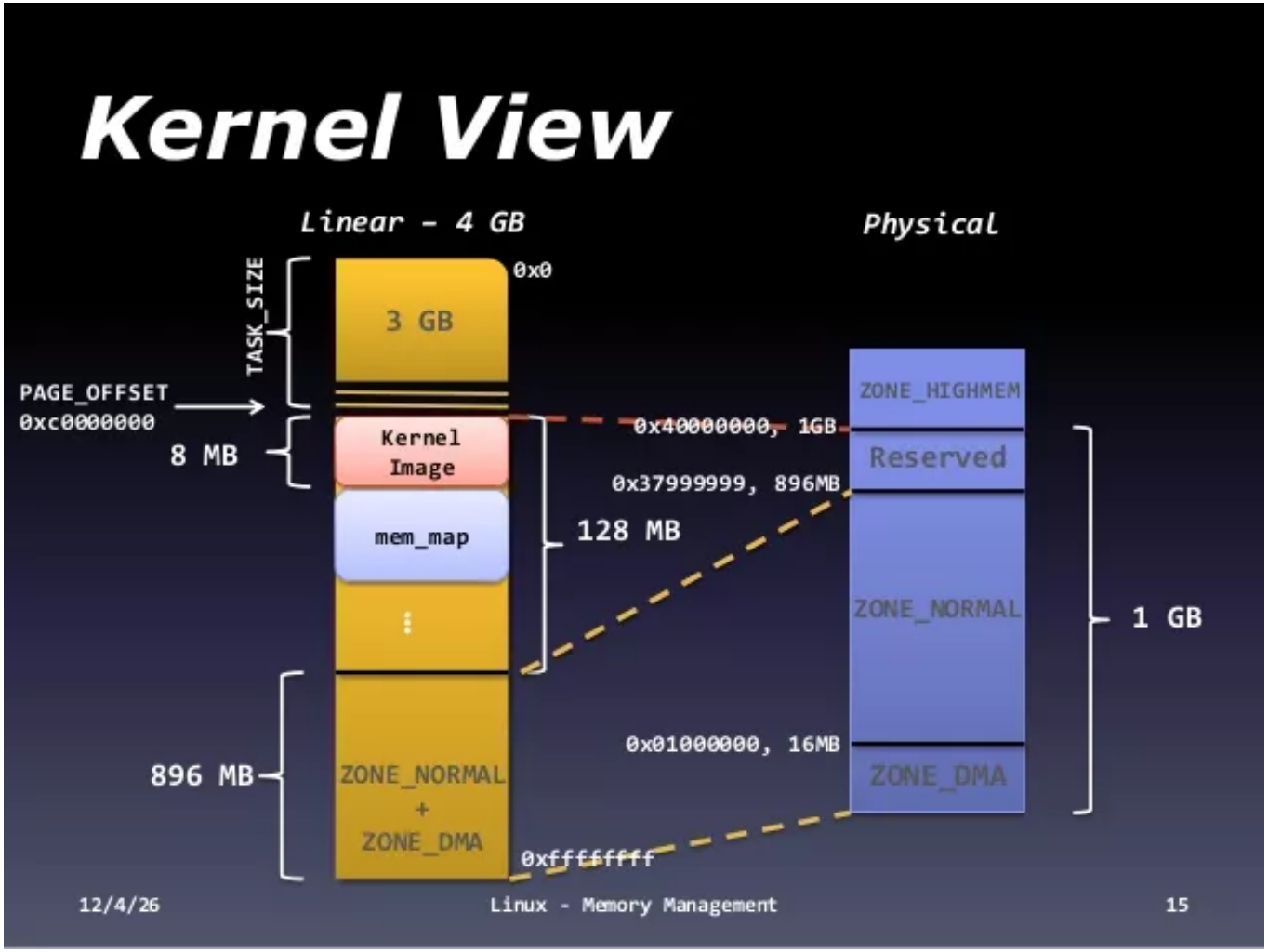
Linux 下找出吃內存的方法總結

終端研發部 今天

來自：Linux就該這麼學

linux下查詢進程佔用的內存方法總結，假設現在有一個「php-cgi」的進程，進程id為「25282」。現在想要查詢該進程佔用的內存大小。linux命令行下有很多的工具進行查看，現總結常見的幾種方式。



通過進程的status

```
[root@web3_u ~]# cat /proc/25282/status
Name: php-cgi
State: S (sleeping)
Tgid: 25282
Pid: 25282
PPid: 27187
```

```
TracerPid: 0
Uid: 99 99 99 99
Gid: 99 99 99 99
Utrace: 0
FDSize: 256
Groups: 99
VmPeak: 496388 kB
VmSize: 438284 kB
VmLck: 0 kB
VmHWM: 125468 kB
VmRSS: 113612 kB
VmData: 92588 kB
VmStk: 100 kB
VmExe: 6736 kB
VmLib: 18760 kB
VmPTE: 528 kB
VmSwap: 0 kB
Threads: 1
SigQ: 0/46155
SigPnd: 0000000000000000
ShdPnd: 0000000000000000
SigBlk: 0000000000000000
SigIgn: 0000000000001000
SigCgt: 0000000184000004
CapInh: 0000000000000000
CapPrm: 0000000000000000
CapEff: 0000000000000000
CapBnd: ffffffffffffffff
Cpus_allowed: f
Cpus_allowed_list: 0-3
Mems_allowed: 00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,00000000,0000
Mems_allowed_list: 0
voluntary_ctxt_switches: 68245
nonvoluntary_ctxt_switches: 15751
```

VmRSS: 113612 kB 表示佔用的物理內存

通過pmap

```
[root@web3_u ~]# pmap -x 25282
25282: /usr/local/php/bin/php-cgi --fpm --fpm-config /usr/local/php/etc/php-fpm.conf
Address Kbytes RSS Dirty Mode Mapping
0000000000400000 6736 2692 0 r-x-- php-cgi
0000000000c93000 264 196 120 rw--- php-cgi
0000000000cd5000 60 48 48 rw--- [ anon ]
. . .
00007fd6226bc000 4 4 4 rw--- ld-2.12.so
00007fd6226bd000 4 4 4 rw--- [ anon ]
00007fff84b02000 96 96 96 rw--- [ stack ]
00007fff84bff000 4 4 0 r-x-- [ anon ]
ffffffffff600000 4 0 0 r-x-- [ anon ]
---------------- ------ ------ ------
total kB 438284 113612 107960
```

關鍵信息點

1、進程ID

2、啟動命令

```
「/usr/local/php/bin/php-cgi --fpm --fpm-config /usr/local/php/etc/php-fpm.conf」
```

3、RSS :佔用的物理內存113612KB

通過smaps

```
[root@web3_u ~]# cat /proc/25282/smaps | grep '^Rss:' \
| awk '{sum +=$2} END{print sum}'
113612
```

求和得到實際佔用物理內存為113612

通過ps 命令

```
[root@web3_u ~]# ps -e -o 'pid,comm,args,pcpu,rsz,vsz,stime,user,uid' \
| awk '$1 ~ /25282/'
25282 php-cgi /usr/local/php/bin/php-cgi 0.0 113612 438284 Oct09 nobody 99
```

awk 過濾25282 進程號，得到第5列「rsz」的內存大小為「113612」

輸出php-cgi進程佔用的物理內存，並從高到低進行排序

```
[root@web3_u ~]# ps -e -o 'pid,comm,args,pcpu,rsz,vsz,stime,user,uid' \
| grep php-cgi | sort -k5nr
```

輸出結果

```
23946 php-cgi /usr/local/php/bin/php-cgi 0.0 129540 440000 Oct06 nobody 99
24418 php-cgi /usr/local/php/bin/php-cgi 0.0 129336 437684 Oct06 nobody 99
18973 php-cgi /usr/local/php/bin/php-cgi 0.0 129268 440176 Oct06 nobody 99
17219 php-cgi /usr/local/php/bin/php-cgi 0.0 126588 439840 Oct06 nobody 99
6996 php-cgi /usr/local/php/bin/php-cgi 0.0 124876 438104 Oct09 nobody 99
23850 php-cgi /usr/local/php/bin/php-cgi 0.0 122984 440036 Oct09 nobody 99
28310 php-cgi /usr/local/php/bin/php-cgi 0.0 122920 436456 Oct09 nobody 99
```

其中rsz為實際內存，上例實現按內存排序，由大到小

TOP 命令輸出的列

```
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
25282 nobody 20 0 428m 110m 93m S 0.0 1.9 0:34.42 php-cgi
```

輸出列信息

1、PID 25282
2、用户 nobody
3、虚拟内存 428M
4、物理内存 110M 110*1024= 112640 「和前面计算出来的值基本一致」
5、共享内存 93M
6、进程使用的物理内存和总内存的百分比 1.9 %

```
PID：进程的ID
USER：进程所有者
PR：进程的优先级别，越小越优先被执行
NInice：值
VIRT：进程占用的虚拟内存
RES：进程占用的物理内存
SHR：进程使用的共享内存
S：进程的状态。S表示休眠，R表示正在运行，Z表示僵死状态，N表示该进程优先值为负数
%CPU：进程占用CPU的使用率
%MEM：进程使用的物理内存和总内存的百分比
TIME+：该进程启动后占用的总的CPU时间，即占用CPU使用时间的累加值。
COMMAND：进程启动命令名称
```

按P

```
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
16036 root 20 0 8902m 8.6g 480 R 100.0 36.6 0:33.15 redis-server
12934 root 20 0 8902m 8.6g 1072 S 5.5 36.6 285:37.81 redis-server
969 root 20 0 0 0 0 D 4.2 0.0 277:14.85 flush-252:16
1304 root 23 3 1689m 50m 3264 S 4.2 0.2 1445:03 xs-searchd
1294 root 20 0 14928 928 584 S 3.5 0.0 635:05.31 xs-indexd
1287 nobody 20 0 12884 772 576 S 2.8 0.0 833:11.42 dnsmasq
1302 root 23 3 1113m 39m 3244 S 0.7 0.2 1437:57 xs-searchd
```

```
4444 www 20 0 280m 43m 884 S 0.7 0.2 27:43.92 nginx
1 root 20 0 19232 1160 868 S 0.0 0.0 0:06.75 init
```

按 P .表示按cpu排序，默认也是按cpu排序

按M

```
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
12934 root 20 0 8902m 8.6g 1072 S 6.0 36.6 285:39.77 redis-server
16036 root 20 0 8902m 8.6g 480 R 100.0 36.6 1:11.42 redis-server
1236 www 20 0 1053m 209m 6556 S 0.0 0.9 4:40.70 php-cgi
1231 www 20 0 1034m 146m 6536 S 0.0 0.6 4:20.82 php-cgi
1184 www 20 0 1043m 119m 6584 S 0.0 0.5 4:21.85 php-cgi
```

按M 。表示按占用内存排序。第一列 redis服务器占用了8.6G的内存 。这个内存和redis info

```
[root@img1_u ~]# redis-cli info memory
# Memory
used_memory_human:8.32G
```

基本相同。

```
[root@img1_u ~]# top -u www
top - 22:09:01 up 67 days, 14:16, 1 user, load average: 0.61, 0.90, 0.98
Tasks: 283 total, 2 running, 281 sleeping, 0 stopped, 0 zombie
Cpu(s): 3.9%us, 1.0%sy, 0.5%ni, 89.7%id, 4.6%wa, 0.0%hi, 0.3%si, 0.0%st
Mem: 24542176k total, 21130060k used, 3412116k free, 1750652k buffers
Swap: 524280k total, 0k used, 524280k free, 4039732k cached

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
681 www 20 0 855m 25m 5796 S 0.0 0.1 0:47.00 php-cgi
1181 www 20 0 887m 57m 6484 S 0.0 0.2 4:41.66 php-cgi
1183 www 20 0 864m 34m 6320 S 0.0 0.1 3:52.39 php-cgi
```

```
1184 www 20 0 1043m 119m 6584 S 0.0 0.5 4:21.85 php-cgi
1185 www 20 0 869m 39m 6376 S 0.0 0.2 3:57.84 php-cgi
1186 www 20 0 886m 56m 6244 S 0.0 0.2 3:44.75 php-cgi
1187 www 20 0 926m 66m 6480 S 0.0 0.3 4:16.12 php-cgi
1188 www 20 0 890m 60m 6288 S 0.0 0.3 4:13.35 php-cgi
1189 www 20 0 892m 62m 6408 S 0.0 0.3 4:06.60 php-cgi
```

-u 指定用户。php-cgi占用的内存在60M左右

按进程消耗内存多少排序的方法

通过 ps 命令

第一种方法

```
ps -e -o 'pid,comm,args,pcpu,rsz,vsz,stime,user,uid' | sort -k5nr
```

第二种方法

```
ps -e -o 'pid,comm,args,pcpu,rsz,vsz,stime,user,uid' --sort -rsz
```

输出结果

```
[root@web3_u ~]# ps -e -o 'pid,comm,args,pcpu,rsz,vsz,stime,user' | sort -k5nr
23946 php-cgi /usr/local/php/bin/php-cgi 0.0 129540 440000 Oct06 nobody
24418 php-cgi /usr/local/php/bin/php-cgi 0.0 129336 437684 Oct06 nobody
18973 php-cgi /usr/local/php/bin/php-cgi 0.0 129268 440176 Oct06 nobody
17219 php-cgi /usr/local/php/bin/php-cgi 0.0 126588 439840 Oct06 nobody
6996 php-cgi /usr/local/php/bin/php-cgi 0.0 125056 438104 Oct09 nobody
23850 php-cgi /usr/local/php/bin/php-cgi 0.0 122984 440036 Oct09 nobody
```

参数解析:

- -e 显示所有进程
- -o 定制显示信息
- pid 进程ID
- comm 进程名
- args 启动命令
- pcpu 占用CPU 百分比
- rsz 占用物理内存大小
- vsz 占用虚拟内存大小
- stime 进程启动时间
- user 启动用户

以第一行为例

```
进程ID 23946
进程名 php-cgi
启动命令 /usr/local/php/bin/php-cgi
占用CPU 0
占用物理内存 129540
占用虚拟内存 440000
启动时间 Oct06
启动用户 nobody
```

通过 top 命令

top命令默认是以CPU排序输出的，按字母「M」，可以按内存占用大小进行排序显示

```
PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
23946 nobody 20 0 429m 126m 107m S 0.0 2.2 1:15.01 php-cgi
24418 nobody 20 0 427m 126m 109m S 0.0 2.2 1:19.56 php-cgi
18973 nobody 20 0 429m 126m 107m S 0.0 2.2 1:20.18 php-cgi
17219 nobody 20 0 429m 123m 104m S 0.0 2.1 1:23.60 php-cgi
6996 nobody 20 0 427m 122m 105m S 0.0 2.1 1:05.27 php-cgi
23850 nobody 20 0 429m 120m 101m S 0.0 2.1 1:02.43 php-cgi
```

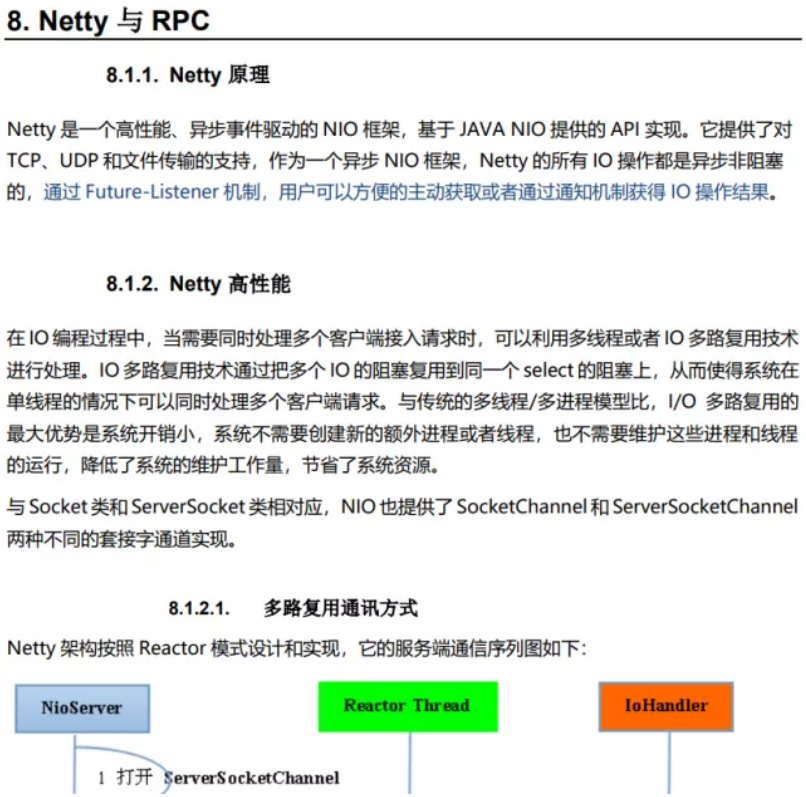
输出参数介绍

- PID：进程的ID
- USER：进程所有者
- VIRT：进程占用的虚拟内存
- RES：进程占用的物理内存
- SHR：进程使用的共享内存
- S：进程的状态。S表示休眠，R表示正在运行，Z表示僵死状态，N表示该进程优先值为负数
- %CPU：进程占用CPU的使用率
- %MEM：进程使用的物理内存和总内存的百分比
- TIME+：该进程启动后占用的总的CPU时间，即占用CPU使用时间的累加值。

通过比较进程ID 「23946」，top 命令和 ps 命令输出的结果基本保持一致

推荐阅读

BAT等大厂Java面试经验总结



8. Netty 与 RPC

8.1.1. Netty 原理

Netty 是一个高性能、异步事件驱动的 NIO 框架，基于 JAVA NIO 提供的 API 实现。它提供了对 TCP、UDP 和文件传输的支持，作为一个异步 NIO 框架，Netty 的所有 IO 操作都是异步非阻塞的，通过 Future-Listener 机制，用户可以方便的主动获取或者通过通知机制获得 IO 操作结果。

8.1.2. Netty 高性能

在 IO 编程过程中，当需要同时处理多个客户端接入请求时，可以利用多线程或者 IO 多路复用技术进行处理。IO 多路复用技术通过把多个 IO 的阻塞复用到同一个 select 的阻塞上，从而使得系统在单线程的情况下可以同时处理多个客户端请求。与传统的多线程/多进程模型比，I/O 多路复用的最大优势是系统开销小，系统不需要创建新的额外进程或者线程，也不需要维护这些进程和线程的运行，降低了系统的维护工作量，节省了系统资源。

与 Socket 类和 ServerSocket 类相对应，NIO 也提供了 SocketChannel 和 ServerSocketChannel 两种不同的套接字通道实现。

8.1.2.1. 多路复用通讯方式

Netty 架构按照 Reactor 模式设计和实现，它的服务端通信序列图如下：

想获取 Java大厂面试题学习资料
扫下方二维码回复「**BAT**」就好了

一些秘密

回复 【加群】 获取github掘金交流群
回复 【电子书】 获取2020电子书教程
回复 【C】 获取全套C语言学习知识手册

回复 【Java】获取java相关的视频教程和资料

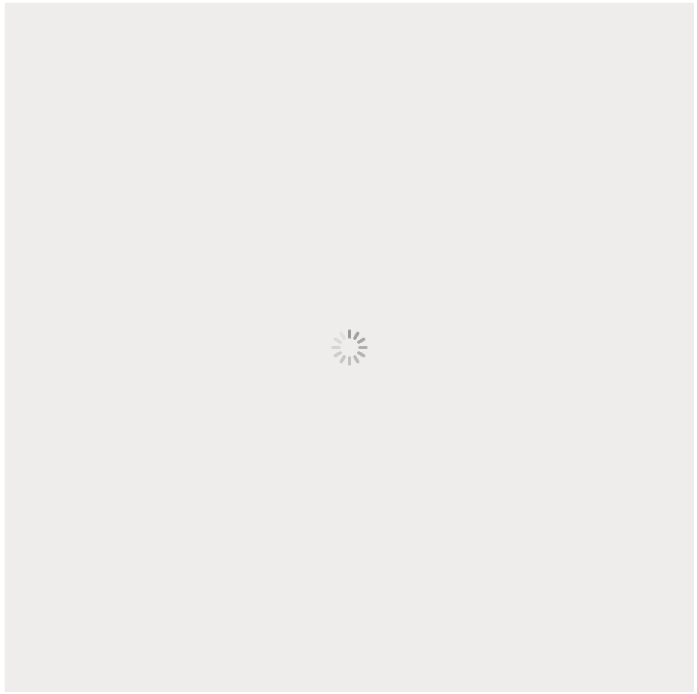
回复 【爬虫】获取SpringCloud相关多的学习资料

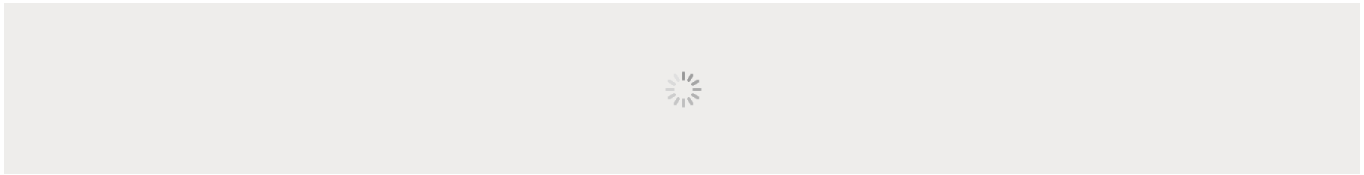
回复 【Python】即可获得Python基础到进阶的学习教程

回复 【idea破解】即可获得intellij idea相关的破解教程

回复 【BAT】即可获得intellij idea相关的破解教程

关注我gitHub掘金，每天发掘一篇好项目，学习技术不迷路!





回复 【idea激活】即可获得idea的激活方式

回复 【Java】获取java相关的视频教程和资料

回复【SpringCloud】获取SpringCloud相关多的学习资料

回复【python】获取全套0基础Python知识手册

回复【2020】获取2020java相关面试题教程

回复【加群】即可加入终端研发部相关的技术交流群

阅读更多

为什么HTTPS是安全的

因为BitMap，白白搭进去8台服务器...

《某廠內部SQL大全》.PDF

字節跳動一面：i++ 是線程安全的嗎？

大家好，歡迎加我微信，很高興認識你！

在華為鴻蒙OS 上嚐鮮，我的第一個“hello world”，起飛！

相信自己，沒有做不到的，只有想不到的

在這裡獲得的不僅僅是技術！

喜歡就給個“在看”

閱讀原文

喜歡此內容的人還喜歡

圖解Docker 架構

架構師