

## Memcached 部署常用设置

### 一、memcached 服务器的典型配置

- CPU: >1GHz
- 内存: >2GB
- 硬盘: >8GB
- 操作系统: Linux (x86\_64)
- 软件环境说明: Centos6.5 x64+libevent+memcached

### 二、Linux 下安装

#### 1、软件下载

**libevent** 下载地址:

<https://github.com/downloads/libevent/libevent/libevent-2.0.21-stable.tar.gz>

**memcached** 下载地址:

<http://memcached.org/files/memcached-1.4.20.tar.gz>

#### 2、软件安装

首先, memcached 是基于 libevent 的, 所以确保你的系统已经装了 libevent, libevent 是个程序库, 它将 Linux 的 epoll、BSD 类操作系统的 kqueue 等事件处理功能封装成统一的接口。即使对服务器的连接数增加, 也能发挥  $O(1)$  的性能。memcached 使用这个 libevent 库, 因此能在 Linux、BSD、Solaris 等操作系统上发挥其高性能。

安装 libevent

```
# tar xzvf libevent-2.0.21-stable.tar.gz ##解压
# cd libevent-2.0.21-stable
# ./configure --prefix=/usr
# make
# make install
```

安装完后可以查看下 /usr/lib 是否有 libevent 等文件(`ls -al /usr/lib | grep libevent`)

```
[root@localhost libevent-2.0.21-stable]# ls -al /usr/lib | grep libevent
lrwxrwxrwx. 1 root root      21 May 13 23:59 libevent-2.0.so.5 -> libevent-2.0.so.5.1.9
-rwxr-xr-x. 1 root root 968690 May 13 23:59 libevent-2.0.so.5.1.9
-rw-r--r--. 1 root root 1571930 May 13 23:59 libevent.a
lrwxrwxrwx. 1 root root      26 May 13 23:59 libevent_core-2.0.so.5 -> libevent_core-2.0.so.5.1.9
-rwxr-xr-x. 1 root root 585233 May 13 23:59 libevent_core-2.0.so.5.1.9
-rw-r--r--. 1 root root 978594 May 13 23:59 libevent_core.a
-rwxr-xr-x. 1 root root      970 May 13 23:59 libevent_core.la
lrwxrwxrwx. 1 root root      26 May 13 23:59 libevent_core.so -> libevent_core-2.0.so.5.1.9
lrwxrwxrwx. 1 root root      27 May 13 23:59 libevent_extra-2.0.so.5 -> libevent_extra-2.0.so.5.1.9
-rwxr-xr-x. 1 root root 404860 May 13 23:59 libevent_extra-2.0.so.5.1.9
-rw-r--r--. 1 root root 593408 May 13 23:59 libevent_extra.a
-rwxr-xr-x. 1 root root      977 May 13 23:59 libevent_extra.la
lrwxrwxrwx. 1 root root      27 May 13 23:59 libevent_extra.so -> libevent_extra-2.0.so.5.1.9
-rwxr-xr-x. 1 root root      935 May 13 23:59 libevent.la
lrwxrwxrwx. 1 root root      30 May 13 23:59 libevent_pthreads-2.0.so.5 -> libevent_pthreads-2.0.so.5.1.9
-rwxr-xr-x. 1 root root 18438 May 13 23:59 libevent_pthreads-2.0.so.5.1.9
-rw-r--r--. 1 root root 18678 May 13 23:59 libevent_pthreads.a
-rwxr-xr-x. 1 root root      998 May 13 23:59 libevent_pthreads.la
lrwxrwxrwx. 1 root root      30 May 13 23:59 libevent_pthreads.so -> libevent_pthreads-2.0.so.5.1.9
lrwxrwxrwx. 1 root root      21 May 13 23:59 libevent.so -> libevent-2.0.so.5.1.9
```

### 安装 memcached

```
# tar xzvf memcached-1.4.20.tar.gz
```

```
# cd memcached-1.4.20
```

```
# ./configure --with-libevent=/usr
```

```
# make
```

```
# make install
```

安装结果(ls -al /usr/local/bin/memcached)如图:

```
[root@localhost lantian]# ls -al /usr/local/bin/memcached
-rwxr-xr-x. 1 root root 341796 Apr 26 19:37 /usr/local/bin/memcached
[root@localhost lantian]#
```

### 启动 memcached

```
#/usr/local/bin/memcached -d -m 1024 -u root -p 11211 -P /tmp/memcached.pid
```

启动后查看进程:

```
[root@localhost ~]# ps -ef|grep memcached
nobody  4009      1  0 16:39 ?        00:00:00 memcached -d -u nobody -m 1024 -p 11211 -c 2048 -P /tmp/memcached.pid -vv
root    4141  2424  0 17:02 pts/0    00:00:00 grep  memcached
```

说明已经正常启动。

### 常用推荐启动配置

方式一：内存为 2G

```
memcached -d -u root -m 1024 -p 11211 -c 2048 -P /tmp/memcached.pid
```

方式二：内存 4G 以上

```
memcached -d -u root -m 3000 -p 11211 -c 30720 -P /tmp/memcached.pid
```

启动过程中所用选项说明如下:

- p, 使用的 TCP 端口。默认为 11211。
- m, 最大内存大小。默认为 64MB。
- vv, 以 very verbose 模式启动, 将调试信息和错误输出到控制台。
- d, 作为守护进程在后台运行。
- c, 最大运行的并发连接数, 默认是 1024, 按照服务器的负载量来设定。
- P, 设置保存 Memcache 的 pid 文件。
- l, 绑定地址(默认:所有都允许,无论内外网或者本机更换 IP, 有安全隐患, 若设置为 127.0.0.1 就只能本机访问)。
- u, 运行 Memcached 的用户, 默认不能由 root 用户启动, 所以当前用户为 root 用户时, 需要利用 -u 参数来指定。
- f, 块大小增长因子, 默认是 1.25, 根据存入的数据的大小可进行调整。

### 三、分布式部署设置:

memcached 虽然称为 “ 分布式 ” 缓存服务器, 但服务器端并没有 “ 分布式 ” 功能。每个服务器都是完全独立和隔离的服务。memcached 的分布式, 则是完全由客户端程序库实现的。 这种分布式是 memcached 的最大特点。所以部署时以相同的方式在不同服务器上安装部署即可。

### 四、监控:

一个简单的命令行工具 memcache-top

下载网址: <http://code.google.com/p/memcache-top/>

下载之后放在 /root/soft 目录下

运行命令:

perl memcache-top-v0.6 --instances=ip:端口号,ip:端口号

两台服务器:

perl /root/soft/memcache-top-v0.6 --instances=192.168.175.100:11211,192.168.175.100:11211

运行效果如下:

```
memcache-top v0.6 (default port: 11211, color: on, refresh: 3 seconds)
```

INSTANCE	USAGE	HIT %	CONN	TIME	EVICT/s	READ/s	WRITE/s
192.168.175.100:11211	0.0%	0.0%	10	0.9ms	0.0	2	359
192.168.175.100:11212	0.0%	0.0%	10	0.7ms	0.0	2	359
192.168.175.101:11211	0.0%	0.0%	10	1.4ms	0.0	2	359
192.168.175.101:11212	0.0%	0.0%	10	1.9ms	0.0	2	359
AVERAGE:	0.0%	0.0%	10	1.2ms	0.0	2	359
TOTAL:	0B/	0.8GB	40	4.8ms	0.0	8	1437

(ctrl-c to quit.)

daemontools: 监视 memcached 进程并自动启动.

网址: <http://cr.yp.to/daemontools.html>

通常情况下 memcached 运行得相当稳定, 但也不排除因为一些不可预知的因素导致 memcached 进程死掉, 而又不能及时的发现重启。架构上保证了即使有几台 memcached 故障 也不会影响服务, 不过对于 memcached 进程死掉的服务器, 只要重新启动 memcached, 就可以正常运行, 所以采用了监视 memcached 进程并自动启动的方法。于是使用了 daemontools。

这里不介绍 daemontools 的安装了。我使用了以下的 run 脚本来启动 memcached。

```
#!/bin/sh
```

```
exec 2>&1
```

```
exec /usr/local/bin/memcached -d -u root -m 1024 -p 11211 -c 2048 -P  
/tmp/memcached.pid
```

## 五、测试

服务端:

两台服务器, 各启动两个进程, 分配 200M 内存

启动参数:

```
/usr/local/bin/memcached -d -m 200 -u root -p 11211
```

```
/usr/local/bin/memcached -d -m 200 -u root -p 11212
```

监控启动:

```
perl/root/soft/memcache-top-v0.6--instances=192.168.175.100:11211,192.168.175.  
100:11212,192.168.175.101:11211,192.168.175.101:11212
```

基于 spymemcached 测试

### 1、打开监控

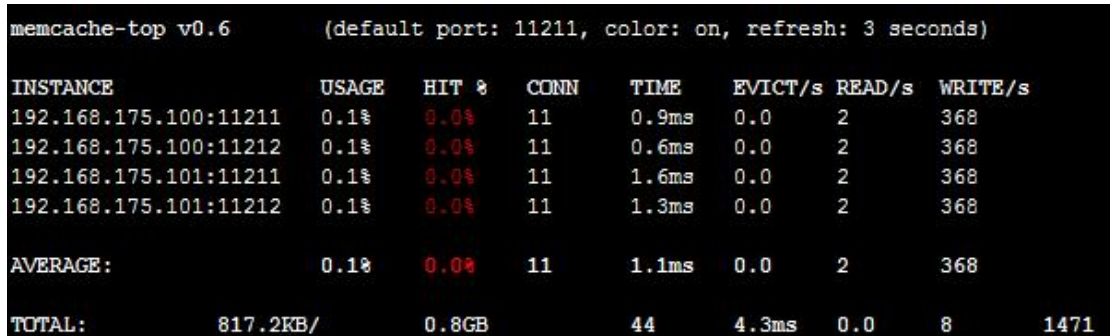
memcache-top v0.6 (default port: 11211, color: on, refresh: 3 seconds)							
INSTANCE	USAGE	HIT %	CONN	TIME	EVICT/s	READ/s	WRITE/s
192.168.175.100:11211	0.0%	0.0%	10	0.9ms	0.0	2	359
192.168.175.100:11212	0.0%	0.0%	10	0.7ms	0.0	2	359
192.168.175.101:11211	0.0%	0.0%	10	1.4ms	0.0	2	359
192.168.175.101:11212	0.0%	0.0%	10	1.9ms	0.0	2	359
AVERAGE:	0.0%	0.0%	10	1.2ms	0.0	2	359
TOTAL:	0B/ 0.8GB		40	4.8ms	0.0	8	1437
(ctrl-c to quit.)							

### 2、往 memcached 插入 10000 条数据

```
MemcachedClient c=new MemcachedClient(  
    AddrUtil.getAddresses(  
"192.168.175.100:11211 192.168.175.100:11212  
192.168.175.101:11211 192.168.175.101:11212"));
```

```
for(int i=0;i<10000;i++){  
c.set("key_"+i, 3600, "value_"+i);  
}
```

插入数据之后监控画面：



```
memcache-top v0.6      (default port: 11211, color: on, refresh: 3 seconds)
```

INSTANCE	USAGE	HIT %	CONN	TIME	EVICT/s	READ/s	WRITE/s
192.168.175.100:11211	0.1%	0.0%	11	0.9ms	0.0	2	368
192.168.175.100:11212	0.1%	0.0%	11	0.6ms	0.0	2	368
192.168.175.101:11211	0.1%	0.0%	11	1.6ms	0.0	2	368
192.168.175.101:11212	0.1%	0.0%	11	1.3ms	0.0	2	368
AVERAGE:	0.1%	0.0%	11	1.1ms	0.0	2	368
TOTAL:	817.2KB/	0.8GB	44	4.3ms	0.0	8	1471

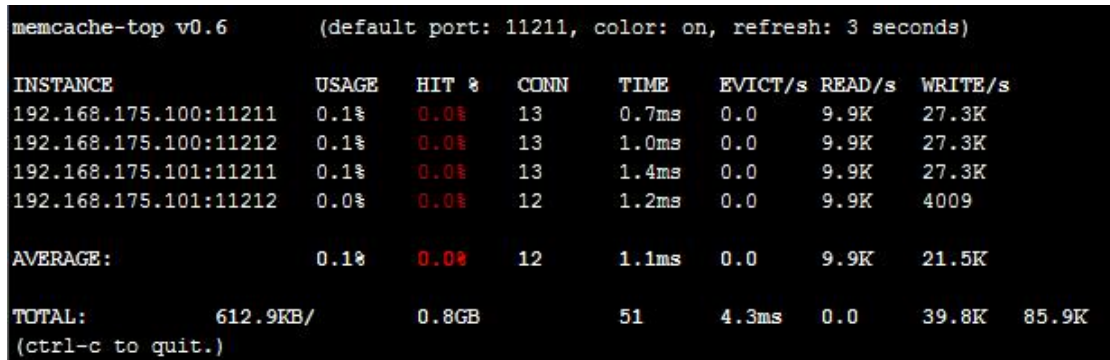
四个 memcached 服务都存在，读取监控也同上图。

取值代码：

```
MemcachedClient c=new MemcachedClient(  
    AddrUtil.getAddresses(  
"192.168.175.100:11211 192.168.175.100:11212  
192.168.175.101:11211 192.168.175.101:11212"));
```

```
for(int i=0;i<10000;i++){  
    String key = "key_"+i;  
    Object obj = c.get(key);  
    System.out.println(key+"="+obj);  
}
```

3、重启服务 101:11212，再读取这 10000 个值，监控结果如下



```
memcache-top v0.6      (default port: 11211, color: on, refresh: 3 seconds)
```

INSTANCE	USAGE	HIT %	CONN	TIME	EVICT/s	READ/s	WRITE/s
192.168.175.100:11211	0.1%	0.0%	13	0.7ms	0.0	9.9K	27.3K
192.168.175.100:11212	0.1%	0.0%	13	1.0ms	0.0	9.9K	27.3K
192.168.175.101:11211	0.1%	0.0%	13	1.4ms	0.0	9.9K	27.3K
192.168.175.101:11212	0.0%	0.0%	12	1.2ms	0.0	9.9K	4009
AVERAGE:	0.1%	0.0%	12	1.1ms	0.0	9.9K	21.5K
TOTAL:	612.9KB/	0.8GB	51	4.3ms	0.0	39.8K	85.9K

(ctrl-c to quit.)

说明某台服务器的数据丢失之后不会影响其它服务器的数据和读取。



4、去掉某台服务，再读取这 10000 个值，监控结果如下

去掉服务 101:11212,获取监控如下:

```
memcache-top v0.6      (default port: 11211, color: on, refresh: 3 seconds)
```

INSTANCE	USAGE	HIT %	CONN	TIME	EVICT/s	READ/s	WRITE/s
192.168.175.100:11211	0.1%	62.5%	11	0.9ms	0.0	9.9K	9972
192.168.175.100:11212	0.1%	76.9%	11	0.5ms	0.0	9.9K	9975
192.168.175.101:11211	0.1%	76.9%	11	1.9ms	0.0	9.9K	9954
192.168.175.101:11212 is DOWN.							
AVERAGE:	0.1%	54.1%	8	0.8ms	0.0	7586	7475
TOTAL:	612.9KB/	0.6GB	33	3.4ms	0.0	29.6K	29.2K
(ctrl-c to quit.)							

增加一个新的服务，101: 11213，获取监控如下:

```
memcache-top v0.6      (default port: 11211, color: on, refresh: 3 seconds)
```

INSTANCE	USAGE	HIT %	CONN	TIME	EVICT/s	READ/s	WRITE/s
192.168.175.100:11211	0.1%	58.4%	13	0.7ms	0.0	4305	4405
192.168.175.100:11212	0.1%	62.8%	13	0.7ms	0.0	4300	4403
192.168.175.101:11211	0.1%	62.8%	13	1.2ms	0.0	4300	4359
192.168.175.101:11212	0.1%	21.1%	12	1.2ms	0.0	4295	4422
192.168.175.101:11213	0.0%	0.0%	11	2.6ms	0.0	4295	1896
AVERAGE:	0.1%	41.0%	12	1.3ms	0.0	4299	3897
TOTAL:	817.2KB/	1.0GB	62	6.3ms	0.0	21.0K	19.0K
(ctrl-c to quit.)							

可见，节点的变化会影响命中率。