<http://blog.csdn.net/yxm659217822/article/details/44162727>

Memcach什么是Memcache

Memcache集群环境下缓存解决方案

Memcache是一个高性能的分布式的内存对象缓存系统，通过在内存里维护一个统一的巨大的hash表，它能够用来存储各种格式的数据，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。

Memcache是danga的一个项目，最早是LiveJournal 服务的，最初为了加速 LiveJournal 访问速度而开发的，后来被很多大型的网站采用。

Memcached是以守护程序方式运行于一个或多个服务器中，随时会接收客户端的连接和操作

为什么会有Memcache和memcached两种名称  
  
其实Memcache是这个项目的名称，而memcached是它服务器端的主程序文件名，知道我的意思了吧。一个是项目名称，一个是主程序文件名，在网上看到了很多人不明白，于是混用了。

Memcached是高性能的，分布式的内存对象缓存系统，用于在动态应用中减少数据库负载，提升访问速度。Memcached由Danga Interactive开发，用于提升LiveJournal.com访问速度的。LJ每秒动态页面访问量几千次，用户700万。Memcached将数据库负载大幅度降低，更好的分配资源，更快速访问。

    上网baidu了很多东西，几乎都差不多，而且基于java的说的很少，所有只有在研究了各个其他语言类的应用后再来尝试在java上进行简单的操作应用。先从memcached上进行说明，memcached的最新版是采用c语言进行开发和设计的，据说旧版的是采用perl语言开发的，而且它是一个应用软件来的，是作为缓存服务器的服务器端运行在服务器上的，需要使用特定的语言编写客户端与其进行通信来进行数据的缓存和获取。通常我们是把memcached安装运行在web服务器上，然后通过对需要的数据进行缓存，据我目前所知，所有数据的缓存设置和存取操作，以及数据的更新后替换操作全部需要程序来进行，而不是自动进行的（自动不知道能不能成功，呵呵）。下面从一个实际的例子来应用memcached。

    如何要下载的话，到<http://danga.com/memcached/>下载memcached。

## [Ubuntu下安装Memcached](http://www.mike.org.cn/articles/ubuntu-install-memcached/" \o "Link to: Ubuntu下安装Memcached" \t "_blank)

编译前，请先确认gcc、make、patch等编译工具是否已安装，并可正常使用。

　　安装Libevent

　　Libevent是一个异步事件处理软件函式库，以BSD许可证释出。Memcached依赖Libevent，因此必须先编译安装Libevent。

检测libevent 安装是否成功，输入：# ls -al /usr/lib | grep libevent  会出现如下结果（不同的机器可能有不同的输出）：

   yangfei@yangfei-laptop:~$ ls -al /usr/lib |grep libevent  
lrwxrwxrwx   1 root    root          21 2009-07-19 08:45 libevent-1.4.so.2 -> libevent-1.4.so.2.1.3  
-rwxr-xr-x   1 root    root      301588 2009-07-19 08:45 libevent-1.4.so.2.1.3  
-rw-r--r--   1 root    root      386638 2009-07-19 08:45 libevent.a  
lrwxrwxrwx   1 root    root          26 2009-07-19 08:45 libevent\_core-1.4.so.2 -> libevent\_core-1.4.so.2.1.3  
-rwxr-xr-x   1 root    root      115721 2009-07-19 08:45 libevent\_core-1.4.so.2.1.3  
-rw-r--r--   1 root    root      151618 2009-07-19 08:45 libevent\_core.a  
-rwxr-xr-x   1 root    root         860 2009-07-19 08:45 libevent\_core.la  
lrwxrwxrwx   1 root    root          26 2009-07-19 08:45 libevent\_core.so -> libevent\_core-1.4.so.2.1.3  
lrwxrwxrwx   1 root    root          27 2009-07-19 08:45 libevent\_extra-1.4.so.2 -> libevent\_extra-1.4.so.2.1.3  
-rwxr-xr-x   1 root    root      239933 2009-07-19 08:45 libevent\_extra-1.4.so.2.1.3  
-rw-r--r--   1 root    root      298406 2009-07-19 08:45 libevent\_extra.a  
-rwxr-xr-x   1 root    root         867 2009-07-19 08:45 libevent\_extra.la  
lrwxrwxrwx   1 root    root          27 2009-07-19 08:45 libevent\_extra.so -> libevent\_extra-1.4.so.2.1.3  
-rwxr-xr-x   1 root    root         825 2009-07-19 08:45 libevent.la  
lrwxrwxrwx   1 root    root          21 2009-07-19 08:45 libevent.so -> libevent-1.4.so.2.1.3

[?](http://www.ericbess.com/ericblog/2008/03/03/wp-codebox/#examples)View Code BASH

|  |  |
| --- | --- |
| 1  2  3  4  5 | <span style="color:#c2cb9;"><strong>wget</strong></span> http:<span style="FONT-WEIGHT: bold">//</span>www.monkey.org<span style="FONT-WEIGHT: bold">/</span>~provos<span style="FONT-WEIGHT: bold">/</span>libevent-2.0.13-stable.tar.gz  <span style="color:#c2cb9;"><strong>tar</strong></span> xzvf libevent-2.0.13-stable.tar.gz  .<span style="FONT-WEIGHT: bold">/</span>configure  <span style="color:#c2cb9;"><strong>make</strong></span>  <span style="color:#c2cb9;"><strong>make</strong></span> <span style="color:#c2cb9;"><strong>install</strong></span> |

1)安装Memcache服务端

sudo apt-get install memcached

安装完Memcache服务端以后，我们需要启动该服务：

memcached -d -m 128 -p 11111 -u root

这里需要说明一下memcached服务的启动参数：

-p 监听的端口  
-l 连接的IP地址, 默认是本机  
-d start 启动memcached服务  
-d restart 重起memcached服务  
-d stop|shutdown 关闭正在运行的memcached服务  
-d install 安装memcached服务  
-d uninstall 卸载memcached服务  
-u 以的身份运行 (仅在以root运行的时候有效)  
-m 最大内存使用，单位MB。默认64MB  
-M 内存耗尽时返回错误，而不是删除项  
-c 最大同时连接数，默认是1024  
-f 块大小增长因子，默认是1.25-n 最小分配空间，key+value+flags默认是48  
-h 显示帮助

查看是否建立成功

telnet测试memcached  
telnet 192.168.1.2 11211

Trying 192.168.1.2...  
Connected to 192.168.1.2.  
Escape character is '^]'

查看版本

version

…

对Memcached缓存服务的状态查询，可以先telnet连接上服务：telnet 127.0.0.1 11211 ，然后使用 stats命令查看缓存服务的状态，会返回如下的数据：      
        time:    1255537291                               服务器当前的unix时间戳   
        total\_items:    54                                     从服务器启动以后存储的items总数量   
        connection\_structures:    19                    服务器分配的连接构造数   
        version:    1.2.6                                        memcache版本   
        limit\_maxbytes:    67108864                    分配给memcache的内存大小（字节）   
        cmd\_get:    1645                                      get命令（获取）总请求次数   
        evictions:    0                                            为获取空闲内存而删除的items数（分配给memcache的空间用满后需   
                                                                         要删除旧的items来得到空间分配给新的items）   
        total\_connections:    19                           从服务器启动以后曾经打开过的连接数   
        bytes:    248723                                      当前服务器存储items占用的字节数   
        threads:    1                                             当前线程数   
        get\_misses:    82                                      总未命中次数   
        pointer\_size:    32                                    当前操作系统的指针大小（32位系统一般是32bit）   
        bytes\_read:    490982                              总读取字节数（请求字节数）   
        uptime:    161                                           服务器已经运行的秒数   
        curr\_connections:    18                             当前打开着的连接数   
        pid:    2816                                               memcache服务器的进程ID   
        bytes\_written:    16517259                     总发送字节数（结果字节数）   
        get\_hits:    1563                                      总命中次数   
        cmd\_set:    54                                          set命令（保存）总请求次数   
        curr\_items:    28                                       服务器当前存储的items数量

Ok,安装memcached1.4.5成功。

初始化：memcache 

**Java代码**[收藏代码](http://dh189.iteye.com/blog/719689)

1. **static** {
2. String[] serverlist = { "server1.com:port", "server2.com:port" };
4. SockIOPool pool = SockIOPool.getInstance();
5. pool.setServers(serverlist);
6. pool.initialize();
7. }

创建一个client对象： 

**Java代码**[收藏代码](http://dh189.iteye.com/blog/719689)

1. MemCachedClient mc = **new** MemCachedClient();

创建一个缓存： 

**Java代码**[收藏代码](http://dh189.iteye.com/blog/719689)

1. MemCachedClient mc = **new** MemCachedClient();
2. String key = "cacheKey1";
3. Object value = SomeClass.getObject();
4. mc.set(key, value);

通过key删除一个缓存： 

**Java代码**[收藏代码](http://dh189.iteye.com/blog/719689)

1. MemCachedClient mc = **new** MemCachedClient();
2. String key = "cacheKey1";
3. mc.delete(key);

通过key获取缓存对象： 

**Java代码**[收藏代码](http://dh189.iteye.com/blog/719689)

1. MemCachedClient mc = **new** MemCachedClient();
2. String key = "key";
3. Object value = mc.get(key);

获取多个缓存对象： 

**Java代码**[收藏代码](http://dh189.iteye.com/blog/719689)

1. MemCachedClient mc = **new** MemCachedClient();
2. String[] keys = { "key", "key1", "key2" };
3. Map<Object> values = mc.getMulti(keys);

刷新全部缓存： 

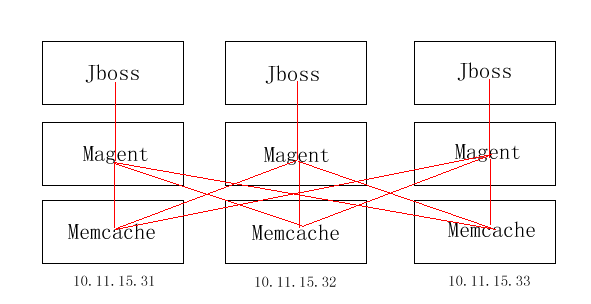
**Java代码**[收藏代码](http://dh189.iteye.com/blog/719689)

1. MemCachedClient mc = **new** MemCachedClient();
2. mc.flushAll();

3. 如何在Java开发中使用Memcache  
  
在Java开发中使用Memcache，一般要用到以下几个程序：  
  
1)      Memcached  
  
该程序用来在Linux或Windows服务器上建立和管理缓存。  
  
其项目网址为：http://danga.com/memcached/。  
  
2)      Magent  
  
Magent是一款开源的Memcached代理服务器软件，使用它可以搭建高可用性的集群应用的Memcached服务，其项目网址为：http://code.google.com/p/memagent/。  
  
3)      Memcached客户端程序  
  
至于Memcached的客户端程序，一般推荐用memcached client for java，为什么推荐用这种客户端，后面会讲到具体的原因，其项目的网址为：http://github.com/gwhalin/Memcached-Java-Client/。  
  
4)      其它程序  
i.              Libevent  
  
在Linux环境下应用Memcache时，Memcache用到了libevent这个库，用于Socket的处理，所以还需要安装libevent。libevent的最新版本是libevent-1.4.13。（如果你的系统已经安装了libevent，可以不用安装）。  
  
官网：http://www.monkey.org/~provos/libevent/  
  
下载：http://www.monkey.org/~provos/libevent-1.4.13-stable.tar.gz  
  
ii.              Windows下的安装程序  
  
Memcache也可以安装在Windows服务器下，安装程序：memcached-1.2.1-win32.zip

可以从这里下载：http://jehiah.cz/projects/memcached-win32/。

四、            原理与部署  
1. magent的hash算法  
  
magent采用的是：Consistent Hashing原理，Consistent Hashing如下所示：首先求出memcached服务器（节点）的哈希值， 并将其配置到0～232的圆（continuum）上。 然后用同样的方法求出存储数据的键的哈希值，并映射到圆上。然后从数据映射到的位置开始顺时针查找，将数据保存到找到的第一个服务器上。 如果超过232仍然找不到服务器，就会保存到第一台memcached服务器上。  
  
从上图的状态中添加一台memcached服务器。余数分布式算法由于保存键的服务器会发生巨大变化 而影响缓存的命中率，但Consistent Hashing中，只有在continuum上增加服务器的地点逆时针方向的第一台服务器上的键会受到影响。  
  
2. 部署示意图  
  
Java开发中的Memcache原理及实现（四）原理与部署



3. 搭建memcache集群服务  
  
利用magent实现对memecache的分布式管理，搭建一套memcache集群服务：  
  
?  前端java对magent的访问跟对memcache访问相同，不需要做任何更改，对于插入的key，magent会把值散列到各个memcache服务上，只操作magent，不用关心后端处理；  
  
?  项目应用：以深圳电信为例，其商呼系统如图部署三台机器做为集群，假设IP分别是：10.11.15.31， 10.11.15.32， 10.11.15.33；  
  
?  每个前端安装memcached服务（大内存机器可以启动多个服务），如端口都为12001，每个前端都安装magent服务，端口都为12000，后端挂载全部机器的memcached服务，  
  
?  启动参数示例：magent -p 12000 -s 10.11.15.31:12001 -s 10.11.15.32:12001 -s 10.11.15.33:12001，这里将三台机器都配置进来，如集集群增加了机器，只需要在启动参数里添加进来即可。所有前端配置都是相同的，任何一个前端只需访问本地端口的magent，这样的memcache集群对应用带来很大便利。  
  
?  这种部署可以解决session共享的应用  
  
项目中多处已经实际应用，magent对memcache的均衡和稳定性都非常不错，推荐使用。

五、            测试Memcached流程  
  
此处以二机集群为例。  
  
1. 启动Memcached及代理  
  
启动两个memcached进程，端口分别为11211和11212：  
  
memcached -m 1 -u root -d -l 127.0.0.1 -p 11211  
  
memcached -m 1 -u root -d -l 127.0.0.1 -p 11212  
  
再启动两个magent进程，端口分别为10000和11000：  
  
magent -u root -n 51200 -l 127.0.0.1 -p 10000 -s 127.0.0.1:11211 -b 127.0.0.1:11212  
  
magent -u root -n 51200 -l 127.0.0.1 -p 11000 -s 127.0.0.1:11212 -b 127.0.0.1:11211  
  
-s 为要写入的memcached， -b 为备份用的memcached。  
  
说明：测试环境用magent和memached的不同端口来实现，在生产环境中可以将magent和memached作为一组放到两台服务器上。也就是说通过magent能够写入两个memcached。  
  
2. 数据读写测试  
  
[root@odb ~]# telnet 127.0.0.1 10000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
set key 0 0 8                       <—在10000端口设置key的值  
  
88888888  
  
STORED  
  
quit  
  
Connection closed by foreign host.  
  
   
  
[root@odb ~]# telnet 127.0.0.1 11211  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
get key                     <—在11211端口获取key的值成功  
  
VALUE key 0 8  
  
88888888  
  
END  
  
quit  
  
Connection closed by foreign host.  
  
   
  
[root@odb ~]# telnet 127.0.0.1 11212  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
get key                     <—在11212端口获取key的值成功  
  
VALUE key 0 8  
  
88888888  
  
END  
  
quit  
  
Connection closed by foreign host.  
  
3. 高可靠性测试  
  
[root@odb ~]# ps aux |grep -v grep |grep memcached  
  
root     23455  0.0  0.0  5012 1796 ?        Ss   09:22   0:00 memcached -m 1 -u root -d -l 127.0.0.1 -p 11212  
  
root     24950  0.0  0.0  4120 1800 ?        Ss   10:58   0:00 memcached -m 1 -u root -d -l 127.0.0.1 -p 11211  
  
[root@odb ~]# ps aux |grep -v grep |grep ‘magent -u’  
  
root     25919  0.0  0.0  2176  484 ?        Ss   12:00   0:00 magent -u root -n 51200 -l 127.0.0.1 -p 10000 -s 127.0.0.1:11211 -b 127.0.0.1:11212  
  
root     25925  0.0  0.0  3004  484 ?        Ss   12:00   0:00 magent -u root -n 51200 -l 127.0.0.1 -p 11000 -s 127.0.0.1:11212 -b 127.0.0.1:11211  
  
   
  
[root@odb ~]# telnet 127.0.0.1 10000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
set stone 0 0 6                      <—在10000端口设置stone的值  
  
123456  
  
STORED  
  
quit  
  
Connection closed by foreign host.  
  
   
  
[root@odb ~]# telnet 127.0.0.1 11000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
set shidl 0 0 6                 <—在11000端口设置shidl的值  
  
666666  
  
STORED  
  
get stone                     <—在11000端口获取stone的值成功  
  
VALUE stone 0 6  
  
123456  
  
END  
  
incr stone 2                   <—在11000端口修改stone的值成功  
  
123458  
  
get stone  
  
VALUE stone 0 6               <—在11000端口验证stone的值，证明上面的修改成功  
  
123458  
  
END  
  
get shidl                     <—在11000端口获取shidl的值成功  
  
VALUE shidl 0 6  
  
666666  
  
END  
  
quit                             <—退出11000端口  
  
Connection closed by foreign host.  
  
   
  
[root@odb ~]# telnet 127.0.0.1 10000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
get stone                     <—在10000端口获取stone的值，已被修改  
  
VALUE stone 0 6  
  
123458  
  
END  
  
get shidl                      <—在10000端口获取shidl的值成功  
  
VALUE shidl 0 6  
  
666666  
  
END  
  
delete shidl                   <—在10000端口删除shidl  
  
DELETED  
  
get shidl                      <—在10000端口删除shidl生效  
  
END  
  
quit  
  
Connection closed by foreign host.  
  
   
  
[root@odb ~]# telnet 127.0.0.1 11000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
get shidl                      <—在11000端口验证删除shidl生效  
  
END  
  
get stone                     <—在11000端口获取stone的值成功  
  
VALUE stone 0 6  
  
123458  
  
END  
  
quit  
  
Connection closed by foreign host.  
  
4. Down机模拟测试1  
1)      Down掉11211端口的memcached  
  
[root@odb ~]# kill -9 24950  
  
[root@odb ~]# telnet 127.0.0.1 10000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
get stone                      <—在10000依然可以获取stone的值  
  
VALUE stone 0 6  
  
123458  
  
END  
  
quit  
  
Connection closed by foreign host.  
  
   
  
[root@odb ~]# telnet 127.0.0.1 11000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
get stone                      <—在11000依然可以获取stone的值  
  
VALUE stone 0 6  
  
123458  
  
END  
  
quit  
  
Connection closed by foreign host.  
  
   
  
5. Down机模拟测试2  
1)      Down掉11000端口的magent  
  
[root@odb ~]# kill -9 25925  
  
[root@odb ~]# telnet 127.0.0.1 10000  
  
Trying 127.0.0.1…  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is ‘^]’.  
  
get stone                      <—在10000依然可以获取stone的值  
  
VALUE stone 0 6  
  
123458  
  
END  
  
quit  
  
Connection closed by foreign host.  
  
   
  
2)      重启11000端口的magent  
  
[root@lh-web-test memcached-1.4.5]# magent -u root -n 51200 -l 127.0.0.1 -p 11000 -s 127.0.0.1:11212 -b 127.0.0.1:11211  
  
[root@lh-web-test memcached-1.4.5]# telnet 127.0.0.1 11000  
  
Trying 127.0.0.1...  
  
Connected to localhost.localdomain (127.0.0.1).  
  
Escape character is '^]'.  
  
get stone                                         <—在11000依然可以获取stone的值  
  
VALUE stone 0 6  
  
123458  
  
END  
  
quit  
  
Connection closed by foreign host.

七、            Memcached客户端程序

Memcached的java客户端已经存在三种了：

?  官方提供的基于传统阻塞io由Greg Whalin维护的客户端

?  Dustin Sallings实现的基于java nio的Spymemcached

?  XMemcached

1. 三种API比较   
1)      memcached client for java

较早推出的memcached JAVA客户端API，应用广泛，运行比较稳定。

2)      spymemcached

A simple, asynchronous, single-threaded memcached client written in java. 支持异步，单线程的memcached客户端，用到了java1.5版本的concurrent和nio，存取速度会高于前者，但是稳定性不好，测试中常报timeOut等相关异常。

3)      xmemcached

XMemcached同样是基于java nio的客户端，java nio相比于传统阻塞io模型来说，有效率高（特别在高并发下）和资源耗费相对较少的优点。传统阻塞IO为了提高效率，需要创建一定数量的连接形成连接池，而nio仅需要一个连接即可（当然,nio也是可以做池化处理），相对来说减少了线程创建和切换的开销，这一点在高并发下特别明显。因此XMemcached与Spymemcached在性能都非常优秀，在某些方面（存储的数据比较小的情况下）Xmemcached比Spymemcached的表现更为优秀，具体可以看这个Java Memcached Clients Benchmark。

2.  建议

由于memcached client for java发布了新版本，性能上有所提高，并且运行稳定，所以建议使用memcached client for java。

XMemcached也使用得比较广泛，而且有较详细的中文API文档，具有如下特点：高性能、支持完整的协议、支持客户端分布、允许设置节点权重、动态增删节点、支持JMX、与Spring框架和Hibernate-memcached的集成、客户端连接池、可扩展性好等。

下面给出这三种客户端的示例程序。

3.  示例程序  
1)      memcached client for java

从前面介绍的Java环境的Memcached客户端程序项目网址里，下载最新版的客户端程序包：java\_memcached-release\_2.5.1.zip，解压后，文件夹里找到java\_memcached-release\_2.5.1.jar，这个就是客户端的JAR包。将此JAR包添加到项目的构建路径里，则项目中，就可以使用Memcached了。

示例代码如下：

package temp;

import com.danga.MemCached.\*;

import org.apache.log4j.\*;

public class CacheTest {

    public static void main(String[] args) {

       /\*\*

        \* 初始化SockIOPool，管理memcached的连接池

        \* \*/

       String[] servers = { "10.11.15.222:10000" };

       SockIOPool pool = SockIOPool.getInstance();

       pool.setServers(servers);

       pool.setFailover(true);

       pool.setInitConn(10);

       pool.setMinConn(5);

       pool.setMaxConn(250);

       pool.setMaintSleep(30);

       pool.setNagle(false);

       pool.setSocketTO(3000);

       pool.setAliveCheck(true);

       pool.initialize();

       /\*\*

        \* 建立MemcachedClient实例

        \* \*/

       MemCachedClient memCachedClient = new MemCachedClient();

       for (int i = 0; i < 1000; i++) {

           /\*\*

            \* 将对象加入到memcached缓存

            \* \*/

           boolean success = memCachedClient.set("" + i, "Hello!");

           /\*\*

            \* 从memcached缓存中按key值取对象

            \* \*/

           String result = (String) memCachedClient.get("" + i);

           System.out.println(String.format("set( %d ): %s", i, success));

           System.out.println(String.format("get( %d ): %s", i, result));

       }

    }

}

2)      spymemcached

spymemcached当前版本是2.5版本，官方网址是：http://code.google.com/p/spymemcached/。可以从地址：<http://spymemcached.googlecode.com/files/memcached-2.5.jar>下载最新版本来使用。

示例代码如下：

package temp;

import java.net.InetSocketAddress;

import java.util.concurrent.Future;

import net.spy.memcached.MemcachedClient;

public class TestSpyMemcache {

    public static void main(String[] args) {

       // 保存对象

       try {

           /\* 建立MemcachedClient 实例，并指定memcached服务的IP地址和端口号 \*/

           MemcachedClient mc = new MemcachedClient(new InetSocketAddress("10.11.15.222", 10000));

           Future<Boolean> b = null;

           /\* 将key值，过期时间(秒)和要缓存的对象set到memcached中 \*/

           b = mc.set("neea:testDaF:ksIdno", 900, "someObject");

           if (b.get().booleanValue() == true) {

              mc.shutdown();

           }

       } catch (Exception ex) {

           ex.printStackTrace();

       }

       // 取得对象

        try {

           /\* 建立MemcachedClient 实例，并指定memcached服务的IP地址和端口号 \*/

           MemcachedClient mc = new MemcachedClient(new InetSocketAddress("10.11.15.222", 10000));

           /\* 按照key值从memcached中查找缓存，不存在则返回null \*/

           Object b = mc.get("neea:testDaF:ksIdno");

           System.out.println(b.toString());

           mc.shutdown();

       } catch (Exception ex) {

           ex.printStackTrace();

       }

    }

}

3)      xmemcached

Xmemcached的官方网址是：[http://code.google.com/p/xmemcached/，可以从其官网上下载最新版本的1.2.4](http://code.google.com/p/xmemcached/%EF%BC%8C%E5%8F%AF%E4%BB%A5%E4%BB%8E%E5%85%B6%E5%AE%98%E7%BD%91%E4%B8%8A%E4%B8%8B%E8%BD%BD%E6%9C%80%E6%96%B0%E7%89%88%E6%9C%AC%E7%9A%841.2.4)来使用。地址是：<http://xmemcached.googlecode.com/files/xmemcached-1.2.4-src.tar.gz>。

示例代码如下：

package temp;

import java.io.IOException;

import java.util.concurrent.TimeoutException;

import net.rubyeye.xmemcached.utils.AddrUtil;

import net.rubyeye.xmemcached.MemcachedClient;

import net.rubyeye.xmemcached.MemcachedClientBuilder;

import net.rubyeye.xmemcached.XMemcachedClientBuilder;

import net.rubyeye.xmemcached.exception.MemcachedException;

public class TestXMemcache {

    public static void main(String[] args) {

       MemcachedClientBuilder builder = new XMemcachedClientBuilder(AddrUtil

              .getAddresses("10.11.15.222:10000"));

       MemcachedClient memcachedClient;

       try {

           memcachedClient = builder.build();

           memcachedClient.set("hello", 0, "Hello,xmemcached");

           String value = memcachedClient.get("hello");

           System.out.println("hello=" + value);

           memcachedClient.delete("hello");

           value = memcachedClient.get("hello");

           System.out.println("hello=" + value);

           // close memcached client

           memcachedClient.shutdown();

       } catch (MemcachedException e) {

           System.err.println("MemcachedClient operation fail");

           e.printStackTrace();

       } catch (TimeoutException e) {

           System.err.println("MemcachedClient operation timeout");

           e.printStackTrace();

       } catch (InterruptedException e) {

           // ignore

       }catch (IOException e) {

           System.err.println("Shutdown MemcachedClient fail");

           e.printStackTrace();

       }

    }

}

(八)  64位机器安装Memcache  
1.   安装  
  
在64位的机器上安装Memcache和在32位的机器上安装的操作是一样的。在安装的过程中，可以使用如下的命令来查看安装是否成功，以进行确认。  
  
1)   确认libevent安装  
  
查看libevent是否安装成功：  
  
# ls -al /usr/lib | grep libevent  
  
在命令行出现如下信息，表明安装成功：  
  
   lrwxrwxrwx   1 root root     21 Mar 22 18:41 libevent-1.2.so.1 -> libevent-1.2.so.1.0.3  
  
   -rwxr-xr-x   1 root root 262475 Mar 22 18:41 libevent-1.2.so.1.0.3  
  
   -rw-r--r--   1 root root 430228 Mar 22 18:41 libevent.a  
  
   -rwxr-xr-x   1 root root    811 Mar 22 18:41 libevent.la  
  
   lrwxrwxrwx   1 root root     21 Mar 22 18:41 libevent.so -> libevent-1.2.so.1.0.3  
  
2)   确认memcache安装  
  
查看memcache是否安装成功：  
  
# ls -al /usr /bin/mem\*  
  
在命令行出现如下信息，表明安装成功：  
  
   -rwxr-xr-x  1 root root 114673 Mar 22 18:52 /usr/local/src/memcached  
  
   -rwxr-xr-x  1 root root 120092 Mar 22 18:52 /usr/local/src/memcached-debug  
  
2.   64位的问题及修复  
1)   问题  
  
安装完成了，现在我们看一下memcache的帮助：  
  
#/usr/local/src/memecached -h  
  
这时候出现了如下错误：  
  
   memcached: error while loading shared libraries: libevent-1.2.so.1: cannot open shared    object file: No such file or directory  
  
2)   修复  
  
下面说下修复过程：  
  
#LD\_DEBUG=libs memcached -v #查看memcached的libs的路径  
  
在命令上出现了如下信息：  
  
5427:     find library=libevent-1.2.so.1 [0]; searching  
  
5427:      search cache=/etc/ld.so.cache  
  
5427: search        path=/lib64/tls/x86\_64:/lib64/tls:/lib64/x86\_64:/lib64:/usr/lib64/tls/x86\_64:/usr/lib64/tls:/usr/lib64/x86\_64:  
  
/usr/lib64              (system search path)  
  
5427:       trying file=/lib64/tls/x86\_64/libevent-1.2.so.1  
  
5427:       trying file=/lib64/tls/libevent-1.2.so.1  
  
5427:       trying file=/lib64/x86\_64/libevent-1.2.so.1  
  
5427:       trying file=/lib64/libevent-1.2.so.1  
  
5427:       trying file=/usr/lib64/tls/x86\_64/libevent-1.2.so.1  
  
5427:       trying file=/usr/lib64/tls/libevent-1.2.so.1  
  
5427:       trying file=/usr/lib64/x86\_64/libevent-1.2.so.1  
  
5427:       trying file=/usr/lib64/libevent-1.2.so.1  
  
5427:            memcached: error while loading shared libraries: libevent-1.2.so.1: cannot open shared object file: No such                   file or directory  
  
现在应该记录下来libs的位置，我选择的是trying file=/usr/lib64/libevent-1.2.so.1，现在我们利用这个来做个符号链接:  
  
# ln -s /usr/lib/libevent-1.4.so.2 /usr/lib64/libevent-1.4.so.2  
  
下面我们继续使用memcached -h做下测试，终于出现了如下信息:  
  
    memcached 1.2.0  
  
    -p <num>      port number to listen on  
  
    -s <file>     unix socket path to listen on (disables network support)  
  
    -l <ip\_addr>  interface to listen on, default is INDRR\_ANY  
  
    -d            run as a daemon  
  
    -r            maximize core file limit  
  
    -u <username> assume identity of <username> (only when run as root)  
  
    -m <num>      max memory to use for items in megabytes, default is 64 MB  
  
    -M            return error on memory exhausted (rather than removing items)  
  
    -c <num>      max simultaneous connections, default is 1024  
  
    -k            lock down all paged memory  
  
    -v            verbose (print errors/warnings while in event loop)  
  
    -vv           very verbose (also print client commands/reponses)  
  
    -h            print this help and exit  
  
    -i            print memcached and libevent license  
  
    -b            run a managed instanced (mnemonic: buckets)  
  
    -P <file>     save PID in <file>, only used with -d option  
  
    -f <factor>   chunk size growth factor, default 1.25  
  
    -n <bytes>    minimum space allocated for key+value+flags, default 48  
  
说明memcached安装成功。（应该是机器是64位的原因，所以将so文件放到了lib64下面，而不是lib下面，使得memcached找不到了so文件）。  
  
下面，我们来启动一个Memcached的服务器端：  
  
# /usr/local/src/memcached -d -m 10  -u root -l 192.168.0.200 -p 12000 -c 256 -P /tmp/memcached.pid

**(九)    Windows下的Memcache安装**1.  安装

在这里简单介绍一下Windows下的Memcache的安装：

1. 下载memcache的windows稳定版，解压放某个盘下面，比如在c:\memcached

2. 在终端（也即cmd命令界面）下输入‘c:\memcached\memcached.exe -d install’安装

3. 再输入：‘c:\memcached\memcached.exe -d start’启动。NOTE: 以后memcached将作为windows的一个服务每次开机时自动启动。这样服务器端已经安装完毕了。

2.  memcached的基本设置

?  -p 监听的端口

?  -l 连接的IP地址, 默认是本机

?  -d start 启动memcached服务

?  -d restart 重起memcached服务

?  -d stop|shutdown 关闭正在运行的memcached服务

?  -d install 安装memcached服务

?  -d uninstall 卸载memcached服务

?  -u 以的身份运行 (仅在以root运行的时候有效)

?  -m 最大内存使用，单位MB。默认64MB

?  -M 内存耗尽时返回错误，而不是删除项

?  -c 最大同时连接数，默认是1024

?  -f 块大小增长因子，默认是1.25

?  -n 最小分配空间，key+value+flags默认是48

?  -h 显示帮助

3.  设置Memcache缓存大小和端口

Memcache的默认启动时的参数可能不满足实际生产环境的需要，于是就想到直接修改windows服务的启动参数，操作如下：

打开注册表，找到：HKEY\_LOCAL\_MACHINE\SYSTEM\CurrentControlSet\Services\memcached Server

其中的ImagePath项的值为： c:\memcached\memcached.exe" -d runservice

改成：c:\memcached\memcached.exe" -p 12345 -m 128 -d runservice

其中，-p就是端口，-m就是缓存大小，以M为单位。

#### 在CentOS 5.6上编译安装Memcached

**1、**由于memcached是基于libevent的，因此需要安装libevent，libevent-devel

view plain   [copy](http://www.sunchis.com/html/db/memcached/2011/0526/333.html)

1. # yum install libevent libevent-devel -y

**2、**下载并解压memcached-1.4.5

memcached官方网站是：<http://memcached.org/>

view plain   [copy](http://www.sunchis.com/html/db/memcached/2011/0526/333.html)

1. # cd /root
2. # wget http://memcached.googlecode.com/files/memcached-1.4.5.tar.gz
3. # tar -xvzf  memcached-1.4.5.tar.gz

**3、**编译安装memcached-1.4.5

view plain   [copy](http://www.sunchis.com/html/db/memcached/2011/0526/333.html)

1. # cd memcached-1.4.5
2. # ./configure --prefix=/etc/memcached
3. # make
4. # make install

**4、**配置环境变量

进入用户宿主目录，编辑.bash\_profile，为系统环境变量LD\_LIBRARY\_PATH增加新的目录，需要增加的内容如下：

# vi .bash\_profile

view plain   [copy](http://www.sunchis.com/html/db/memcached/2011/0526/333.html)

1. MEMCACHED\_HOME=/etc/memcached
2. export LD\_LIBRARY\_PATH=$LD\_LIBRARY\_PATH:$MEMCACHED\_HOME/lib

刷新用户环境变量：# source .bash\_profile

**5、**编写memcached服务启停脚本

# cd /etc/init.d

vi memcached，脚本内容如下：

view plain   [copy](http://www.sunchis.com/html/db/memcached/2011/0526/333.html)

1. #!/bin/sh
2. #
3. # Startup script for the server of memcached
4. #
5. # processname: memcached
6. # pidfile: /etc/memcached/memcached.pid
7. # logfile: /etc/memcached/memcached\_log.txt
8. # memcached\_home: /etc/memcached
9. # chkconfig: 35 21 79
10. # description: Start and stop memcached Service
12. # Source function library
13. . /etc/rc.d/init.d/functions
15. RETVAL=0
17. prog="memcached"
18. basedir=/etc/memcached
19. cmd=${basedir}/bin/memcached
20. pidfile="$basedir/${prog}.pid"
21. #logfile="$basedir/memcached\_log.txt"
23. # 设置memcached启动参数
24. ipaddr="192.168.1.201"    # 绑定侦听的IP地址
25. port="11211"                    # 服务端口
26. username="root"                 # 运行程序的用户身份
27. max\_memory=64                   # default: 64M | 最大使用内存
28. max\_simul\_conn=1024             # default: 1024 | 最大同时连接数
29. #maxcon=51200
30. #growth\_factor=1.3              # default: 1.25 | 块大小增长因子
31. #thread\_num=6                   # default: 4
32. #verbose="-vv"                  # 查看详细启动信息
33. #bind\_protocol=binary           # ascii, binary, or auto (default)
35. start() {
36. echo -n $"Starting service: $prog"
37. $cmd -d -m $max\_memory -u $username -l $ipaddr -p $port -c $max\_simul\_conn -P $pidfile
38. RETVAL=$?
39. echo
40. [ $RETVAL -eq 0 ] && touch /var/lock/subsys/$prog
41. }
43. stop() {
44. echo -n $"Stopping service: $prog  "
45. run\_user=`whoami`
46. pidlist=`ps -ef | grep $run\_user | grep memcached | grep -v grep | awk '{print($2)}'`
47. for pid in $pidlist
48. do
49. #           echo "pid=$pid"
50. kill -9 $pid
51. if [ $? -ne 0 ]; then
52. return 1
53. fi
54. done
55. RETVAL=$?
56. echo
57. [ $RETVAL -eq 0 ] && rm -f /var/lock/subsys/$prog
58. }
60. # See how we were called.
61. case "$1" in
62. start)
63. start
64. ;;
65. stop)
66. stop
67. ;;
68. #reload)
69. #    reload
70. #    ;;
71. restart)
72. stop
73. start
74. ;;
75. #condrestart)
76. #    if [ -f /var/lock/subsys/$prog ]; then
77. #        stop
78. #        start
79. #    fi
80. #    ;;
81. status)
82. status memcached
83. ;;
84. \*)
85. echo "Usage: $0 {start|stop|restart|status}"
86. exit 1
87. esac
89. exit $RETVAL

设置脚本可被执行：# chmod +x memcached

**6、**设置memcached随系统启动

view plain   [copy](http://www.sunchis.com/html/db/memcached/2011/0526/333.html)

1. # chkconfig --add memcached
2. # chkconfig --level 35 memcached on

#### 启动memcached

view plain   [copy](http://www.sunchis.com/html/db/memcached/2011/0526/333.html)

1. # service memcached start
2. //启动的时候实际上是调用了下面的这个命令，以守护进程的方式来启动memcached
3. /etc/memcached/bin/memcached -d -m 64 -u root -l 192.168.1.201
4. \-p 11211 -c 1024 -P /etc/memcached/memcached.pid

查看memcached是否启动：

# ps -ef | grep memcached

#### memcached命令参数解释

| 参数 | 参数解释及说明 |
| --- | --- |
| -p <num> | 监听的端口 |
| -l <ip\_addr> | 连接的IP地址,，默认是本机。-l选项可以不使用，此时表示在所有网络接口地址上监听。建议是-l <ip\_addr>指定一个内部网络IP地址，以避免成为外部网络攻击的对象 |
| -d start | 启动memcached 服务 |
| -d restart | 重起memcached 服务 |
| -d stop|shutdown | 关闭正在运行的memcached 服务 |
| -d install | 安装memcached 服务 |
| -d uninstall | 卸载memcached 服务 |
| -u <username> | 以<username>的身份运行 (仅在以root运行的时候有效) |
| -m <num> | 最大内存使用，单位MB。默认64MB |
| -M | 内存耗尽时返回错误，而不是删除项 |
| -c <num> | 最大同时连接数，默认是1024 |
| -f <factor> | 块大小增长因子，默认是1.25 |
| -n <bytes> | 最小分配空间，key+value+flags默认是48 |
| -h | 显示帮助 |