知识点整理

# 修订记录

|  |  |  |  |
| --- | --- | --- | --- |
| 日期 | 修订版本 | 修改描述 | 作者 |
| **2015-12-04** | V1.0 | 初始版本 | maxluo |
| **2015-12-05** | V1.1 | 脑图内容word化 | devirdai(戴礼松) |

目录

[Linux常用诊断命令和方法 1](#_Toc437022642)

[1. 修订记录 2](#_Toc437022643)

[1操作系统 3](#_Toc437022644)

[1.1CPU 3](#_Toc437022645)

[1.2 内存 7](#_Toc437022646)

[1.3 磁盘I/O 8](#_Toc437022647)

[1.4 网络I/O 10](#_Toc437022648)

[1.5 系统 14](#_Toc437022649)

[1.6 监控 19](#_Toc437022650)

[2 调试工具 19](#_Toc437022651)

[2.1 编译阶段 19](#_Toc437022652)

[2.2 运行阶段 23](#_Toc437022653)

[2.3 proc文件系统 27](#_Toc437022654)

[3 程序 28](#_Toc437022655)

[3.1 进程CPU 28](#_Toc437022656)

[3.2 pgrep查看进程ID 28](#_Toc437022657)

[3.3 内存 28](#_Toc437022658)

[3.4 进程磁盘I/O 29](#_Toc437022659)

[3.5 查看进程中的线程数 29](#_Toc437022660)

[3.6 进程堆栈 29](#_Toc437022661)

[3.7 lsof命令，可以组合使用 29](#_Toc437022662)

[3.8 gstack pid 30](#_Toc437022663)

[3.9 strace 30](#_Toc437022664)

[3.10内存泄漏诊断 30](#_Toc437022665)

[3.11 查看进程的运行堆栈信息 31](#_Toc437022666)

[3.12 生成core文件 31](#_Toc437022667)

[3.13性能诊断 31](#_Toc437022668)

## 1操作系统

### 1.1CPU

#### 1.1.1 vmstat

-f ,显示系统自启动以来执行的fork次数

-s,显示内存相关的统计信息以及多种系统活动的数量（比如CPU上下文切换次数）

-d，显示磁盘相关的统计信息。

vmstat 1 每隔一秒输出系统状态，数据来源/pro/stat，输出各列的含义

|  |  |
| --- | --- |
| 列 | 含义 |
| procs | r,运行队列  b,阻塞队列，为block与就绪之和 |
| memory | swpd ， 虚拟内存已使用的大小，最好为0，或者在一定时间内保持不变，swpd大小一般为内存的2倍，腾讯的服务器基本上都为2G  free， 空闲的物理内存的大小  buff，Linux/Unix系统是用来存储，目录里面有什么内容，权限等的缓存，为了降低高低速设备的差异而存在如内存与磁盘  cache，直接用来记忆我们打开的文件,给文件做缓冲，已申请过的内存，但未被使用  可用物理内存free+buff+cache |
| swap | si 每秒从磁盘读入虚拟内存的大小  so 每秒虚拟内存写入磁盘的大小  值过大，说明内存不够用 |
| io | bi 块设备每秒接收的块数量，这里的块设备是指系统上所有的磁盘和其他块设备，读磁盘  bo 块设备每秒发送的块数量，例如我们写文件，写磁盘  （相对于内存来说的in或者out）,值太大，可能是在读写大量文件 |
| system | in 每秒CPU的中断次数，包括时间中断  cs 每秒上下文切换次数，值太大，多半是活跃的进程数过多 |
| cpu- | 用户CPU时间,值太大则是计算密集型  sy 系统CPU时间，值太大，较多的系统调用，比如I/O操作  id 空闲 CPU时间  wt 等待IO CPU时间，值太大，说明I/O频繁或者出现瓶颈 |

vmstat -S M 1 2 输出流量数据单位为M

统计的是实体机，如果虚拟机执行，则不太准，cs和r常常比较大，没有参考价值

#### 1.1.2 mpstat

查看每个CPU的运行状态，数据来源/pro/stat

mpstat 1/mpstat -P ALL 1 每隔一秒输出一次数据 ， 常用诊断参数有

%idle，数字较低时，出现CPU瓶颈，或者是CPU繁忙

%iowait,数字较大时，则I/O出现问题，每进行一次I/O等待时间较长

#### 1.1.3 top

|  |  |
| --- | --- |
| 参数 | 含义 |
| up 266 days | 本机已经运行时间 |
| load average: 1.43, 0.91, 0.78 | CPU平均负载，值一般单个逻辑CPU小于1比较好 |
| Tasks | 在运行任务情况 |
| Cpu(s) | CPU相关参数，具体含义与vmstat一致， 可以通过键盘输入“1”，来查看每个CPU的详细情况 |
| Mem | 内存使用情况 |
| Swap | 虚拟内存 |

具体进程情况，主要关注

|  |  |
| --- | --- |
| 参数 | 含义 |
| VIRT | 进程使用的虚拟内存总量，单位kb |
| RES | 进程使用的、未被换出的物理内存大小，单位kb |
| SHR | 共享内存大小，单位kb |
| %MEM | 进程使用的物理内存百分比 |
| Mem | 内存使用情况 |
| Swap | 虚拟内存 |

按键"1"，查询每个CPU负载情况，wa为I/O阻塞（网络和磁盘I/O）且无其他进程竞争，关注id,较低时，负载较高。

查看指定进程情况： top -p 31497 -d 3 查看进程运行状态，每隔3秒输出

#### 1.1.4 uptime

查看负载，运行时间等

#### 1.1.5 sar

sar -u 3 5 查看CPU负载

#### 1.1.6 watch

每隔3秒查看机器负载 watch -n 3 'cat /proc/loadavg'

#### 1.1.7 查看逻辑CPU个数

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

#### 1.1.8 查看物理CPU个数

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

#### 1.1.9 每个物理CPU内核个数

grep "cpu cores" /proc/cpuinfo|uniq

#### 1.1.10 每个物理CPU上逻辑CPU个数

grep "siblings" /proc/cpuinfo|uniq

#### 1.1.11 超线程

1， 逻辑CPU个数 > 物理CPU个数 \* CPU内核数

2， cat /proc/cpuinfo中的flags存在ht

### 1.2 内存

#### 1.2.1 free

free -m可用内存应该是free+buffers+cached

消息来源 cat /proc/meminfo

#### 1.2.2 cat /proc/slabinfo

查看内存总体使用情况

### 1.3 磁盘I/O

#### 1.3.1 iostat

iostat -d -x 2 2秒统计一次磁盘I/O数据

iostat /dev/sda4 -x 2 查看指定磁盘I/O情况

|  |  |
| --- | --- |
| 参数 | 含义 |
| r/s | 每秒完成的读 I/O 设备次数 |
| w/s | 每秒完成的写 I/O 设备次数 |
| rsec/s | 每秒读扇区数 |
| wsec/s | 每秒写扇区数 |
| rkB/s | 每秒读K字节数。是 rsect/s 的一半，因为扇区大小为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) |
| avgqu-sz | 平均I/O队列长度。 |
| await | 平均每次设备I/O操作的等待时间 (毫秒) |
| svctm | 平均每次设备I/O操作的服务时间 (毫秒) |
| %util | 一秒中有百分之多少的时间用于 I/O 操作 |

|  |  |
| --- | --- |
| 常见问题 | 含义 |
| %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 比较接近await | 说明I/O 几乎没有等待时间 |
| await 远大于svctm | 说明I/O队列太长，应用得到的响应时间变慢 |
| %util很大，而rkB/s和wkB/s很小 | 一般是因为磁盘存在较多的磁盘随机读写，最好把磁盘随机读写优化成顺序读写。 |

常见诊断及解决办法：

1，想区分写压力 则扩容

2，想解决读压力 则增加缓存

MYSQL：

select 读

update 读+写+读

insert 写

则可以根据磁盘读写数据比值来判断mysql的操作

#### 1.3.2 sar

sar -d t1 t2

t1时间统计一次磁盘情况，一共统计输出t2次

#### 1.3.3 iotop

与top命令类似 ，数据来源 /proc/diskstats

### 1.4 网络I/O

#### 1.4.1 tcpdump

|  |  |
| --- | --- |
| 命令、用法 | 含义 |
| /usr/sbin/tcpdump -i eth1 port 53101 and host 10.198.15.176 and \(10.198.21.24 or 10.198.19.179 or 10.224.153.12 or 10.223.25.101 or 10.198.18.47\) | 主机 10.198.15.176 与其他机器之间的通信 |
| sudo /usr/sbin/tcpdump -i lo src 10.6.223.152 and port 53101 and dst 10.6.223.152 -nn -s0 -w "/tmp/maxluo/javatest/tcp.txt" |  |
| tcpdump -ieth1 -c 100 -w filename.txt | 转100次网路包记录到filename.txt中 |

#### 1.4.2 netstat

|  |  |
| --- | --- |
| 用途、用法 | 解释 |
| 查看端口占用情况  netstat -anp|grep xxx | -n参数是不展示端口别名  端口别名配置文件 /etc/services |
| netstat -s | 查看网络收发情况  UDP关注的另两个值是： RcvbufErrors: 0 SndbufErrors: 0 这两个值有计数，说明接收缓存和发送缓存不够用，导致数据包被丢弃。 |
| netstat –i | 通过该命令组合检测网络接口状况 |
| netstat –r | 通过该命令组合检测系统的路由表信息 |
| 查询端口使用的进程ID | 然后ls -lt /proc/pid/exe 查询具体程序的启动方式 |
| lsof -Pnl +M -i4 | 查看网络端口和连接情况  与netstat -anp类似 |
| netstat -a |grep -v 'unix\|udp'|awk '{print $6}'|sort|uniq -c |sort -rk 1 | 查看各状态连接数 |

#### 1.4.3 sar

网络流量sar -n DEV 5 2

每隔5秒，共输出2次，网络情况

rxbyt/s txbyt/s，接收和发送的数据

网络错误sar -n EDEV 5 2

网络连接数sar -n SOCK 5 2

tcpsck，totalsck 分别为TCP，和总连接数

#### 1.4.4 ss

需要root权限

ss -pt|grep 27408 查看指定进程的socket

ss –s 列出当前socket详细信息

ss -t -a 显示所有tcp socket

ss -u -a 显示所有的UDP Socekt

ss -l 显示本地打开的所有端口

#### 1.4.5 ping

查看网络是否连通

#### 1.4.6 telnet

telnet ip port

查看对应机器上，监听端口port的服务是否存在

#### 1.4.7 curl

通过代理访问外网命令

curl -x"172.23.28.199:8080" http://www.baidu.com

curl -H "Host:www.baidu.com" http://172.23.28.199:8080

#### 1.4.8 dmidecode

#### 1.4.9 dmesg和cat /var/log/messages

关注参数

Out of Memory

allocation failure

table full

time wait bucket table overflow

#### 1.4.10 lscpu

lscpu -p

#### 1.4.11 lspci

一个用来显示系统中所有PCI总线设备或连接到该总线上的所有设备的工具

#### 1.4.12 getconf -a

数据来源 /proc/net/dev

流量包量 /proc/net/snmp

TCP连接数，UDP发送和接收数据量

#### 1.4.13 监控机器流量和TCP连接数

while :; do Transmit\_bytes1=`cat /proc/net/dev|grep eth1|awk -F ' ' '{print $9 }'` ; receive\_bytes1=`cat /proc/net/dev|grep eth1|awk -F ' ' '{print $1 }'|awk -F ':' '{print$2}'` ; sleep 1;Transmit\_bytes2=`cat /proc/net/dev|grep eth1|awk -F ' ' '{print $9 }'`;receive\_bytes2=` cat /proc/net/dev|grep eth1|awk -F ' ' '{print $1 }'|awk -F ':' '{print$2}'`; currEsb=`cat /proc/net/snmp|grep -i "tcp"|tail -n 1|awk -F ' ' '{print $10}'`;echo "TransmitKbps/s: " $(($(($Transmit\_bytes2-$Transmit\_bytes1))\*8/1000 )) ;echo " receiveKbps: " $(($(($receive\_bytes2-$receive\_bytes1))\*8/1000));echo "currEsb: " $currEsb;echo '======================';sleep 5 ;done

### 1.5 系统

#### 1.5.1 查看位数

getconf LONG\_BIT

#### 1.5.2 查看支持线程版本

getconf GNU\_LIBPTHREAD\_VERSION

#### 1.5.3 查看系统版本

cat /proc/version

uname -a

#### 1.5.4 系统内核参数调优

cat /proc/sys/fs/file-max 最多打开文件数 655290

cat /proc/sys/kernel/pid\_max 系统最多进程数 32768

cat /proc/sys/kernel/threads-max 系统最多线程数 125889

共享内存

|  |  |
| --- | --- |
| cat /proc/sys/kernel/shmmax | 64000000000  单个共享内存断大小 |
| cat /proc/sys/kernel/shmmni | 4096  系统最大共享内存块数 |
| cat /proc/sys/kernel/shmall | 系统可使用的共享内存总量，非常大，与内存一样大  1152921504606846720  表示系统共享内存总大小，以页为单位。 一个32位的Linux系统，8G的内存，可以设置kernel.shmall = 2097152，即为： 2097152\*4k/1024/1024 = 8G就是说可用共享内存一共8G，这里的4K是32位操作系统一页的大小，即4096字节。 |
| ipcs -m/ipcrm -m shmid | 常用命令 |

消息队列

|  |  |
| --- | --- |
| cat /proc/sys/kernel/msgmax | 20480000  单个最大消息队列容量 |
| cat /proc/sys/kernel/msgmnb | 41943040  消息队列中的最大字节数 |
| cat /proc/sys/kernel/msgmni | 4096000  系统最大消息队列数 |
| ipcs -q | 常用命令 |

信号量:

cat /proc/sys/kernel/sem #2500 32000 320 1280

ipcs -s|head #常用命令

管道:

cat /proc/sys/fs/pipe-size-max #有问题

端口:

cat /proc/sys/net/ipv4/ip\_local\_port\_range #21000 61000 可使用的端口范围

#### 1.5.5 调优方法

将以下参数值加入/etc/sysctl.conf文件，然后执行命令sysctl -p，生效

|  |  |
| --- | --- |
| 参数值 | 具体含义 |
| net.ipv4.tcp\_max\_tw\_buckets = 6000 | 选项用来设定timewait的数量，默认是180000，这里设为6000。 |
| net.ipv4.ip\_local\_port\_range = 1024 65000 | 选项用来设定允许系统打开的端口范围。在高并发情况否则端口号会不够用。 |
| net.ipv4.tcp\_tw\_recycle = 1 | 选项用于设置启用timewait快速回收. |
| net.ipv4.tcp\_tw\_reuse = 1 | 选项用于设置开启重用，允许将TIME-WAIT sockets重新用于新的TCP连接。 |
| net.ipv4.tcp\_syncookies = 1 | 选项用于设置开启SYN Cookies，当出现SYN等待队列溢出时，启用cookies进行处理。 |
| net.core.somaxconn = 262144 | 选项的默认值是128， 这个参数用于调节系统同时发起的tcp连接数，在高并发的请求中，默认的值可能会导致链接超时或者重传，因此，需要结合并发请求数来调节此值。 |
| net.core.netdev\_max\_backlog = 262144 | 选项表示当每个网络接口接收数据包的速率比内核处理这些包的速率快时，允许发送到队列的数据包的最大数目 |
| net.ipv4.tcp\_max\_orphans = 262144 | 选项用于设定系统中最多有多少个TCP套接字不被关联到任何一个用户文件句柄上。如果超过这个数字，孤立连接将立即被复位并打印出警告信息。这个限制只是为了防止简单的DoS攻击。不能过分依靠这个限制甚至人为减小这个值，更多的情况下应该增加这个值。 |
| net.ipv4.tcp\_max\_syn\_backlog = 262144 | 选项用于记录那些尚未收到客户端确认信息的连接请求的最大值。对于有128MB内存的系统而言，此参数的默认值是1024，对小内存的系统则是128。 |
| net.ipv4.tcp\_synack\_retries = 1 | 参数的值决定了内核放弃连接之前发送SYN+ACK包的数量。 |
| net.ipv4.tcp\_syn\_retries = 1 | 选项表示在内核放弃建立连接之前发送SYN包的数量。 |
| net.ipv4.tcp\_fin\_timeout = 1 | 选项决定了套接字保持在FIN-WAIT-2状态的时间。默认值是60秒。正确设置这个值非常重要，有时即使一个负载很小的Web服务器，也会出现大量的死套接字而产生内存溢出的风险。 |
| net.ipv4.tcp\_keepalive\_time = 30 | 选项表示当keepalive启用的时候，TCP发送keepalive消息的频度。默认值是2（单位是小时）。 tcp\_orphan\_retries，FIN\_WAIT1状态的时间 |
| 其他 | 如果发送端要求关闭套接字，net.ipv4.tcp\_fin\_timeout选项决定了套接字保持在FIN-WAIT-2状态的时间。接收端可以出错并永远不关闭连接，甚至意外宕机。 |
| 其他 | net.ipv4.tcp\_fin\_timeout的默认值是60秒。需要注意的是，即使一个负载很小的Web服务器，也会出现因为大量的死套接字而产生内存溢出的风险。FIN-WAIT-2的危险性比FIN-WAIT-1要小，因为它最多只能消耗1.5KB的内存，但是其生存期长些。 |

### 1.6 监控

#### 1.6.1 watch

watch -n2 "xxxx"

while:;do xxxx;done

### 1.7 源码下载

## 2 调试工具

### 2.1 编译阶段

#### 2.1.1 nm

获取二进制文件里面包含的符号

|  |  |
| --- | --- |
| 参数 | 含义 |
| -C | 把C++函数签名转为可读形式 |
| -A | 列出符号名的时候同时显示来自于哪个文件 |
| -a | 列出所有符号（这将会把调试符号也列出来。 默认状态下调试符号不会被列出） |
| -l | 列出符号在源代码中对应的行号（指定这个参数后， nm将利用调试信息找出文件名以及符号的行号。 对于一个已定义符号，将会找出这个符号定义的行号， 对于未定义符号,显示为空） |
| -n | 根据符号的地址来排序（默认是按符号名称的字母顺序排序的） |
| -u | 只列出未定义符号 |

常用nm -CAln xxx

等价于objdump xxx -t -C|grep -P "\\*UND\\*"

#### 2.1.2 strings

获取二进制文件里面的字符串常量

-a不只是扫描目标文件初始化和装载段, 而是扫描整个文件。

-f在显示字符串之前先显示文件名。

常用于查看glibc支持的版本

strings main|grep GLIBC

#### 2.1.3 strip

可执行程序减肥（通常只在已经调试和测试过的生成模块上，因为不能调试了）

#### 2.1.4 c++filt

可以根据符号还原对应的函数调用名，命令空间，类名

#### 2.1.5 反编译，反跟踪

nm 程序可用于列举符号及其类型和值， 但是，要更仔细地研究目标文件中这些命名段的内容， 需要使用功能更强大的工具。 其中两种功能强大的工具是objdump和readelf。

#### 2.1.6 readelf

readelf工具使用来显示一个或多个ELF格式文件信息

|  |  |
| --- | --- |
| 选项 | 含义 |
| -a | 显示所有ELF文件的信息 |
| -h | 显示ELF文件的文件头 |
| -l | 显示程序头（program-header） 和程序段（segment）和段下面的节 |
| -S | 显示较为详细的节信息（section） |
| -s | 显示符号信息 |
| -r | 显示重定位信息(如果有) |
| -n | 显示标识信息（如果有） |
| -u | 显示展开函数信息（如果有） |
| -d | 显示动态节信息，一般是动态库的信息 |

常用 readelf -h xxx

#### 2.1.7 objdump

反汇编目标文件或者可执行文件的命令

objdump -S xx.o

|  |  |
| --- | --- |
| 选项 | 含义 |
| -C | 源码于反汇编出来的指令对照 |
| -f | 显示文件头信息 |
| -h | 显示Section Header |
| -x | 显示全部Header信息 |

#### 2.1.8 addr2line

根据地址查找代码行

注意点

1.该可执行程序用-g编译， 使之带调试信息。

2. 如果crash在一个so里面， 那addr2line不能直接给出代码行。

用法：

-a 在显示函数名或文件行号前显示地址

-b 指定二进制文件格式

-C 解析C++符号为用户级的名称,可指定解析样式

-e 指定二进制文件

-s 仅显示文件的基本名，而不是完整路径

-f 同时显示函数名称

-i 展开内联函数

-j 读取相对于指定节的偏移而不是绝对地址

-p 每个位置都在一行显示

eg：addr2line –e exe addr

如果core文件没有调试方法

1，dmesg | grep program\_name或者cat /var/log/dmesg，查询IP后面的地址

2，如果是段错误不在加载的so, addr2line -e ./test\_prog 0000000000400634 -f| 如果是在加载的so，找到偏移，ip减去spp\_module\_test.so后面的基址 即7f41e04eecc0-7f41e04eb000=3CC0， addr2line -e spp\_module\_test.so 3CC0

错误信息描述： qzhttp[32464]: segfault at ffffffff89000001 ip 00007f41e04eecc0 sp 00007fffa99da788 error 6 in spp\_module\_test.so[7f41e04eb000+5000]

### 2.2 运行阶段

#### 2.2.1 调试程序的常见步骤

1、确定运行时间主要花在用户态还是内核态 （比较土的一个方法：程序暂时屏蔽daemon()调用， hardcode收到n个请求后exit(0)，time一下程序……）。

2、如果是用户态，则使用gprof进行性能分析。

3、如果是内核态，则使用strace进行性能分析， 另外可以使用其他工具（比如ltrace等）辅助。

#### 2.2.2 ldd

显示程序需要使用的动态库和实际使用的动态库

ldd -r ./common\_async\_cgi.so

#### 2.2.3 strace

跟踪当前系统调用

|  |  |
| --- | --- |
| 选项 | 含义 |
| -p <pid> | attach到一个进程 |
| -c | 最后统计各个system call的调用情况 |
| -T | 打印system call的调用时间 |
| -o <file> | 将strace的输出定向到file中。 |
| -e expr | 指定一个表达式，用来控制如何跟踪 |
| -e open | 等价于-e trace=open，表示只跟踪open调用  使用 strace –e open ./prg 来看程序使用了哪些配置文件或日志文件，很方便。 |
| -e trace=<set> | 只跟踪指定的系统调用 |
| -e trace=open,close,rean,write | 表示只跟踪这四个系统调用 |
| -e trace=file | 只跟踪有关文件操作的系统调用 |
| -e trace=process | 只跟踪有关进程控制的系统调用 |
| -e trace=network | 跟踪与网络有关的所有系统调用 |
| -e strace=signal | 跟踪所有与系统信号有关的系统调用 |
| -e trace=ipc | 跟踪所有与进程通讯有关的系统调用 |

常用：strace -p 5157 -T -ttt -c -o strace.txt

strace -p 5157 -T -ttt -c -e trace=process

#### 2.2.4 ltrace

跟踪当前库函数

参数和strace很接近,基本一致

常用法：ltrace -p 5157 -T -ttt -c ltrace -p 5157 -T -ttt -c -o ltrace.txt

#### 2.2.5 time

查看程序执行时间、用户态时间、内核态时间

time只跟踪父进程，所以不能fork

#### 2.2.6 gprof

显示用户态各函数执行时间

使用步骤

1、使用 -pg 编译和链接应用程序

gcc -pg -o exec exec.c

如果需要库函数调用情况: gcc -lc\_p -gp -o exec exec.c

2、执行应用程序使之生成供gprof 分析的数据gmon.out

3、使用gprof 程序分析应用程序生成的数据

gprof exec gmon.out > profile.txt

注意：

程序必须通过正常途径退出（exit()、main返回），kill无效。 对后台常驻程序的调试——我的比较土方法是，屏蔽daemon()调用， 程序hardcode收到n个请求后exit(0)。

有时不太准。只管了用户态时间消耗，没有管内核态消耗。

#### 2.2.6 gdb

调试可执行程序，或者调试core文件

gdb <program> <core文件>

#### 2.2.7 oprofile

使用oprofile进行cpu使用情况检测， 需要经过初始化、启动检测、导出检测数据、 查看检测结果等步骤，以下为常用的oprofile命令。

#### 2.2.8 valgrind

检查内存错误

valgrind --tool=memcheck --leak-check=full ./spp\_worker ../etc/spp\_worker1.xml

#### 2.2.9 mtrace

mtrace是glibc內提供的工具，原理很简单， 就是把你程序中malloc()和free()的位置全部下來， 最后两辆配对，沒有配对到的就是memory leak。

#### 2.2.10 perf

常用

1，perf record -g -e cycles -p pid或者perf record -g -e cycles -a ./xxx（应用名）

2，perf report -g --stdio -i perf.data

perf stat/record /report

perf list

perf top -p pid

perf stat -p pid

perf record -g -e cycles -p pid

### 2.3 proc文件系统

内核的窗口

proc文件系统是一个伪文件系统， 它存在内存当中，而不占用外存空间。

用户和应用程序可以通过proc得到系统的信息， 并可以改变内核的某些参数。

## 3 程序

### 3.1 进程CPU

pidstat

数据来源/pro/$pid/stat

top -p $pid

### 3.2 pgrep查看进程ID

pgrep pidname

pidstat -p $(pidof memtest) -r 1

### 3.3 内存

常用命令

pidstat -p $pid -r 1

ps -o majflt,minflt -p pid

参数含义

minflt/s 缺页中断，需要分配物理内存

majflt/s，缺页中断，需要向磁盘读写数据

VSZ ，虚拟内存占有

RSS，实际占有物理内存

SHM共享内存

实际物理内存为RSS-SHM

可以根据VSZ,RSS的值是否一直上升来判断是否内存泄漏

watch -n2 'ps -o vsz,rss -p $pid'

### 3.4 进程磁盘I/O

pidstat -p $pid -d 1

参数含义

kB\_rd/s，读数据

kB\_wr/s，写数据

### 3.5 查看进程中的线程数

ls /proc/PID/task | wc -l

### 3.6 进程堆栈

cat /proc/$pid/maps

pmap -d $pid

### 3.7 lsof命令，可以组合使用

|  |  |
| --- | --- |
| 命令 | 含义 |
| lsof -u username | 列出某个用户打开的文件信息 |
| lsof -c mysql | 列出某个程序所打开的文件信息 |
| lsof -p pid | 通过某个进程号显示该进行打开的文件  cat /proc/pid/fd |
| lsof -i | 列出所有的网络连接 |
| lsof -i tcp | 列出所有tcp 网络连接信息 |
| lsof -i tcp:53101 | 特定的tcp端口 |
| lsof /filepath/file ，  lsof /dev/sda3|head | 查看谁正在使用某个文件或者文件系统 |
| lsof | 列出所有打开的文件: |
| lsof -u webadmin | 列出某个用户打开的文件信息 |

### 3.8 gstack pid

查询进程实时堆栈信息

watch -n2 "gstack pid"

### 3.9 strace

常用来跟踪进程执行时的系统调用和所接收的信号

strace -p 17510 -T –emsgrcv 查看进程对应的msgrcv操作

strace -p 17510 查看进程系统调用

strace -T -r -c -p 2302 统计系统调用

### 3.10内存泄漏诊断

valgrind

valgrind --tool=memcheck --log-file=test.txt --leak-check=full --show-reachable=yes ./pledgeseller

### 3.11 查看进程的运行堆栈信息

gstack thread-pid

### 3.12 生成core文件

gcore pid

kill -6 pid

### 3.13性能诊断

- Linux下的系统性能调优工具