<https://www.cnblogs.com/lixuwu/p/7446170.html>

* [1 了解memcache](https://www.cnblogs.com/lixuwu/p/7446170.html#_label0)
* [2 在java中使用memcache](https://www.cnblogs.com/lixuwu/p/7446170.html#_label1)

背景：公司项目中使用java和memcache相结合来搭建缓存，所以要了解下缓存的基础知识！

[回到顶部](https://www.cnblogs.com/lixuwu/p/7446170.html#_labelTop)

# 1 了解memcache

## 1.1 基础知识

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](http://lib.csdn.net/base/mysql)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。

## 1.2 linux服务器使用说明

**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

[复制代码](javascript:void(0);)

-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

[复制代码](javascript:void(0);)

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。

**linux下的Memcache安装**

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：

[复制代码](javascript:void(0);)

-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

[复制代码](javascript:void(0);)

查看是否建立成功

telnet[测试](http://lib.csdn.net/base/softwaretest)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：

[复制代码](javascript:void(0);)

　　　　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）

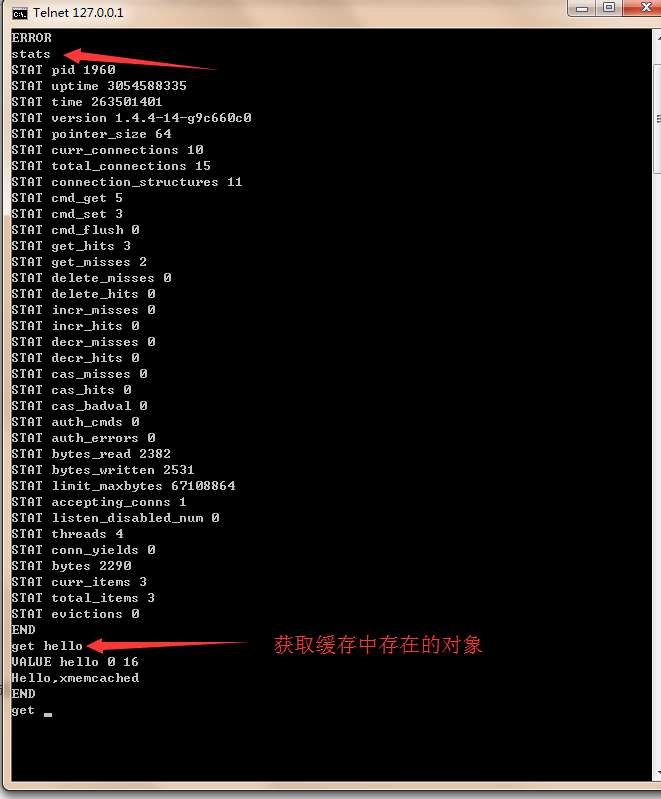
get\_hits: 1563 总命中次数

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

[复制代码](javascript:void(0);)

ps：windows下同样可以。



Ok,安装memcached1.4.5成功。

## 1.3 数据读写测试

一些memcache相关命令可以慢慢搜索积累。

[复制代码](javascript:void(0);)

[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.

[复制代码](javascript:void(0);)

[回到顶部](https://www.cnblogs.com/lixuwu/p/7446170.html#_labelTop)

# 2 在java中使用memcache

## 2.1 基本使用方法

初始化memcache：

[复制代码](javascript:void(0);)

static {

String[] serverlist = { "server1.com:port", "server2.com:port" };

SockIOPool pool = SockIOPool.getInstance();

pool.setServers(serverlist);

pool.initialize();

}

[复制代码](javascript:void(0);)

创建一个client对象：

MemCachedClient mc = new MemCachedClient();

创建一个缓存：

MemCachedClient mc = new MemCachedClient();

String key = "cacheKey1";

Object value = SomeClass.getObject();

mc.set(key, value);

通过key删除一个缓存：

MemCachedClient mc = new MemCachedClient();

String key = "cacheKey1";

mc.delete(key);

通过key获取缓存对象：

MemCachedClient mc = new MemCachedClient();

String key = "key";

Object value = mc.get(key);

获取多个缓存对象：

MemCachedClient mc = new MemCachedClient();

String[] keys = { "key", "key1", "key2" };

Map<Object> values = mc.getMulti(keys);

刷新全部缓存：

MemCachedClient mc = new MemCachedClient();

mc.flushAll();

## 2.2 memcache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](http://lib.csdn.net/base/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](http://lib.csdn.net/base/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](http://lib.csdn.net/base/javaee)框架和[hibernate](http://lib.csdn.net/base/javaee)-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。

## 2.3 三种api的示例程序

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：

[复制代码](javascript:void(0);)

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));

}

}

}

[复制代码](javascript:void(0);)

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：

[复制代码](javascript:void(0);)

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();

}

}

}

[复制代码](javascript:void(0);)

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：

[复制代码](javascript:void(0);)

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();

}

}

}

[复制代码](javascript:void(0);)

如果您觉得阅读本文对您有帮助，请点一下“**推荐**”按钮，让更多的人能够享受到获取知识的快乐！因为本人初入职场，鉴于自身阅历有限，**所以本博客内容大部分来源于网络中已有知识的汇总**，欢迎各位转载，评论，大家一起学习进步！如有侵权，请及时和我联系，切实维护您的权益！