## L1 题目

### 下面那个不能描述进程状态( D)

A: R

B: S

C: D

D: W

### 通过uptime命令我们可以获取系统的负载长度,这三个信息分别表示的是 (A )

$ uptime

19:55:38 up 35 days, 5:21, 1 user, load average: 0.27, 0.47, 0.52

A: 过去一分钟,五分钟,十五分钟

B： 过去十五分钟,五分钟,一分钟

C： 过去一分钟,五分钟,十分钟

D： 过去十分钟,五分钟,一分钟

### 通过下面这条命令能获取的信息是 ( A )

cat /proc/cpuinfo | grep processor | wc –l

A: 系统逻辑CPU颗数

B： 系统物理CPU颗数

C： 系统内核CPU颗数

D： 系统CPU封装数

### 通过vmstat监控的（A ）可以统计系统这段时间的负载

A: r

B: b

C: r+b

D si+so

### 通过vmstat监控发现很大可以看出系统物理内存不足 AB

A: si

si 每秒从磁盘读入虚拟内存的大小，如果这个值大于0，表示物理内存不够用或者内存泄露了，要查找耗内存进程解决掉。我的机器内存充裕，一切正常

B: so

so 每秒虚拟内存写入磁盘的大小，如果这个值大于0，同上

C: bi

D: bo

### 通过vmstat监控发现（CD）很大可以看出系统IO存在读瓶颈

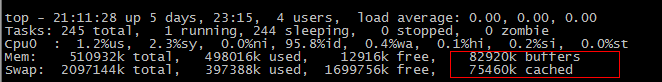
A: si

B: so

C: bi

D: bo

### 如下图所示,top命令获取的buffer表示的含义是( C)



A: swap 分区用来给块设备做缓冲

B: swap 分区用来给文件做缓冲

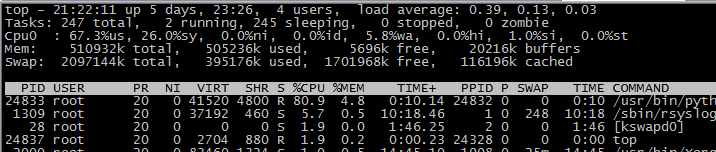
C: 物理内存用来给设备做缓冲

D: 物理内存用来给文件做缓冲

 **buffers**：用于存放要输出到disk（块存储）的数据，在这里buff是指被OS buffer住的内存（written to disk）；

 **cached**：存放从disk上读出的数据；buffer和cache是为了提高IO性能并由OS管理。

### top命令获取的TIME表示的是 ( D)



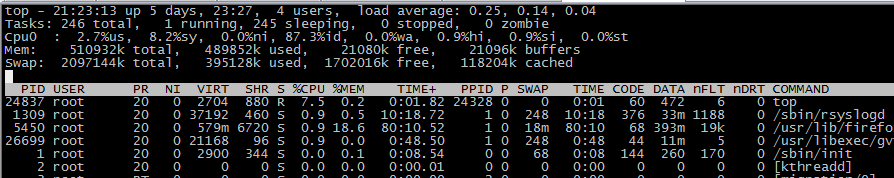
A: 进程存活时间

B: 进程处于运行状态的时间

C: 进程占用的用户态CPU时间

D: 进程占用的CPU时间

### 下图哪列信息描述了进程实际使用的内存大小 ( A)



A: VIRT

B: SHR

C: CODE

D: 没有答案

VIRT— 进程使用的虚拟内存。进程使用的虚拟内存总量，单位kb。VIRT=SWAP+RES

RES— 驻留内存大小。驻留内存是任务使用的非交换物理内存大小。进程使用的、未被换出的物理内存大小，单位kb。RES=CODE+DATA

SHR— SHR是进程使用的共享内存。共享内存大小，单位kb

### 在top命令中通过( C ) 可以计算系统当前使用的cpu信息

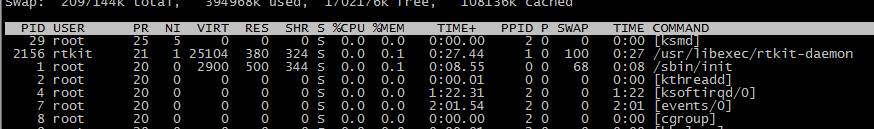
A: us + sys

B: idle

C: 1 - idle

D: us + sys + hi + si

### 如下图,进程的按优先级排序的顺序是 （ A ）



A PID 1 > PID 2156 > PID 29

B PID29 > PID 2156 > PID 1

C PID 1 > PID 29 > PID 2156

D 没有正确答案

**什么是NICE值?**

NICE值应该是熟悉Linux/UNIX的人很了解的概念了，它是反应一个进程“优先级”状态的值，其取值范围是-20至19，一共40个级别。

**这个值越小，表示进程”优先级”越高，而值越大“优先级”越低。**

PR=NI+20 pr越小优先级越高

### 想通过sar命令每3秒监控CPU0,总共监控5次的命令是 （ AB）

A sar -u 0 3 5

B sar -u 0 3 5

C sar -P 0 3 5

D 没有正确答案

### 通过(ABCD) 命令可以列出对应的线程信息

A ps aux

B ps -ef

C ps –eLf

D ps axjf

### 可以得出错误的结论是: （C ）

A: 通过vmstat我们可以查询inactive,active内存信息

B：Active是活跃的内存

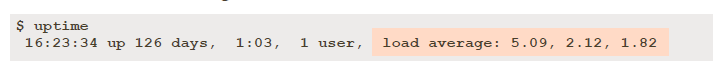
C: Inactive的内存没有被申请,它的计数在free里面

D: Inactive内存是有一段时间没有被使用的内存

**Memory（内存）：**

inact: 非活跃内存大小（当使用-a选项时显示）  
active: 活跃的内存大小（当使用-a选项时显示）

### 通过下面这个信息,我们可以判断结论1,2分别是 (A )



1. 过去1分钟,五分钟,15分钟的平均运行的进程为5.09,2.12,1.82, 目前的系统负载很忙
2. 当负载长度很大的时候,线程之间的调度可能会受到影响,甚至有延迟

A: T T

B: T F

C: F T

D: F F

### 下面看机器是几核逻辑CPU错误的方法是( B )

A: top 按下 1

B: sar –P ALL

C: sar –u ALL

D: grep processor /proc/cpuinfo | wc -l

### 下面哪个命令不可以看到进程内存映像信息 (D)

A: pmap -x 1

B: pmap -d 1

C: cat /proc/1/maps

D: cat /proc/1/statm

### 定位内存相关的问题,我们不能用的sar命令是 ( A)

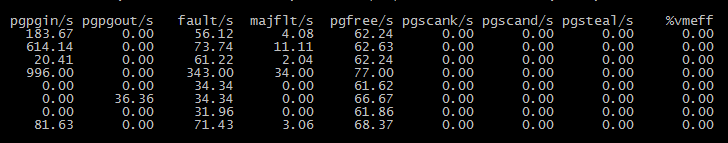
A: sar –q

B: sar -W

C: sar –B

D: sar -r

### `如下图所示,做内存分析时,哪些结论不正确 (C )



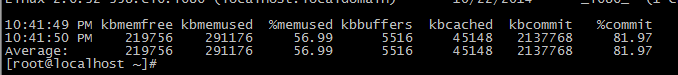
A: pgpgin/s：表示每秒从内存置换到磁盘或SWAP的字节数,当内存不足时,计数可能会很大

B： majflt/s表示每秒钟系统产生的主缺页数,当内存不足时,这个计数可能会很大

C: pgfree/s：每秒被放入空闲队列中的页个数

D: %vmeff 很低以为这内存不足

### 如下所示,结论不正确的是 (C )



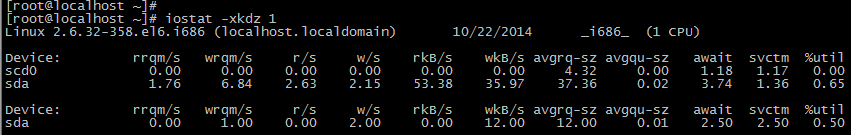
A: Kbmemfree = mem.free

B： Memused= (mem.used + swap.used )% mem.total

C: commit = kbcommit / (mem.total + swap.total)

D: 以上答案都正确

### 如下所示,下面哪个结论是正确的 ( B)



A: await表示的是设备发出 I/O 请求经过的平均时间,单位是秒

B： 第一次取的数据不能作为参考

C: avgrq-sz 表示的是读请求大小

D: %util 有可能大于100%

### 可以用来监控进程的IO信息的命令有 （A）

A: iotop

B：pidstat

C: dmesg -c

D: 上面命令都可以

### 下面那个参数sar命令不能用来监控系统IO （ D ）

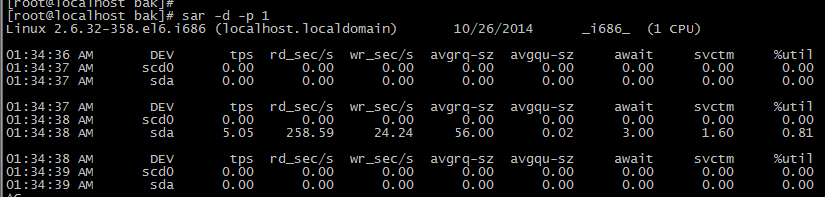
A -B

B -b

C -d

D: -n NFS

### 关于sar 命令获取设备使用情况,正确的是 ( C)



avgqu-sz 的值较低时，因为我们磁盘请求的队列少，所以设备的利用率较高,但要注意的是它是一个按照单位时间来计算的平均值，不能直接反应瞬间的I/O

A : avgqu-sz发给设备请求的平均扇区数,它反映了单位时间里平均排队的队列

B svctm表示了设备I/0 请求的平均服务时间,包括在请求队列中消耗的时间.

C await表示设备 I/O 请求平均等待时间（单位毫秒）,包括请求队列等待时间和服务时间.

D: wr\_sec/s表示每秒向设备写扇区的次数,单位是1K.（512）

### 关于ifconfig命令描述不正确的是 ( A)

A : 可以查询网卡实时收发包速率

B 可以查询网卡累计收发包计数

C 可以查询网卡地址信息

D: 可以查询网卡丢包计数

### 关于sar命令监听网络接口,不正确的是 (D)

A: sar –n DEV 能监听接口实时收发包速率

B: sar –n EDEV能监听接口实时丢包速率

C: sar –n SOCK 能监听当前TCP SOCKET的计数

D: sar命令不能查询IP 分片报文的计数

### 关于ping命令的描述, 不正确的是 (C)

A: ping命令可以统计网络延迟

B: 可以指定ping的原接口

C: 最大可以发出的ping报文大小是9000B

D: 可以修改服务器,让对方禁ping

### 通过tcpdump获取接口eth0,主机road发送的报文 ( B )

A: tcpdump –i eth0 dst host road

B: tcpdump –i eth0 src host road

C: tcpdump –p eth0 dst host road

D: tcpdump –p eth0 src host road

### 关于路由的信息,下面说法不正确的是 (A)

A: 通过route命令添加的路由能永久生效

B: 路由包括静态路由,动态路由,默认路由

C: route命令是用于操作基于内核ip路由表

D: 通过ip route show可以查看当前机器路由信息

## 判断题,判断下面说法的正确错误

### 频繁的磁盘读不写会增加对内存的使用 （X）

### 大量的网络吞吐,意味着有一定的CPU消耗 （Y）

### 可用物理内存的减少可能会换出到虚拟内存 （Y）

### 进程状态S和D都处于睡眠状态且都能被提前唤醒（y）

### Vmstat命令中的 r b 表示的分别是运行的线程数和不可中断的线程数 （x）

### 关于中断它包含两种分别是软件中断和硬件中断（y）

### 我们可以修改/proc/irq/ 下面的中断配置,来打散CPU对于中断的亲和力 （y）

### Ksoftirqd/0 表示的是cpu0用来处理软件中断的应用程序 （y）

### 对于IO的效率排序应该是 内存 > 虚拟内存 > 磁盘 （y）

### 内存回收机制我们用的是LRU, 既最近不被使用的内存,应最先被回收 （y）

### 当发现swap 有used计数,说明内存存在问题了 （y）

### 系统使用虚拟内存的技术利用磁盘来对内存空间进行扩展 （y）

### 通过swap in, swap out 这两个速率可以看出物理内存是否存在瓶颈 （y）

### 系统会将一些大块的内存写入磁盘来释放更多的可用内存.当需要的时候再重新加载到内存中 （y）

### 缓存计数主要是允许系统把存放在磁盘上的某些数据保留在内存中,目的是下次可以不需要再访问磁盘 （Y）

### 脏页是指的进程对缓冲区修改后的数据,且为了做标记所定义的名称 （Y）

脏页是Linux内核中的概念，因为硬盘的读写速度远赶不上内存的速度，系统就把读写比较频繁的数据事先放到内存中，以提高读写速度，这就叫高速缓存，linux是以页作为高速缓存的单位，当进程修改了高速缓存里的数据时，该页就被内核标记为脏页，内核将会在合适的时间把脏页的数据写到磁盘中去，以保持高速缓存中的数据和磁盘中的数据是一致的。

### 脏页生存周期有限,可以修改内核配置加快和磁盘之间的同步 （Y）

### 缺页分主缺页和次缺页,通过次缺页可以看出内存是否存在瓶颈 （Y）

## 三、缺页中断

可分为下面两种

* 主缺页中断（Major Page Fault）  
  要从磁盘读取数据而产生的中断是主缺页中断
* 次缺页中断（Minor Page Fault）  
  数据已经被读入内存并被缓存起来，从内存缓存区中而不是直接从硬盘中读取数据而产生的中断是次缺页中断。

### 一般情况下是不允许使用过多的内存,但内核里面有个配置可允许系统commit更多的内存 （Y）

### 当物理内存不够用的时候,可以强行把物理内存缓存数据同步到磁盘,并释放部分物理内存 （Y）

1. **Swap用途**
2. Swap意思是交换分区，通常我们说的虚拟内存，是从硬盘中划分出的一个分区。当物理内存不够用的时候，内核就会释放缓存区（buffers/cache）里一些长时间不用的程序，然后将这些程序临时放到Swap中，也就是说如果物理内存和缓存区内存不够用的时候，才会用到Swap。

### 我们可以用sar,vmstat,dstat等命令分析进程的内存使用情况 （Y）

dstat命令是一个用来替换vmstat、iostat、netstat、nfsstat和ifstat这些命令的工具，是一个全能系统信息统计工具。与sysstat相比，dstat拥有一个彩色的界面，在手动观察性能状况时，数据比较显眼容易观察；而且dstat支持即时刷新，譬如输入dstat 3即每三秒收集一次，但最新的数据都会每秒刷新显示。和sysstat相同的是，dstat也可以收集指定的性能资源，譬如dstat -c即显示CPU的使用情况。

### IO 分顺序IO和随机IO, 虽然略有不同,但对应性能指标是差不多（x）

刷盘分随机IO和顺序IO，两者性能相差很大

### 当IO出现瓶颈的时候,同样swap也会出现一定的瓶颈 （y）

### 通过每秒的读写请求数就能看出IO性能是否有问题了 (y )

### %util数值越高,证明系统IO越正常 （x）

iostat

%util 表示磁盘忙碌情况，一般该值超过80%表示该磁盘可能处于繁忙状态。

rrqm/s:          每秒进行 merge 的读操作数目。即 delta(rmerge)/s  
wrqm/s:         每秒进行 merge 的写操作数目。即 delta(wmerge)/s  
r/s:            每秒完成的读 I/O 设备次数。即 delta(rio)/s  
w/s:            每秒完成的写 I/O 设备次数。即 delta(wio)/s  
rsec/s:         每秒读扇区数。即 delta(rsect)/s  
wsec/s:         每秒写扇区数。即 delta(wsect)/s  
rkB/s:          每秒读K字节数。是 rsect/s 的一半，因为每扇区大小为512字节。(需要计算)  
wkB/s:          每秒写K字节数。是 wsect/s 的一半。(需要计算)  
avgrq-sz:       平均每次设备I/O操作的数据大小 (扇区)。delta(rsect+wsect)/delta(rio+wio)  
avgqu-sz:       平均I/O队列长度。即 delta(aveq)/s/1000 (因为aveq的单位为毫秒)。  
await:          平均每次设备I/O操作的等待时间 (毫秒)。即 delta(ruse+wuse)/delta(rio+wio)  
svctm:          平均每次设备I/O操作的服务时间 (毫秒)。即 delta(use)/delta(rio+wio)  
%util:          一秒中有百分之多少的时间用于 I/O 操作，或者说一秒中有多少时间 I/O 队列是非空的。即 delta(use)/s/1000 (因为use的单位为毫秒)

如果 %util 接近 100%，说明产生的I/O请求太多，I/O系统已经满负荷，该磁盘可能存在瓶颈。

### 当统计IO的平均时间时,只需要查看对应svctm的数值就好 （x）

### 内核中有相关debug IO的信息,到时可以通过dmesg命令从内核中实时获取对应进程的IO利用率 （x）

dmesg用来显示内核环缓冲区（kernel-ring buffer）内容，内核将各种消息存放在这里。在系统引导时，内核将与硬件和模块初始化相关的信息填到这个缓冲区中。内核环缓冲区中的消息对于诊断系统问题 通常非常有用。在运行dmesg时，它显示大量信息。通常通过less或grep使用管道查看dmesg的输出，这样可以更容易找到待查信息。例如，如果发现硬盘性能低下，可以使用dmesg来检查它们是否运行在DMA模式：

$dmesg | grep DMA

...

ide0: BM-DMA at 0xf000-0xf007, BIOS settings: hda:DMA, hdb:DMA

ide1: BM-DMA at 0xf008-0xf00f, BIOS settings: hdc:DMA, hdd:DMA

...

### ifconfig命令可以查看单个或全部接口收发包的累计大小,且会一直递增 （y）

### 针对过来的网络报文,除了需要考虑正常的包外,分片包,组包等情况也是需要进一步考虑,其中分片包只可能是发送的包的大小大于接口的mtu值 （y）

片，从字面上也可以猜到是什么意思，就是当一个skb包长度大于传输设备或者链路上物理设备的mtu时，会根据一定的方式进行切割，从而使报文得以发送出去。但是这里需要说明，分片又分为IP和TCP分片两种，由于tcp报文有自己的机制去分片，不需要依赖IP层分片；而对于udp或者icmp等报文，只能依赖IP层去分片。

### tcpdump可以捕获接口的报文,且可以实时捕获 （x）

tcpdump采用命令行方式对接口的数据包进行筛选抓取，其丰富特性表现在灵活的表达式上

### netstat 命令可以查看当前udp,tcp协议的所有端口使用情况 （y）

### 通过netstat命令查看连接数的信息,可以帮我们做初步的Dos攻击防护 （y）

### 通过route命令可以永久的把地址路由给添加进系统 （x）

### 网络报文吞吐量我们可以通过sar –n DEV 来实时统计接口收发包计数 （y）

# sar -n DEV 1 1

Linux 2.6.32-358.el6.x86\_64 (host\_linux)01/17/2018 \_x86\_64\_(2 CPU)  
01:39:36 PM     IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s  
01:39:37 PM        lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00  
01:39:37 PM      eth0     23.23      1.01      1.55      0.10      0.00      0.00      0.00  
Average:        IFACE   rxpck/s   txpck/s    rxkB/s    txkB/s   rxcmp/s   txcmp/s  rxmcst/s  
Average:           lo      0.00      0.00      0.00      0.00      0.00      0.00      0.00  
Average:         eth0     23.23      1.01      1.55      0.10      0.00      0.00      0.00

sar -n DEV  输出网络设备状态的统计信息

例如：sar -n DEV 1 2  将显示lo、eth0、eth1等信息   
IFACE：就是网络设备的名称；   
rxpck/s：每秒钟接收到的包数目   
txpck/s：每秒钟发送出去的包数目   
rxbyt/s：每秒钟接收到的字节数   
txbyt/s：每秒钟发送出去的字节数   
rxcmp/s：每秒钟接收到的压缩包数目   
txcmp/s：每秒钟发送出去的压缩包数目   
txmcst/s：每秒钟接收到的多播包的包数目

### ethtool命令可以用来修改网卡支持的模式,且可以扩展所有网卡的最大速率到千兆 (y )

ethtool 是用于查询及设置网卡参数的命令。

**1）[root@linux /]# which ethtool**        //查询 ethtool 存放的路径  
    /sbin/ethtool  
**2）[root@linux /]# rpm -qf /sbin/ethtool**        //查询 ethtool 的版本信息  
   ethtool-1.6-5  
**3）将 ethtool 设置永久保存在网络设备的方法**  
        解决方法一:  
        ethtool 设置可通过 /etc/sysconfig/network-scripts/ifcfg-ethX 文件保存，从而在设备下次启动时激活选项。  
例如：ethtool -s eth0 speed 100 duplex full autoneg off  
此指令将eth0设备设置为全双工自适应，速度为100Mbs。若要eth0启动时设置这些参数, 修改文件/etc/sysconfig/network-scripts/ifcfg-eth0 ，添加如下一行:  
ETHTOOL\_OPTS="speed 100 duplex full autoneg off"  
         解决方法二:  
         将ethtool设置写入/etc/rc.d/rc.local之中。

## 问答题

### 从/proc/cpuinfo里面生成了下表信息，请回答如下问题

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Processor | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| physical id | 0 | 1 | 2 | 3 | 0 | 1 | 2 | 3 |
| core id | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| siblings | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |
| cpu cores | 2 | 2 | 2 | 2 | 2 | 2 | 2 | 2 |

1. 通过上面的表格，我们能确认哪些信息

物理CPU个数：=4

cat /proc/cpuinfo | grep "physical id" | sort -u | wc -l

#逻辑CPU个数=8

cat /proc/cpuinfo | grep "processor" | wc –l

#每个物理CPU中Core的个数：=2

cat /proc/cpuinfo | grep "cpu cores" | uniq | awk -F: '{print $2}'

查看core id的数量,即为所有物理CPU上的core的个数

cat /proc/cpuinfo | grep "core id" | uniq |  wc -l

#是否为超线程？

#如果有两个逻辑CPU具有相同的”core id”，那么超线程是打开的。或者siblings数目比cpu cores数目大。

#每个物理CPU中逻辑CPU(可能是core, threads或both)的个数：

cat /proc/cpuinfo | grep "siblings"

1. 如何快速的统计cpu的物理颗数，逻辑cpu核数值

如上

grep "physical id" /proc/cpuinfo |sort |uniq|wc –l

grep process /proc/cpuinfo |wc -l



### 如何查看系统能支持安装的rpm包版本?

　 abc-1.2.3-4.i386.rpm

　　abc-1.2.3-4.i586.rpm

　　abc-1.2.3-4.i686.rpm

abc-1.2.3-4.i786.rpm

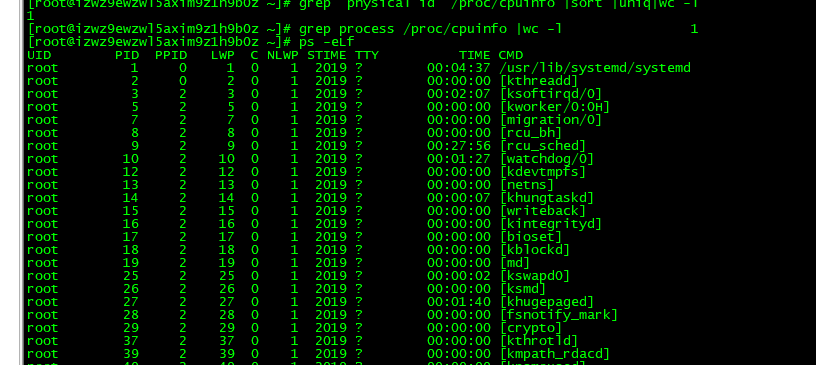
答案：

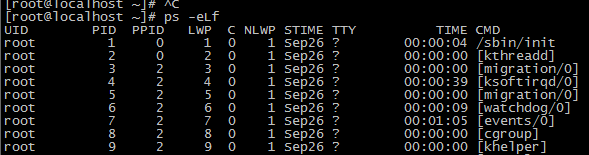


从/proc/cpuinfo的”cpu family” 可以看出它支持的rpm版本有 386,586,686 不支持安装786的rpm包

### 如何查看系统线程的信息,以及对应的父进程ID

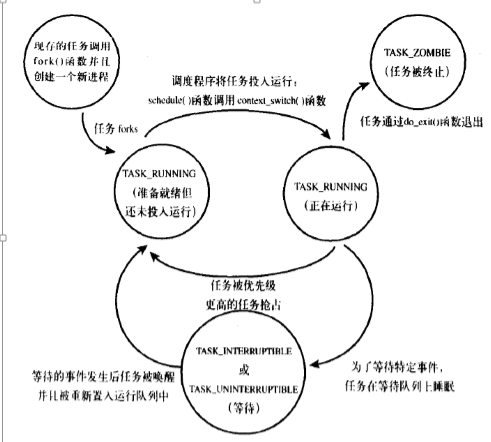
**答案：**





### 请描述进程状态R D S 之间的状态轮转

**答案：**

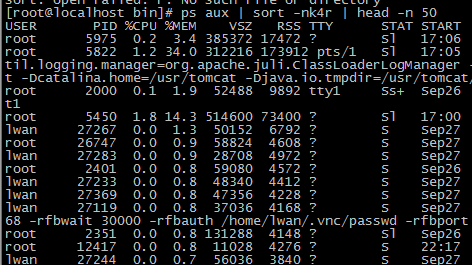


### 找出java相关的进程并把它给kill掉： 取进程号的方法是通过管道以及 awk ‘{print $2}’。

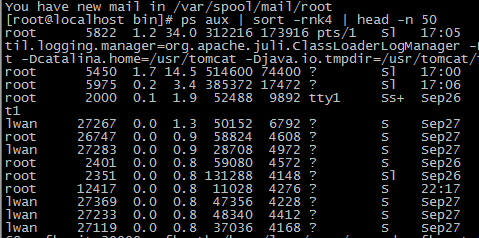
答案：



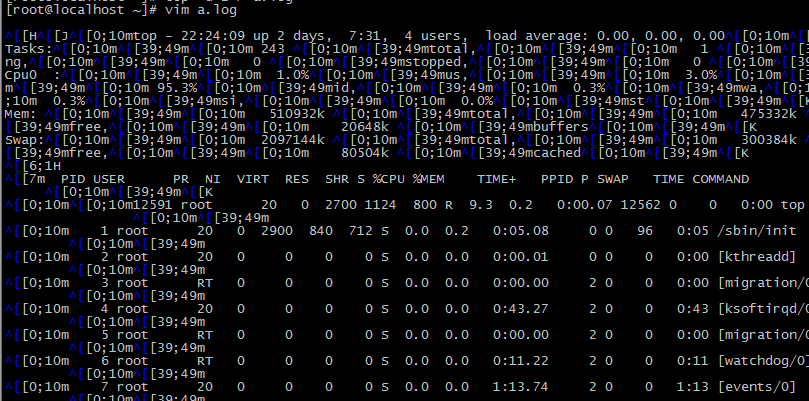
### 如下图所示,找出为什么内存排序失败的原因,该如何写才能内存排序正确



**答案：**



### 下图为测试人员监控到的top log, 请分析会出现这类日志的原因,我们该如何修复



不会

### 看图并回答-/+ buffer/cache的意义以及推算方法



-/+ buffers/cache的意思相当于：  
-buffers/cache 的内存数：892 (等于第1行的 used - buffers - cached)  
+buffers/cache 的内存数: 1155 (等于第1行的 free + buffers + cached)  
  
可见-buffers/cache反映的是被程序实实在在吃掉的内存，而+buffers/cache反映的是可以挪用的内存总数。

### 查看下图内存的使用率,并回答下面的问题



如上图所示,请解释free命令中 free, buffers, cached的含义

上图是否意味着内存不够用了? 并说明原因

-/+ buffers/cache的意思相当于：  
-buffers/cache 的内存数：892 (等于第1行的 used - buffers - cached)  
+buffers/cache 的内存数: 1155 (等于第1行的 free + buffers + cached)  
  
可见-buffers/cache反映的是被程序实实在在吃掉的内存，而+buffers/cache反映的是可以挪用的内存总数。

### 请描述内存buffer和cache的区别

磁盘是一个块设备，可以划分为不同的分区；在分区之上再创建文件系统，挂载到某个目录，之后才可以在这个目录中读写文件。

其实 Linux 中“一切皆文件”，而提到的“文件”是普通文件，磁盘是块设备文件，可以执行 "ls -l <路径>" 查看它们的区别。

**在读写普通文件时，会经过文件系统，由文件系统负责与磁盘交互；而读写磁盘或者分区时，就会跳过文件系统，也就是所谓的“裸I/O“。**

这两种读写方式所使用的缓存是不同的，也就是 Cache 和 Buffer 区别。

* Buffer 既可以用作“将要写入磁盘数据的缓存”，也可以用作“从磁盘读取数据的缓存”。
* Cache 既可以用作“从文件读取数据的页缓存”，也可以用作“写文件的页缓存”。

**简单来说，Buffer 是对磁盘数据的缓存，而 Cache 是文件数据的缓存，它们既会用在读请求中，也会用在写请求中**

### 假定被测设备是一个8核cpu,下图为通过vmstat监控得到的数据,请分析当前系统是否存在瓶颈,以及下一步定位问题的方法.

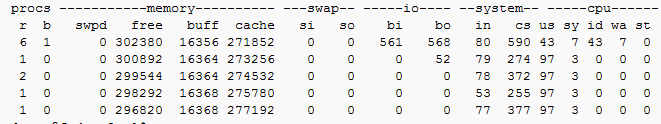
<https://blog.csdn.net/wzyzzu/article/details/50395502>

r（运行队列）展示了正在执行和等待CPU资源的任务个数。当这个值超过了CPU数目，就会出现CPU瓶颈了。

Linux下查看CPU核心数的命令：  
cat /proc/cpuinfo|grep processor|wc -l

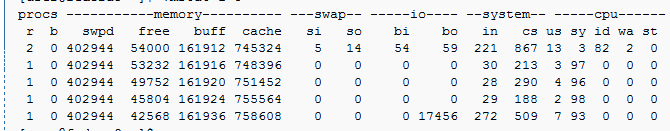
当r值超过了CPU个数，就会出现CPU瓶颈，解决办法大体几种：

1. 最简单的就是增加CPU个数和核数  
2. 通过调整任务执行时间，如大任务放到系统不繁忙的情况下进行执行，进尔平衡系统任务  
3. 调整已有任务的优先级



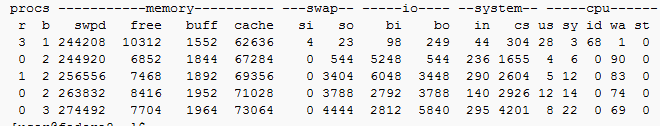
### 假定被测设备是一个8核cpu,下图为通过vmstat监控得到的数据,请分析当前系统是否存在瓶颈,以及下一步定位问题的方法.

同上

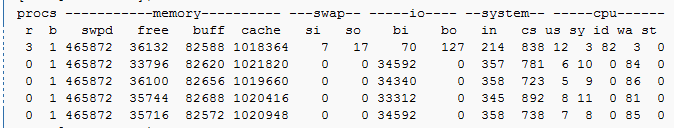


### 假定被测设备是一个8核cpu,下图为通过vmstat监控得到的数据,请分析当前系统是否存在瓶颈,以及下一步定位问题的方法.

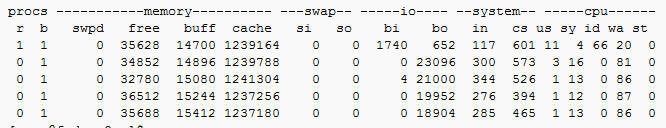
同上



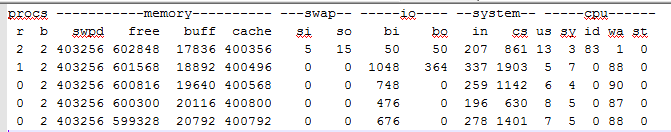
### 假定被测设备是一个8核cpu,下图为通过vmstat监控得到的数据,请分析当前系统是否存在瓶颈,以及下一步定位问题的方法.



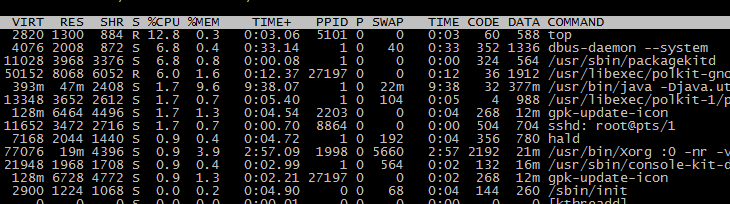
### 假定被测设备是一个8核cpu,下图为通过vmstat监控得到的数据,请分析当前系统是否存在瓶颈,以及下一步定位问题的方法.



### 假定被测设备是一个8核cpu,下图为通过vmstat监控得到的数据,请分析当前系统是否存在瓶颈,以及下一步定位问题的方法.



### 如下图所示,请填写下表通过top命令获取进程内存的这些值的含义:



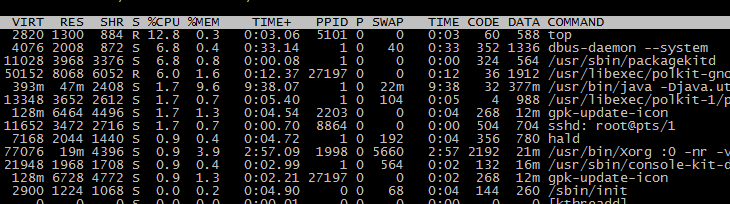
|  |  |
| --- | --- |
| VIRT | VIRT 进程使用的虚拟内存总量，单位kb。VIRT=SWAP+RES |
| SWAP | **进程使用的虚拟内存中，被换出的大小，单位kb** |
| RES | **进程使用的、未被换出的物理内存大小，单位kb。RES=CODE+DATA** |
| CODE | **可执行代码占用的物理内存大小，单位kb** |
| DATA | **可执行代码以外的部分(数据段+栈)占用的物理内存大小，单位kb** |
| SHR | **共享内存大小，单位kb** |

答案：

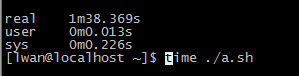
|  |
| --- |
| VIRT |
| SWAP |
| RES |
| CODE |
| DATA |
| SHR |

### 请描述下图top数据中进程”hald”对应的TIME的含义是？

**TIME进程使用的CPU时间总计，单位秒**



### 如下图所示, 通过time命令可以统计进程从开始和结束所占用的时间,请分析者三个时间分别表示的含义



## 实际使用时间（real time）

## 用户态使用时间（the process spent in user mode）

## 内核态使用时间（the process spent in kernel mode）

### 上图三个时间,real一定大于其他两个时间吗

是的

## 常见误区

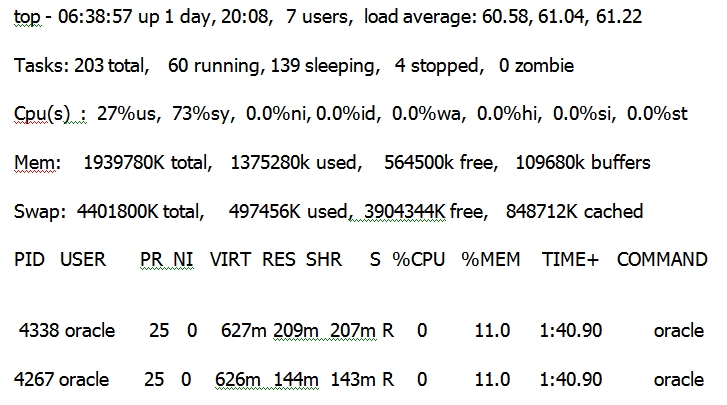
* realtime = user + sys
* realtime > user + sys  
  由于有一些程序需要等待I/O等导致realtime与user+sys不相等  
  当在多核CPU的情况下第二种情况不成立

1. **top命令能监控到所有进程使用的资源信息吗？**

不会

### 当我们使用top命令看到某个进程的CPU过高时,下一步的分析步骤是？

### 下图为一测试人员通过监控数据得出的结论，请判断每个结论是否正确，错误的给出理由以及正确的结论分析。

****

load average值过高说明当前有大量在进入等待序列的线程，CPU出现了阻塞。

答案：

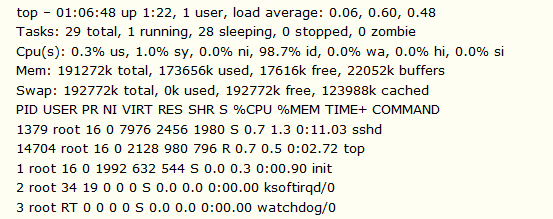
### 89. 上图可以看出1分钟、5分钟、15分钟的cpu利用率几乎无差距，而当前的cpu只达到27%，说明系统CPU比较平稳。

答案：

### 从上图可以看出虽然swap已经使用，但是使用较少，且cached和buffers也使用了比较多的内存，说明内存正常

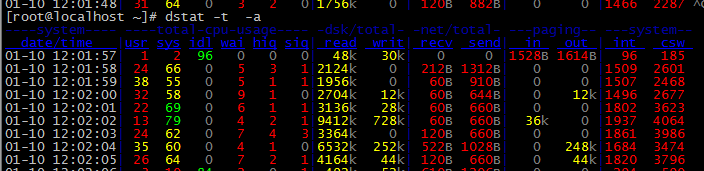
答案：

### 下图为一测试人员通过监控数据得出的结论，请判断每个结论是否正确，错误的给出理由以及正确的结论分析。

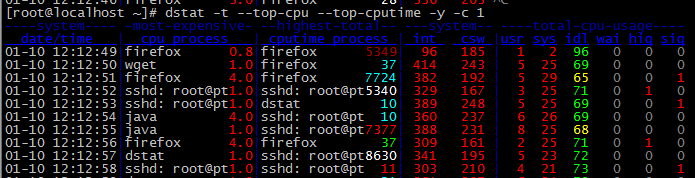
**上图可以看出1分钟和5分钟、15分钟相差较大，且5分钟的数值较高，说明系统是处在由高负载到无负载的状态过度。而有当前的cpu和1分钟的比例来看，高峰时期的cpu预估小于80%，或者大于80%的时间较短**

错误

### 下图是一段时间内机器的性能数据,请问有问题吗？机器可能存在的瓶颈是？



1. cpu：hiq、siq分别为硬中断和软中断次数。
2. system：int、csw分别为系统的中断次数（interrupt）和上下文切换（context switch）。



### 什么是主缺页和次缺页？通过什么命令我们可以查询到相关的缺页信息？在什么情况下我们认为缺页意味系统存在瓶颈？

sar -B 3 10

输出项说明：

* pgpgin/s：表示每秒从磁盘或SWAP置换到内存的字节数(KB)
* pgpgout/s：表示每秒从内存置换到磁盘或SWAP的字节数(KB)
* fault/s：每秒钟系统产生的缺页数,即主缺页与次缺页之和(major + minor)
* majflt/s：每秒钟产生的主缺页数.
* pgfree/s：每秒被放入空闲队列中的页个数
* pgscank/s：每秒被kswapd扫描的页个数
* pgscand/s：每秒直接被扫描的页个数
* pgsteal/s：每秒钟从cache中被清除来满足内存需要的页个数
* %vmeff：每秒清除的页(pgsteal)占总扫描页(pgscank+pgscand)的百分比

## 缺页中断

可分为下面两种

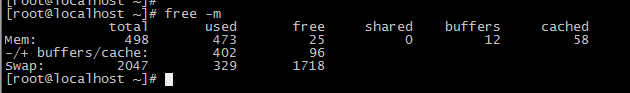
* 主缺页中断（Major Page Fault）  
  要从磁盘读取数据而产生的中断是主缺页中断
* 次缺页中断（Minor Page Fault）  
  数据已经被读入内存并被缓存起来，从内存缓存区中而不是直接从硬盘中读取数据而产生的中断是次缺页中断。

### 在内存调度过程中,可能会分别会产生内部碎片和外部碎片.请解释这两个碎片的含义是？ 如何有效降低内存碎片的产生？

内部碎片的产生：因为所有的内存分配必须起始于可被 4、8 或 16 整除（视处理器体系结构而定）的地址或者因为MMU的分页机制的限制，决定内存分配算法仅能把预定大小的内存块分配给客户。假设当某个客户请求一个43字节的内存块时，因为没有适合大小的内存，所以它可能会获得 44字节、48字节等稍大一点的字节，因此由所需大小四舍五入而产生的多余空间就叫内部碎片。

外部碎片的产生：频繁的分配与回收物理页面会导致大量的、连续且小的页面块夹杂在已分配的页面中间，就会产生外部碎片。假设有一块一共有100个单位的连续空闲内存空间，范围是0~99。如果你从中申请一块内存，如10个单位，那么申请出来的内存块就为0~9区间。这时候你继续申请一块内存，比如说5个单位大，第二块得到的内存块就应该为10~14区间。如果你把第一块内存块释放，然后再申请一块大于10个单位的内存块，比如说20个单位。因为刚被释放的内存块不能满足新的请求，所以只能从15开始分配出20个单位的内存块。现在整个内存空间的状态是0~9空闲，10~14被占用，15~24被占用，25~99空闲。其中0~9就是一个内存碎片了。如果10~14一直被占用，而以后申请的空间都大于10个单位，那么0~9就永远用不上了，变成外部碎片。

### 下图为使用free命令获取的系统信息,通过这个我们能得出的内存公式有？



Total =

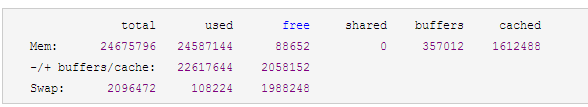
Used=

Free=

下面先解释一下输出的内容：  
**Mem** 行(第二行)是内存的使用情况。  
**Swap** 行(第三行)是交换空间的使用情况。  
**total** 列显示系统总的可用物理内存和交换空间大小。  
**used** 列显示已经被使用的物理内存和交换空间。  
**free** 列显示还有多少物理内存和交换空间可用使用。  
**shared** 列显示被共享使用的物理内存大小。  
**buff/cache** 列显示被 buffer 和 cache 使用的物理内存大小。  
**available** 列显示还可以被应用程序使用的物理内存大小。

<https://www.cnblogs.com/ultranms/p/9254160.html>

### 如下图所示,free内存已经很小了,在不重启环境的条件下，如何让系统释放更多的内存？



答案在此<https://blog.csdn.net/Gavinmiaoc/article/details/80527717>

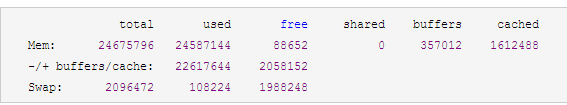
echo 3 > /proc/sys/vm/drop\_caches

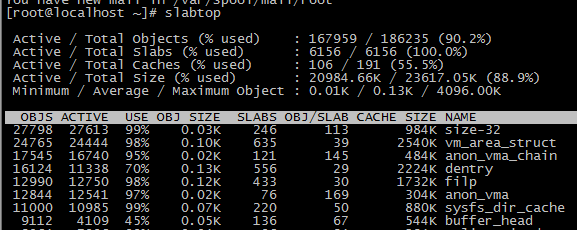
drop\_caches的值可以是0-3之间的数字，代表不同的含义：  
0：不释放（系统默认值）  
1：释放页缓存  
2：释放dentries和inodes

3：释放所有缓存

### 如下图所示,物理内存有24G, 但top却得出全部进程所使用的内存只有18G,剩下的内存我们该如何找回？如何进行这部分内存的定位？请列出你的思路。

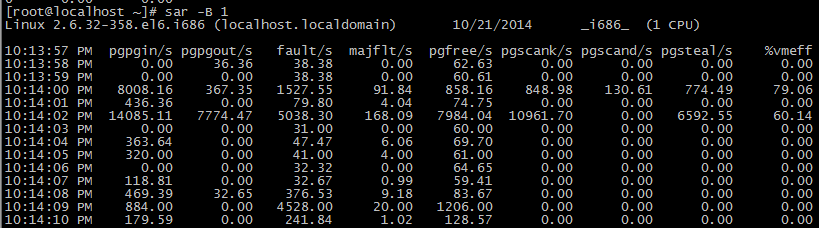
可能的答案在此<https://blog.csdn.net/Gavinmiaoc/article/details/80527717>



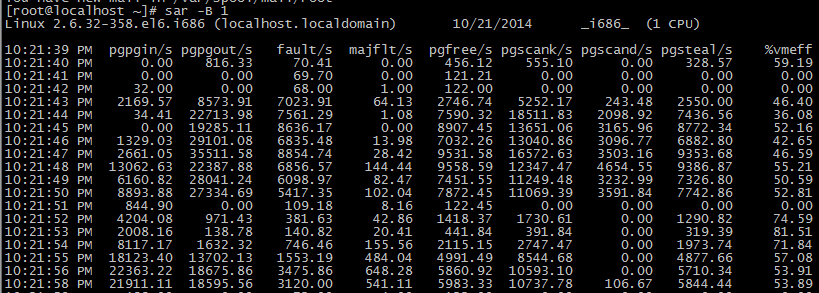


### 通过top命令我们可以查询到系统内存的使用情况,当我们内存不足时,应该如何排查问题？

### 看图回答系统是否存在瓶颈以及导致的原因



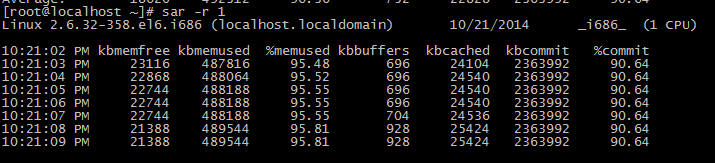
### 看图回答系统是否存在瓶颈以及导致的原因



<https://www.jianshu.com/p/ea7ed85918ac>

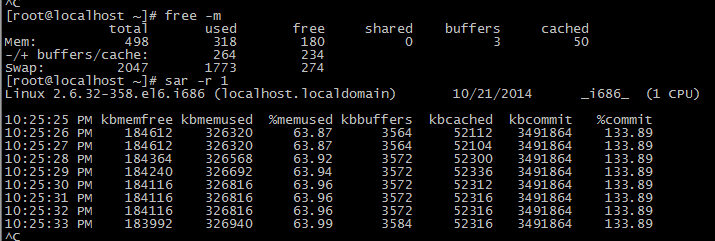
http://passport.ouchn.cn/.well-known/openid-configuration

### 如下图所示,显示了系统的commit内存信息,推算实际使用的物理内存计数



%commit：这个值是kbcommit与内存总量(包括swap)的一个百分比.

### 如下图所示, 下图系统存在瓶颈吗？当前%commit的数据正常吗？



输出项说明：

kbmemfree：这个值和free命令中的free值基本一致,所以它不包括buffer和cache的空间.

kbmemused：这个值和free命令中的used值基本一致,所以它包括buffer和cache的空间.

%memused：这个值是kbmemused和内存总量(不包括swap)的一个百分比.

kbbuffers和kbcached：这两个值就是free命令中的buffer和cache.

kbcommit：保证当前系统所需要的内存,即为了确保不溢出而需要的内存(RAM+swap).

%commit：这个值是kbcommit与内存总量(包括swap)的一个百分比.

原文链接：https://blog.csdn.net/mig\_davidli/java/article/details/52149993