GNU/Linux-Memcached

Memcached 配置与应用

GNU/Linux-Memcached

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

Memcached 同时是一个开源的、高性能、具有分布式内存对象的缓存系统。它通过减轻数据库负载加速动态 WEB 应用。

GNU/Linux-Memcached

Memcached

缓存一般用来保存一些常用存取的对象或数据，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。

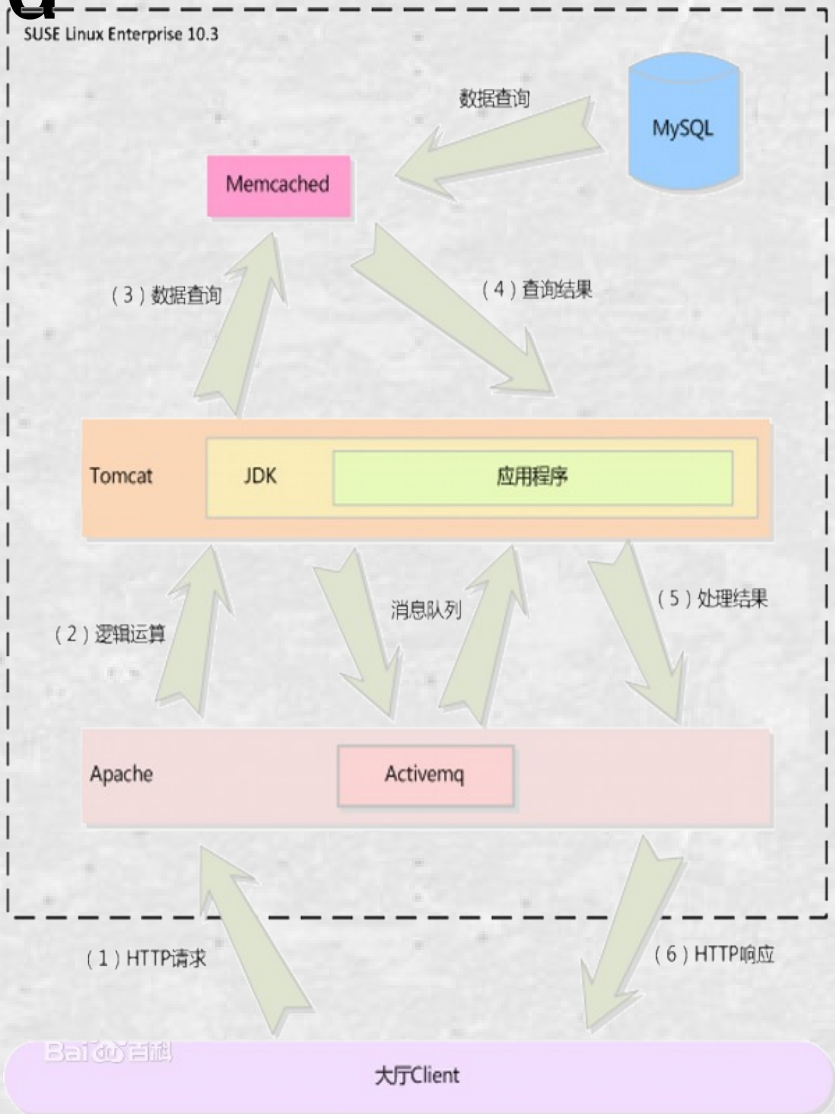
GNU/Linux-Memcached

Memcached

Memcached 通过缓存经常被存取的对象或数据，来减轻数据库的压力，提高网站的响应速度，构建速度更快的可扩展的 WEB 应用。

GNU/Linux-Memcached

Memcached



GNU/Linux-Memcached

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

Memcached

Memcached 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

Memcached

Memcached 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

Memcached

Memcached 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

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

Memcached 特征

2. 基于 libevent 的事件处理

libevent 利用 C 开发的程序库，它将 BSD 系统的 kqueue,Linux 系统的 epoll 等事件处理功能封装成为一个接口，确保即使服务器端的链接数增加也能发挥很好的性能。

Memcached 利用这个库进行异步事件处理。

GNU/Linux-Memcached

Memcached 特征

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。

Memecached 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

Memcached 特征

4. 互不通信的分布式

各个 Memecached 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

1. 准备 EPEL 源，并测试通过

2. 安装 Memcached

```
#yum --enablerepo=epel  install memcached -y
```

GNU/Linux-Memcached

Memcached 环境建立

3. 启动 Memcached
#systemctl start memcached

#systemctl enable memcached

GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

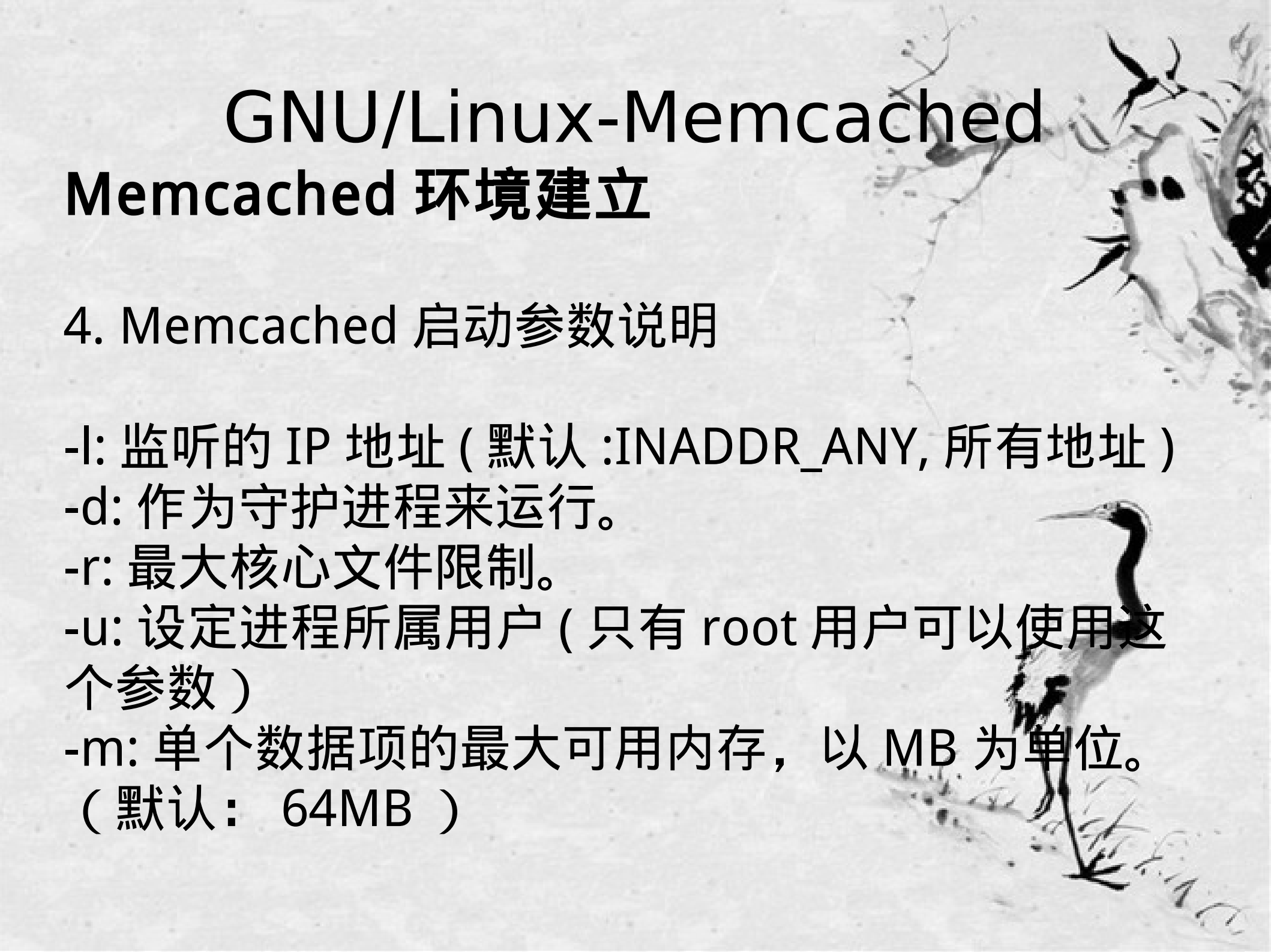
-p: 监听的 TCP 端口 (默认 : 11211)
-U: 监听的 UDP 端口 (默认 : 11211, 0 表示不监听)
-s: 用于监听的 UNIX 套接字路径 (禁用网络支持)
-a:UNIX 套接字访问掩码，八进制数字 (默认 : 0700)

GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

-l: 监听的 IP 地址 (默认 :INADDR_ANY, 所有地址)
-d: 作为守护进程来运行。
-r: 最大核心文件限制。
-u: 设定进程所属用户 (只有 root 用户可以使用这个参数)
-m: 单个数据项的最大可用内存，以 MB 为单位。（ 默认： 64MB ）

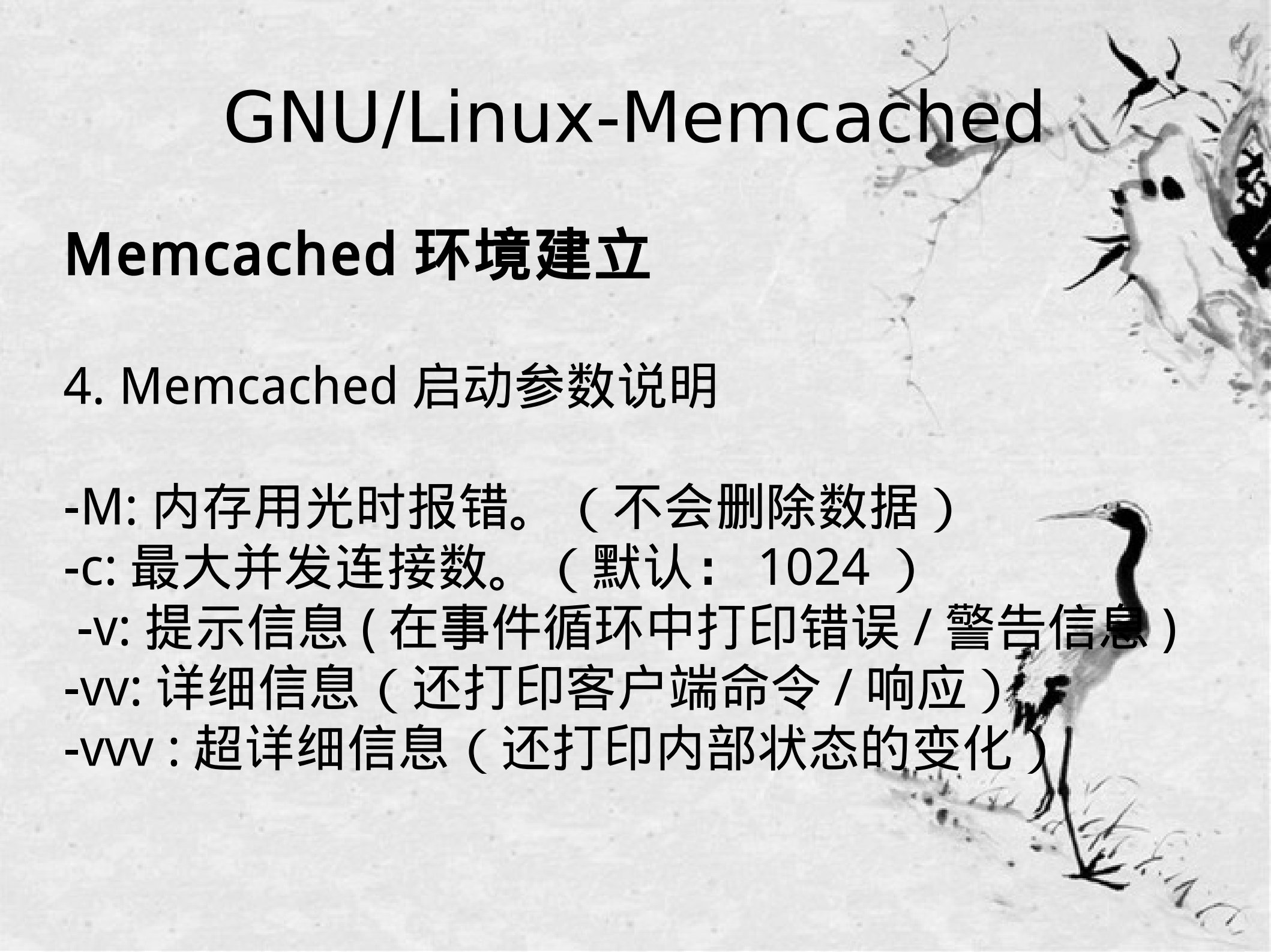


GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

-M: 内存用光时报错。（不会删除数据）
-c: 最大并发连接数。（默认： 1024 ）
-v: 提示信息 (在事件循环中打印错误 / 警告信息)
-vv: 详细信息（还打印客户端命令 / 响应）
-vvv : 超详细信息（还打印内部状态的变化）



GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

-k: 锁定所有内存页。注意你可以锁定的内存上限 . 试图分配更多内存会失败的，所以留意启动守护进程时所用的用户可分配的内存上限 . 不是前面的 -u 参数；在 sh 下，使用命令 "ulimit -S -l NUM_KB" 来设置)

GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

-h: 打印这个帮助信息并退出。
-i: 打印 memcached 和 libevent 的许可。
-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。
-f: 块大小增长因子。（默认： 1.25 ）
-n : 分配给 key+value+flags 的最小空间 (默认 :48)

GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

-L: 尝试使用大内存页（如果可用的话）。 提高内存页尺寸可以减少 " 页表缓冲（ TLB ）" 丢失次数，提高运行效率。 为了从操作系统获得大内存页， memcached 会把全部数据项分配到一个大区块

GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

-D: 使用 x 作为前缀和 ID 的分隔符。这个用于按前缀获得状态报告。默认是 ":"（冒号）。如果指定了这个参数，则状态收集会自动开启；如果没指定，则需要用命令 "stats detail on" 来开启。

GNU/Linux-Memcached

Memcached 环境建立

4. Memcached 启动参数说明

-t: 使用的线程数（默认： 4 ）
-R: 每个连接可处理的最大请求数。
-C : 禁用 CAS 。
-b: 设置后台日志队列的长度（默认： 1024 ）
-B: 绑定协议 - 可能值： ascii,binary,auto(默认)
-I: 重写每个数据页尺寸。调整数据项最大尺寸。

GNU/Linux-Memcached

Memcached 环境建立

5. 示例
//* 启动 memecached 守护进程 (-d), 分配 Memecached 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)

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

GNU/Linux-Memcached

Memcached 环境建立

6. 查看 Memcached 状态

```
# telnet localhost 11211
Trying ::1...
Connected to localhost.
Escape character is '^]'.
stats ← 输入命令
```

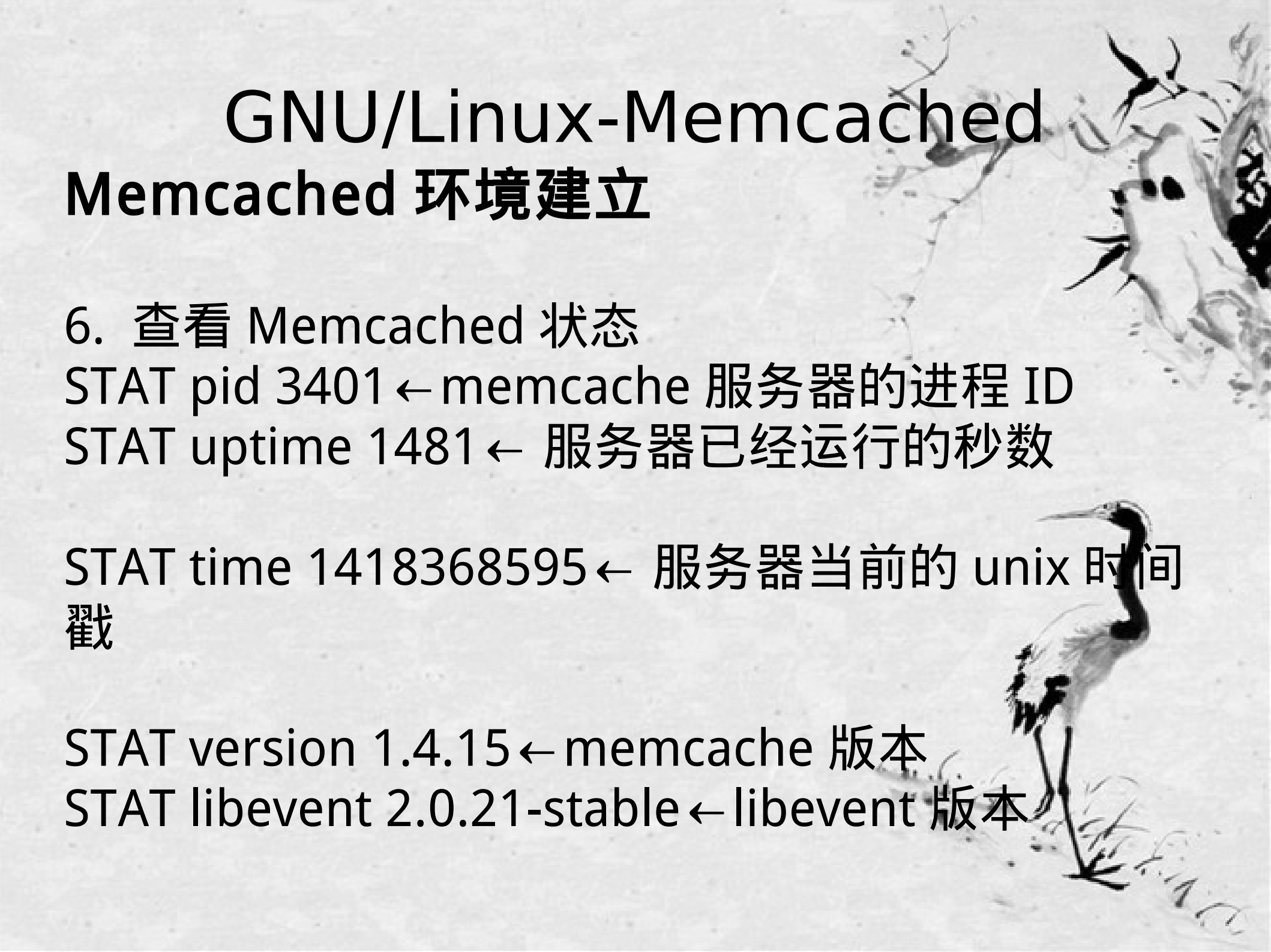
GNU/Linux-Memcached

Memcached 环境建立

6. 查看 Memcached 状态
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 1418368595← 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

Memcached 环境建立

6. 查看 Memcached 状态

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

Memcached 环境建立

6. 查看 Memcached 状态

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

Memcached 环境建立

6. 查看 Memcached 状态
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

Memcached 环境建立

6. 查看 Memcached 状态

STAT cmd_flush 0←flush 命令请求次数

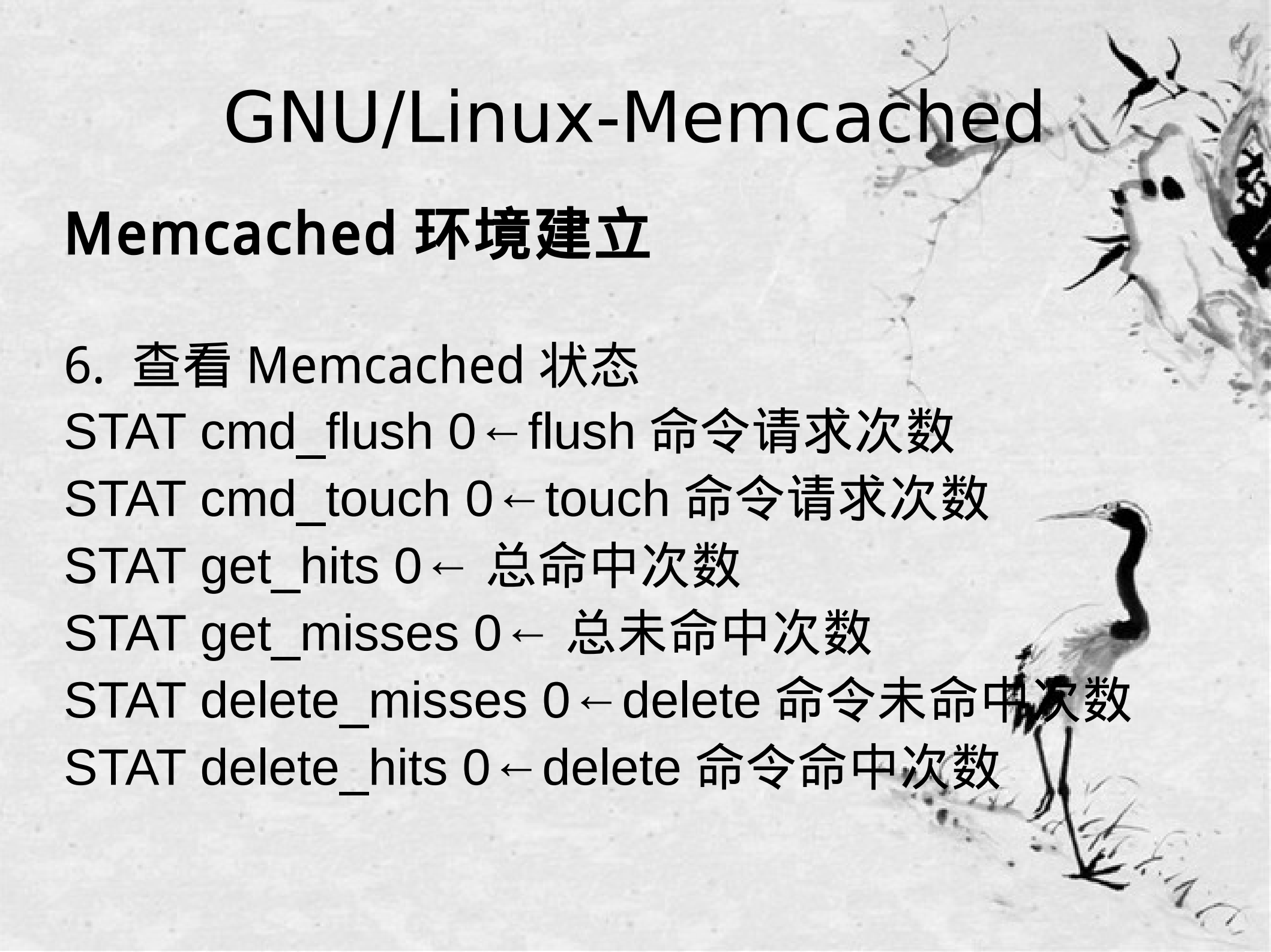
STAT cmd_touch 0←touch 命令请求次数

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

Memcached 环境建立

6. 查看 Memcached 状态

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

Memcached 环境建立

6. 查看 Memcached 状态

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

Memcached 环境建立

6. 查看 Memcached 状态

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目

GNU/Linux-Memcached

Memcached 环境建立

6. 查看 Memcached 状态

STAT hash_power_level 16
STAT hash_bytes 524288
STAT hash_is_expanding 0
STAT malloc_fails 0

GNU/Linux-Memcached

Memcached 环境建立

6. 查看 Memcached 状态

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

Memcached 环境建立

6. 查看 Memcached 状态

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

Memcached 环境建立

6. 查看 Memcached 状态

STAT crawler_reclaimed 0

STAT lrutail_reflocked 0

END

GNU/Linux-Memcached

Memcached 的 PHP 扩展

1）安装 Memcached 的 PHP 扩展插件

```
# yum --enablerepo=epel install libmemcached -y

#yum install httpd php php-mbstring php-pear -y
```

GNU/Linux-Memcached

Memcached 的 PHP 扩展

2. PHP 扩展配置

2) 确认 memcached 模块已被扩展
#vi /etc/php.d/z_memcached.ini
//* 确认第 2 行
extension = “memcached.so”

GNU/Linux-Memcached

Memcached 的 PHP 扩展

2. PHP 扩展配置

3) 确认 memcached 模块已被扩展
#vi /etc/php.d/z-memcached.ini
//* 确认第 2 行
extension = “memcached.so”

GNU/Linux-Memcached

Memcached 的 PHP 扩展

2. PHP 扩展配置

4) 编辑 index.php
#cd /var/www/html
#vi index.php
<?php phpinfo() ?>

GNU/Linux-Memcached

Memcached 的 PHP 扩展

2. PHP 扩展配置

5) 启动 apache
#systemctl restart httpd

6) 打开浏览器查看 php 信息，在页面中找” memcached”, 如果存在则证明扩展成功。

GNU/Linux-Memcached

监控 Memcached

1. Nagios[使用 check_tcp(mixi) 方法]

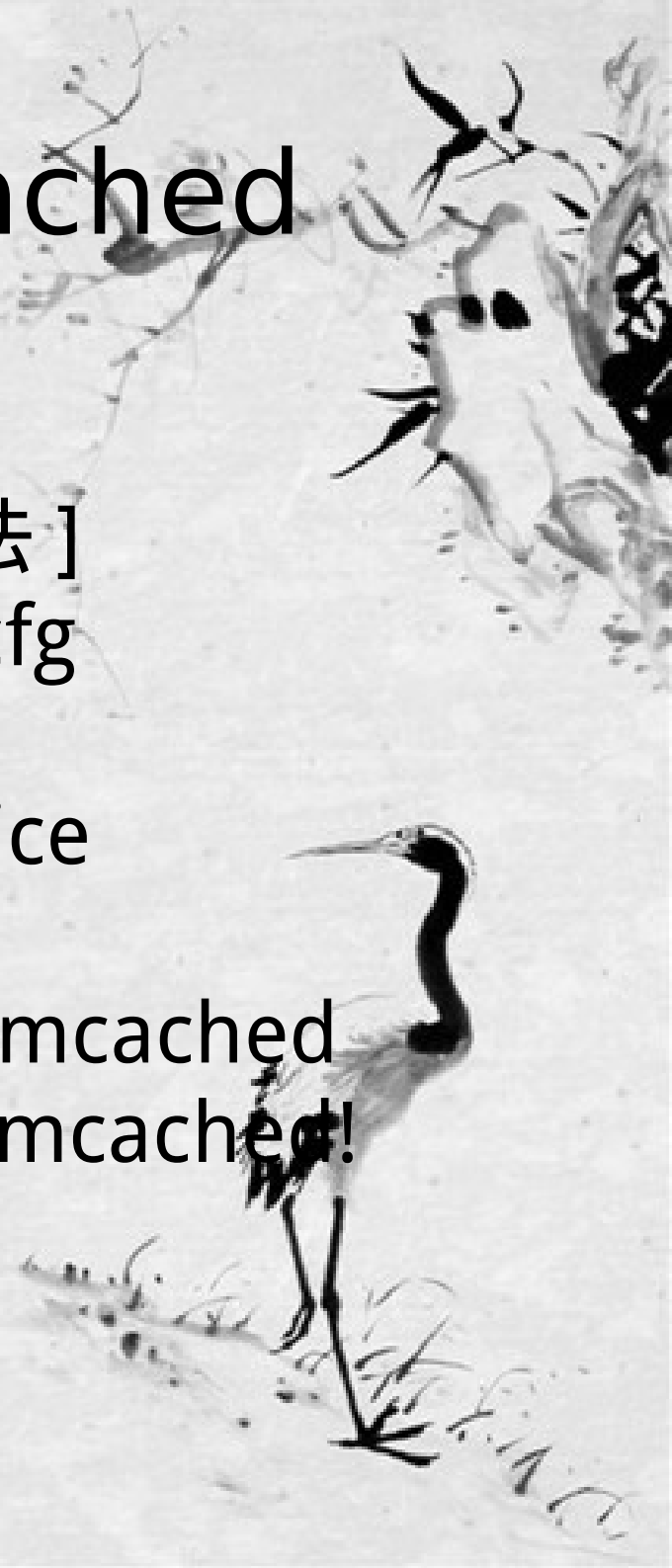
```
#  vi /etc/nagios/objects/commands.cfg
define command{
        command_name   check_memcached
        command_line    $USER1$/check_tcp -H
$HOSTADDRESS$ -p 11211 -t 5 -E -s
'stats\r\nquit\r\n' -e 'uptime' -M crit
        }
```

GNU/Linux-Memcached

监控 Memcached

1. Nagios[使用 check_tcp(mixi) 方法]

```
#vi  /etc/nagios/objects/localhost.cfg
define service{
        use                             local-service
        host_name                       localhost
        service_description             check_memcached
        check_command                   check_memcached!
192.168.188.111
        }
```



GNU/Linux-Memcached

监控 Memcached

1. Nagios[使用 check_tcp(mixi) 方法]

#systemctl restart nagios

浏览器查看

GNU/Linux-Memcached

监控 Memcached

2. Nagios[使用 check_memcached 监控插件方法]

//* 下载 Nagios-Plugins-Memcached-0.02.tar.gz 并安装

GNU/Linux-Memcached

监控 Memcached

2. Nagios[使用 check_memcached 监控插件方法]

```
#  vi /etc/nagios/objects/commands.cfg
define command{
        command_name   check_memcached
        command_line
$USER1$/check_memcached -H $HOSTADDRESS$
-w 80 -c 90
        }
```

GNU/Linux-Memcached

监控 Memcached

2. Nagios[使用 check_memcached 监控插件方法]

```
#vi  /etc/nagios/objects/localhost.cfg
define service{
        use                             local-service
        host_name                       localhost
        service_description  check_memcached
        check_command      check_memcached!
192.168.188.111
        }
```

GNU/Linux-Memcached

监控 Memcached

3. Cacti
//*cacti 即具备一整套 Memcached 当前状态信息的模板 . 模板可从

http://dealnews.com/developers/cacti/memcached.html

下载

GNU/Linux-Memcached

监控 Memcached

3. Cacti
//* 下载后解压数据包

// 安装
#python setup.py install

//*cacti 页面查看