

一个后台开发工程师必须掌握的那些Linux命令——性能监测篇

原创 陈同学在搬砖 陈同学在搬砖

2020-02-25
22:54

本篇文章是Linux命令系列的第一篇文章

又挖了一个新坑~

本系列将按照上面五大模块对Linux命令讲解

让你知道一个后台开发工程师

除了cd ls rm这些基本的操作

真正到生产环境下，还需要掌握哪些linux命令

建立了一个仓库

将本系列所涉及的命令都放到github仓库里了

会保证一直更新

欢迎大家多提issue 多多交流

(顺手也可以给个star，哈哈~)

点击文末 " 阅读原文 "

获取仓库地址

性能监测相关命令

mpstat

功能:

显示CPU的状态信息
这些信息存放在`/proc/stat`文件中。

在多CPUs系统里，

其不但能查看所有CPU的平均状况信息，

而且能够查看特定CPU的信息。

输入语法:

`mpstat` (选项) (参数)

选项

`-A` : 此选项等效于 `# mpstat -I ALL -u -P ALL`

`-I {SUM | CPU | ALL}` : 报告中断统计信息。 使用SUM关键字, `mpstat`命令报告每个处理器的中断总数。使用CPU关键字, 显示CPU

`-P {cpu [, ...] | ON | ALL}` : 指示要报告统计信息的处理器编号。cpu是处理器号。注意, 处理器0是第一个处理器。ON关键字

参数

间隔时间: 每次报告的间隔时间 (秒) ;

次数: 显示报告的次数。

输出信息:

user 在internal时间段里, 用户态的CPU时间 (%) , 不包含 nice值为负 进程 (usr/total)*100
nice 在internal时间段里, nice值为负进程的CPU时间 (%) (nice/total)*100
system 在internal时间段里, 内核态的CPU时间 (%) (system/total)*100
iowait 在internal时间段里, 硬盘IO等待时间 (%) (iowait/total)*100
irq 在internal时间段里, 硬中断时间 (%) (irq/total)*100
soft 在internal时间段里, 软中断时间 (%) (softirq/total)*100
idle 在internal时间段里, CPU除去等待磁盘IO操作外的因为任何原因而空闲的时间闲置时间 (%) (idle/total)*100

实例:

mpstat

显示开机到现在以来cpu的平均状态信息

```
Linux 4.15.0-88-generic (kyle)      2020年02月25日      _x86_64_      (4 CPU)

20时12分09秒  CPU      %usr      %nice      %sys %iowait      %irq      %soft      %steal      %guest      %gnice      %idle
20时12分09秒  all      27.16      0.55      9.85      0.34      0.00      0.43      0.00      0.00      0.00      61.66
```

mpstat -P ALL 2 3

每隔2秒显示一次 所有cpu的状态信息

一共产生3个间隔的信息

最后给出这3次间隔的平均信息

```
kylechen@kyle:~$ mpstat -P ALL 2 3
Linux 4.15.0-88-generic (kyle)      2020年02月25日      _x86_64_      (4 CPU)

20时12分43秒 CPU      %usr    %nice    %sys %iowait    %irq    %soft  %steal  %guest  %gnice   %idle
20时12分45秒 all       9.96    0.00    3.57    0.00    0.00    0.12    0.00    0.00    0.00    86.35
20时12分45秒  0        8.96    0.00    2.99    0.00    0.00    0.00    0.00    0.00    0.00    88.06
20时12分45秒  1        8.87    0.00    3.45    0.49    0.00    0.49    0.00    0.00    0.00    86.70
20时12分45秒  2        9.76    0.00    2.44    0.00    0.00    0.00    0.00    0.00    0.00    87.80
20时12分45秒  3       12.32    0.00    4.43    0.00    0.00    0.00    0.00    0.00    0.00    83.25

20时12分45秒 CPU      %usr    %nice    %sys %iowait    %irq    %soft  %steal  %guest  %gnice   %idle
20时12分47秒 all       13.43    0.00    4.68    0.00    0.00    0.12    0.00    0.00    0.00    81.77
20时12分47秒  0       16.59    0.00    5.69    0.00    0.00    0.47    0.00    0.00    0.00    77.25
20时12分47秒  1       12.08    0.00    3.86    0.00    0.00    0.00    0.00    0.00    0.00    84.06
20时12分47秒  2       12.80    0.00    4.74    0.00    0.00    0.00    0.00    0.00    0.00    82.46
20时12分47秒  3       12.44    0.00    5.26    0.00    0.00    0.00    0.00    0.00    0.00    82.30

20时12分47秒 CPU      %usr    %nice    %sys %iowait    %irq    %soft  %steal  %guest  %gnice   %idle
20时12分49秒 all       8.44    0.00    2.02    0.00    0.00    0.13    0.00    0.00    0.00    89.42
20时12分49秒  0        7.69    0.00    1.54    0.00    0.00    0.51    0.00    0.00    0.00    90.26
20时12分49秒  1        6.47    0.00    2.99    0.00    0.00    0.50    0.00    0.00    0.00    90.05
20时12分49秒  2       10.66    0.00    1.02    0.00    0.00    0.00    0.00    0.00    0.00    88.32
20时12分49秒  3        8.46    0.00    2.49    0.00    0.00    0.00    0.00    0.00    0.00    89.05

平均时间:  CPU      %usr    %nice    %sys %iowait    %irq    %soft  %steal  %guest  %gnice   %idle
平均时间:  all       10.65    0.00    3.44    0.00    0.00    0.12    0.00    0.00    0.00    85.78
平均时间:   0       11.20    0.00    3.46    0.00    0.00    0.33    0.00    0.00    0.00    85.01
平均时间:   1        9.17    0.00    3.44    0.16    0.00    0.33    0.00    0.00    0.00    86.91
平均时间:   2       11.09    0.00    2.77    0.00    0.00    0.00    0.00    0.00    0.00    86.13
平均时间:   3       11.09    0.00    4.08    0.00    0.00    0.00    0.00    0.00    0.00    84.83
```

mpstat -P ALL -I SUM

查看cpu中断的统计(各个cpu分开显示)

```
linux 4.15.0-88-generic (kyle)      2020年02月25日      _x86_64_      (4 CPU)

20时13分24秒 CPU      intr/s
20时13分24秒 all       2057.27
20时13分24秒  0        473.14
20时13分24秒  1        591.17
20时13分24秒  2        687.38
20时13分24秒  3        493.70
```

mpstat -I SUM

查看cpu中断的统计(各个cpu合并显示)

```
kylechen@kyle:~$ mpstat -I SUM
Linux 4.15.0-88-generic (kyle)      2020年02月25日      _x86_64_      (4 CPU)

20时13分47秒 CPU      intr/s
20时13分47秒 all       2036.32
```

free

功能

free指令会显示内存的使用情况，
包括实体内存，

虚拟的交换文件内存，
共享内存区段，
以及系统核心使用的缓冲区等。

输入语法

```
free [-bkmotV][ -s <间隔秒数>]
参数说明：

-b    以Byte为单位显示内存使用情况。
-k    以KB为单位显示内存使用情况。
-m    以MB为单位显示内存使用情况。
-h    以合适的单位显示内存使用情况，最大为三位数，自动计算对应的单位值。单位如下：

B = bytes
K = kilos
M = megas
G = gigas
T = teras
-t    显示内存总和列。
-V    显示版本信息。
-o    不显示缓冲区调节列。
-s<间隔秒数>持续观察内存使用状况。
```

输出信息

```
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。(进程间的共享内存)
buff/cache 列显示被 磁盘缓存 使用的物理内存大小。(buffer表示写缓冲 cache表示读缓冲)
available 列显示还可以被应用程序使用的物理内存大小

注意点
1.当应用程序需要内存时，如果没有足够的 free 内存可以用，内核就会从 buffer 和 cache 中回收内存来满足应用程序的请求。所以从应用程序的角度来说，available = free + buffer + cache。
    请注意，这只是一个很理想的计算方式，实际中的数据往往有较大的误差。

2.swap space 是磁盘上的一块区域，可以是一个分区，也可以是一个文件。
    所以具体的实现可以是 swap 分区也可以是 swap 文件。
    当系统物理内存吃紧时，Linux 会将内存中不常访问的数据保存到 swap 上，
    这样系统就有更多的物理内存为各个进程服务，而当系统需要访问 swap 上存储的内容时，
    再将 swap 上的数据加载到内存中，这就是常说的换出和换入。
    交换空间可以在一定程度上缓解内存不足的情况，但是它需要读写磁盘数据，所以性能不是很高。
```

实例

```
free 不指定单位显示内存使用情况

      总计      已用      空闲      共享      缓冲/缓存      可用
内存:   3920116   2680416   179976   596300   1059724      408108
交换:    999420    519916   479504

free -h以合适的单位显示内存使用情况
kylechen@kyle:~$ free -h
      总计      已用      空闲      共享      缓冲/缓存      可用
内存:     3.7G     2.6G     178M     573M     1.0G     401M
交换:     975M     507M     468M
```

free -th以合适的单位显示内存使用情况,同时显示总量列

kylechen@kyle:~\$ free -th

	总计	已用	空闲	共享	缓冲/缓存	可用
内存:	3.7G	2.5G	200M	570M	1.0G	424M
交换:	975M	507M	468M			
总量:	4.7G	3.0G	668M			

free -h -s 1 以1秒为间隔显示内存使用情况

kylechen@kyle:~\$ free -h -s 1

	总计	已用	空闲	共享	缓冲/缓存	可用
内存:	3.7G	2.5G	198M	569M	1.0G	421M
交换:	975M	507M	468M			

	总计	已用	空闲	共享	缓冲/缓存	可用
内存:	3.7G	2.5G	196M	571M	1.0G	419M
交换:	975M	507M	468M			

	总计	已用	空闲	共享	缓冲/缓存	可用
内存:	3.7G	2.5G	195M	571M	1.0G	419M
交换:	975M	507M	468M			

	总计	已用	空闲	共享	缓冲/缓存	可用
内存:	3.7G	2.5G	196M	571M	1.0G	420M
交换:	975M	507M	468M			

	总计	已用	空闲	共享	缓冲/缓存	可用
内存:	3.7G	2.5G	199M	568M	1.0G	423M
交换:	975M	507M	468M			

top

功能

top是系统管理员最重要的工具之一。被广泛用于监视服务器的负载。
它可以动态的查看系统当前正在运行的进程情况，内存使用情况，cpu使用情况
也就是说上面mpstate 和free能实现的功能 它都内在的包含了

输入语法

很简单 直接输入top

输出信息

```
任务: 298 total, 1 running, 246 sleeping, 0 stopped, 0 zombie
%Cpu(s): 5.7 us, 1.6 sy, 0.0 ni, 92.6 id, 0.1 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 3920116 total, 301948 free, 2514912 used, 1103256 buff/cache
KiB Swap: 999420 total, 480272 free, 519148 used. 618380 avail Mem
```

进程id	USER	PR	NI	VIRT	RES	SHR		%CPU	%MEM	TIME+	COMMAND
1590	kylechen	20	0	3974220	265424	72628	S	7.3	6.8	5:33.19	gnome-shell
1446	kylechen	20	0	521960	38412	19292	S	6.3	1.0	3:28.27	Xorg
3008	kylechen	20	0	780520	50972	32452	S	4.6	1.3	0:17.32	gnome-term+
2779	kylechen	20	0	1201196	318344	79024	S	1.3	8.1	11:48.81	chrome
1955	kylechen	20	0	1505112	399200	243564	S	1.0	10.2	5:32.38	chrome
1	root	20	0	225908	4064	1584	S	0.7	0.1	0:08.15	systemd
1036	gdm	20	0	3492528	89676	62056	S	0.7	2.3	0:11.70	gnome-shell
1993	kylechen	20	0	556488	55780	24440	S	0.7	1.4	0:34.27	chrome
8	root	20	0	0	0	0	I	0.3	0.0	0:08.76	rcu_sched
983	mysql	20	0	1358648	0	0	S	0.3	0.0	0:03.85	mysqld
1571	kylechen	20	0	220792	156	0	S	0.3	0.0	0:03.21	at-spi2-re+
2192	kylechen	20	0	486420	5828	0	S	0.3	0.1	0:01.88	sogou-qimp+
5330	kylechen	20	0	2621992	450080	80948	S	0.3	11.5	1:13.24	XMind
5708	root	20	0	0	0	0	I	0.3	0.0	0:01.65	kworker/u8+
6974	kylechen	20	0	880400	185648	107616	S	0.3	4.7	0:45.80	chrome
7245	root	20	0	0	0	0	I	0.3	0.0	0:00.22	kworker/2:0
7760	kylechen	20	0	51360	4004	3332	R	0.3	0.1	0:00.19	top

输出信息如上所示 一大坨 很繁琐 咱们一行一行来分析

第一行表示 当前的进程状态 总共有298个进程 1个处理runing 246个处于sleeping 0个处于stopped状态

```
任务: 298 total, 1 running, 246 sleeping, 0 stopped
```

第二行表示 当前的cpu状态 这里显示的状态参数其实和mpstat那里说的是一样的 具体可以看上面的mpstat命令讲解

```
%Cpu(s): 5.7 us, 1.6 sy, 0.0 ni, 92.6 id, 0.1 wa, 0.0 hi,
```

**第三四行表示 **当前的内存状态 这里显示的状态参数其实和free那里说的是一样的 具体可以看上面的free命令讲解

```
KiB Mem : 3920116 total, 301948 free, 2514912 used, 1103256 buff/cache
KiB Swap: 999420 total, 480272 free, 519148 used. 618380.1 wa, 0.0 hi,
```

后面几行其实 表示的就是每个进程具体占用的系统资源 会随着时间动态变化

PID

进程ID, 进程的唯一标识符

USER

进程所有者的实际用户名。

PR

进程的调度优先级。这个字段的一些值是'rt'。这意味这这些进程运行在实时态。

NI

进程的nice值（优先级）。越小的值意味着越高的优先级。

VIRT

进程使用的虚拟内存。

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。

SHR

SHR是进程使用的共享内存。

S

这个是进程的状态。它有以下不同的值:

D – 不可中断的睡眠态。

R – 运行态

S – 睡眠态

T – 被跟踪或已停止

Z – 僵尸态

I - 空闲状态 (idle)

%CPU

自从上一次更新时到现在任务所使用的CPU时间百分比。

%MEM

进程使用的可用物理内存百分比。


TIME+

任务启动后到现在所使用的全部CPU时间，精确到百分之一秒。

COMMAND

运行进程所使用的命令。

还有许多在默认情况下不会显示的输出，它们可以显示进程的页错误、有效组和组ID和其他更多的信息。

进程id	USER	PR	NI	VIRT	RES	SHR		%CPU	%MEM	TIME+	COMMAND
1590	kylechen	20	0	3974220	265424	72628	S	7.3	6.8	5:33.19	gnome-shell
1446	kylechen	20	0	521960	38412	19292	S	6.3	1.0	3:28.27	Xorg
3008	kylechen	20	0	780520	50972	32452	S	4.6	1.3	0:17.32	gnome-term+
2779	kylechen	20	0	1201196	318344	79024	S	1.3	8.1	11:48.81	chrome
.....											

实例

实例1:top

```
top - 21:25:55 up 1:27, 1 user, load average: 1.51, 1.87, 1.49
任务: 305 total, 2 running, 250 sleeping, 0 stopped, 0 zombie
%Cpu(s): 22.7 us, 1.8 sy, 0.0 ni, 75.1 id, 0.1 wa, 0.0 hi, 0.3 si, 0.0 st
KiB Mem : 3920116 total, 121880 free, 2859016 used, 939220 buff/cache
KiB Swap: 999420 total, 333328 free, 666092 used. 182088 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
2779 kylechen 20 0 1191.7m 314.5m 59.2m S 17.1 8.2 20:45.14 chrome
1590 kylechen 20 0 3881.1m 242.1m 70.3m R 4.8 6.3 7:19.67 gnome-shell
1446 kylechen 20 0 444.6m 31.1m 11.8m S 1.3 0.8 4:42.48 Xorg
8711 kylechen 20 0 622.0m 17.4m 5.6m S 0.7 0.5 0:02.43 gnome-termi+
1955 kylechen 20 0 1458.6m 348.2m 189.5m S 0.5 9.1 7:31.10 chrome
1989 kylechen 20 0 653.0m 112.1m 47.2m S 0.5 2.9 7:06.33 chrome
8736 kylechen 20 0 50.2m 1.5m 0.8m R 0.2 0.0 0:05.51 top
1036 gdm 20 0 3410.7m 82.2m 55.3m S 0.1 2.1 0:15.10 gnome-shell
1484 kylechen 20 0 442.8m 48.0m 8.9m S 0.1 1.3 0:40.19 fcitx
1993 kylechen 20 0 551.4m 48.9m 17.8m S 0.1 1.3 0:45.63 chrome
3299 kylechen 20 0 633.0m 20.9m 5.1m S 0.1 0.5 0:17.90 chrome
1 root 20 0 220.6m 3.0m 0.6m S 0.0 0.1 0:10.02 systemd
2 root 20 0 0.0m 0.0m 0.0m S 0.0 0.0 0:00.00 kthreadd
4 root 0 -20 0.0m 0.0m 0.0m I 0.0 0.0 0:00.00 kworker/0:0H
6 root 0 -20 0.0m 0.0m 0.0m I 0.0 0.0 0:00.00 mm_percpu_wq
7 root 20 0 0.0m 0.0m 0.0m S 0.0 0.0 0:00.22 ksoftirqd/0
8 root 20 0 0.0m 0.0m 0.0m I 0.0 0.0 0:11.86 rcu_sched
```

实例2：交互式命令

top是一个交互式的命令
所以用top调出动态显示的进程状态以后
在界面上继续用键盘输入指令
会继续在界面上执行对应的操作

交互命令1：回车/空格

top命令默认在一个特定间隔(3秒)后刷新显示。
要手动刷新，用户可以输入回车或者空格。

交互命令2：B

一些重要信息会以加粗字体显示。
这个命令可以切换粗体显示。

交互命令3：d 或s

当按下'd'或's'时，你
将被提示输入一个值（以秒为单位），
它会以设置的值作为刷新间隔。
如果你这里输入了1，top将会每秒刷新。

交互命令4：'R'

切换反向/常规排序。

交互命令5：'V'

切换树视图。

交互命令6：'k'

top命令中最重要的一个命令之一。
用于发送信号给任务（通常是结束任务）。

交互命令7：'e'

切换显示的单位
依次以k ->m->g->t->p单位选择

我是陈同学
让技术
有温度

你的支持是我搬砖的动力

▼
往期精彩回顾
▼

[你的微信消息是怎么发出去的？](#)

[一个小时学会Git](#)

[Leetcode面试高频题汇总--链表](#)

[Leetcode面试高频题汇总--数组](#)

[【设计模式】可能是东半球最透彻的单例模式讲解](#)

点击 " 阅读原文 " 获取仓库地址

听说点击在看的今年都会暴富脱单，升职加薪