**Linux系统性能测试与分析**

# 1、前言

通过对系统中和性能相关的各个环节的介绍，使大家知道出现性能问题时可以从那些方面入手去查，而分析典型应用对系统资源使用的特点，让大家对应用和系统资源的依赖有了更直观的认识。大多数的硬件性能问题主要和CPU、磁盘、内存相关，还没有遇到因为开发语言的运行效率对整个应用的性能造成影响，而应用程序设计的缺陷和数据库查询的滥用反倒是最最常见的性能问题。需要注意的是，大多数情况下，虽然性能瓶颈的起因是程序性能差或者是内存不足或者是磁盘瓶颈等各种原因，但最终表现出的结果就是CPU耗尽，系统负载极高，响应迟缓，甚至暂时失去响应，因此我们观察服务器状况时，最先看的就是系统负载和CPU空闲度。当你阅读完了这遍文档以后就会有一个对系统分析的思路。

**2、性能分析的目的**

## 2.1找出系统性能瓶颈

1.硬件瓶颈

2.软件瓶颈

## 2.2提供性能优化方案

1.升级硬件

2.改进系统结构

达到合理的硬件和软件配置，使系统资源使用达到平衡。但遗憾的是解决一个性能瓶颈，往往又会出现另外的瓶颈或者其他问题，所以性能优化更加切实的目标是做到在一定范围内使系统的各项资源使用趋向合理和保持一定的平衡。系统运行良好的时候恰恰也是各项资源达到了一个平衡体，任何一项资源的过渡使用都会造成平衡体系破坏，从而造成系统负载极高或者响应迟缓。比如CPU过渡使用会造成大量进程等待 CPU资源，系统响应变慢，等待会造成进程数增加，进程增加又会造成内存使用增加，内存耗尽又会造成虚拟内存使用，使用虚拟内存又会造成磁盘IO增加和CPU开销增加 （用于进程切换、缺页处理的CPU开销）

**3、性能相关的各个环节**

**3.1 硬件资源**

### 3.1.1、CPU

   ⒈ 是否使用SMP。

   ⒉ 单颗CPU的性能对依赖CPU的某些应用的影响很严重，比如数据库的查询处理。

### 3.1.2、内存

  1. 物理内存

      物理内存不够时会使用交换内存。

   2. 交换内存

      使用交换内存会带来磁盘IO和CPU的开销增加。

### 3.1.3、存储系统

SCSI磁盘

ATA/SATA磁盘

RAID磁盘阵列(RAID0, RAID1, RAID5, RAID0+1)

一些经验：

1. 小文件读写的性能瓶颈是磁盘的寻址（随机读写性能更差），评估的标准是tps

2. 大文件读写的性能瓶颈是带宽，评估的标准是持续的读写速度

3. Linux可以利用空闲内存作文件系统访问的cache，因此系统内存越大存储系统  的性能也越好

### 3.1.4、带宽

1. 网络带宽

⒉ SCSI总线带宽

大文件访问时SCSI的带宽瓶颈

3. 系统总线带宽

**3.2 操作系统**

SMP性能

VM性能

IO性能(存储设备、网络设备、异步IO)

文件系统性能(大文件优化、小文件优化、写优化、读优化、网络文件系统)

多线程性能

**3.3 服务器软件**

根据各个服务器的不同而定制不同的软件，这里就不多说了，服务器软件实在是太多了。

**3.4 开发平台/中间件软件/框架软件**

没有什么规范的定义，按情况而定。

**3.5 应用程序**

一些应用软件。

**4、典型应用对系统资源使用的特点**

**4.1动态内容为主的Web应用**

⒈ 频繁执行程序，如 Perl, PHP, Java 等，消耗CPU严重。

⒉ 提供并发用户访问，因此系统进程数多，消耗内存多，当内存不足时，使用交换内存也会增加

   CPU的开销。

⒊ 磁盘的写IO比较频繁（主要为随机写），比如生成cache文件，更新session文件等。

⒋ 内存充足时读取的内容可以被cache住，cache的命中率和文件更新的频繁程度成反比，磁盘的 读IO相对较小 。

**4.2静态内容为主的Web应用 (如Squid Cache)**

⒈ 网络带宽瓶颈。

⒉ 小文件的随机读取频繁，内存充足时可以缓解磁盘随机读的压力。

⒊ 系统内存不足时磁盘IO量会比较大（读、写、交换内存），因此增加CPU的开销。

**4.3数据库应用**

1．数据库查询语句复杂，大量的 where 子句，order by, group by 排序等，CPU容易出现瓶颈。

2．表太大时，查询遍历全表造成磁盘读的IO量大，容易出现读IO等待的情况。

3．数据更新量大或者更新频繁时，造成磁盘写的IO量大。

4．内存不足时频繁使用交换内存。

**4.4软件下载**

1．网络带宽瓶颈

2．存储系统带宽瓶颈(读)

**4.5流媒体服务**

1．网络带宽瓶颈

2．存储系统带宽瓶颈(读)

**5、系统使用和优化的原则**

**5.1系统监控和数据数据**

**对资源的使用状况作长期的监控和数据收集**

Snmp+MRTG（网络流量监控）

Sar（系统资料监控及收集）

**5.2优化建议**

程序的优化和系统结构的优化比硬件的性能优化更有效

**5.3限制使用系统资源**

避免不受限制的使用系统资源

设置各项服务对资源的使用限额，如Apache, MySQL,PHP等

**5.4保留空闲资源及资源均衡**

**始终保留一定量的空闲资源**

多少合适？根据应用的特点，比如是否有突发性使用增长？

日常情况下，保留至少 60% 的系统资源，以应付突发使用增长。

日常情况下，资源使用率达到 80% 时，你必须有所行动了，尤其是web应用。

系统硬件达到合理的配置（以适合应用的特点为依据，资源消耗均衡为目标）

系统性能的水桶理论

**应用软件对资源的使用要均衡（理想目标）**

怎么样就算是均衡了？我也在摸索中……

理想状况为：CPU消耗到50%的时候，磁盘的带宽也到50%，磁盘的tps也到50%，内存使用也到50%(除去可以提供给cache的内存)

**6、附件**

**6.1常用工具详解**

**6.1.1 Free**

|  |  |
| --- | --- |
| **Free:** | |
| 名称： | 显示系统内存 的使用状况（物理内存和交换内存） |
| 格式： | free [-b|-k|-m|-g] [-l] [-s delay] |
| 描述： | 通过这个命令我们可以看到系统进程实际使用的物理内存，buffer和cache  使用的物理内存 |
| 常用选项： | -b,-k,-m,-g  分别以字节、KB、MB、GB为单位显示内存使用情况。  -s delay：显示每隔多少秒数来显示一次内存使用情况  -l 逐条的显示Low和High内存统计表 |
| 输出信息简介： | •  free命令输出的第二行(Mem)     这行分别显示了物理内存的总量(total)、已使用的(used)、空闲的(free)、     共享的(shared)、buffer、cache的内存。  •  free命令输出的第三行(-/+ buffers/cache)。这行最容易让人迷惑。     它显示的第一个值（used这一列）是这样得来的：     Mem行used列 - Mem行buffers列 - Mem行cached列     它显示的第二个值（free这一列）是这样得来的：     Mem行free列 + Mem行buffers列 + Mem行cached列  •  free命令输出的第四行(Swap)     这行显示交换内存的总量、已使用量、空闲量  通常buffer和cache可以使用的内存空间越大，系统IO和文件系统访问的性能  越好。 |

**6.1.2 uptime**

|  |  |
| --- | --- |
| **uptime:** | |
| 名称： | 显示系统正常运行多少时间 |
| 格式： | uptime |
| 描述： | · **uptime** 命令打印当前时间、系统正常运行的时间长度、联机用户数目和平均负载.     负载平均值是以 1 分钟、5 分钟、15 分钟时间间隔开头的可运行的进程。  · 最简便的查看系统负载的工具，系统负载越小，系统运行状况越好，对于系统负载处  在什么范围内比较合适，我想是没有定论的，我介绍一下我的习惯。  · 我一般以15分钟负载的值来评估系统的健康度，以10为这个值的临界点，     如果系统负载持续高于10，通常是存在某个资源长期紧张的原因，我们需     要重视，并且得开始着手解决这个问题了。如果偶尔高于10，应该开始留     意它出现的频度，这往往是前面一种状况的先兆。 |

**6.1.3 vmstat**

|  |  |  |
| --- | --- | --- |
| **vmstat:** | |  |
| 名称： | 报告虚拟内存的统计信息 |  |
| 格式： | vmstat [-n] [延时[次数]] |  |
| 描述： | · vmstat是一个很全面的性能分析工具，可以观察到系统的进程状态、     内存使用、虚拟内存使用、磁盘的IO、中断、上下问切换、CPU使用等。     系统性能分析工具中，我使用最多的是这个，除了 sysstat 工具包外，     这个工具能查看的系统资源最多。  · 对于 Linux 的性能分析，100%理解 vmstat 输出内容的含义，那你     对系统性能分析的能力就算是基本掌握了。  · 我这里主要说明一下这个命令显示出的部分数据代表的含义，和它反     映出系统相关资源的状况。输出内容共有 6 类，分别说明如下。 | |
| 常用选项： | **-**n 开关令第一次的信息只显示一次而不是周期地产生。    -S 单位大小    -a 显示活跃和不活跃的内存信息    -s 显示相关虚拟内存统计表 | |
| 输出信息简介： | **• Procs**  r: 运行的和等待(CPU时间片)运行的进程数，这个值也可以判断是否需要增加CPU(长期大于1)      b: 处于不可中断状态的进程数，常见的情况是由IO引起的  • **Memory**     swpd: 切换到交换内存上的内存(默认以KB为单位)       • 如果 swpd 的值不为0，或者还比较大，比如超过100M了，但是         si, so 的值长期为0，这种情况我们可以不用担心，不会影响系统         性能。     free: 空闲的物理内存     buff: 作为buffer cache的内存，对块设备的读写进行缓冲     cache: 作为page cache的内存, 文件系统的cache          • 如果 cache 的值大的时候，说明cache住的文件数多，如果频            繁访问到的文件都能被cache住，那么磁盘的读IO bi 会非常小。  **• Swap**     si: 交换内存使用，由磁盘调入内存     so: 交换内存使用，由内存调入磁盘         • 内存够用的时候，这2个值都是0，如果这2个值长期大于0时，           系统性能会受到影响。磁盘IO和CPU资源都会被消耗。         • 我发现有些朋友看到空闲内存(free)很少或接近于0时，就认           为内存不够用了，实际上不能光看这一点的，还要结合si,so，           如果free很少，但是si,so也很少(大多时候是0)，那么不用担           心，系统性能这时不会受到影响的。  **• Io**     bi: 从块设备读入的数据总量(读磁盘) (KB/s)     bo: 写入到块设备的数据总理(写磁盘) (KB/s)         •  随机磁盘读写的时候，这2个 值越大（如超出1M），能看            到CPU在IO等待的值也会越大  **• System**     in: 每秒产生的中断次数     cs: 每秒产生的上下文切换次数         • 上面这2个值越大，会看到由内核消耗的CPU时间会越多  **• Cpu**     us: 用户进程消耗的CPU时间百分比       • us 的值比较高时，说明用户进程消耗的CPU时间多，但是如         果长期超过50% 的使用，那么我们就该考虑优化程序算法         或者进行加速了(比如 PHP/Perl)     sy: 内核进程消耗的CPU时间百分比       • sy 的值高时，说明系统内核消耗的CPU资源多，这并不是良性         的表现，我们应该检查原因。  id: CPU处在空闲状态时间百分比     wa: IO等待消耗的CPU时间百分比       • wa 的值高时，说明IO等待比较严重，这可能是由于磁盘大量         作随机访问造成，也有可能是磁盘的带宽出现瓶颈(块操作)。  **• 情景分析**     这个vmstat的输出那些信息值得关注？       • Procs r: 运行的进程比较多，系统很繁忙       • Io bo: 磁盘写的数据量稍大，如果是大文件的写，10M以内基本         不用担心，如果是小文件写2M以内基本正常       • Cpu us: 持续大于50，服务高峰期可以接受       • Cpu wa: 稍微有些高       • Cpu id:持续小于50，服务高峰期可以接受 |  |

**6.1.4 top**

|  |  |  |
| --- | --- | --- |
| **top:** | | |
| 名称： | 显示执行中的程序进程 | |
| 格式： | top [－] [d delay] [q] [c] [S] [s] [i] [n] | |
| 描述： | •  这个命令可以查看系统中运行的进程的状况，CPU使用状况，系统负载，     内存使用等。它是检查系统进程运行状况最方便的工具了，它默认     显示部分活动的进程，并且按照进程使用CPU的多少排序。它可以显示     全部CPU的使用状况，也可以显示每个进程都运行在那个CPU上面。  •  我习惯使用这个命令查看那些进程或者那类进程占用CPU和内存资源最多，     以此迅速定位存在性能问题的进程，以及运行异常的进程。 | |
| 常用选项： | -d：指定更新的间隔，以秒计算。    -q：没有任何延迟的更新。如果使用者有超级用户，则top命令将会以        最高的优先序执行。    -c：显示进程完整的路径与名称。    -S：累积模式，会将己完成或消失的子行程的CPU时间累积起来。    -s：安全模式。    -i：不显示任何闲置(Idle)或无用(Zombie)的行程。    -n：显示更新的次数，完成后将会退出top。 |  |
| Top命令使过  程中，还可以使用一些交互的命令来完成其它参数的功能。这些命令是通过快捷键启动的。 | <空格>：立刻刷新。     P：根据CPU使用大小进行排序。     T：根据时间、累计时间排序。     q：退出top命令。     m：切换显示内存信息。     t：切换显示进程和CPU状态信息。     W：将当前设置写入~/.toprc文件中。这是写top配置文件的推荐方法。     c：切换显示命令名称和完整命令行。     U：切换为指定用户的进程运行界面     M：根据使用内存大小进行排序。 |  |
| 输出信息简介： | load average：命令输出的最后内容表示在过去的1、5、15分钟内运行队列中的平均  进程数量。数据是每隔5秒钟检查一次活跃的进程数，然后根据这个数值算出来的。一  般来说只要每个CPU的当前活动进程数不大于3，那么系统的性能就是良好的，如果每个  CPU的任务数大于5，那么就表示这台机器的性能有严重问题。对于上面的例子来说，假设  系统有两个CPU，那么其每个CPU的当前任务数为：8.13/2=4.065。这表示该系统的性  能是可以接受的。  • 用 top 看到的进程所处的几种状态(STT列)。    -D：不可中断休眠，通常是 IO 操作所处的状态    -R：正在执行的或者处在等待执行的进程队列中    -S：休眠中    -T：暂停刮起的（比如Ctrl+Z），也可能是被 strace 命令调用中的状态    -Z：僵尸进程，进程执行完成，但由于其父进程没有销毁该进程，而被        init进程接管进行销毁。    -W：没有使用物理内存，所占用的物理内存被切换到交换内存    -<：高优先级的进程    -N：低优先级    有时候一个进程会有多个状态的标志，比如SWN，SW  • 在图中，第一行表示的项目依次为当前时间、系统启动时间、当前系统    登录用户数目、平均负载。  • 第二行显示的是所有启动的进程、目前运行的、挂起(Sleeping)    的和无用(Zombie)的进程。  • 第三行显示的是目前CPU的使用情况，包括系统占用的比例、    用户使用比例、闲置(Idle)比例。  • 第四行显示物理内存的使用情况，包括总的可以使用的内存、已用内存、    空闲内存、缓冲区占用的内存。  • 第五行显示交换分区使用情况，包括总的交换分区、使用的、空闲的    和用于高速。缓存的大小。  • 第六行显示的项目最多，下面列出了详细解释。   PID（Process ID）：进程标示号。   USER：进程所有者的用户名。   PR：  进程的优先级别。   NI：  进程的优先级别数值。   VIRT：进程占用的虚拟内存值。   RES： 进程占用的物理内存值。   SHR： 进程使用的共享内存值。   S：   进程的状态，其中S表示休眠，R表示正在运行，Z表示僵死状态，N  表示该进程优先值是负数。   %CPU：该进程占用的CPU使用率。   %MEM：该进程占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。   Command：进程启动的启动命令名称，如果这一行显示不下，进程会有一个完整的命令行。 |  |

**6.1.5 sysstat工具包**

|  |  |
| --- | --- |
| **Sysstat: sa1、sa2** | |
| 名称： | Sa1:收集并存储每天系统动态信息到一个二进制的文件中  Sa2:把每天的系统活跃性息写入总结性的报告中 |
| 格式： | /usr/lib/sa/sa1  /usr/lib/sa/sa2 |
| 描述： | Sa1工具负责收集并存储每天系统动态信息到一个二进制的文件中。它是通过  计划任务工具cron来运行， 是为sadc所设计的程序前端程序。  Sa2工具负责把每天的系统活跃性息写入总结性的报告中。它是为sar所设计的前端 ，要通过cron来调用。    sa1, sa2 这2个命令以配置在cron中定期执行，把系统当时的运行状况信息保存在磁盘上，每日存在一个文件中，因为有这个功能，因此 sar 工具不单是一个性能分析的工具，这2个命令的使用说明如下：  sa1 配置在cron中可以实现系统状态收集，比如10分钟运行一次  sa2 配置在cron中可以实现每日状态的汇总报告      你可以在系统crontab中添加如下配置：      \*/10 \* \* \* \* root /usr/lib/sa/sa1 1 1      53 23 \* \* \* root /usr/lib/sa/sa2 -A |
| **Sysstat: sadc** | |
| 名称： | 系统动态数据收集 |
| 格式 | /usr/lib/sa/sadc [ -d ] [ -F ] [ -I ] [ -L ] [ -V ] [ interval [ count ] ] [ outfile ] |
| 描述： | sadc 是系统动态数据收集工具，收集的数据被写一个二进制的文件中，它被用作sar工具的后端。如果想查看数据内容，需要用sadf工具来显示 |
| 常用选项： | -d 报告硬盘设置的相关统计； -F 强制把数据写入文件。 -I 报告所有系统中断数据。 interval 表示时间间隔，单位是秒，比如3 。 count 统计数据的次数，也是一个数字。 outfile 输出统计到outfile文件。 |
| **Sysstat:iostat** | |
| 名称： | 报告磁盘活动统计情况，同时也会汇报出CPU使用情况 |
| 格式： | iostat [ -c | -d ] [ -k ] [ -t ] [ -V ] [ -x [ device ] ] [ interval] [count] |
| 描述： | iostat是I/O statistics（输入/输出统计）的缩写，iostat工具将对系统的磁盘  操作活动进行监视。它的特点是汇报磁盘活动统计情况，同时也会汇报出CPU使用  情况。同vmstat一样，iostat也有一个弱点，就是它不能对某个进程进行深入分  析，仅对系统的整体情况进行分析。 |
| 常用选项： | -c为汇报CPU的使用情况；  -d为汇报磁盘的使用情况；  -k表示每秒按kilobytes字节显示数据；  -p显示所有存储设备分区信息。  -t为打印汇报的时间；  -v表示打印出版本信息和用法；  -x device指定要统计的设备名称，默认为所有的设备；   interval指每次统计间隔的时间；   count指按照这个时间间隔统计的次数。 |
| 输出信息简介： | CPU：表示机器内所有的CPU； %user 表示CPU的利用率； %nice 表示CPU在用户层优先级的百分比，0表示正常； %system 表示当系统运行时，在用户应用层上所占用的CPU百分比； %iowait 表示请求硬盘I/0数据流出时，所占用CPU的百分比； %idle 表示空闲CPU百分比，值越大系统负载越低  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。  wkB/s: 每秒写K字节数。是 wsect/s 的一半。  avgrq-sz: 平均每次设备I/O操作的数据大小 (扇区)。即 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。 |
| 情景分析： | 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。   svctm 一般要小于 await (因为同时等待的请求的等待时间被重复计算了)，  svctm 的大小一般和磁盘性能有关，CPU/内存的负荷也会对其有影响，请求过多  也会间接导致 svctm 的增加。await 的大小一般取决于服务时间(svctm) 以及  I/O 队列的长度和 I/O 请求的发出模式。如果 svctm 比较接近 await，说明  I/O 几乎没有等待时间；如果 await 远大于 svctm，说明 I/O 队列太长，应用  得到的响应时间变慢，如果响应时间超过了用户可以容许的范围，这时可以考虑  更换更快的磁盘，调整内核 elevator 算法，优化应用，或者升级 CPU。   队列长度(avgqu-sz)也可作为衡量系统 I/O 负荷的指标，但由于 avgqu-sz 是  按照单位时间的平均值，所以不能反映瞬间的 I/O 洪水。 |
| **Sysstat:mpstat** | |
| 名称： | 多处理器系统中的CPU的利用率的统计 |
| 格式： | mpstat [ -P { cpu | ALL } ] [ -V ] [ interval [ count ] ] |
| 描述： | mpstat是Multiprocessor Statistics的缩写，是实时系统监控工具。其报告与CPU的一些统计信息，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。 |
| 常用选项： | -P {|ALL} 表示监控哪个CPU， cpu在[0,cpu个数-1]中取值  internal 相邻的两次采样的间隔时间  count 采样的次数，count只能和delay一起使用 |
| 输出信息简介： | Cpu cpu Id号  user 在internal时间段里，用户态的CPU时间（%） ，不包含 nice值为负进程。  nice 在internal时间段里，nice值为负进程的CPU时间（%）  system 在internal时间段里，核心时间（%）  iowait 在internal时间段里，硬盘IO等待时间（%）  irq 在internal时间段里，软中断时间（%）  soft 在internal时间段里，软中断时间（%）  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 （%）  intr/s 在internal时间段里，每秒CPU接收的中断的次数 |
| **Sysstat:sar** | |
| 名称： | 收集、显示、保存系统活跃信息 |
| 格式： | sar [ -A ] [ -b ] --- [ -n { DEV | EDEV | NFS | NFSD | SOCK | ALL }  ] --- |
| 描述： | sar 工具比较强大，既能收集系统CPU、硬盘、动态数据，也能显示动态显示，  更能查看二进制数据文件；sar 的应用比较多，而且也比较复杂，数据更为  精确。 |
| 常用选项： | -A 显示所有历史数据，通过读取/var/log/sar 目录下的所有文件，并把它们     分门别类的显示出来； -b 通过设备的I/O中断读取设置的吞吐率； -B 报告内存或虚拟内存交换统计； -c 报告每秒创建的进程数； -d 报告物理块设备（存储设备）的写入、读取之类的信息，如果直观一点，     可以和p参数共同使用，-dp -f 从一个二进制的数据文件中读取内容，比如 sar -f filename -i interval 指定数据收集的时间，时间单位是秒； -n 分析网络设备状态的统计，后面可以接的参数有 DEV、EDEV、NFS、NFSD、     SOCK等。比如-n DEV -o 把统计信息写入一个文件，比如 -o filename ； -P 报告每个处理器应用统计，用于多处理器机器，并且启用SMP内核才有效； -p 显示友好设备名字，以方便查看，也可以和-d 和-n 参数结合使用，比如     -dp 或-np -r 内存和交换区占用统计； -R -t 这个选项对从文件读取数据有用，如果没有这个参数，会以本地时间为标     准 读出； -u 报告CPU利用率的参数； -v 报告inode, 文件或其它内核表的资源占用信息； -w 报告系统交换活动的信息； 每少交换数据的个数； -W 报告系统交换活动吞吐信息； -x 用于监视进程的，在其后要指定进程的PID值； -X 用于监视进程的，但指定的应该是一个子进程ID； |
| 输出信息简介： | proc/s 在internal时间段里，每秒上下文切换率  cswch/s 在internal时间段里，每秒上下文切换率  intr/s 在internal时间段里，每秒CPU接收的中断的次数  runq-sz 采样时，运行队列中任务的个数，不包括vmstat 进程procs\_running-1  plist-sz 采样时，系统中活跃的任务的个数 nr\_threads  ldavg-1 采样的前一分钟系统的负载(%) lavg\_1  ldavg-5 采样的5分钟系统的负载(%) lavg\_5  ldavg-15 采样的前15分钟系统的负载(%) lavg\_15  CPU：表示机器内所有的CPU； %user 表示CPU的利用率； %nice 表示CPU在用户层优先级的百分比，0表示正常； %system 表示当系统运行时，在用户应用层上所占用的CPU百分比； %iowait 表示请求硬盘I/0数据流出时，所占用CPU的百分比； %idle 表示空闲CPU百分比，值越大系统负载越低；  第一字段：时间；  IFACE：设备名；  rxpck/s: 每秒收到的包；  rxbyt/s：每秒收到的所有包的体积；  txbyt/s：每秒传输的所有包的体积；  rxcmp/s：每秒收到数据切割压缩的包总数；  txcmp/s: 每秒传输的数据切割压缩的包的总数；  rxmcst/s:每秒收到的多点传送的包；    bread/s： 每秒从硬盘读入系统缓冲区buffer的物理块数。  lread/s： 平均每秒从系统buffer读出的逻辑块数。  %rcache： 在buffer cache中进行逻辑读的百分比。  bwrit/s： 平均每秒从系统buffer向磁盘所写的物理块数。  lwrit/s： 平均每秒写到系统buffer逻辑块数。  %wcache： 在buffer cache中进行逻辑读的百分比。  pread/s： 平均每秒请求物理读的次数。  pwrit/s： 平均每秒请求物理写的次数  device： sar命令正在监视的块设备的名字。  %busy： 设备忙时，传送请求所占时间的百分比。  avque： 队列站满时，未完成请求数量的平均值。  r+w/s： 每秒传送到设备或从设备传出的数据量。  blks/s： 每秒传送的块数，每块512字节。  avwait： 队列占满时传送请求等待队列空闲的平均时间。  avserv： 完成传送请求所需平均时间（毫秒）。  如果想知道网络设备错误报告，也就就是用来查看设备故障的。应该用EDEV |