# Linux 常见异常分析

Honest1y 高效运维

## 1 常用的 Load 分析方法

#### CPU高、Load高

通过 top 命令查找占用CPU最高的进程PID；

通过top -Hp PID查找占用CPU最高的线程TID;

对于java程序，使用jstack打印线程堆栈信息；

通过printf %x tid打印出最消耗CPU线程的十六进制；

#### CPU低、Load高

产生的原因一句话总结就是：等待磁盘I/O完成的进程过多，导致进程队列长度过大，但是CPU运行的进程却很少，这样就体现到负载过大了，cpu使用率低。

通过top命令查看CPU等待IO时间，即%wa；

通过iostat -d -x -m 1 10查看磁盘IO情况；(安装命令 yum install -y sysstat)

通过sar -n DEV 1 10查看网络IO情况；

通过如下命令查找占用IO的程序；

ps -e -L h o state,cmd | awk '{if($1=="R"||$1=="D"){print $0}}' | sort | uniq -c | sort -k 1nr

## 2 CPU高、Load高情况分析

使用vmstat 查看系统纬度的 CPU 负载；

使用 top 查看进程纬度的 CPU 负载；

##### 2.1 使用 vmstat 查看系统纬度的 CPU 负载

可以通过 vmstat 从系统维度查看 CPU 资源的使用情况

格式：vmstat -n 1 -n 1 表示结果一秒刷新一次

[root@VM-1-14-centos ~]# vmstat -n 1procs -----------memory---------- ---swap-- -----io---- -system-- ------cpu----- r b swpd free buff cache si so bi bo in cs us sy id wa st 1 0 0 250304 163472 2154300 0 0 1 16 0 4 1 0 98 0 0 0 0 0 250412 163472 2154332 0 0 0 0 937 1439 1 1 99 0 0 0 0 0 250428 163472 2154332 0 0 0 4 980 1329 0 0 100 0 0 0 0 0 250444 163472 2154332 0 0 0 0 854 1227 0 0 99 0 0 0 0 0 250444 163472 2154332 0 0 0 68 832 1284 0 1 99 1 0 0 0 0 250016 163472 2154332 0 0 0 0 929 1389 1 1 99 0 0

返回结果中的主要数据列说明：

**r**：表示系统中 CPU 等待处理的线程。由于 CPU 每次只能处理一个线程，所以，该数值越大，通常表示系统运行越慢。

**b**：表示阻塞的进程,这个不多说，进程阻塞，大家懂的。

**us**：用户CPU时间，我曾经在一个做加密解密很频繁的服务器上，可以看到us接近100，r运行队列达到80(机器在做压力测试，性能表现不佳)。

**sy**：系统CPU时间，如果太高，表示系统调用时间长，例如是IO操作频繁。

**wa**：IO 等待消耗的 CPU 时间百分比。该值较高时，说明 IO 等待比较严重，这可能磁盘大量作随机访问造成的，也可能是磁盘性能出现了瓶颈。

**id**：处于空闲状态的 CPU 时间百分比。如果该值持续为 0，同时 sy 是 us 的两倍，则通常说明系统则面临着 CPU 资源的短缺。

**常见问题及解决方法：**

如果 r 经常大于4，且id经常少于40，表示cpu的负荷很重。

如果pi，po长期不等于0，表示内存不足。

如果disk经常不等于0，且在b中的队列大于3，表示io性能不好。

##### 2.2 使用 top 查看进程纬度的 CPU 负载

可以通过 top 从进程纬度来查看其 CPU、内存等资源的使用情况。

top - 19:49:59 up 36 days, 23:15, 3 users, load average: 0.11, 0.04, 0.05Tasks: 133 total, 1 running, 131 sleeping, 0 stopped, 1 zombie%Cpu(s): 3.1 us, 3.1 sy, 0.0 ni, 93.8 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 stKiB Mem : 3880188 total, 241648 free, 1320424 used, 2318116 buff/cacheKiB Swap: 0 total, 0 free, 0 used. 2209356 avail Mem PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND 1793 mysql 20 0 1608796 236708 9840 S 6.7 6.1 83:36.23 /usr/sbin/mysqld 1 root 20 0 125636 3920 2444 S 0.0 0.1 4:34.13 /usr/lib/systemd/systemd 2 root 20 0 0 0 0 S 0.0 0.0 0:00.90 [kthreadd] 4 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 [kworker/0:0H] 6 root 20 0 0 0 0 S 0.0 0.0 0:15.46 [ksoftirqd/0] 7 root rt 0 0 0 0 S 0.0 0.0 0:12.02 [migration/0]

默认界面上第三行会显示当前 CPU 资源的总体使用情况，下方会显示各个进程的资源占用情况。

可以直接在界面输入大小字母 P，来使监控结果按 CPU 使用率倒序排列，进而定位系统中占用 CPU 较高的进程。最后，根据系统日志和程序自身相关日志，对相应进程做进一步排查分析，以判断其占用过高 CPU 的原因。

## 3 CPU低、Load高

**问题描述**

Linux 系统没有业务程序运行，通过 top 观察，类似如下图所示，CPU 很空闲，但是 load average 却非常高：

**问题分析**

CPU低而负载高也就是说等待磁盘I/O完成的进程过多，就会导致队列长度过大，这样就体现到负载过大了，但实际是此时CPU被分配去执行别的任务或空闲，具体场景有如下几种：

场景一：磁盘读写请求过多就会导致大量I/O等待

上面说过，cpu的工作效率要高于磁盘，而进程在cpu上面运行需要访问磁盘文件，这个时候cpu会向内核发起调用文件的请求，让内核去磁盘取文件，这个时候会切换到其他进程或者空闲，这个任务就会转换为不可中断睡眠状态。当这种读写请求过多就会导致不可中断睡眠状态的进程过多，从而导致负载高，cpu低的情况。

场景二：MySQL中存在没有索引的语句或存在死锁等情况

我们都知道MySQL的数据是存储在硬盘中，如果需要进行sql查询，需要先把数据从磁盘加载到内存中。当在数据特别大的时候，如果执行的sql语句没有索引，就会造成扫描表的行数过大导致I/O阻塞，或者是语句中存在死锁，也会造成I/O阻塞，从而导致不可中断睡眠进程过多，导致负载过大。具体解决方法可以在MySQL中运行show full processlist命令查看线程等待情况，把其中的语句拿出来进行优化。

场景三：外接硬盘故障，常见有挂了NFS，但是NFS server故障

比如我们的系统挂载了外接硬盘如NFS共享存储，经常会有大量的读写请求去访问NFS存储的文件，如果这个时候NFS Server故障，那么就会导致进程读写请求一直获取不到资源，从而进程一直是不可中断状态，造成负载很高。

**处理办法**

load average 是对 CPU 负载的评估，其值越高，说明其任务队列越长，处于等待执行的任务越多。

出现此种情况时，可能是由于僵死进程导致的。可以通过指令ps -axjf查看是否存在 D 状态进程。

D 状态是指不可中断的睡眠状态。该状态的进程无法被 kill，也无法自行退出。只能通过恢复其依赖的资源或者重启系统来解决。

等待 I/O 的进程通过处于 uninterruptible sleep 或 D 状态；通过给出这些信息我们就可以简单的查找出处在wait状态的进程。

ps -e -L h o state,cmd | awk '{if($1=="R"||$1=="D"){print $0}}' | sort | uniq -c | sort -k 1nr