**内存数据库系列课程**

# 一、MemCached缓存技术

## （一）什么是MemCached

Memcache是一套开源,高性能的分布式的内存对象缓存系统，目前被许多网站使用以提升网站的访问速度，尤其对于一些大型的、需要频繁访问数据库的网站访问速度提升效果十分显著 。

Memcache将所有数据存储在内存中，并在内存里维护一个统一的巨大的Hash表，它能存储任意类型的数据，包括图像、视频、文件以及数据库检索的结果等。简单的说就是将数据调用到内存中，然后从内存中读取，从而大大提高读取速度。

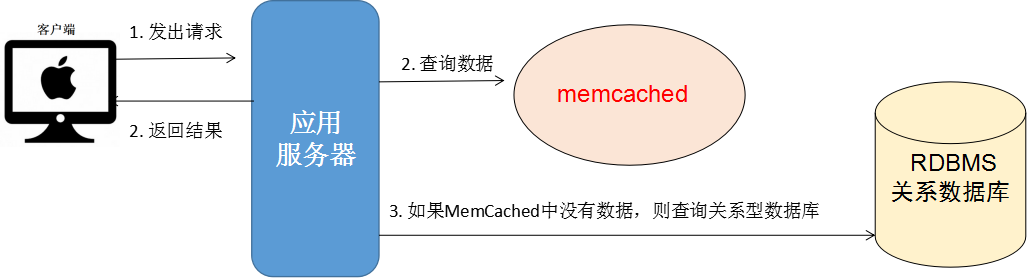
## （二）为什么要用MemCached？

请大家先看一个问题,这个问题在大并发，高负载的网站中必须考虑！大家思考如何让速度更快。

解决方案：

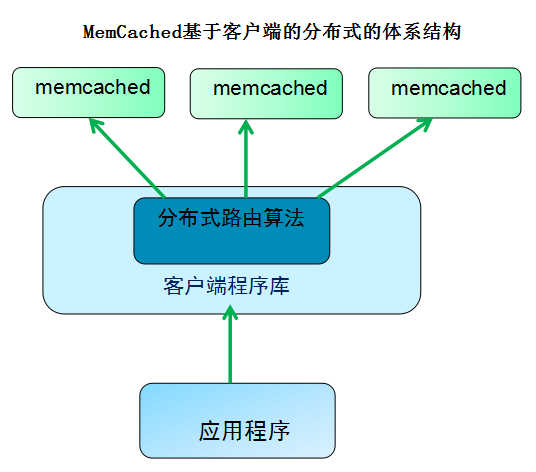
* 传统的RDBMS
* 页面静态化
* MemCached缓存技术

**基于Memcached的网站架构**



## （三）MemCached的基本原理和体系结构

简单的说: memcached就是在内存中维护一张巨大的hash表,通过自己的一套**路由算法**来维护数据的操作。



## （四）MemCached数据的存储方式和过期

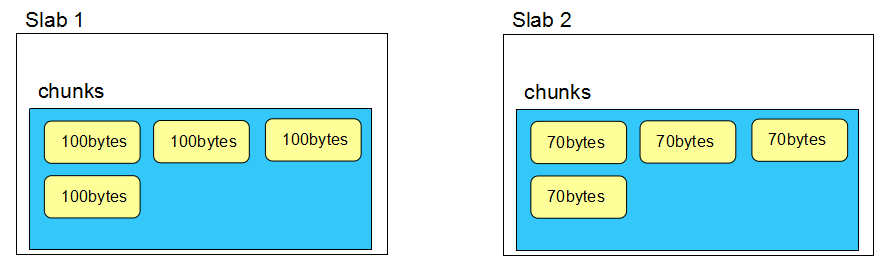
**1、数据存储方式：Slab Allocation即：按组分配内存**

（1）每次先分配一个Slab，相当于一个page，大小1M。

（2）然后在1M的空间里根据内容再划分相同大小的chunk

（3）优点是：最大限度的利用内存，避免产生内存碎片

（4）缺点是：会造成内存的浪费



**2、数据过期方式**

（1）懒过期方式（Lazy Expiration）：

memcached内部不监视数据是否过期，而是get时查看记录时间，检查是否已经过期，这叫惰性过期。

（2）LRU算法：采用最近最少使用算法淘汰内存中的数据

## （五）MemCached安装与部署

**实验环境：**

* OracleLinux-R6-U6-Server-i386-dvd.iso
* memcached-1.4.25.tar.gz
* memcached-1.2.8-repcached-2.2.tar.gz
* libevent-2.0.21-stable.tar.gz
* spymemcached-2.10.3.jar

**1、安装GCC编译器所需的rpm包**

rpm -ivh libgomp-4.4.7-11.el6.i686.rpm

rpm -ivh libstdc++-devel-4.4.7-11.el6.i686.rpm

rpm -ivh kernel-headers-2.6.32-504.el6.i686.rpm

rpm -ivh glibc-headers-2.12-1.149.el6.i686.rpm

rpm -ivh glibc-devel-2.12-1.149.el6.i686.rpm

rpm -ivh ppl-0.10.2-11.el6.i686.rpm

rpm -ivh cloog-ppl-0.15.7-1.2.el6.i686.rpm

rpm -ivh mpfr-2.4.1-6.el6.i686.rpm

rpm -ivh cpp-4.4.7-11.el6.i686.rpm

rpm -ivh gcc-4.4.7-11.el6.i686.rpm

rpm -ivh gcc-c++-4.4.7-11.el6.i686.rpm

rpm -ivh telnet-0.17-48.el6.i686.rpm

**2、安装Libevent**

\*、确认是否已经安装？并删除旧版本

rpm -qa | grep libevent

rpm -e libevent-1.4.13-4.el6.i686 --nodeps

\*、解压：tar -zxvf libevent-2.0.21-stable.tar.gz

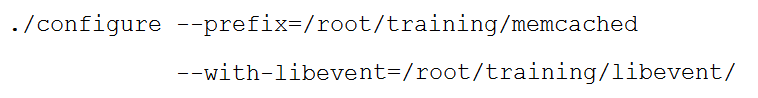
\*、安装到/root/training/libevent目录

./configure --prefix=/root/training/libevent

make

make install

**3、安装MemCached**



**4、启动MemCached：**memcached -h 帮助信息

**启动命令：./memcached -u root -d -m 128**

-p：指定端口 默认：11211

-u：指定用户名（root用户必须使用该选项）

-m：分配的内存，默认：64M

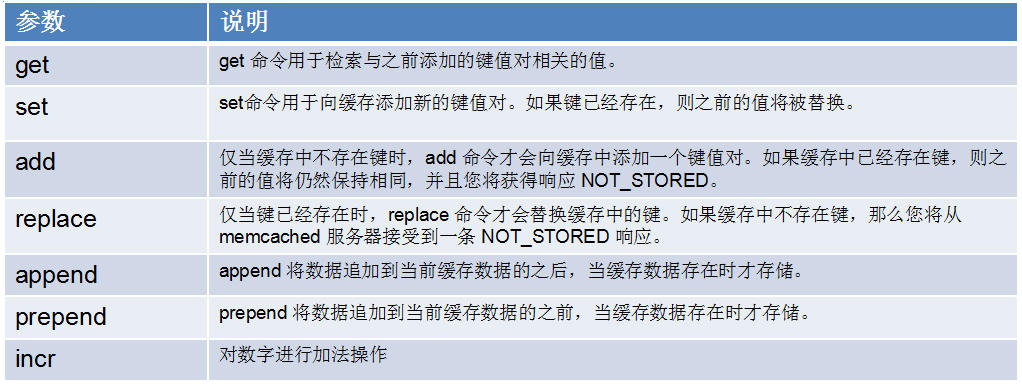
-c：最大并发连接，默认1024

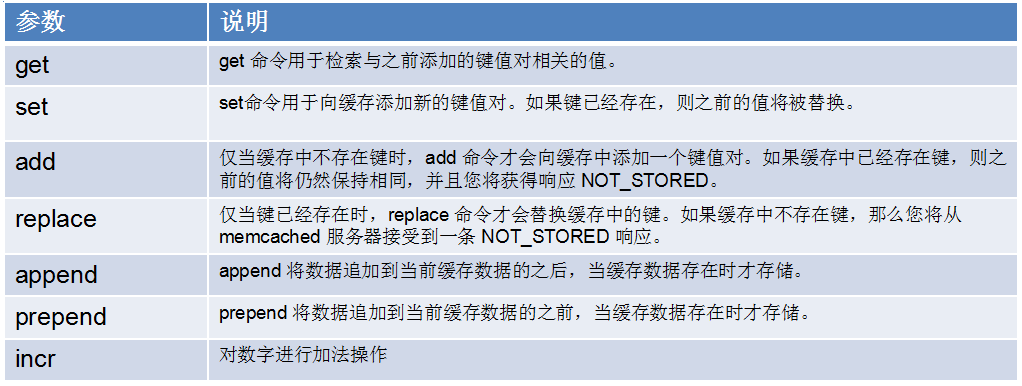
-d：启动一个守护进程



## （六）操作MemCached

**1、Telnet方式（命令行方式）**





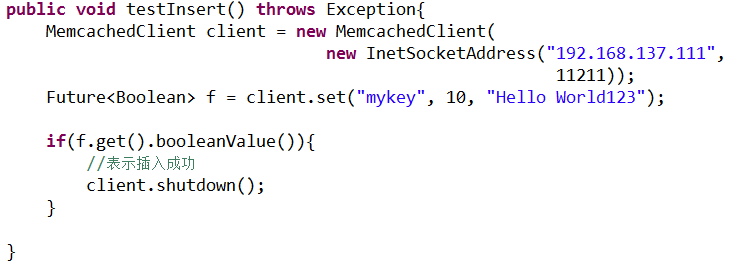
**统计信息的命令：**

* + - stats
    - stats items
    - stats slabs

**2、Java客户端方式**



* **插入数据**



* **查询数据**



* **基于客户端的分布式插入数据**



## （七）MemCached的路由算法

**1、求余数hash算法**

用key做hash运算得到一个整数，根据余数路由。

例如：服务器端有三台MemCached服务器

根据key，做hash运算

7%3=1，那么就路由到第2台服务器。

6%3=0，那么路由到第1台服务器

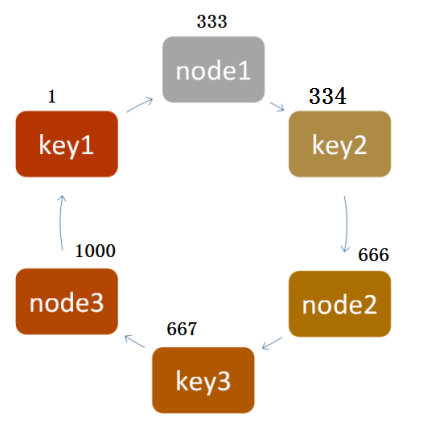
5%3=2，那么路由到第3台服务器

优点：数据分布均衡在多台服务器中，适合大多数据需求。

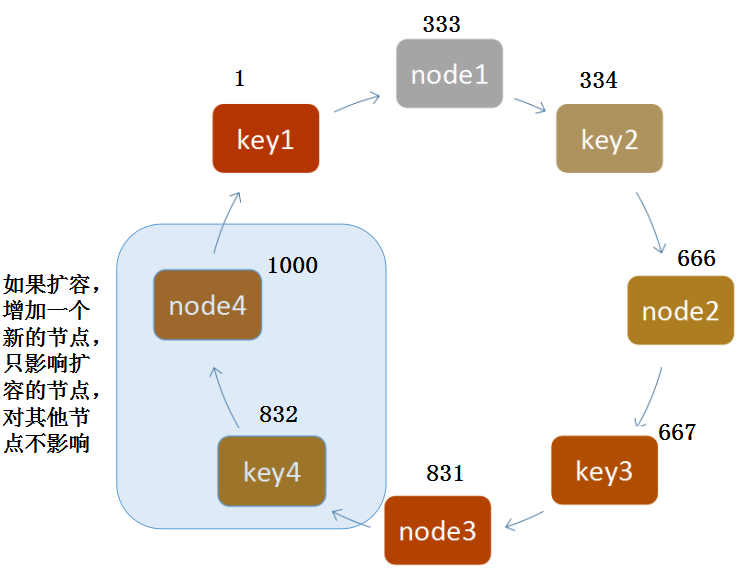
缺点：如果需要扩容或者有宕机的情况，会造成数据的丢失。

**2、一致性hash算法**

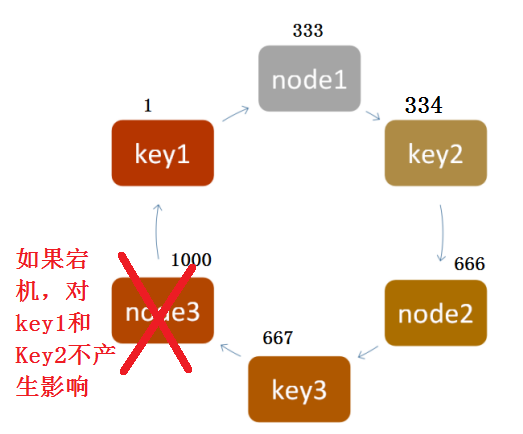
* + - * 基本原理：



* + - * 一致性hash算法下扩容

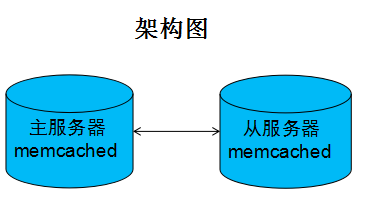


* + - * 一致性hash算法下DOWN机



## （八）MemCached的主主复制和HA

**1、Memcached主主复制**



* 安装具有复制功能的memcached版本

tar zxvf memcached-1.2.8-repcached-2.2.tar.gz

cd memcached-1.2.8-repcached-2.2

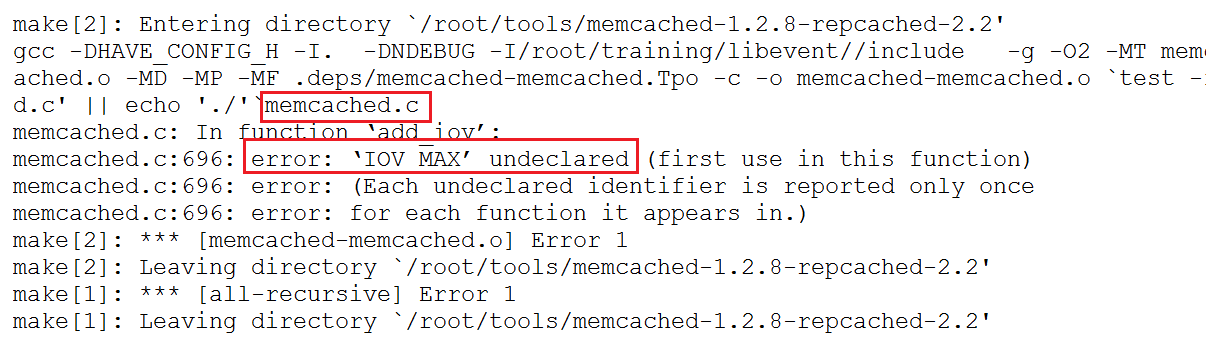
./configure --prefix=/root/training/memcached\_replication

--with-libevent=/root/training/libevent/ --enable-replication

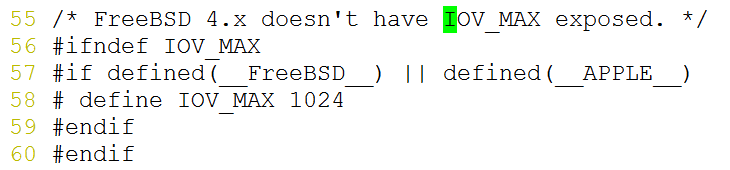
make

make install

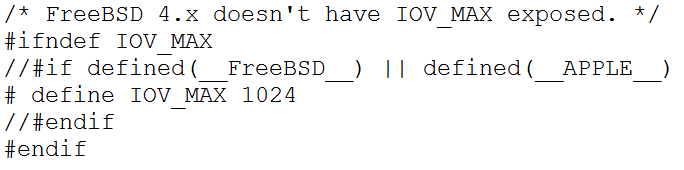
**出现以下错误：**



**解决办法，编辑memcached.c文件如下：**



修改成如下形式：



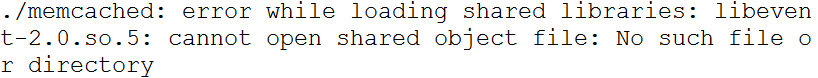
* 启动第一台MemCached，使用-x指定对端服务器的地址

./memcached -d -u root -m 128 -x **192.168.137.12**

* 启动第二台MemCached，使用-x指定对端服务器的地址

./memcached -d -u root -m 128 -x **192.168.137.11**

**出现以下错误：**

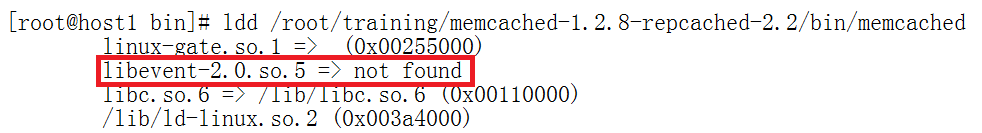


**解决办法：**

* 查找 libevent-2.0.so.5

whereis libevent-2.0.so.5

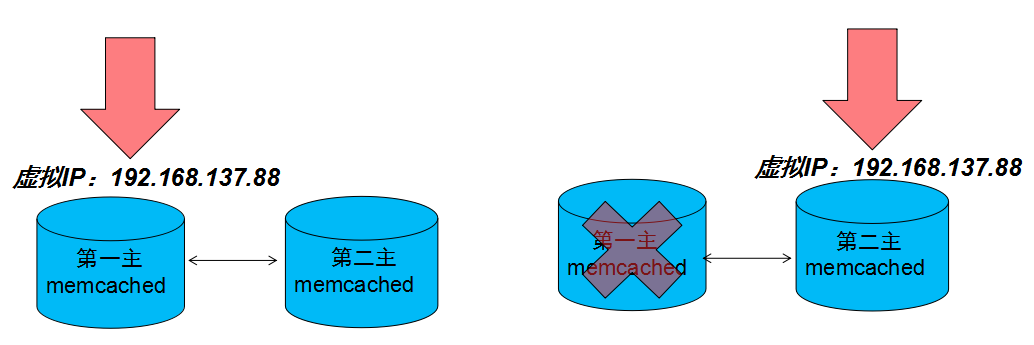
* 使用ldd命令查看memcached命令，发现找不到



* 建立软连接

ln -s /root/training/libevent/lib/libevent-2.0.so.5 /usr/lib/libevent-2.0.so.5

**2、Memcached的HA（High Availablity）**



Keepalived是一个交换机制的软件。Keepalived的作用是检测服务器的状态，如果有一台web服务器死机，或工作出现故障，Keepalived将检测到，并将有故障的服务器从系统中剔除，同时使用其他服务器代替该服务器的工作，当服务器工作正常后Keepalived自动将服务器加入到服务器群中，这些工作全部自动完成，不需要人工干涉，需要人工做的只是修复故障的服务器。

利用Keepalived实现MemCached的主主复制高可用架构：

* + - * Keepalived在memcached主服务器产生一个虚拟IP（VIP）
      * Keepalived可以通过不断的检测memcached主服务器的11211端口是否正常工作，
      * 如果发现memcached Down机，虚拟IP就从主服务器移到从服务器

**配置Keepalived（每台机器都要配置）:**

* rpm -ivh keepalived-1.2.13-4.el6.i686.rpm
* 配置：主从节点都要配置，配置文件：
  + /etc/keepalived/keepalived.conf

***（主节点配置信息） （从节点配置信息）***

! Configuration File for keepalived

global\_defs {

notification\_email {

collen\_training@126.com

}

notification\_email\_from collen\_training@126.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 100

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

virtual\_ipaddress {

192.168.137.88

}

}

! Configuration File for keepalived

global\_defs {

notification\_email {

collen\_training@126.com

}

notification\_email\_from collen\_training@126.com

smtp\_server 127.0.0.1

smtp\_connect\_timeout 30

router\_id LVS\_DEVEL

}

vrrp\_instance VI\_1 {

state MASTER

interface eth0

virtual\_router\_id 51

priority 101

advert\_int 1

authentication {

auth\_type PASS

auth\_pass 1111

}

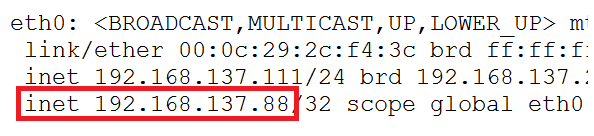
virtual\_ipaddress {

192.168.137.88

}

}

**验证Keepalived: 使用命令 ip ad sh 查看虚拟ip地址**



# 二、Redis高性能内存数据库

## （一）什么是Redis?

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，包括string(字符串)、list(链表)、set(集合)、zset(sorted set --有序集合)和hash（哈希类型）。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会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。

Redis 是一个高性能的key-value数据库。Redis的出现，很大程度补偿了memcached这类key/value存储的不足，在部分场合可以对关系数据库起到很好的补充作用。它提供了Java，C/C++，C#，PHP，JavaScript，Perl，Object-C，Python，Ruby，Erlang等客户端，使用很方便。[1]

Redis支持主从同步。数据可以从主服务器向任意数量的从服务器上同步，从服务器可以是关联其他从服务器的主服务器。

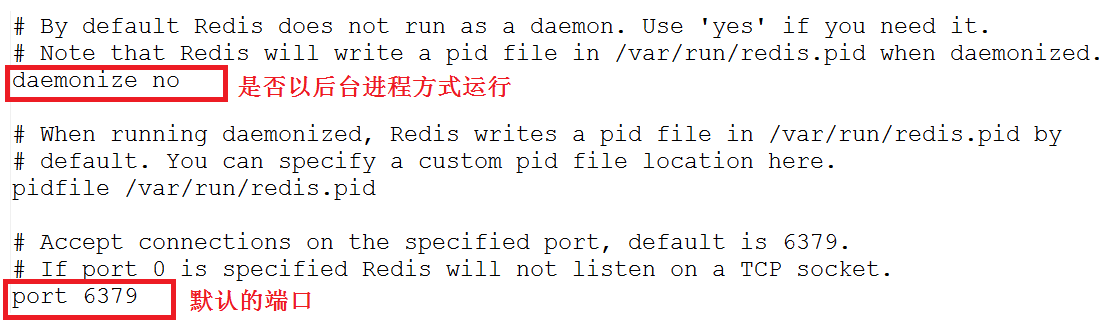
## （二）Redis与Memcached的区别

* + - 持久化：
      * Redis可以用来做缓存，也可以做存储；支持ADF和RDB两种持久化方式
      * Memcached只能缓存数据
    - 数据结构：
      * Redis有丰富的数据类型：字符串、链表，Hash、集合，有序集合
      * Memcached一般就是字符串和对象

## （三）Redis的安装与配置

1. 解压：tar -zxvf redis-3.0.5.tar.gz
2. make
3. make PREFIX=/root/training/redis install
4. cp ~/tools/redis-3.0.5/redis.conf /root/training/redis/etc/

**Redis的核心配置文件：redis.conf**



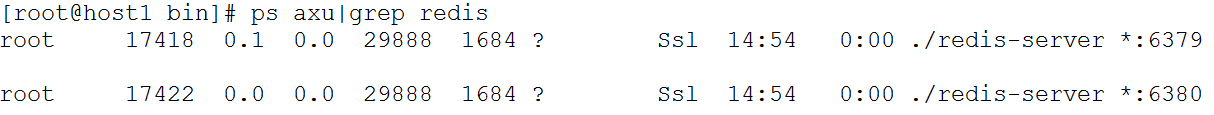
**Redis的命令脚本：**

* redis-benchmark 性能测试工具
* redis-check-aof 检查AOF日志
* redis-check-dump 检查RDB日志
* redis-cli 启动命令行客户端
* redis-sentinel
* redis-server 启动Redis服务

**启动Redis：**

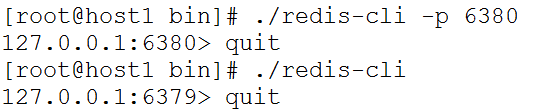
* ./redis-server ../etc/redis6379.conf
* ./redis-server ../etc/redis6380.conf

这样就在6379和6380端口上，各自启动了一个Redis实例；也可以通过ps命令查看：



**启动Redis的客户端：redis-cli**

* 默认连接6739端口，也可以通过-p指定连接的端口号：



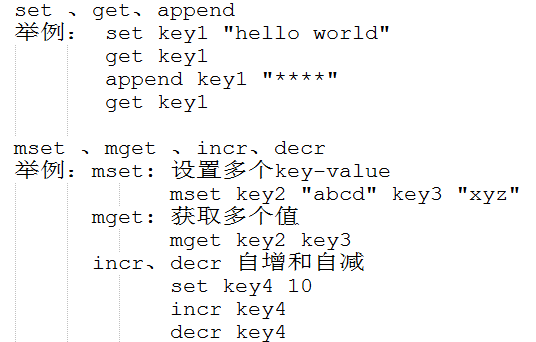
* ./redis-cli --help显式帮助信息

## （四）Redis的操作

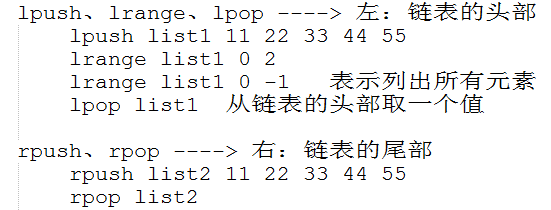
1. **键值操作**

|  |  |
| --- | --- |
| **键值命令** | **示例** |
| keys pattern |  |
| randomkey |  |
| exists key |  |
| type key |  |
| expire key |  |
| pexpire key |  |
| persist key |  |

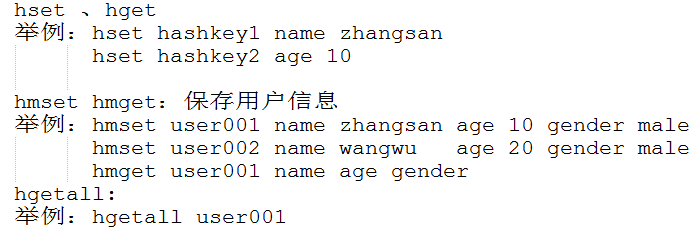
1. **数据类型**
2. 字符串



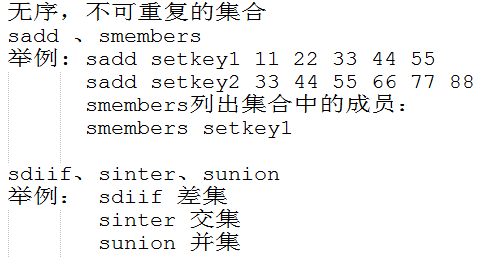
1. 链表



1. Hash



1. 无序集合

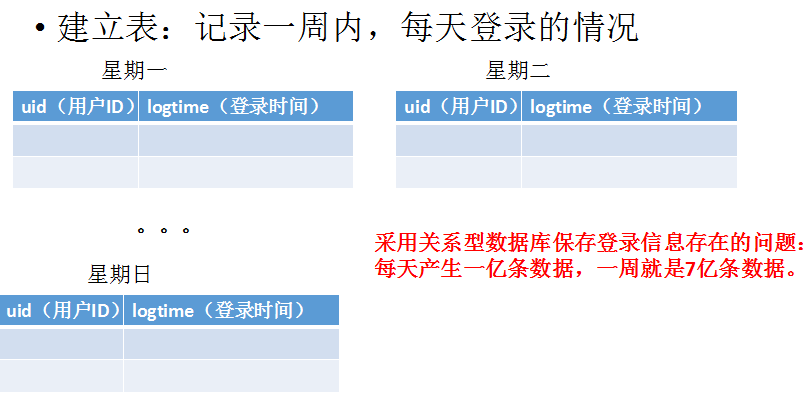


1. 有序集合



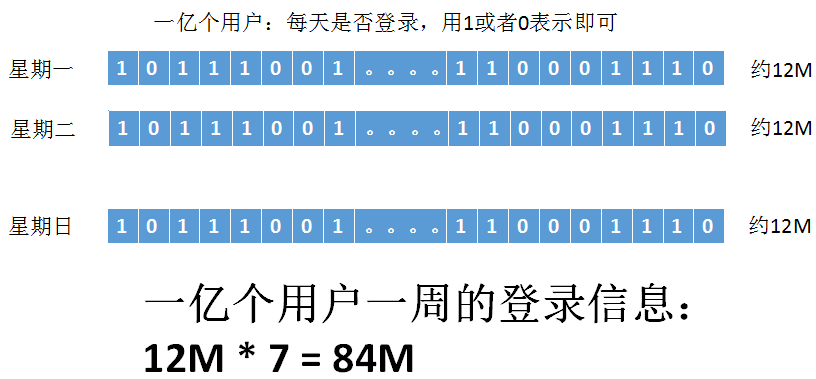
1. Redis数据类型案例分析：网站统计用户登录的次数
2. 1亿个用户，有经常登录的，也有不经常登录的
3. 如何来记录用户的登录信息
4. 如何查询活跃用户：比如：一周内，登录3次的

* ***解决方案一：采用关系型数据库***



* ***解决方案二：采用Redis存储登录信息***

**可以使用Redis的setbit，登录与否：有1和0就可以表示**



1. **Java客户端**
2. 基本操作



1. 连接池



1. 使用Redis实现分布式锁

使用Maven搭建工程：

<dependency>

<groupId>redis.clients</groupId>

<artifactId>jedis</artifactId>

<version>2.9.0</version>

</dependency>

## （五）Redis的事务和消息机制

1. **Redis的事务**

Redis对事务的支持目前还比较简单。redis只能保证一个client发起的事务中的命令可以连续的执行，而中间不会插入其他client的命令。 由于redis是单线程来处理所有client的请求的所以做到这点是很容易的。一般情况下redis在接受到一个client发来的命令后会立即处理并 返回处理结果，但是当一个client在一个连接中发出multi命令有，这个连接会进入一个事务上下文，该连接后续的命令并不是立即执行，**而是先放到一个队列中**。当从此连接受到exec命令后，redis会顺序的执行队列中的所有命令。并将所有命令的运行结果打包到一起返回给client.然后此连接就 结束事务上下文。

**Oracle数据库中的事务和Redis的事务对比**

|  |  |  |
| --- | --- | --- |
|  | **Oracle** | **Redis** |
| 开启事务的方式 | 自动开启事务 | multi |
| 操作 | DML语句 | Redis命令 |
| 提交事务 | commit | exec |
| 回滚事务 | rollback | discard |

1. **Redis的事务示例：银行转账**

* 从Tom转100块钱给Mike

set tom 1000

set mike 1000

multi

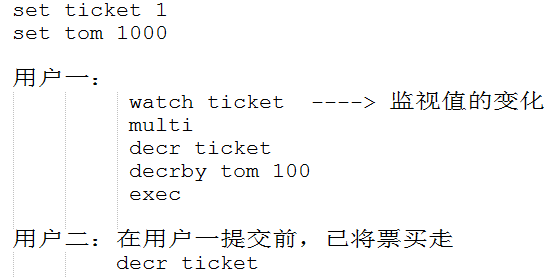
decrby tom 100

incrby mike 100

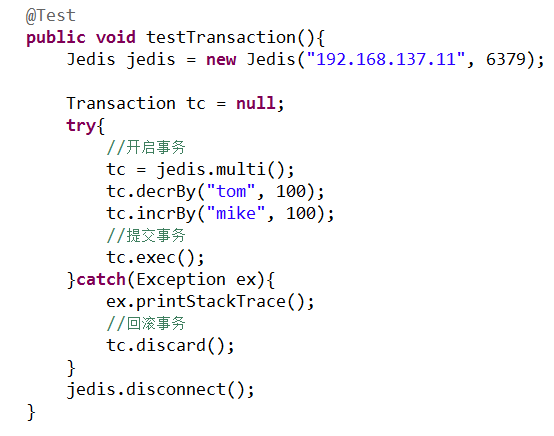
exec

1. **Redis的锁机制：watch**

* 举例：买票



1. **Java应用程序中的事务和锁**
2. 事务

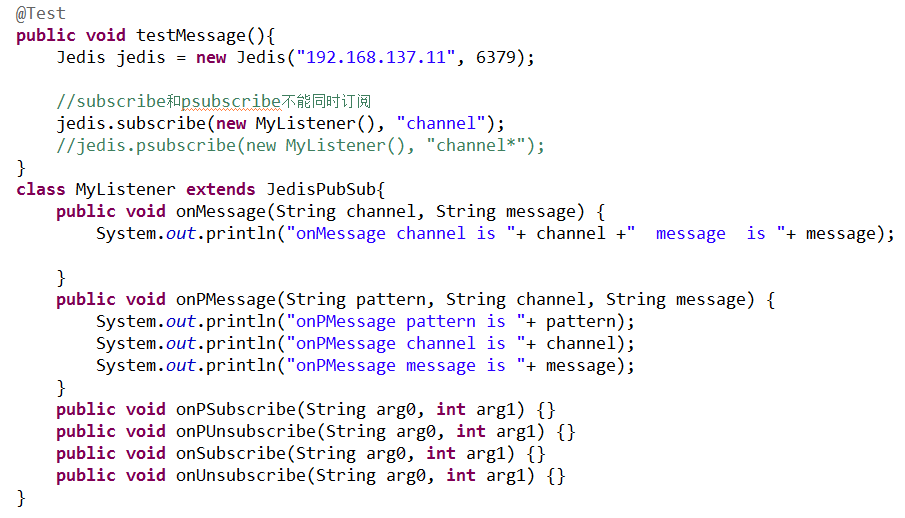


1. 锁



1. **Redis的消息机制：消息的发布与订阅，适合做在线聊天**

* publish:发布消息
  + 格式：publish channel名称 “消息内容”
* subscribe: 订阅消息
  + 格式：subscribe channel名称
* psubscribe: 使用通配符定义消息
  + 格式：psubscribe channel\*名称
* 使用Java程序实现消息的发布与订阅，需要继承**JedisPubSub**类



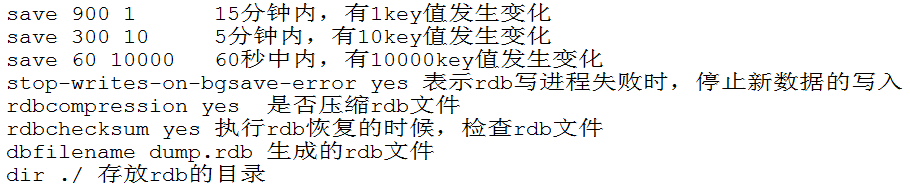
## （六）Redis的持久化

Redis 提供了多种不同级别的持久化方式：

* + - **RDB** 持久化可以在指定的时间间隔内生成数据集的时间点快照（point-in-time snapshot）。
    - **AOF （Append-only file）**持久化记录服务器执行的所有写操作命令，并在服务器启动时，通过重新执行这些命令来还原数据集。 AOF 文件中的命令全部以 Redis 协议的格式来保存，新命令会被追加到文件的末尾。 Redis 还可以在后台对 AOF 文件进行重写（rewrite），使得 AOF 文件的体积不会超出保存数据集状态所需的实际大小。
    - Redis 还可以同时使用 AOF 持久化和 RDB 持久化。 在这种情况下， 当 Redis 重启时， 它会优先使用 AOF 文件来还原数据集， 因为 AOF 文件保存的数据集通常比 RDB 文件所保存的数据集更完整。
    - 你甚至可以关闭持久化功能，让数据只在服务器运行时存在。

1. **RDB**

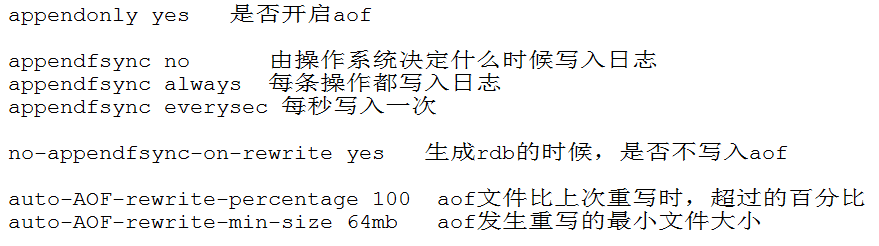
* 工作原理：每隔一定时间给内存照一个快照，将内存中的数据写入文件（rdb文件）
* 配置参数：redis.conf文件



* RDB示例测试：可以使用redis-benchmark进行压力测试
  + ./bin/redis-benchmark -n 100000 表示执行100000个操作
* RDB的缺点：
  + 在两次快照之间，如果发生断电，数据会丢失
  + 举例：在生成rdb后，插入新值。突然断电，数据可能会丢失

1. **AOF：通过日志的方式**

* 工作原理：记录操作的命令
* 配置参数：



* 什么是AOF的重写：rewrite
  + 将内存中的key逆向生成命令，如同一个可以，反复操作了100次，aof文件会记录100次操作，这样会导致AOF文件过大

例如：

set age 0

incr age

incr age

... 100次

最后 age的值是100

经过重写后，直接执行： set age 100

* + 可以通过观察aof日志文件的大小

1. **Redis持久化注意的问题**

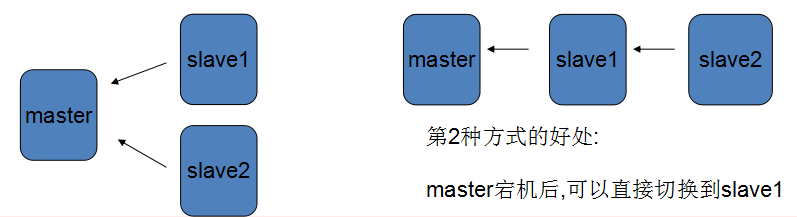
* RDB恢复的速度快
* **如果RDB和AOF都有，默认使用AOF进行恢复**

## （七）Redis的集群

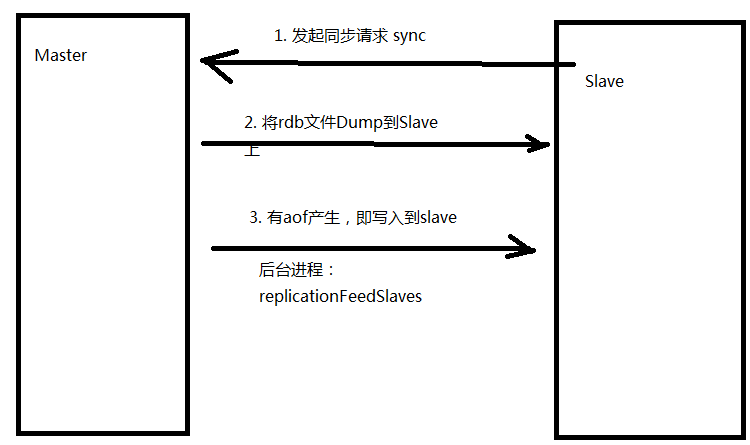
1. **集群的作用**

* 主从备份 防止主机宕机
* 读写分离,分担master的任务
* 任务分离,如从服分别分担备份工作与计算工作

1. **Redis集群的两种部署方式**



1. **Redis主从服务的通信原理**



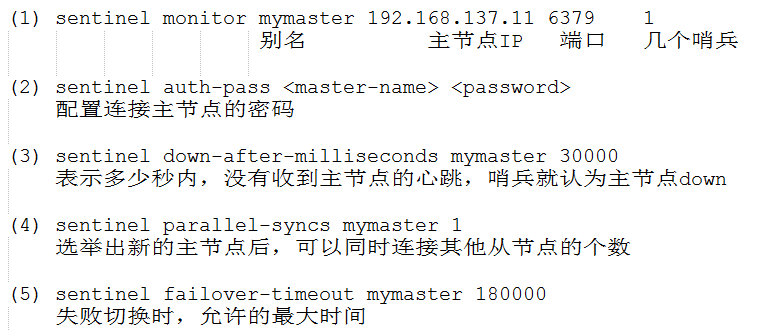
1. **配置Redis的集群（主从模式）**

* 主节点：关闭rdb和aof即可
* 从节点：slaveof localhost 6379

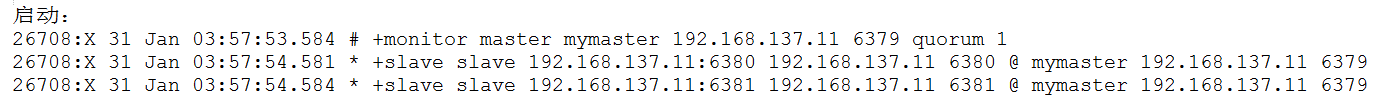
开启rdb和aof

1. **Redis集群的高可用性**

* Redis 2.4+自带了一个HA实现Sentinel
* 配置文件：sentinel.conf

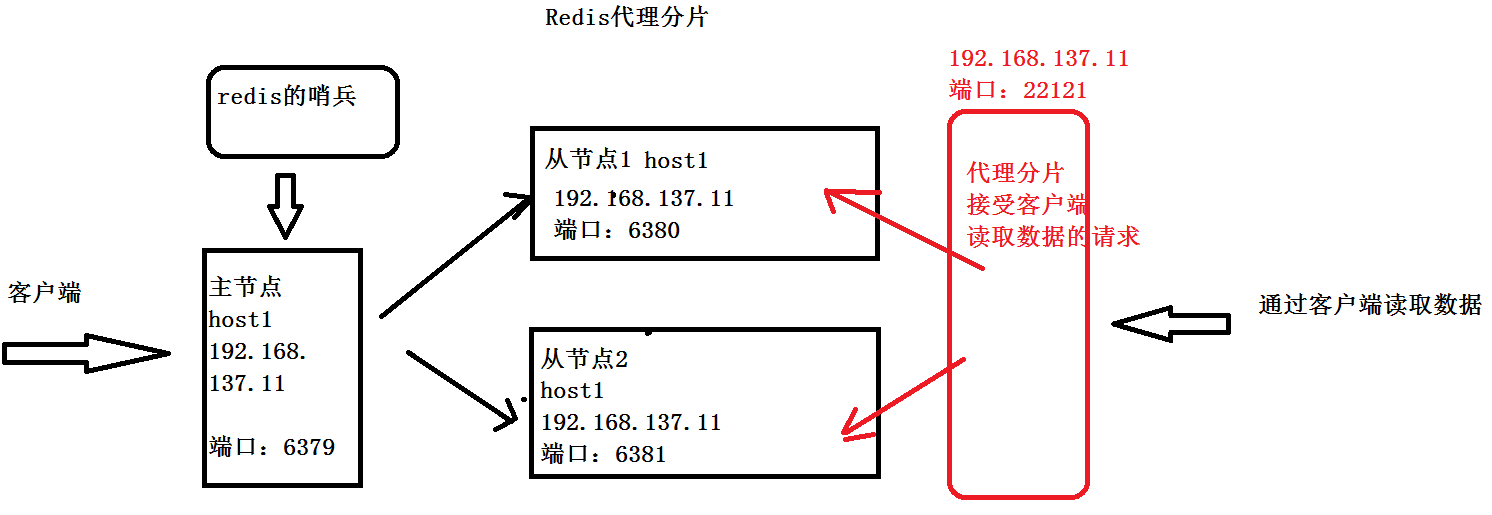


* redis-sentinel ../etc/sentinel.conf
* 查看日志：



1. **实现Redis的代理分片**

* Twemproxy是一种代理分片机制，由Twitter开源。
* Twemproxy作为代理，可接受来自多个程序的访问，按照路由规则，转发给后台的各个Redis服务器，再原路返回。该方案很好的解决了单个Redis实例承载能力的问题。



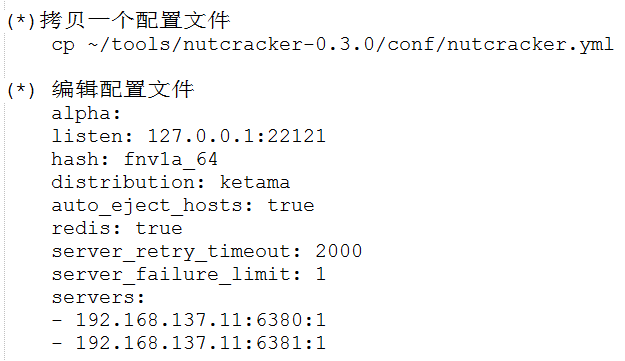
* 安装

./configure --prefix=/root/training/proxy

make

make install

* 配置文件



* 检查配置文件是否正确

./nutcracker -t conf/nutcracker.yml

* 启动代理服务器

./nutcracker -d -c conf/nutcracker.yml