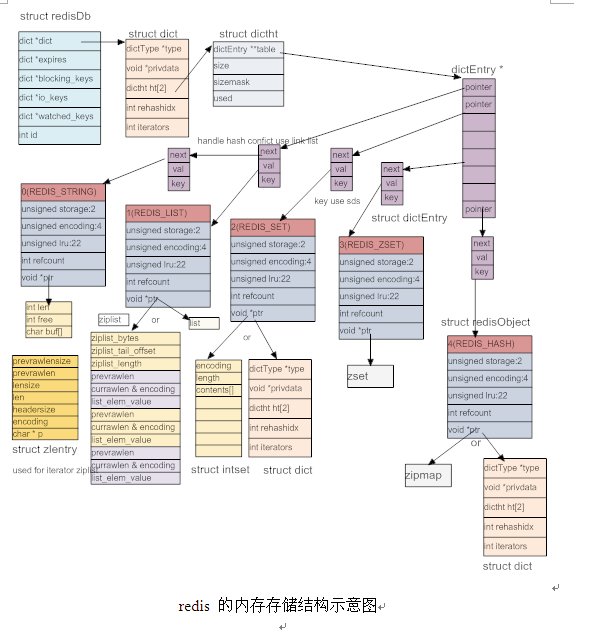
   unsigned encoding:4;    // 编码

  unsigned lru:REDIS\_LRU\_BITS; // 对象最后一次被访问的时间 /\* lru time (relative to server.lruclock) \*/

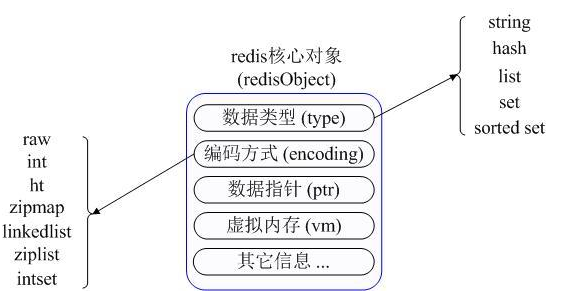
**int** refcount;    // 引用计数

**void** \*ptr;    // 指向实际值的指针

} robj;



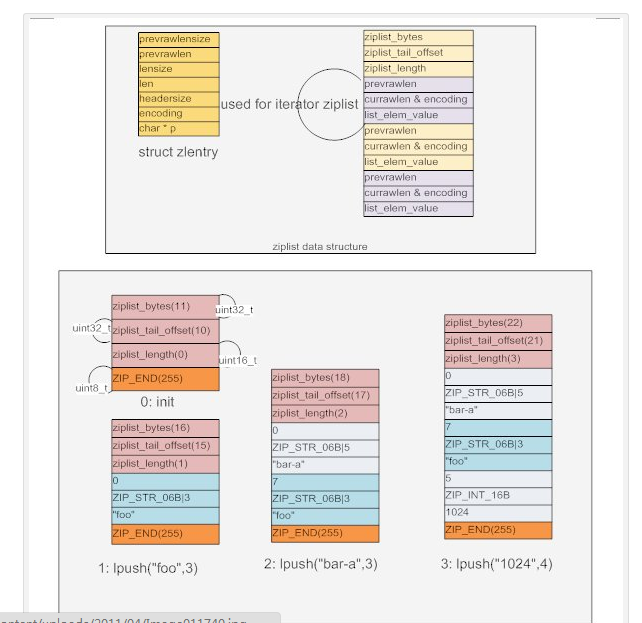
内部编码优化：http://blog.csdn.net/freebird\_lb/article/details/7733994



a. 字符串类型：sds(Simple Dynamic Strings)，使用sdshdr类型存储字符串，而redisObject的ptr字段指向的是该变量的地址，当健值内容可以用64位有符号整数表示时，redis会将键值转化为long类型来存储，redis启动后会预先建立10000个分别存储0到9999这些数字的redisObject类型变量作为共享对象。

struct sdshdr {   
　　　　int len; // buf 中已被使用的字符串空间数量   
　　　 int free; 　// buf 中预留字符串空间数量   
　　　　char buf[]; // 实际储存字符串的地方   
　　};

b. 散列类型： redis-encodeing-ziplist：



c. 集合类型

typedef struct intset {  
  uint32\_t encoding;  
  uint32\_t length;  
   int8\_t contents[];  
} intset;

encoding有三种:

#define INTSET\_ENC\_INT16 (sizeof(int16\_t))  
#define INTSET\_ENC\_INT32 (sizeof(int32\_t))  
#define INTSET\_ENC\_INT64 (sizeof(int64\_t))

1. 持久化
2. Save 命令： 当执行save命令时，同步进行快照操作
3. Bgsave命令： 当执行bgsave命令时，异步进行快照操作
4. Flushall命令： 只要自动快照条件不为空时，执行flushall命令时，就进行快照操作
5. 当设置了主从模式时，redis会在初始化时进行自动快照
6. 哨兵
7. 主从配置

1). 从数据库的配置文件中加入“slaveof 主数据库地址 主数据库端口” 或命令行slaveof主数据库地址 主数据库端口

2). 原理：作为从数据库向主数据库发送ping命令，确认是否连接-----》replconf listening-port 自己的端口-----》向主数据库发送sync命令开始同步（同步的是rdb快照文件）

Slave-server-stale-data 参数no使从数据库在同步完成前对所有命令（除info和slaveof）都回复错误

3). 手动将从数据库设置成主数据库：slaveof no one

1. 哨兵

1). 建立配置文件sentinel.conf,内容：sentinel monitor master-name ip redis-port quorum

Quorum: 表示执行故障恢复操作前至少需要几个哨兵节点同意

2). 启动哨兵： redis-sentinel /path/to/sentinel.conf

3). 哨兵工作原理：a）. 发送INFO命令使得哨兵可以获取当前数据库信息，解析返回的结果得知从数据库列表，从而对每个从数据库建立两个连接。b）.哨兵向主从数据库的\_sentinel\_:hello频道发送信息来与同样监控该数据库的哨兵分享自己的信息，发送内容：<哨兵的地址>,<哨兵的端口>,<哨兵的运行ID>,<哨兵的配置版本>,<主数据库的名字>,<主数据库的地址>,<主数据库的端口>,<主数据库的配置版本>

c). 哨兵会每隔一定时间向数据库发送PING, 根据down-after-millisecond 设置相关，当大于1s时，每隔1s，小于1s时，按down-after-millisecond指定时间。

1. 集群
2. 集群配置（每个集群至少需要3个主数据库才能正常运行）

1).在配置文件中打开“cluster-enabled yes”

2).使用辅助工具redis-trib.rb (执行gem install redis 来安装)： /path/to/redis-trib.rb create –replicas 1 ip:port ip:port ip:port……

--replicas 1 表示每个主数据库拥有的从数据库为1

B. 向集群中加入新节点:cluster meet ip port(ip和port是集群中的任意一个节点

C. 插槽分配(在一个集群中，所有的建都会被分配给16384个插槽cluster slots 查看插槽分配情况)

1). 插槽之前没有被分配过的，现在想分配给制定节点

在该节点执行命令Cluster addslots slot1[…]

2). 插槽之前被分配过，现在想移动到制定节点

Cluster setslot 插槽号 NODE 新节点的运行ID

a）. 手动获取插槽中的建：cluster getkeysinslot 插槽号 要返回的建的数量

b). 使用migrate 命令将其迁移（这种形式花费的时间较长，可能存在键值找不到）：

migrate 目标节点地址 目标节点端口 键名 数据库号 超时时间 【copy】【replace】

c）. redis 提供了两个命令实现在集群不下线的情况下迁移数据

cluster setslot 插槽号 migrating 新节点的运行ID

cluster setslot 插槽号 importing 原节点的运行ID

3). Redis-trib.rb 的执行过程（把0从A迁移到B）

a). 在B上执行cluster setslot 0 importing A

b）. 在A上执行Cluster setslot 0 migraing B

c）. 执行cluster getkeysinslot 0

d）. 对第三步获取的每个键执行migrate 命令

e）. 执行cluster setslot 0 node B完成迁移