

**Memcached**



# GNU/Linux-Memcached

**Memcached** 是一个高性能的分布式内存对象缓存系统，用于动态 **Web** 应用，可以减轻数据库负载。

它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提高动态、数据库驱动网站的速度。

**Memcached** 基于一个存储键 / 值对的 **hashmap**。其守护进程（**daemon**）是用 **C** 写的，但是客户端可以用任何语言来编写，并通过 **memcached** 协议与守护进程通信。

# GNU/Linux-Memcached

**Memcached** 是一个高性能的分布式内存对象缓存系统，用于动态 **Web** 应用，可以减轻数据库负载。

它通过在内存中缓存数据和对象来减少读取数据库的次数，从而提高动态、数据库驱动网站的速度。

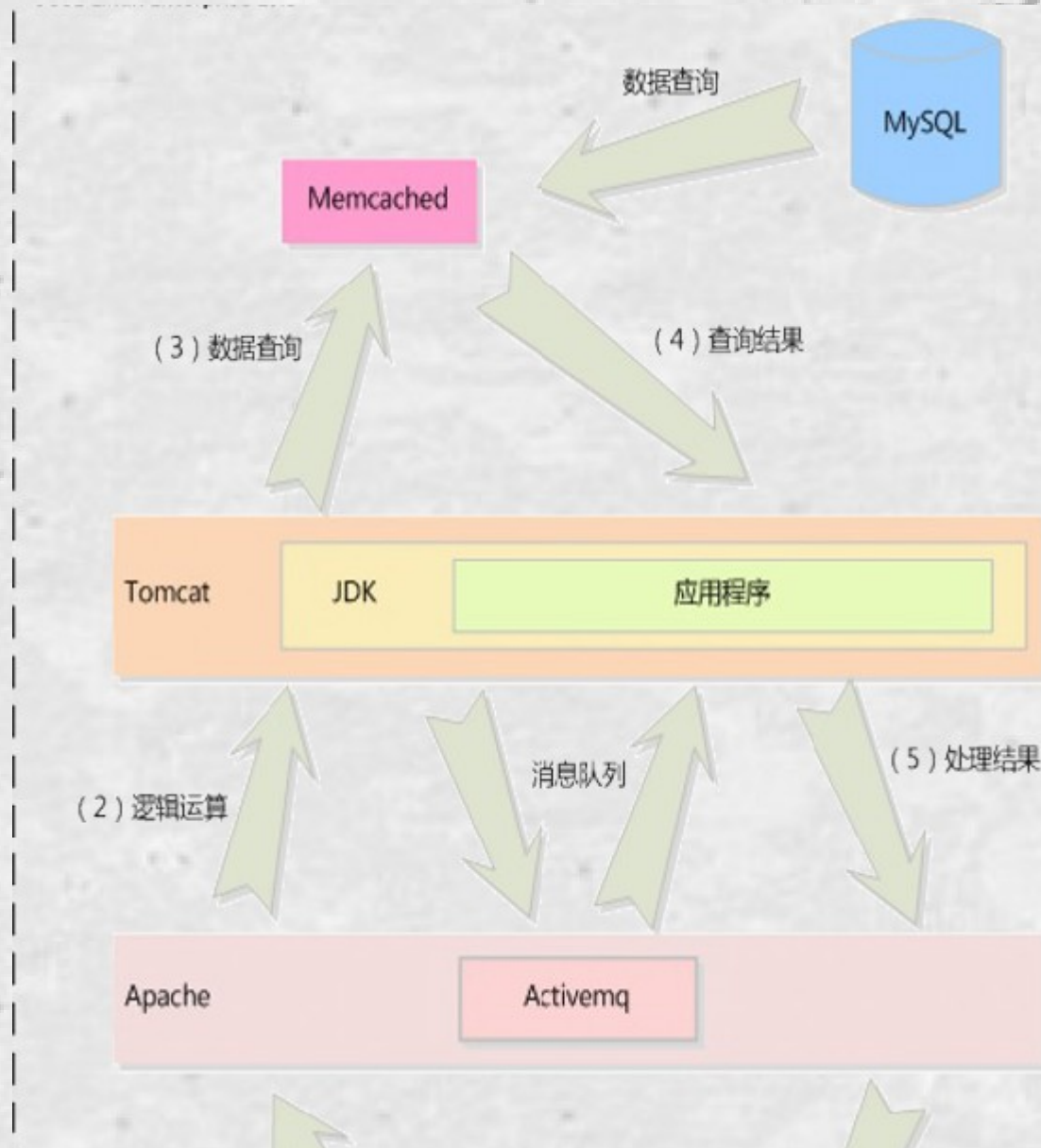
**Memcached** 基于一个存储键 / 值对的 **hashmap**。其守护进程（**daemon**）是用 **C** 写的，但是客户端可以用任何语言来编写，并通过 **memcached** 协议与守护进程通信。

# GNU/Linux-Memcached


缓存一般用来保存一些使用频率高的对象或数据，通过缓存来存取对象或数据要比磁盘存取快。

Memcached是一种内存缓存，把经常需要存取的对象或数据缓存在内存中，内存中缓存的这些数据通过API的方式被存取，数据就像一张大的HASH表一样，以Key-value对的方式存在。

# GNU/Linux-Memcached




# GNU/Linux-Memcached



## Memcache与数据库写作的流程

1.检查客户端请求的数据是否在Memcache中，如果存在，直接将请求的数据返回，不在对数据进行任何操作。



# GNU/Linux-Memcached

## **Memcache**与数据库写作的流程

2.如果请求的数据不在Memcache中，就去数据库查询，把从数据库中获取的数据返回给客户端，同时把数据缓存一份Memcache中。

# GNU/Linux-Memcached

## **Memcache与数据库写作的流程**

3. 每次更新数据库的同时更新Memcache中的数据库。确保数据信息一致性。



# GNU/Linux-Memcached

## Memcache与数据库写作的流程

4. 当分配给Memcache内存空间用完后，会使用LRU(least Recently Used，最近最少使用)策略加到其失效策略，失效的数据首先被替换掉，然后在替换掉最近未使用的数据。

# GNU/Linux-Memcached

## Memcached 特征

1. 协议简单
2. 基于libevent的事件处理
3. 内置的内存管理方式
4. 互不通信的分布式



# GNU/Linux-Memcached

## 1. 协议简单

其使用基于文本行的协议，能直接通过telnet在Memcached服务器上存取数据

# GNU/Linux-Memcached

## 2. 基于libevent的事件处理

libevent是个程序库，memcached使用这个libevent库，因此能在Linux、BSD等操作系统上发挥其高性能。Memcached利用这个库进行异步事件处理。



# GNU/Linux-Memcached

## 3. 内置的内存管理方式

Memcached有一套自己管理内存的方式，这套方式非常高效，所有的数据都保存在Memcached内置的内存中，当存入的数据占满空间时，使用LRU算法自动删除不使用的缓存，即重用过期的内存空间。

Memcached不考虑数据的容灾问题，一旦重启所有数据全部丢失。

# GNU/Linux-Memcached

## 4. 互不通信的分布式

各个Memcached服务器之间互不通信，都是独立的存取数据，不共享任何信息。通过对客户端的设计，让Memcached具有分布式，能支持海量缓存和大规模应用。

# GNU/Linux-Memcached

## Memcached 环境建立



# GNU/Linux-Memcached

## 1. 安装

```
#yum install memcached -y
```

## 2. 启动服务

```
#systemctl enable memcached
```

```
#systemctl start memcached
```





# GNU/Linux-Memcached

## 3. 启动memcached

```
#memcached -d -m 256 -u root -p 11211 -c  
1024 -P /tmp/memcached.pid
```

参数解释：

启动memcached守护进程(-d),分配Memcached内存使用量为256M,以root身份运行(-u),监听端口为11211,接收最大并发连接数为1024个(-c)。pid文件位置为/tmp目录下(-P)

# GNU/Linux-Memcached

## 3. 启动memcached

```
#memcached -d -m 256 -u root -p 11211 -c  
1024 -P /tmp/memcached.pid
```

参数解释：

启动memcached守护进程(-d),分配Memcached内存使用量为256M,以root身份运行(-u),监听端口为11211,接收最大并发连接数为1024个(-c)。pid文件位置为/tmp目录下(-P)

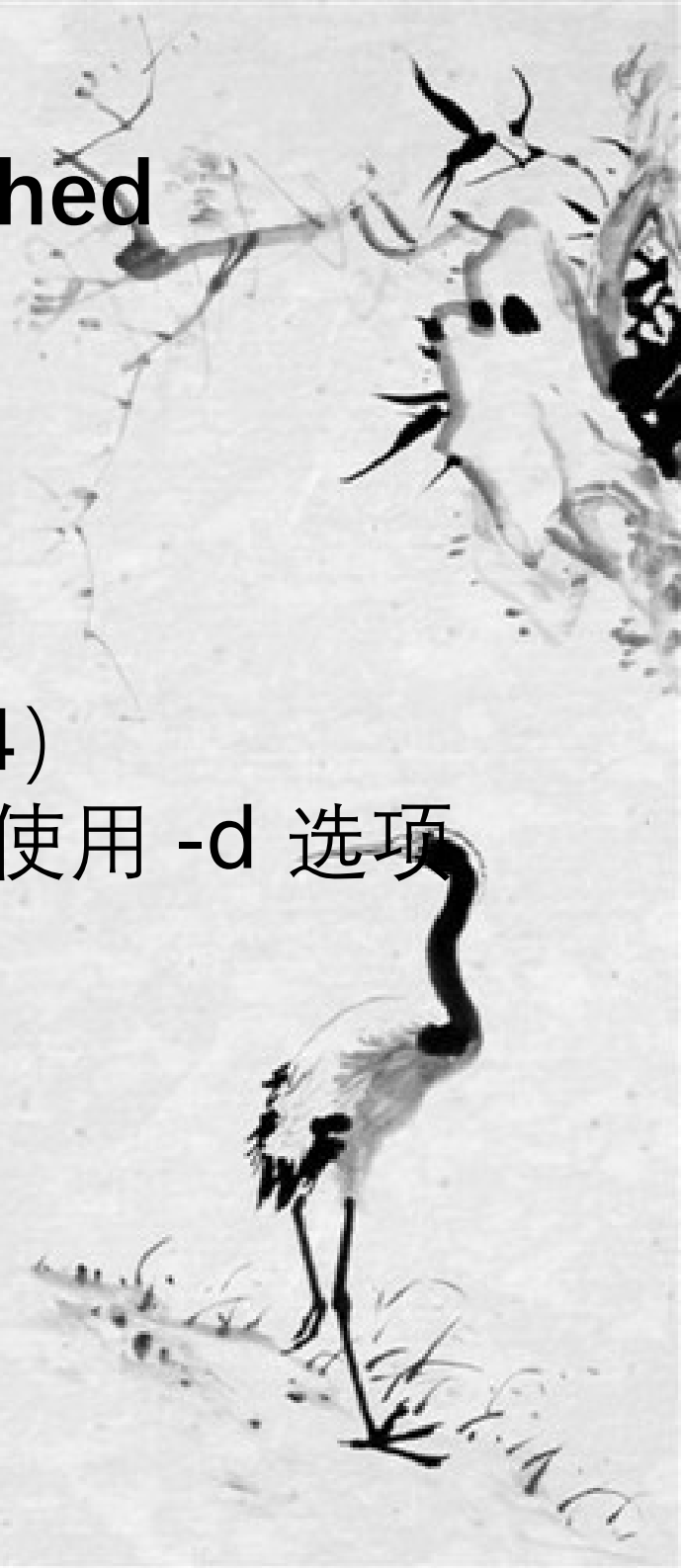
# GNU/Linux-Memcached

- d:作为守护进程来运行。
- m:单个数据项的最大可用内存，以MB为单位。  
(默认：64MB )
- u:设定进程所属用户(只有root用户可以使用这个参数)



# GNU/Linux-Memcached

- p:监听的TCP端口(默认: 11211)
- c:最大并发连接数。(默认: 1024)
- P:保存进程ID到指定文件, 只有在使用 -d 选项的时候才有意义。



# GNU/Linux-Memcached

- p:监听的TCP端口(默认: 11211)
- c:最大并发连接数。(默认: 1024)
- P:保存进程ID到指定文件, 只有在使用 -d 选项的时候才有意义。



# GNU/Linux-Memcached

## 4. 查看Memcached状态（需要安装telnet）

# telnet localhost 11211

```
[root@localhost ~]# telnet localhost 11211
Trying ::1...
Connected to localhost.
Escape character is '^]'.
stats
STAT pid 36148
STAT uptime 14795
STAT time 1492784955
STAT version 1.4.15
STAT libevent 2.0.21-stable
STAT pointer_size 64
STAT rusage_user 0.732471
STAT rusage_system 0.575793
STAT curr_connections 10
STAT total_connections 13
STAT connection_structures 11
STAT reserved_fds 20
STAT cmd_get 2
STAT cmd_set 2
STAT cmd_flush 0
STAT cmd_touch 0
STAT get_hits 2
```



# GNU/Linux-Memcached

STAT pid 3401←memcache服务器的进程ID

STAT uptime 1481←服务器已经运行的秒数

STAT time 1\*\*\*5←服务器当前的unix时间戳

STAT version 1.4.15←memcache版本

STAT libevent 2.0.21-stable←libevent版本

# GNU/Linux-Memcached

STAT pointer\_size 64 ← 当前操作系统的指针大小  
(32位系统一般是32bit, 64就是64位操作系统)

STAT rusage\_user 0.014997 ← 进程的累计用户时间

STAT rusage\_system 0.022996 ← 进程的累计系统时间



# GNU/Linux-Memcached

STAT curr\_connections 10 ← 服务器当前存储的  
items数量

STAT total\_connections 12 ← 从服务器启动以后  
存储的items总数量

STAT connection\_structures 11 ← 服务器分配的  
连接构造数



# GNU/Linux-Memcached

STAT reserved\_fds 20

STAT cmd\_get 0 ← get命令 ( 获取 ) 总请求次数

STAT cmd\_set 0 ← set命令 ( 保存 ) 总请求次数

# GNU/Linux-Memcached

STAT cmd\_flush 0←flush命令请求次数

STAT cmd\_touch 0←touch命令请求次数

STAT get\_hits 0←总命中次数

STAT get\_misses 0←总未命中次数

STAT delete\_misses 0←delete命令未命中次数

STAT delete\_hits 0←delete命令命中次数



# GNU/Linux-Memcached

STAT incr\_misses 0←incr命令未命中次数  
STAT incr\_hits 0←incr命令命中次数  
STAT decr\_misses 0←decr命令未命中次数  
STAT decr\_hits 0←decr命令命中次数  
STAT cas\_misses 0←cas命令未命中次数  
STAT cas\_hits 0←cas命令命中次数



# GNU/Linux-Memcached

STAT cas\_badval 0 ← 使用擦拭次数

STAT touch\_hits 0 ← touch命令未命中次数

STAT touch\_misses 0 ← touch命令命中次数

STAT auth\_cmds 0 ← 认证命令处理的次数

STAT auth\_errors 0 ← 认证失败数目

STAT bytes\_read 13 ← 总读取字节数(请求字节数)

# GNU/Linux-Memcached

STAT limit\_maxbytes 10485760 ←分配给  
memcache的内存大小（字节）

STAT accepting\_conns 1 ←服务器是否达到过最大  
连接（0/1）

STAT listen\_disabled\_num 0 ←失效的监听数

STAT threads 4 ←当前线程数

STAT conn\_yields 0 ←连接操作主动放弃数



# GNU/Linux-Memcached

STAT hash\_power\_level 16  
STAT hash\_bytes 524288  
STAT hash\_is\_expanding 0  
STAT malloc\_fails 0



# GNU/Linux-Memcached

STAT bytes 0 ← 当前存储占用的字节数

STAT curr\_items 0 ← 当前存储的数据总数

STAT total\_items 0 ← 启动以来存储的数据总数

STAT expired\_unfetched 0

STAT evicted\_unfetched 0





# GNU/Linux-Memcached

STAT 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）

STAT reclaimed 0 ← 已过期的数据条目来存储新数据的数目

# GNU/Linux-Memcached

```
STAT crawler_reclaimed 0  
STAT lrutail_reflocked 0  
END
```



# GNU/Linux-Memcached

## Memcached 的PHP 扩展



# GNU/Linux-Memcached

## 1. 安装php扩展插件

```
#Yum install libmemcached php-pecl-  
memcache -y
```

```
#Yum httpd php php-mbstring php-pear
```

## 2. 确认memcached模块已被扩展

```
#vim /etc/php.d/memcache.ini
```

```
1 ; ----- Enable memcache extension module  
2 extension=memcache.so  
3
```



# GNU/Linux-Memcached

3.编辑index.php

```
#cd /var/www/html
```

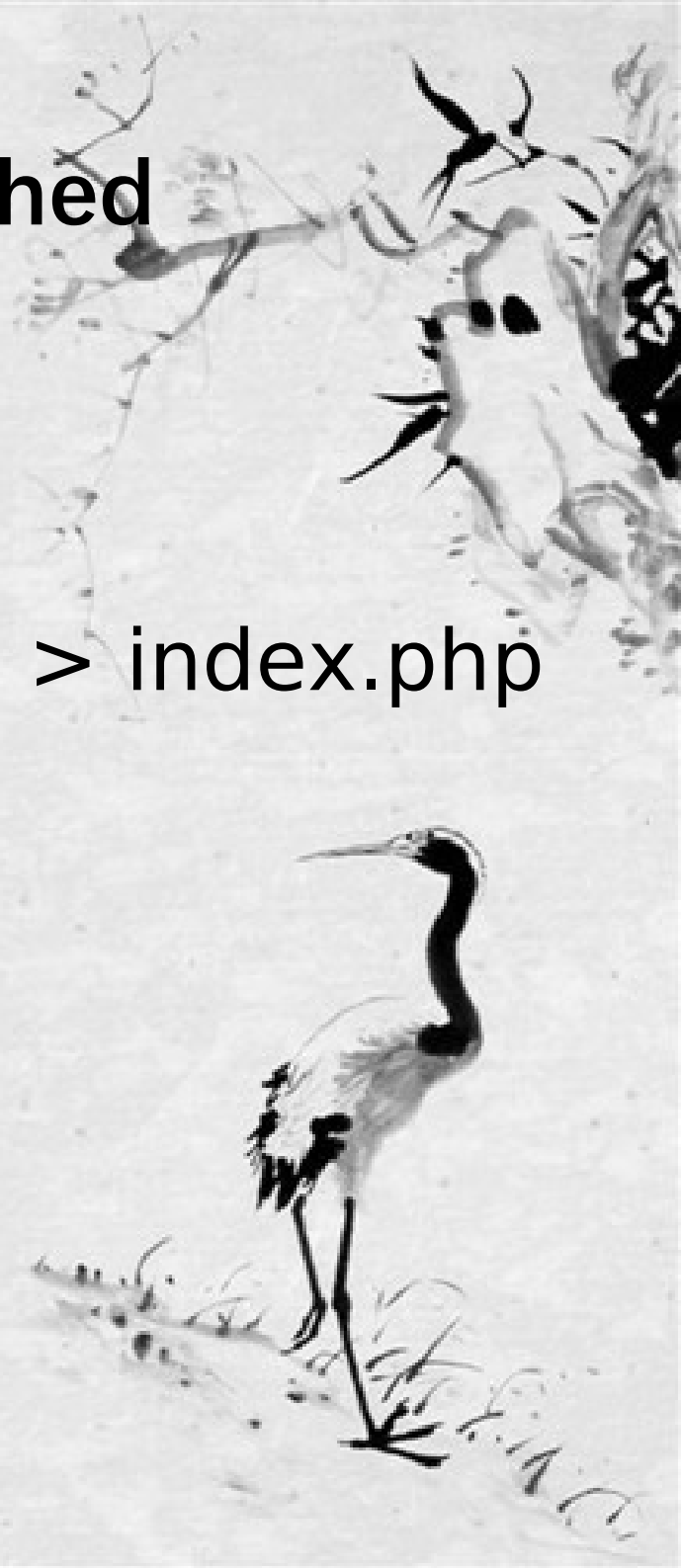
```
#echo "<?php phpinfo() ?>" > index.php
```

```
#cd /etc/httpd/conf/
```

```
#vim httpd.conf
```

```
162 #  
163 <IfModule dir_module>  
164     DirectoryIndex index.html index.php  
165 </IfModule>
```

```
#systemctl restart httpd
```



# GNU/Linux-Memcached

## 4 浏览器查看

### memcache

memcache support	enabled
Version	3.0.8
Revision	\$Revision: 329835 \$

Directive	Local Value	Master Value
memcache.allow_failover	1	1
memcache.chunk_size	32768	32768
memcache.compress_threshold	20000	20000
memcache.default_port	11211	11211
memcache.hash_function	crc32	crc32
memcache.hash_strategy	consistent	consistent
memcache.lock_timeout	15	15
memcache.max_failover_attempts	20	20
memcache.protocol	ascii	ascii
memcache.redundancy	1	1
memcache.session_redundancy	2	2

# GNU/Linux-Memcached

## memcached集群

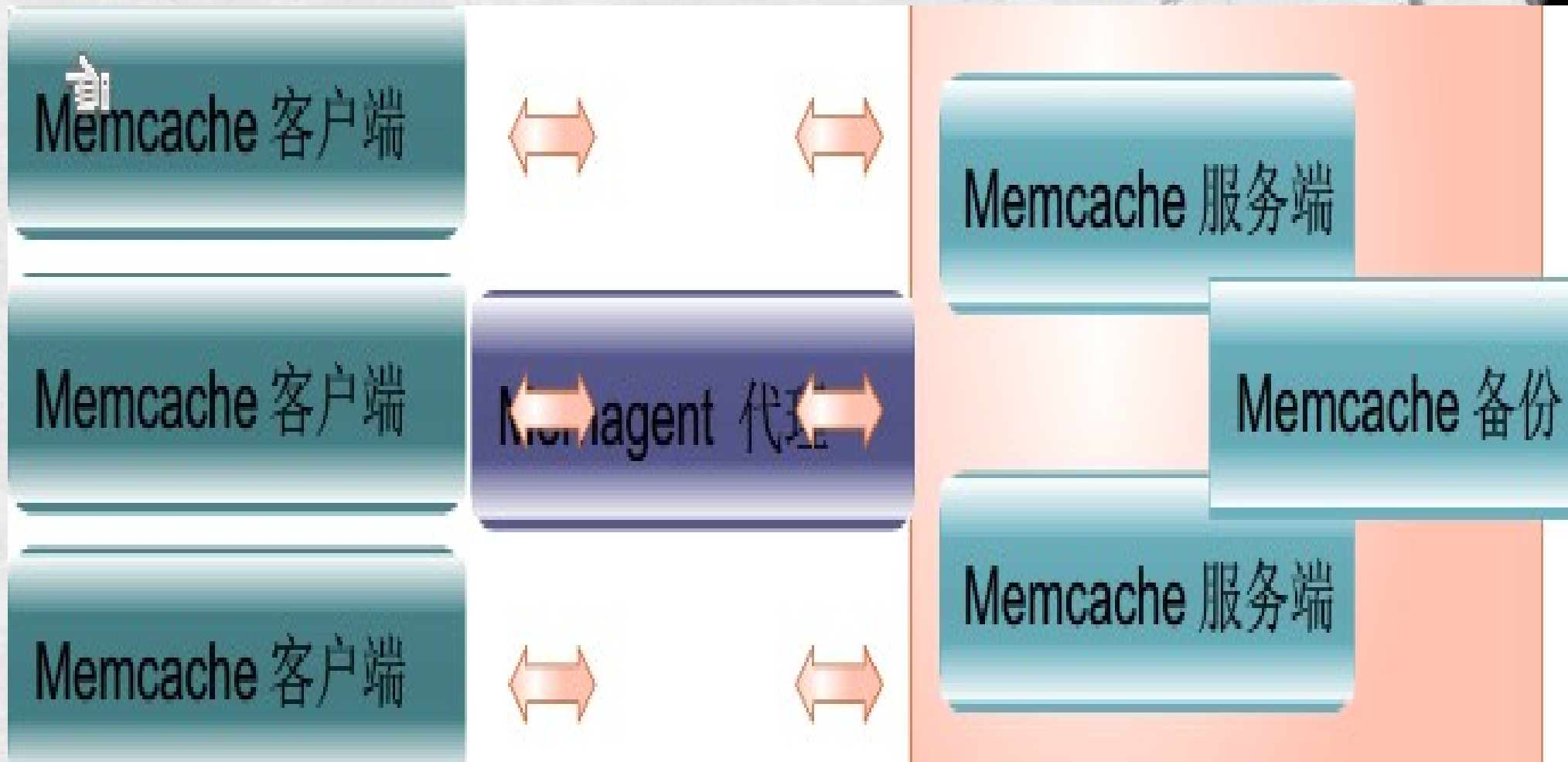


# GNU/Linux-Memcached

集群可以定义为两台或两台以上计算机组合工作，对外表现为一个整体系统。memcached的集群实现，采用一个代理程序memagent，memagent工作在memcached服务器和memcached客户端之间，当客户端请求数据时先通过memagent，memagent通过算法从memcached中读取，实现了数据请求的负载均衡。

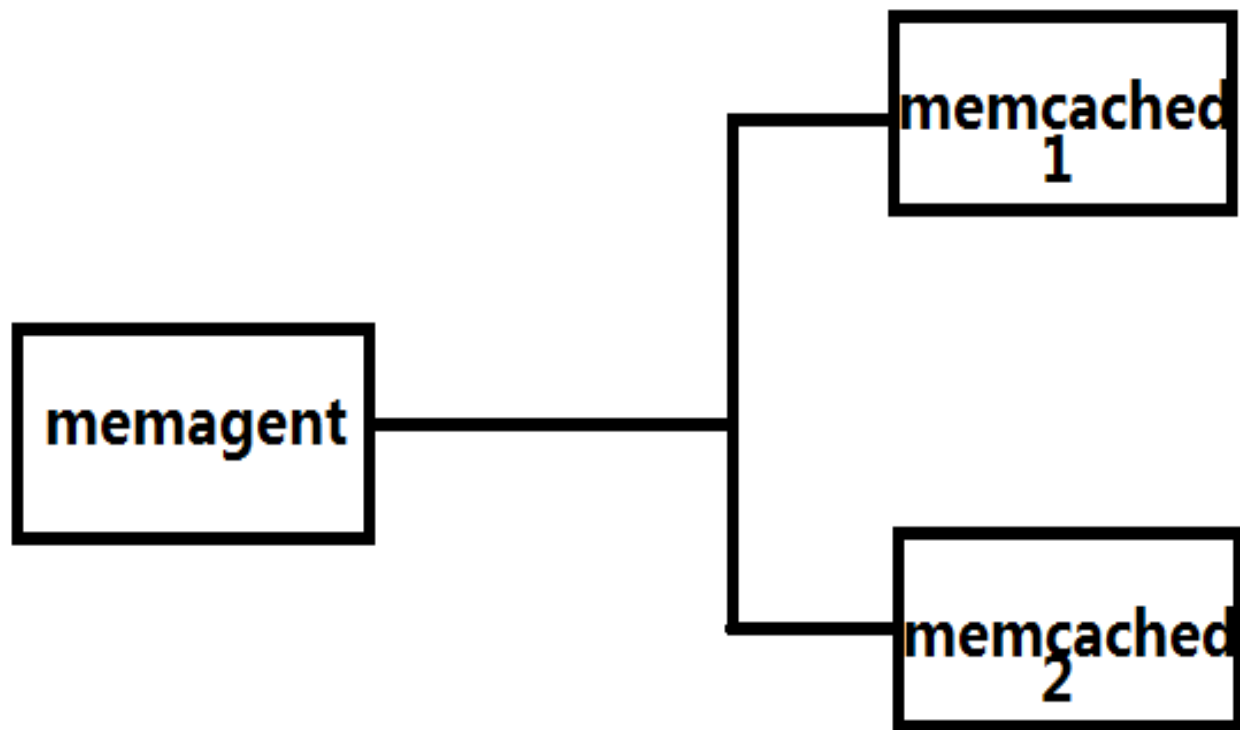


# GNU/Linux-Memcached



# GNU/Linux-Memcached

## Memcached 集群实验



# GNU/Linux-Memcached

## Memcached 集群实验 代理节点操作

1. 安装 libevent, Memcached 基于 libevent 驱动来处理 IO

```
#yum install libevent libevent-devel -y
```

2. 下载 memagent 源码包

```
#wget https://storage.googleapis.com/google-code-archive-downloads/v2/code.google.com/memagent/magent-0.5.tar.gz
```

# GNU/Linux-Memcached

memcached 集群实验

3. 安装编译环境

```
#yum install gcc make glibc glib2 glib2-* -y
```

4. 安装 memagent 源码包

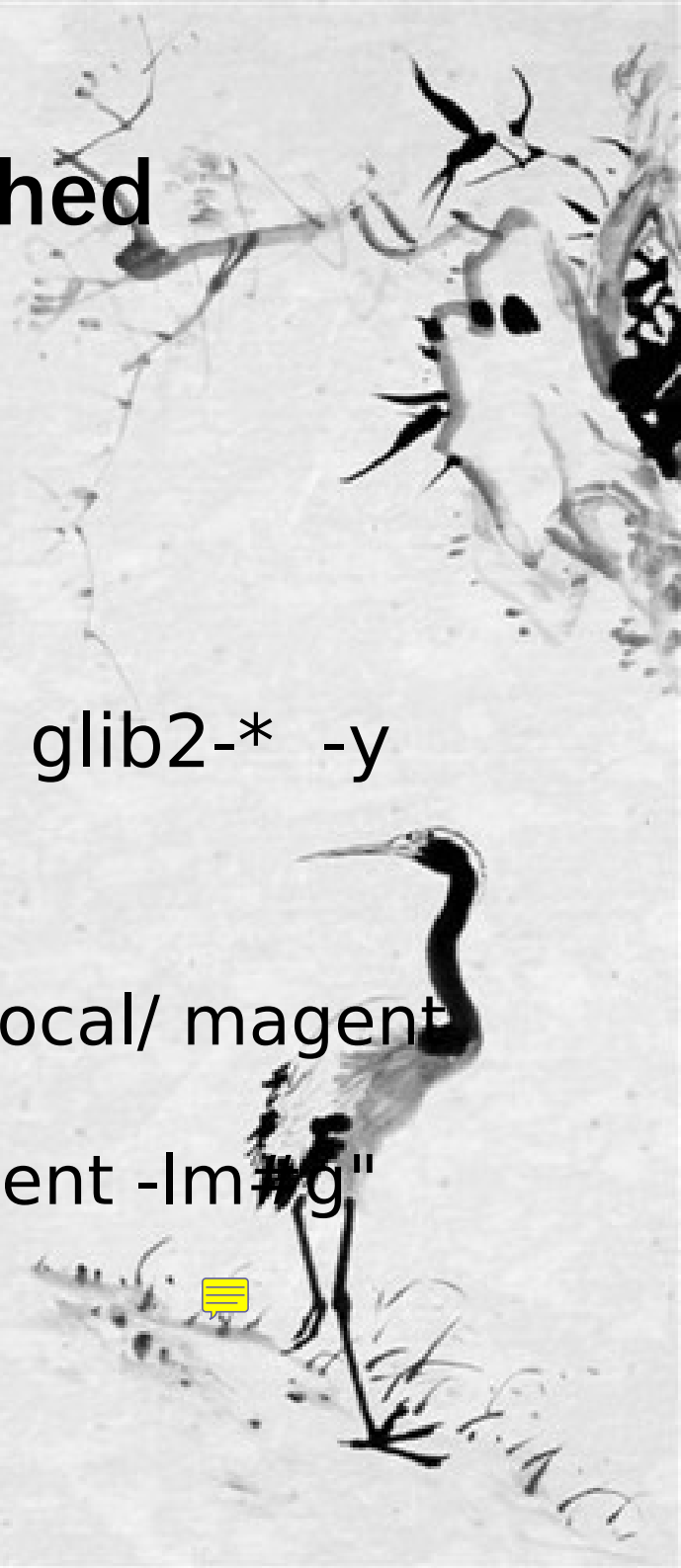
```
#mkdir -p /usr/local/magent/
```

```
#tar -zxf magent-0.5.tar.gz -C /usr/local/ magent
```

```
#cd /usr/local/ magent
```

```
#sed -i "s/LIBS = -levent/LIBS = -levent -lm#g"
```

```
Makefile
```



# GNU/Linux-Memcached

## memcached 集群实验

```
[root@localhost memagent]# vim ketama.h
```

```
#ifndef SSIZE_MAX  
# define SSIZE_MAX      32767  
#endif  
#ifndef _KETAMA_H  
#define _KETAMA_H
```

```
#make
```

```
#cp magent /usr/bin/magent
```



# GNU/Linux-Memcached

memcached 集群实验

5.memcached 节点配置同上

6. 测试

```
#magent -u root -n 51200
```

```
-l 192.168.100.130  注:magent IP
```

```
-p 11219
```

```
-s 192.168.100.128:11211  注:memcached IP
```

```
-s 192.168.100.133:11212  注:memcached IP
```



# GNU/Linux-Memcached

## memcached 集群实验

代理节点查看连接的 memcached 节点

#telnet 192.168.100.130 11 11219

```
[root@localhost memagent]# telnet 192.168.100.130 11219
Trying 192.168.100.130...
Connected to 192.168.100.130.
Escape character is '^]'.
stats
memcached agent v0.4
matrix 1 -> 192.168.100.128:11211, pool size 1
matrix 2 -> 192.168.100.133:11211, pool size 1
END
```



# GNU/Linux-Memcached

## memcached 集群实验

在代理节点测试（如图）

```
set key1 0 0 8
```

```
tiantian
```

```
Set ley2 0 0 8
```

```
wangwang
```

```
set key1 0 0 8
```

```
tiantian
```

```
STORED
```

```
set key2 0 0 8
```

```
wangwang
```

```
STORED
```





# GNU/Linux-Memcached

## memcached 集群实验

memcached1 节点取值 (如图)

```
#telnet 192.168.100.128 11211
```

```
get key1
```

```
get key2
```

```
get key1  
END  
get key2  
VALUE key2 0 8  
wangwang  
END
```

# GNU/Linux-Memcached

## memcached 集群实验

memcached2 节点取值 (如图)

```
#telnet 192.168.100.133 11211
```

```
get key1
```

```
get key2
```

```
VALUE key1 0 8
tiantian
END
get key2
END
```

