Java 开发中的 Memcache 原理及实现

作者：jiaxiaoyuan1204

整理：chaijunkun

来源：http://jiaxiaoyuan1204.blog.163.com/

一、概述

1. Memcache 是什么

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。

Memcache 官方网站：http://www.danga.com/memcached，更多详细的信息可以来这里了解。

2. 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。

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/。

二、服务器端安装

1. 编译安装 libevent

wget http://monkey.org/~provos/libevent-1.4.9-stable.tar.gz
tar zxvf libevent-1.4.9-stable.tar.gz
cd libevent-1.4.9-stable/
./configure --prefix=/usr
make && make install
cd ../

2. 编译安装 Memcached

Memcached 的最新版本是 1.4.5，安装包为：memcached-1.4.5.tar.gz。
wget http://memcached.googlecode.com/files/memcached-1.4.5.tar.gz
tar zxvf memcached-1[1].4.5.tar.gz
cd memcached-1.4.5/
./configure --with-libevent=/usr
make && make install
cd ../

3. 编译安装 magent

Magent 的最新版本是 0.6，安装包为：magent-0.6.tar.gz。
mkdir magent
cd magent/
wget http://memagent.googlecode.com/files/magent-0.6.tar.gz
tar zxvf magent-0.5.tar.gz
/sbin/ldconfig
sed -i "s#LIBS = -levent#LIBS = -levent -lm#g" Makefile
make
cp magent /usr/bin/magent
cd ../

三、启动和结束服务

1. 启动一个 Memcache 的服务器端

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

? -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，

2. 结束一个 Memcache 进程

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。

3. 启动 Magent 代理

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

命令参数：

? -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 ...

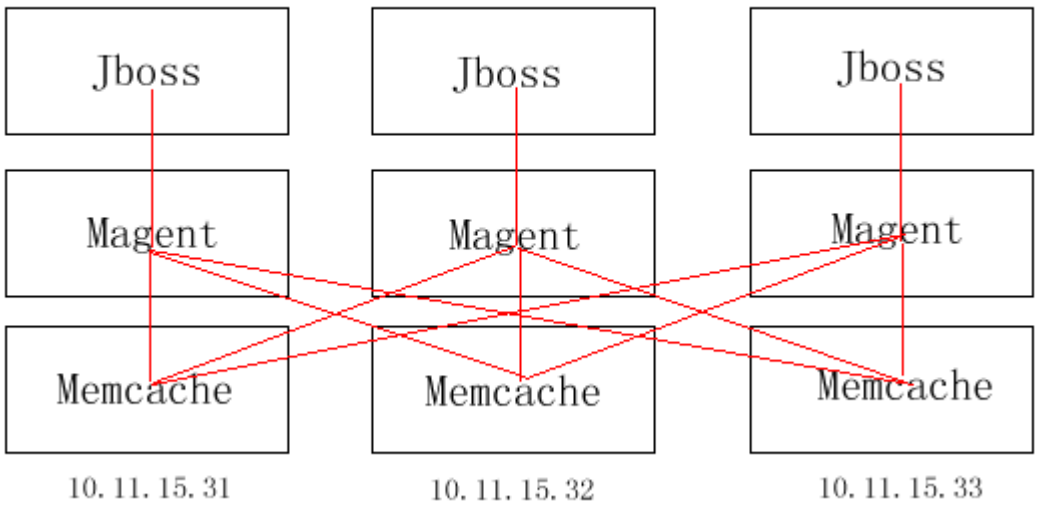
四、原理与部署

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



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.
```

六、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

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 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。