AIXCHINA

CPU 和内存性能监测工具

sar&vmstat

www.aixchina.com

CPU和内存性能监测工具
sar&vmstat

本文详细介绍 CPU 和内存的性能监测工具，通过参阅随附的例子，可以帮助大家学习如何从这些工具的输出中提取有用的信息。

1 sar 命令

sar（系统活动报告）命令收集数据的方式有两种：一种是实时的观测系统信息；另一种是读取先前捕获的数据。

sar 命令往往是管理员首先执行的监测命令，虽然从它的输出中可以获得绝大多数的系统信息，不过针对某些局部的信息，还需要使用专门的工具以获得更精确的报告。

1.1 sar 命令使用实例

不带任何参数执行 sar 命令，将输出 sa1 命令采集的当天数据，crontab 文件中包含采集时间间隔的设置，下面例子采用/var/spool/cron/crontabs/adm 文件中的缺省设置：

```
# sar
08:00:00    %usr    %sys    %wio    %idle
08:20:00    0       0       0       100
08:40:00    0       0       0       100
09:00:00    0       0       0       100
Average     0       0       0       100
```

只带时间间隔和显示记录数执行 sar 命令的输出见下面的例子。这与执行 sar –u 1 10 命令的输出是一样的，1 代表以秒为单位的时间间隔，10 表示捕获多少条数据项。

```
# sar 1 10
AIX server2 3 4 000FA17D4C00 06/30/00
09:14:57 %usr    %sys    %wio    %idle
09:14:58 54      18      28      0
09:14:59 40      20      40      0
09:15:00 44      19      38      0
09:15:01 82      14      4       0
09:15:02 66      16      18      0
09:15:03 45      12      43      0
09:15:04 60      17      23      0
09:15:05 47      16      37      0
09:15:06 65      12      23      0
09:15:07 48      8       44      0
Average 55       15      30      0
```

sar –a 命令反映了文件访问系统调用的使用情况，报告中包含几类文件访问调用的每秒执行次数。

```
# sar -a 1 10
```

AIX server2 3 4 000FA17D4C00 06/30/00

09:28:44	*iget/s*	*lookuppn/s*	*dirblk/s*
09:28:45	*0*	*1169*	*277*
09:28:46	*0*	*15*	*0*
09:28:47	*0*	*50*	*0*
09:28:48	*0*	*559*	*19*
09:28:49	*0*	*390*	*20*
09:28:50	*0*	*1467*	*137*
09:28:51	*0*	*1775*	*153*
09:28:52	*0*	*2303*	*74*
09:28:53	*0*	*2832*	*50*
09:28:54	*0*	*883*	*44*
Average	*0*	*1144*	*77*

sar –c 命令统计系统调用情况。

 # sar -c 1 10

AIX server2 3 4 000FA17D4C00 06/30/00

09:33:04	*scall/s*	*sread/s*	*swrit/s*	*fork/s*	*exec/s*	*rchar/s*	*wchar/s*
09:33:05	*1050*	*279*	*118*	*0.00*	*0.00*	*911220*	*5376749*
09:33:06	*186*	*19*	*74*	*0.00*	*0.00*	*3272*	*3226417*
09:33:07	*221*	*19*	*79*	*0.00*	*0.00*	*3272*	*3277806*
09:33:08	*2996*	*132*	*400*	*0.00*	*0.00*	*314800*	*2284933*
09:33:09	*3304*	*237*	*294*	*0.00*	*0.00*	*167733*	*848174*
09:33:10	*4186*	*282*	*391*	*0.00*	*0.00*	*228196*	*509414*
09:33:11	*1938*	*109*	*182*	*1.00*	*1.00*	*153703*	*1297872*
09:33:12	*3263*	*179*	*303*	*0.00*	*0.00*	*242048*	*1003364*
09:33:13	*2751*	*172*	*258*	*0.00*	*0.00*	*155082*	*693801*
09:33:14	*2827*	*187*	*285*	*0.00*	*0.00*	*174059*	*1155239*
Average	*2273*	*162*	*238*	*0.10*	*0.10*	*235271*	*1966259*

sar –d 命令给出硬盘读写活动以及平均传输块大小的统计值。该参数属于内部参数，AIX 的文档中没有说明，而是用 iostat 命令代替之。

 # sar -d 5 3

AIX server2 3 4 000FA17D4C00 06/30/00

10:08:19	*device*	*%busy*	*avque*	*r+w/s*	*blks/s*	*avwait*	*avserv*
10:08:24	*hdisk0*	*0*	*0.0*	*0*	*4*	*0.0*	*0.0*
	hdisk1	*0*	*0.0*	*0*	*3*	*0.0*	*0.0*
	cd0	*0*	*0.0*	*0*	*0*	*0.0*	*0.0*
10:08:29	*hdisk0*	*44*	*1.0*	*366*	*3569*	*0.0*	*0.0*
	hdisk1	*36*	*0.0*	*47*	*2368*	*0.0*	*0.0*
	cd0	*0*	*0.0*	*0*	*0*	*0.0*	*0.0*
10:08:34	*hdisk0*	*84*	*2.0*	*250*	*1752*	*0.0*	*0.0*
	hdisk1	*16*	*1.0*	*19*	*950*	*0.0*	*0.0*
	cd0	*0*	*0.0*	*0*	*0*	*0.0*	*0.0*
Average	*hdisk0*	*42*	*1.0*	*205*	*1775*	*0.0*	*0.0*
	hdisk1	*17*	*0.3*	*22*	*1107*	*0.0*	*0.0*

cd0 0 0.0 0 0 0.0 0.0

sar –q 命令报告运行队列的统计情况。

 # sar -q 1 10

AIX server2 3 4 000FA17D4C00 06/30/00

11:08:33 runq-sz %runocc swpq-sz %swpocc

11:08:34 1.0 100

11:08:35 1.0 100

11:08:36 1.0 100

11:08:37 1.0 100

11:08:38 1.0 100 1.0 100

11:08:39 1.0 100 1.0 100

11:08:40 1.0 100 1.0 100

11:08:41 1.0 100

11:08:42 1.0 100

11:08:43 1.0 100

Average 1.0 50 1.0 80

sar –r 命令统计调页数据。

 # sar -r 1 10

AIX server2 3 4 000FA17D4C00 06/30/00

11:16:11	*slots*	*cycle/s*	*fault/s*	*odio/s*
11:16:12	*130767*	*0.00*	*472.82*	*66.02*
11:16:13	*130767*	*0.00*	*989.00*	*800.00*
11:16:14	*130767*	*0.00*	*44.00*	*1052.00*
11:16:15	*130767*	*0.00*	*43.00*	*1040.00*
11:16:16	*130767*	*0.00*	*47.00*	*1080.00*
11:16:17	*130767*	*0.00*	*43.00*	*808.00*
11:16:18	*130767*	*0.00*	*40.00*	*860.00*
11:16:19	*130767*	*0.00*	*46.00*	*836.00*
11:16:20	*130767*	*0.00*	*47.00*	*852.00*
11:16:21	*130767*	*0.00*	*48.00*	*836.00*
Average	*130767*	*0*	*183*	*821*

sar –v 命令统计进程、核心线程、I 节点和文件表的状态信息。

 # sar -v 1 5

AIX server2 3 4 000FA17D4C00 06/30/00

11:12:39	*proc-sz*	*inod-sz*	*file-sz*	*thrd-sz*
11:12:40	*49/262144*	*229/42942*	*315/511*	*59/524288*
11:12:41	*46/262144*	*221/42942*	*303/511*	*56/524288*
11:12:42	*45/262144*	*220/42942*	*301/511*	*55/524288*
11:12:43	*45/262144*	*220/42942*	*301/511*	*55/524288*
11:12:44	*45/262144*	*220/42942*	*301/511*	*55/524288*

sar –y 命令统计每秒的 TTY 设备活动情况

 # sar -y 1 10

AIX server2 3 4 000FA17D4C00 06/30/00

11:48:36	*rawch/s*	*canch/s*	*outch/s*	*rcvin/s*	*xmtin/s*	*mdmin/s*
11:48:37	*0*	*0*	*104*	*63*	*60*	*0*

11:48:38	*0*	*0*	*58*	*9*	*60*	*0*
11:48:39	*0*	*0*	*58*	*69*	*61*	*0*
11:48:40	*0*	*0*	*58*	*68*	*60*	*0*
11:48:41	*0*	*0*	*58*	*69*	*3*	*0*
11:48:42	*0*	*0*	*58*	*68*	*52*	*0*
11:48:43	*0*	*0*	*58*	*69*	*60*	*0*
11:48:44	*0*	*0*	*58*	*25*	*60*	*0*
11:48:45	*0*	*0*	*58*	*42*	*23*	*0*
11:48:46	*0*	*0*	*58*	*68*	*9*	*0*
Average	*0*	*0*	*63*	*55*	*45*	*0*

上面介绍了一部分主要的 SAR 命令使用方法，在使用时还可以几个参数混合执行，以获取需要的分析数据，例如：

 # sar -y -r 1 5

AIX server2 3 4 000FA17D4C00 06/30/00

11:48:56	*rawch/s*	*canch/s*	*outch/s*	*rcvin/s*	*xmtin/s*	*mdmin/s*
	slots	*cycle/s*	*fault/s*	*odio/s*		
11:48:57	*0*	*0*	*147*	*67*	*3*	*0*
	130767	*0.00*	*3.96*	*0.00*		
11:48:58	*0*	*0*	*102*	*69*	*58*	*0*
	130767	*0.00*	*0.00*	*0.00*		
11:48:59	*0*	*0*	*102*	*68*	*60*	*0*
	130767	*0.00*	*0.00*	*0.00*		
11:49:00	*0*	*0*	*102*	*69*	*17*	*0*
	130767	*0.00*	*0.00*	*0.00*		
11:49:01	*0*	*0*	*102*	*68*	*3*	*0*
	130767	*0.00*	*1.00*	*4.00*		
Average	*0*	*0*	*111*	*68*	*28*	*0*
Average	*130767*	*0*	*1*	*1*		

1.2 sar 命令语法说明

sar 命令把操作系统中关于系统活动的计数信息输出到标准输出，它的使用语法如下：

sar [{ -A | [-a] [-b] [-c] [-k] [-m] [-q] [-r] [-u] [-v] [-w] [-y] }] [-P *ProcessorIdentifier*, ... | ALL] [-e*hh* [:*mm* [:*ss*]]] [-f*File*] [-i*Seconds*] [-o*File*] [-s*hh* [:*mm* [:*ss*]]] [*Interval* [*Number*]]

记账系统根据 Interval 和 Number 参数，每隔指定的秒数输出指定次数的统计信息。缺省的取样间隔为 1 秒，如果带-o 参数，收集的数据同时保存到-o 指定的文件中。

如果 CPU 的利用率接近 100%（user+system），那么所测试的系统瓶颈在 CPU 上。如果大部分的时间都花在 I/O 等待上，那么 CPU 中的进程就会因等待磁盘 I/O 而挂起，这些 I/O 可能是频繁的文件访问，也可能是由于内存不足而产生的频繁调页。

注意：

系统等待远程文件访问所耗费的时间没有计算在 I/O 等待时间内。如果 CPU 利用率和 I/O 等待时间都很低，而系统响应时间又很慢，就有必要考察系统花了多少时间在等待远程 I/O 上。由于没有什么高级命令能提供远程 I/O 等待的统计，所以只有使用 TRACE 数据来分析了。

sar 命令通过调用 sadc 命令来访问系统数据，cron 命令定时调用/usr/lib/sa/sa1 和/usr/lib/sa/sa2 两个脚本程序来收集每日的统计报告。/var/spool/cron/crontabs/adm 下的 crontab 文件中包含了运行这两个脚本的例子（但已经注释掉了）。用这种方式收集数据有助于了解系统的使用情况，发现峰值使用时段。

表 1 中包含了 SAR 常用的命令参数：

命令参数	描述
-A	如果不另带-P 参数，该参数相当于同时指定-abckmqruvwy；如果另带-P，该参数相当于同时指定-acmuw。
-a	报告文件访问系统调用的使用情况，给出几类文件访问调用的每秒执行次数。如果同时带上-P 参数，将按每个 CPU 列出使用情况的信息；否则提供全系统的数据，输出信息包括下面内容： dirblk/s：目录搜索函数读入多少个 512 字节块，以找到指定的文件； iget/s：I 节点查询函数的调用次数。iget 函数返回一个指向文件或设备的 I 节点结构指针； lookuppn/s：搜索 v 节点地址的目录搜索函数的调用次数。
-b	报告缓冲区的传输、访问活动，以及每秒的缓存命中率。在 AIX 版本 3 中，大多数的文件访问没有经过核心缓存，所以不增加这部分统计数据。但程序在打开块或字符设备进行 I/O 操作时，仍然使用传统的工作模式（有经过核心缓存），所以能产生有意义的统计数据。输出信息中包含下面内容： bread/s、bwrit/s：报告块 I/O 操作的次数，这些 I/O 往往由核心产生，用来管理块缓冲区中的缓存数据； lread/s、lwrit/s：报告逻辑 I/O 请求的次数。在对块设备进行读写时，实际传输的数据尺寸可能小于一个完整的块。系统是按完整的块与物理设备交换数据，因此会把用户请求的数据放入核心缓冲区（预留出来就是用于这个目的）。这部分缓存由核心管理，所以很多用户的逻辑读写请求直接从缓存中取数据，没有发生实际的 I/O 操作。应用程序对块设备的读写请求统计记录在逻辑读写项内；而核心管理缓存执行的块 I/O 操作统计记录在块读写项内。 pread/s、pwrit/s：报告对裸设备的 I/O 操作次数。对裸设备的 I/O 请求不象对块设备那样经过缓存，而是直接对设备操作。 %rcache、%wcache：报告缓存的效率（即命中率）。这是通过公式：[(100)*(lreads-breads)/(lreads)]计算得出的。
-c	报告系统调用统计情况，如果同时带上-P 参数，显示数据将按每个 cpu 分列，否则，只显示全系统的总值。输出信息中包含下面内容：

	exec/s，fork/s：报告 fork 和 exec 系统调用的执行次数； sread/s，swrit/s：报告读写系统调用的执行次数； rchar/s，wchar/s：报告读写调用传输的字符数； scall/s：报告系统调用的总数。
-e hh[:mm[:ss]]	设置报告的结束时间，缺省的结束时间是 18:00
-f File	从 File 文件中获取数据（该文件可以通过-o File 生成）。缺省的 File 参数指向当前每日收集数据文件（/var/adm/sa/sadd）
-i Seconds	按 Seconds 指定的数值报告数据，否则命令报告数据文件中每秒的数据
-k	报告核心的活动情况。输出中包含下面的内容： kexit/s：报告每秒结束的核心进程数； kproc-ov/s：报告因为到达进程阀值限制而无法创建核心进程的次数； ksched/s：报告每秒分配任务的核心进程的数目
-m	报告每秒的（发送和接受的）消息以及（创建、使用或删除的）信号灯活动。如果带-P 参数，输出按每个处理器分列，否则给出的是全系统的总值。输出中包含下面内容： msg/s：报告 IPC 消息的数目； sema/s：报告 IPC 信号灯的数目
-o File	把采集的数据以二进制格式保存到 File 文件中。每次采集的数据形成但读一条记录，并带有时间标记。
-P <处理器 id〉| ALL	报告指定处理器的统计情况，如果带上 ALL 参数，则报告每个处理器的数据以及全系统的总值，只有-a、-c、-m、-u 和-w 参数才可以与-P 合在一起使用
-q	报告队列统计数据。输出中包含以下内容： runq-sz：报告运行队列中核心线程的平均数量； %runocc：报告运行队列被占用时间的百分比； swpq-sz：报告等待调入的核心线程的平均数量； %swpocc：报告对换队列被占用时间的百分比。
-r	报告调页的统计数据。输出中包含以下内容： cycle/s：报告每秒钟页面更换循环的次数；

	fault/s：报告每秒缺页的数目，这并不意味着会产生 I/O，因为有些缺页不通过 I/O 操作就能解决。 slots：报告调页空间上空闲页面的数量； odio/s：报告每秒非调页操作引起的磁盘 I/O 数量；
-s hh[:mm[:ss]]	设置数据读取的开始时间，sar 命令将从指定的时间开始解析数据，缺省值是 08:00。
-u	报告每个处理器或全系统处理器使用的统计数据。如果带上-P 参数，信息按每个处理器分列；否则给出系统的全值。由于-u 参数反映的是百分比，所以全系统的数据来自每个处理器数据的平均值。输出中包含下面内容： %idle：报告 CPU 空闲的百分比； %sys：报告 CPU 正在处理系统（核心）进程的百分比； %usr：报告 CPU 正在处理用户（应用）进程的百分比； %wio：报告 CPU 正在等待磁盘 I/O 的百分比，对于全系统的统计，如果大多数处理器空闲（比较少见），该值反映出来的信息会有些夸大。
-v	报告进程、核心线程、I 节点和文件表的状态。输出中包含下面内容： file-sz、inod-sz、proc-sz、thrd-sz：报告系统核心中这几类表中的记录项数量。
-w	报告系统切换活动的统计数据。如果带上-P 参数，输出将按每个处理器分列；否则给出的是全系统的全值。输出中包含下面内容： pswch/s：报告每秒钟上下文相关切换发生的次数。
-y	报告 tty 设备活动统计。 canch/s：报告 tty 规范输入队列中的字符数，在 AIX 4 以上的版本该字段总是 0； mdmin/s：报告 tty modem 中断次数； outch/s：报告 tty 输出队列中的字符数； rawch/s：报告 tty 输入队列中的字符数； revin/s：报告 tty 接收中断次数； xmtin/s：报告 tty 发送中断次数。

注意：

- ***根据执行的频度，sar 命令本身会产生可观的读写量，在系统不生产时执行 sar 命令，可以看出 sar 命令对统计数据的贡献。***
- ***如果不带别的参数，sar 命令报告系统的活动统计。***

1.3 sadc 命令

sadc 命令提供一个系统数据收集器。使用的语法为：

sadc [Interval Number] [Outfile]

该命令按指定的时间间隔（Interval）定时对系统数据进行指定次数（Number）的取样，采集的数据已二进制的格式写入到指定的文件（outfile）或标准输出。如果取样时间间隔和次数都没有指定，sadc 将写入一条空记录，用在系统启动时，要来标示计数器（从零）开始的时间。sadc 用作 sar 命令的后台程序。

AIX 包含一系列的计数器，每当发生相关的系统活动就会增加计数值，这些系统活动计数器包括：

- 系统单元利用情况计数器
- 缓冲区使用计数器
- 硬盘和磁带 I/O 活动计数器
- TTY 设备活动计数器
- 切换和子调用计数器
- 文件访问计数器
- 队列活动计数器
- 进程间通讯计数器

1.4 sa1 和 sa2 命令

sa1 命令是 sadc 命令的 SHELL 脚本版本，用来处理 sadc 命令的每个参数设置。sa1 命令采集数据并保存在/var/adm/sa/sa$DATE 文件中，$DATE 为当天号数。

sa1 的使用语法如下：

sa1 [interval Number]

Interval 和 Number 参数指定取样的时间间隔和次数，如果都没有指定，sa1 命令将写下一条空记录（与 sadc 命令一样）。

sa1 通过 cron 命令自动启动，如果 cron 没有每天运行该命令，那么执行 sar 命令时会报告/usr/lib/sa/sa1 的数据文件不存在。

sa2 命令是 sar 命令的 SHELL 脚本版本，用来生成每天的报告，报告放在/var/adm/sa/sar$dd 文件中，$dd 当天的号数。

sa2 通过 cron 命令自动启动，与 sa1 命令一起运行，它的使用语法如下：

sa2

2 vmstat 命令

vmstat 命令能够报告的统计信息包括：核心线程、虚拟内存、磁盘访问、终端调用和 CPU 活动等。这些统计信息包含采样数据的平均或统计值，常常用来作为调整系统负载平衡的依据。

vmstat 命令的语法如下：

vmstat [-f] [-i] [-s] [PhysicalVolume] [Interval [Count]]

不带任何参数执行 vmstat 命令，将输出一条自系统启动以来的统计值，见下面的例子：

```
# vmstat
kthr   memory             page              faults        cpu
----- -----------   ------------------------   ------------   -----------------
r b    avm    fre    re    pi po    fr    sr cy    in    sy    cs      us    sy    id    wa
0 0 15982 1388   0    0    0    8    22 0    113 281 36        1     0    98    1
```

如果带上-f 参数，则报告自系统启动以来 fork 调用次数的统计值，建下面的例子：

```
# vmstat -f
51881 forks
```

PhysicalVolume 参数限定报告中的物理卷，Interval 参数指定取样的时间间隔。输出报告中的第一行仍然是自系统启动以来的统计值，之后的内容包括指定时间间隔之内的系统信息统计数据。Count 参数限定输出记录的笔数（不允许为 0），如果不指定 Count 参数，vmstat 将按照 Interval 指定的时间间隔持续的输出统计信息。见下面的例子：

```
# vmstat 1 5
kthr memory             page                faults           cpu
----- ------------- ------------------------ ---------------- --------------
r b    avm    fre    re pi po fr sr cy        in sy cs         us sy id wa
0 0 15982 1388   0 0 0 8 22 0              113 281 36    1 0 98 1
0 0 15982 1387   0 0 0 0 0 0              108 4194 31   2 3 95 0
0 0 15982 1387   0 0 0 0 0 0              109 286 30    0 0 99 0
0 0 15982 1387   0 0 0 0 0 0              108 285 26    0 0 99 0
0 0 15982 1387   0 0 0 0 0 0              111 286 32    0 0 99 0
```

vmstat 命令通过使用 knlist 调用和访问/dev/kmem 伪设备，得到核心线程、调页和中断活动的统计信息。磁盘输入输出的数据来自磁盘的驱动程序，其中磁盘的活动时间和传输次数决定了平均传输率，而磁盘活动时间百分比值是通过取样期间磁盘的繁忙程度计算出来的。

vmstat 可以带上一个磁盘设备名作为附加参数：

```
# vmstat hdisk1
kthr    memory             page              faults        cpu          disk xfer
-----   ----------- ------------------------ ------------ ----------------- ----------------
r b     avm    fre     re pi po fr sr cy    in    sy    cs    us sy    id wa       1 2 3 4
```

0 0 16273 8385 0 0 0 9 22 0 115 284 39 1 1 98 1 0

vmstat 命令输出中每列数据的含义是：

- kthr：取样时间间隔内，核心线程状态每秒钟变化次数，其中：
  - r：放入运行队列中的核心线程个数；
  - b：放入等待队列中的核心线程个数（等待资源或输入/输出）。
- Memory：包含虚存和实存的使用情况。虚存页一经分配就被认为处于激活状态，一页数据为 4096 字节。
  - avm：活动的虚存页。进程运行时，工作段所使用的内存在调页设备上同时分配一份（用于后备）。该值可用来计算当前全部运行程序分配的调页空间，avm 除于 256 的得到就是以兆字节（MB）为单位的已分配调页空间数量，这与 lsps –a 命令的输出应该是一致的。当调页设备的可分配空间少于 128 页时，系统就会开始杀进程以释放出足够的调页空间，所以系统中应该分配足够大的调页空间。
  - fre：内存空闲表的尺寸。系统自动维护一个名为空闲表的内存页缓冲池，用于 VMM 在需要额外空间满足其需求。空闲表的正常大小与系统的实存有关，实存大于 64MB 的系统，空闲表最小值（MINFREE）为 120 页，小于 64MB 的系统，空闲表最小值是内存大小的两倍减去 8，例如：内存为 32MB，该系统的 MINFREE 等于 56。空闲表的 MINFREE 和 MAXFREE 限值可以通过 vmtune 命令（/usr/samples/kernel 目录下）显示或设值。

注意：

由于文件系统有可能使用大量的实存作为文件缓冲区，所以空闲页常常会变得很小

- Page：包含缺页和调页活动的信息（以秒为单位）。
  - re：调页程序的输出/输出列表；
  - pi：从调页空间中调入的页面数；
  - po：从调页空间中调出的页面数；
  - fr：释放的页面数；
  - sr：调页算法扫描的页面数；
  - cy：调页算法使用的时钟周期
- Fault：每秒的中断调用次数。
  - in：设备中断
  - sy：系统调用
  - cs：核心线程上下文切换
- CPU：CPU 使用不同分类所占的百分比。
  - us：用户态使用时间
  - sy：系统态使用时间
  - id：CPU 空闲时间
  - wa：进程挂起等待磁盘 I/O 的 CPU 时钟周期
- Disk xfer：取样周期内指定物理卷的每秒传输次数，最多允许指定四个物理卷，传输次数统计信息的输出按指定的顺序分列。这项统计值是针对物理设备的，由于几个逻辑请求可能被合成一个物理读写，所以该值并不意味着数据的读写次数。

带-s 参数执行 vmstat 命令，将输出自系统启动以来的发生调页事件的统计值，-s 选项不能和命令的其他选项混用。下面是使用-s 参数的例子：

```
# vmstat -s
      8765020 total address trans. faults
      4832918 page ins
      2989263 page outs
           19 paging space page ins
            7 paging space page outs
            0 total reclaims
      5417148 zero filled pages faults
        12633 executable filled pages faults
     15031850 pages examined by clock
          118 revolutions of the clock hand
      6086090 pages freed by the clock
       105808 backtracks
            0 lock misses
            0 free frame waits
            0 extend XPT waits
      2025516 pending I/O waits
      3031667 start I/Os
      3031667 iodones
     24786000 cpu context switches
     77240518 device interrupts
            0 software interrupts
            0 traps
    191650677 syscalls
```

可能发生的调页事件说明如下：

- address trans：每次发生地址转换缺页（逻辑地址翻译成物理地址后，发现该物理地址的页面不在实存中）即增加该值，解决这种缺页不一定需要额外的 I/O（正确的页面可能还在实存中）。存储保护缺页不计入该值；
- page ins：虚存管理器（VMM）每次读入一个页面即增加该值，该计数包含从调页空间和文件系统空间读入的页面，它与 page out 统计值一起标示了 VMM 产生的 I/O 数量；
- page out：每次 VMM 写回一个页面即增加该值，该计数包含了写回到调页空间和文件系统空间的所有页面，它与 page in 统计值一起标示了 VMM 产生的 I/O 数量；
- paging space page ins：每次 VMM 从调页空间读入一个页面即增加该值；
- paging space page outs：每次 VMM 写回一个页面到调页空间即增加该值；
- total reclaims：每次不需要产生额外 I/O 就能解决地址转换缺页即增加该值。当别的内存请求已经要求 VMM 读入页面但尚未完成、VMM 根据预读算法已经预先读入但隐藏在发生缺页的数据段之外、或者该页面已经被放入空闲页表但尚未重用等几种情况下，都可以直接解决缺页。
- zero filled pages faults：每次能够用一个空白页解决工作段缺页（进程申请新的页面）即增加该值；
- executable filled pages faults：每次代码页缺页即增加该值；
- pages examined by clock：VMM 使用一种时钟算法实现最少使用（LRU）页面更新调度，用时钟检查的次数标示页面的新旧，每次时钟检查一个页面即增加该值。
- revolutions of the clock hand：每次 VMM 时钟走一周（也就是说每次扫描完全部

内存）即增加该值；

- pages freed by the clock：每次时钟算法选择释放一个页面即增加该值；
- backtracks：每当在解决前一个缺页时又发生一个缺页即增加该值（即先得解决新的缺页才能回朔解决初始的缺页）
- lock misses：VMM 通过加锁禁止指向某个页面来控制页面的并发处理，由于无法加锁会导致一次缺页，每次发生这种情况即增加该值；
- free frame waits：每次进程等待 VMM 收集空闲页即增加该值；
- extend XPT waits：每次进程等待 VMM 提交所访问的内存段即增加该值；
- pending I/O waits：每次进程等待 VMM 完成页面调入操作即增加该值；
- start I/Os：每次 VMM 产生一个读写 I/O 即增加该值；
- iodones：每次 VMM 产生的 I/O 请求完成即增加该值；
- CPU context switches：每次 CPU 上下文切换（调入一个新进程）即增加该值；
- device interrupts：每次硬件中断即增加该值；
- software interrupts：每次软件中断即增加该值，软件中断是与硬件中断类似的机器指令，也会通过服务程序保存当前状态和程序分支，系统调用通过软件中断指令转入系统处理分支；
- traps：AIX 操作系统未用；
- syscalls：每次系统调用即增加该值。

下面先给出的是一个空闲系统的 vmstat 输出的例子，然后加上负载，再次执行 vmstat，通过分析输出的统计信息调查潜在的问题所在。

不带任何负载的 vmstat 输出：

```
# vmstat 1 5
 kthr     memory             page              faults        cpu
 -----  -----------  ------------------------  ----------  -----------
r    b   avm    fre  re  pi  po  fr  sr  cy  in   sy  cs  us  sy  id  wa
0    0   16057  1291  0   0   0   8   22  0   113  281 36  1   0   98  1
0    0   16057  1290  0   0   0   0   0   0   108  472 25  0   0   99  0
0    0   16057  1290  0   0   0   0   0   0   109  282 32  0   0   99  0
0    0   16057  1290  0   0   0   0   0   0   109  285 26  0   0   99  0
0    0   16057  1290  0   0   0   0   0   0   108  282 29  0   0   99  0
```

输出的第一行是系统启动以来的平均值，这与不带参数运行 vmstat 的输出是一样的。出于练习的目的，vmtune 命令设置如下：

```
# /usr/samples/kernel/vmtune
vmtune: current values:
   -p       -P        -r          -R          -f        -F        -N          -W
minperm  maxperm  minpgahead  maxpgahead  minfree   maxfree  pd_npages  maxrandwrt
 26007    104028       2           8         120       128      524288        0
   -M       -w        -k          -c          -b        -B        -u          -l
maxpin   npswarn   npskill    numclust   numfsbufs  hd_pbuf_cnt lvm_bufcnt lrubucket  de
104851    4096      1024          1          93        80          9       131072
      -s                 -n          -S                -h
sync_release_ilock  nokillroot  v_pinshm  strict_maxperm
         0              0          0           0
number of valid memory pages = 131063 maxperm=79.4% of real memory
maximum pinable=80.0% of real memory minperm=19.8% of real memory
```

number of file memory pages = 101629 numperm=77.5% of real memory

加上负载后的 vmstat 命令输出：

 # vmstat 1 15

kthr		*memory*			*page*					*faults*			*cpu*			
r	*b*	*avm*	*fre*	*re*	*pi*	*po*	*fr*	*sr*	*cy*	*in*	*sy*	*cs*	*us*	*sy*	*id*	*wa*
0	*0*	*16299*	*1749*	*0*	*0*	*0*	*8*	*21*	*0*	*113*	*281*	*36*	*1*	*0*	*98*	*1*
1	*1*	*16299*	*1529*	*0*	*0*	*0*	*0*	*0*	*0*	*301*	*8707*	*382*	*52*	*13*	*0*	*35*
1	*1*	*16299*	*1398*	*0*	*0*	*0*	*0*	*0*	*0*	*185*	*6557*	*238*	*91*	*8*	*0*	*1*
1	*1*	*16299*	*1227*	*0*	*0*	*0*	*0*	*0*	*0*	*225*	*6705*	*257*	*85*	*15*	*0*	*0*
1	*0*	*16299*	*1049*	*0*	*0*	*0*	*0*	*0*	*0*	*246*	*6587*	*334*	*71*	*10*	*0*	*19*
1	*1*	*16299*	*861*	*0*	*0*	*0*	*0*	*0*	*0*	*250*	*9051*	*317*	*72*	*19*	*0*	*9*
0	*1*	*16265*	*653*	*0*	*0*	*0*	*0*	*0*	*0*	*342*	*10304*	*516*	*37*	*21*	*0*	*43*
4	*0*	*16284*	*121*	*0*	*0*	*0*	*16*	*35*	*0*	*253*	*2432*	*375*	*36*	*6*	*43*	*15*
0	*0*	*16284*	*120*	*0*	*0*	*0*	*432*	*1066*	*0*	*265*	*302*	*246*	*31*	*4*	*54*	*11*
1	*0*	*16284*	*121*	*0*	*0*	*0*	*160*	*389*	*0*	*221*	*1184*	*239*	*8*	*5*	*77*	*10*
0	*1*	*16284*	*120*	*0*	*0*	*0*	*576*	*1447*	*0*	*394*	*2377*	*525*	*28*	*9*	*39*	*24*
0	*0*	*16284*	*122*	*0*	*0*	*0*	*232*	*480*	*0*	*277*	*1185*	*346*	*21*	*5*	*63*	*11*
0	*0*	*16284*	*122*	*0*	*0*	*0*	*384*	*1630*	*0*	*326*	*1081*	*400*	*16*	*12*	*51*	*21*
0	*0*	*16284*	*126*	*0*	*0*	*0*	*336*	*784*	*0*	*284*	*742*	*326*	*20*	*3*	*59*	*18*
0	*1*	*16284*	*126*	*0*	*0*	*0*	*761*	*1615*	*0*	*336*	*1032*	*420*	*36*	*4*	*48*	*12*

从上面的输出可以看出，kthr（核心线程）列中的运行线程数和等待线程数基本保持稳定并且取值较小，运行线程数总是小于 5，而等待线程数接近 0。

在 memory 列，avm（活动虚存）保持相对稳定，但 fre（空闲实存）从 1749 减少到最小值 120。如果 fre 长时间低于 120，系统将持续的进行页面的调入和调出，从而导致系统的性能问题。

在 page 列，re、pi、po 和 cy 值保持相对稳定，fr 和 sr 列则逐渐增长，pi 率不高于 5。然而每次调入操作的发生，总伴随着之前的一次调出操作，就像在一个内存紧张的环境中，每次调入操作被迫进行偷页（因此调出被偷的页面）。如果系统持续读入大量的页面，看到 po 值增长但没有相应的 pi 值增长，这种情况并不意味着系统颠簸，而应该检查应用系统的数据存取模式。fr 列表示释放出来的页面数，而 sr 列表时页面调度算法扫描的页面数。对于稳定的无碎片的内存来说，页面扫描率和页面释放率应该相接近。对于多进程的系统，进程各自使用不同页面，页面更容易发生变化，并且相互不连续，这种情况下页面扫描率就可能大大超过页面释放率。

在 faults 列，in、sy 和 cs 值起伏很大，由于开销极小，对于这些值没有什么固定的限制，并且也很难说出哪些值属于过量。唯一要记住的是 in 值总是大于 100。

在 cpu 列，us、sy、id 和 wa 值起伏也很大。这部分的输出是 cpu 利用率的百分比值。us 部分是进程用户态部分所消耗的 cpu 时间，sy 则是进程核心态部分所消耗的 cpu 时间。最优化的使用是让 cpu 100%繁忙，但这只对不需要分享 cpu 的单用户系统有意义。通常情况下，如果单用户系统的 us+sy 低于 90%，就认为该系统不存在 cpu 瓶颈；然而一旦多用户系统的 us+sy 值高于 80%，进程就常常得在运行队列中等待，系统的等待时间和吞吐率将大受影响。id 表示 cpu 的空闲时间，wa 表示 cpu 空闲，但正在等待本地的磁盘输入/输出。wa 值高于 40%表明磁盘子系统工作不均衡，或存在大量的磁盘访问。这四项值相加应等于 100%的 cpu 利用率。