Memcached

# 概述

## Memcache

Memcache(Memcached)是集群环境下的缓存解决方案。Memcache 是danga.com的一个项目，最早是为 LiveJournal 服务的，目前全世界不少人使用这个缓存项目来构建自己大负载的网站，来分担数据库的压力。它可以应对任意多个连接，使用非阻塞的网络IO。它的工作机制是 在内存中开辟一块空间，然后建立一个HashTable，Memcached自管理这些HashTable。

## 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。

## 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

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/](http://github.com/gwhalin/Memcached-Java-Client/downloads)。

## 4)      其它程序

### 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。

### Windows

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/>。

# 相关命令行

## 启动一个memcached

### 命令实例

进入到memcached的安装目录，如：

#cd /usr/local/memcached-1.4.5

# ./memcached -d -m 10 -u root -l 10.11.15.222 -p 12000 -c 256 -P /tmp/memcached.pid

### 命令详解

?  -d  选项是启动一个守护进程，

?  -m  是分配给Memcache使用的内存数量，单位是MB，我这里是10MB，

?  -u  是运行Memcache的用户，我这里是root，

?  -l  是监听的服务器IP地址，我这里指定了服务器的IP地址10.11.15.222，

?  -p  是设置Memcache监听的端口，我这里设置了12000，最好是1024以上的端口，

?  -c  是最大运行的并发连接数，默认1024，这里设置了256，按照服务器的负载量来设定

?  -P  是设置保存Memcache的pid文件，我这里是保存在/tmp/memcached.pid，

## 结束一个memcached进程

### 命令实例

如果要结束Memcache进程，执行：

# kill `cat /tmp/memcached.pid`

?  注意，上面命令中的符号是 `，不是单引号’

也可以启动多个守护进程，不过端口不能重复。

## 启动Magent代理

### 命令实例

Magent已保存到/usr/bin目录下，可以直接执行该命令。如：

#magent -u root -n 51200 -l 127.0.0.1 -p 12000 -s 127.0.0.1:11211 -s 127.0.0.1:11212 -b 127.0.0.1:11213

### 命令详解

?  -h  this message

?  -u  uid

?  -g  gid

?  -p  port, default is 11211. (0 to disable tcp support)

?  -s  ip:port, set memcached server ip and port

?  -b  ip:port, set backup memcached server ip and port

?  -l  ip, local bind ip address, default is 0.0.0.0

?  -n  number, set max ...

# 原理与部署

## magent

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。

## 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

## 测试memcache实战

### 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。

### 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 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.

### 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

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.

### Down

## Memcached java客户端

* 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

### 三种API比较

* 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。
* 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。
* 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。

### 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。

# FAQ

## 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。