链滴

运维高手第九课： 典型系统故障 - 快速排错操作系统问题进程

作者：chenteng

原文链接：https://ld246.com/article/1636556617975

来源网站：链滴

典型系统故障：快速排错操作系统问题进程

一、常见进程问题及其对业务的影响

下面这张表格里面列出的是问题进程常见的一些类型。这个表格里一共有 6 个类型，前面 4 个类型可总结为对资源使用过度，主要是 CPU、内存、 IO 以及对操作系统上文件句柄使用过度。这样的问题影响到业务服务的稳定运行，同时可能会因为资源的占用，造成操作系统上面其他服务进程出现问题。

进程问题	影响	产生原因举例
CPU资源使用过渡	影响业务服务稳定性 响应主机资源分配	1. 实际请求超过处理承载能力 2. 代码逻辑效率性差 3. 资源抢占
内存资源使用过渡	影响业务服务稳定性 响应主机资源分配	1. 实际请求超过处理承载能力 2. 代码对内存资源申请、销毁分配 3. 资源抢占
IO资源使用过渡	影响业务服务稳定性 响应主机资源分配	1. 实际请求超过处理承载能力 2. 代码逻辑效率性差 3. 资源抢占
进程占用过多描述符	影响业务服务稳定性 响应主机资源分配	1. 实际请求超过处理承载能力 2. 代码逻辑问题
僵尸状态进程问题	阻碍系统正常运行	1. 进程结束，父进程没有等待
进程不可中断的睡眠状态	业务服务不可用	1. 程序代码问题

造成问题的常见原因有很多：

1. 由于外部请求或外部访问，超过了自身进程所能够承载的负荷，导致系统的资源消耗过多；
2. 进程内部的问题，即进程的程序代码设计不合理，效率比较低，出现了资源分配不合理的地方，导

在自己的资源上消耗过多；

3. 部署不合理，比如我们在部署的时候，两个进程抢占同一份资源；

4. 安全性的问题，比如被攻击等。

接下来是第 2 类进程问题的类型：进程状态的问题。这里主要介绍两个我们运维会常见到的问题进程态，一个是僵尸状态（Z），另外一个是进程不可中断的睡眠状态（D），这些都会导致自己业务服出现问题。

这一类的问题通常是进程本身造成的，代码逻辑导致的情况居多。

以上就是我罗列的一些常见的进程问题类型，针对这些问题，接下来我罗列了一张表格，看看可以通哪些命令来分析进程问题。

二、进程分析命令

命令名称	命令说明	用例
top	实时显示系统中各个进程的资源占用状况	top、top –p pid
ps	显示瞬间行程的状态	ps –ef、ps -aux
strace	跟踪进程中的系统调用	strace -f -F -o ~/straceout.txt ls
lsof	列出某个程序进程所打开的文件信息	lsof -p 11968
free	用来查看系统可用内存	free –m、free -h
iostat	动态监视系统的磁盘操作活动	iostat 2 3
vmstat	实时动态监视操作系统的虚拟内存、进程、CPU活动	vmstat 5 5
ldd	用来查看程式运行所需的共享库	ldd test

1. top 命令可以做到实时显示系统中各个进程的资源占用状况，后面是它的使用用例，如 top 或 top p pid 对应的 pid 来展示某一个进程的使用状态。

2. ps 命令，它也用来展示进程的状态，不过它是显示瞬间进程的状态 。我们可以通过 ps –ef、ps -a x 参数把某一个时态的操作系统的进程状态全部获取出来。

3. strace 命令主要用来跟踪一个进程调用系统底层模块的过程，我们可以通过 strace+ 具体进程执的命令，去跟踪对应进程对系统的调用。

4. free 命令统计系统内存使用情况。

5. iostat 主要用来监视操作系统上，磁盘操作的活动情况。

6. vmstat 主要用来监控内存、进程、CPU 的活动状态。

7. ldd 用来监控一个进程在启动运行时所需要的一些共享库。后面有一个用例，比如我们这里执行 ldd test 这个命令，那么它就会查看 test 进程在启用时，需要调用到操作系统的哪些共享库。

以上就是我们在分析问题进程时可能会经常用到的一些命令。接下来我们来讲讲，刚介绍的这些问题程产生的原因是什么？以及我们具体应该通过什么样的方式去查看和分析这些问题进程？

三、问题进程排查方式

3.1、top命令查找进程使用 CPU 资源过度

首先对于查找进程 CPU 资源使用过度的场景，这里最常用到的就是 top 命令。top 命令可以实时展

某一个进程或系统上所有进程的资源使用情况。可以在 Linux 操作系统上先输入 top 命令，如果我们要去分析哪一个进程使用的资源更多，需要按对 CPU 的利用率由大到小进行排序，这时我们可以在行 top 命令后，同时按住键盘上的“shift+p”键，就可以使进程对 CPU 的资源使用率这一列，按照大到小来进行排序。

```
Tasks:     total,     running,     sleeping,     stopped,     zombie
%Cpu(s):      us,       sy,       ni,       id,       wa,      hi,      si,      st
KiB Mem :           total,          free,          used,          buff/cache
KiB Swap:           total,          free,          used.          avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU %MEM     TIME+ COMMAND
16303 root      20   0  676804  13264   1492 S   3.1  0.7 554:55.95 barad_agent
    1 root      20   0  125480   2712   1340 S   0.0  0.1  34:34.34 systemd
    2 root      20   0       0      0      0 S   0.0  0.0   0:00.85 kthreadd
    3 root      20   0       0      0      0 S   0.0  0.0   8:09.59 ksoftirqd/0
    5 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kworker/0:0H
    7 root      rt   0       0      0      0 S   0.0  0.0   0:00.00 migration/0
    8 root      20   0       0      0      0 S   0.0  0.0   0:00.00 rcu_bh
    9 root      20   0       0      0      0 S   0.0  0.0  12:10.42 rcu_sched
   10 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 lru-add-drain
   11 root      rt   0       0      0      0 S   0.0  0.0   0:48.44 watchdog/0
   13 root      20   0       0      0      0 S   0.0  0.0   0:00.00 kdevtmpfs
   14 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 netns
   15 root      20   0       0      0      0 S   0.0  0.0   0:02.36 khungtaskd
   16 root       0 -20       0      0      0 S   0.0  0.0   0:00.25 writeback
   17 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 kintegrityd
   18 root       0 -20       0      0      0 S   0.0  0.0   0:00.00 bioset
```


这样我们会更加直观地看到，在当前操作系统上，哪一个进程（command），使用的 CPU 是排名前的，我们可以定位到导致系统 CPU 占用率过高的进程。

3.2、PS命令获取CPU 资源排行

第 2 个是 ps 命令，它可以瞬间把操作系统的进程状态提取出来。通过 ps -aux --sort=-%cpu 这个数，我们同样也可以实现按照 CPU 的使用率，由大到小进行排序。跟 top 相比，ps 并不是实时去展的，而是提取出瞬间的进程信息。我们可以再加一个管道符过滤一下，也就是写成“ps -aux --sort=%cpu|head -n 10”，表示显示对系统 CPU 资源使用率排名 Top10 的进程。

```
[root@vipumi_com ~]# ps -aux --sort=-%cpu|head -n 10
USER       PID %CPU %MEM    VSZ   RSS TTY      STAT START   TIME COMMAND
root       374  0.9  2.0 343348 39500 ?        Sl   00:00  12:08 /usr/bin/python3 /opt/codes/Jtrac/manage.py runserver 127.0.0.1:1080
root     16303  0.2  0.7 676804 13264 ?        Sl    2019 554:57 barad_agent
root     12567  0.1  1.4 163736 27856 ?        Ssl  14:16   0:45 /usr/local/qcloud/YunJing/YDEyes/YDService
root         1  0.0  0.1 125480  2712 ?        Ss    2019  34:34 /usr/lib/systemd/systemd --switched-root --system --deserialize 22
root         2  0.0  0.0      0     0 ?        S     2019   0:00 [kthreadd]
```

可以看到这张图的样例是这样进行展示的，top 可以做实时分析，而 ps 的命令是瞬间展示，相对于 top 命令而言，ps 命令的优势在于它本身不会占用操作系统太多资源，而 top 由于实时提取计算操作统的进程信息，所以消耗性能情况会比使用 ps命令更多。所以如果我们只是想通过 ps 命令，快速地当前操作系统进程上面的进程占用率做一个排序，提取出来，而不想对操作系统有额外的消耗，就可通过 ps 命令来操作。

刚刚讲到的是对 CPU 的资源使用率的操作，接下来我会讲到对内存资源使用率的操作。

3.3、查找进程使用内存/IO资源过多

如果是通过 top 命令来分析进程对内存使用率情况，那么我们可以按住“shift+m”，这个时候它就以按照内存这一列的使用率进行排序，我们可以把操作系统上当前占用率更高的进程整体罗列出来。s 命令也是一样，只不过是在 sort 排序这一列里，改成了 -%MEM，按照这一列来进行由大到小的数排序。

```
[root@vipumi_com ~]# top

top - 22:46:44 up 150 days,  9:22,  2 users,  load average: 0.25, 0.19, 0.19
Tasks:      total,      running,      sleeping,      stopped,      zombie
%Cpu(s):       us,       sy,       ni,       id,       wa,       hi,       si,       st
KiB Mem :           total,           free,           used,           buff/cache
KiB Swap:           total,           free,           used.           avail Mem

  PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
16875 mysql     20   0 1154936 179756   4440 S   0.0   9.6   8:43.58 mysqld
 2963 root      20   0  744056  44680   8272 S   0.0   2.4  12:20.71 rsyslogd
  374 root      20   0  343348  39500   6376 S   1.5   2.1  12:30.94 python3
  366 root      20   0  268972  38852   6344 S   0.0   2.1   0:00.56 python3
12567 root      20   0  165560  29608   3148 S   0.0   1.6   0:49.55 YDService
 1313 root      20   0   55608  15888  15552 S   0.0   0.8  11:10.02 systemd-journal
16303 root      20   0  676804  13264   1492 S   0.0   0.7 555:03.99 barad_agent
 2945 root      20   0  573924  11328    164 S   0.0   0.6  16:34.83 tuned
 2492 polkitd   20   0  614440   9232    440 S   0.0   0.5   8:13.11 polkitd
16302 root      20   0  183856   9028   1544 S   0.0   0.5 106:27.46 barad_agent
21602 nginx     20   0  128660   8404    540 S   0.0   0.4   0:06.44 nginx
16297 root      20   0  157412   6784    244 S   0.0   0.4   2:06.50 barad_agent
```


以上就是排查进程对操作系统的内存资源使用过度的方式，接下来讲解是进程使用的磁盘 IO分析场，我们首先用 yum install sysstat 安装软件包，再在操作系统的 Terminal 终端上面执行 iostat 命。

```
[root@vipumi_com ~]# iostat  2 1
Linux 3.10.0-957.21.3.el7.x86_64 (vipumi_com)   2020年03月09日   _x86_64_        (1 CPU)

avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           3.35    0.00    1.00    0.10    0.00   95.55

Device:            tps    kB_read/s    kB_wrtn/s    kB_read    kB_wrtn
vda               3.10         7.04        20.79   91335980  269741996
scd0              0.00         0.00         0.00      39004          0
```


可以看到我在执行 iostat 命令后面加入两个数值的参数，分别是 2 和 1，2 表示刷新的频率，间隔周，1 表示总共的次数，我这里总共只执行了一次。

执行 iostat 的展示情况就是这样，我们重点需要关注的是磁盘设备（我这里是 vda 设备）的读写速，在上图中，前面是读（KB_read/s），后面是写（KB_wrtn/s）。

如果你觉得刚展示的这些数值还不够详细和形象的话，我们可以加入一个 -x 选项，就可以展示出更详细的对于磁盘 IO 的操作信息。比如 %util，这个值表示一秒钟有多少时间用于 IO 操作，可以反映 IOPS 的情况，并且非常形象地展示出它的百分比，从而能够清晰地看出当前磁盘的负荷状态。

```
Total DISK READ :       4.06 M/s | Total DISK WRITE :     122.03 M/s
Actual DISK READ:       4.40 M/s | Actual DISK WRITE:     125.50 M/s
  TID  PRIO  USER     DISK READ  DISK WRITE  SWAPIN      COMMAND
16525 be/4 root        0.00 B/s  121.90 M/s  0.00 % 63.29 % dd if=/dev/zero of=./test.img
 2977 be/4 root      137.12 K/s    3.92 K/s  0.00 % 48.66 % rsyslogd -n [rs:main Q:Reg]
16048 be/4 root        3.89 M/s    0.00 B/s  0.00 % 13.88 % python /usr/sbin/iotop
  374 be/4 root       11.75 K/s    0.00 B/s  0.00 % 12.75 % python3 /opt/codes/Jtrac/manage.py runserver 127.0.0.1:1080
20685 be/4 root        3.92 K/s    0.00 B/s  0.00 %  0.09 % [kworker/u2:1]
12986 be/4 root        3.92 K/s    0.00 B/s  0.00 %  0.00 % YDService
12571 be/4 root        3.92 K/s    0.00 B/s  0.00 %  0.00 % YDService
 1313 be/4 root        0.00 B/s  121.45 K/s  0.00 %  0.00 % systemd-journald
```


只用 iostat 命令还不够，有时候我们还会想更加具体地了解每一个进程使用 IO 的情况。这个时候我只需要去安装另外一个包：iotop。它能够展示每个进程对 IO 的使用。值得注意的是，如果你的系统O 操作非常频繁，这个命令可能会占用比较大的操作系统性能，所以你还是需要合理使用它。iotop 把每一个进程信息的 IO 使用率进行罗列，并展示出来。我们看到这里有一个展示的效果图示：

刚刚讲到的查找进程 IO 资源使用情况，接下来我将为你介绍进程占用文件句柄。

3.4、进程占用文件描述符问题

之前的课时里我们讲过文件句柄的相关知识，这个课时里我们主要补充，如何查看操作系统当前的文句柄以及它的每一个限制。

首先就是操作系统上允许所有进程打开文件句柄的总数限制，我们可以通过 cat 命令路径下的 file-ma ，查看状态信息，这个数值代表操作系统上所有进程允许打开的最大 fd 数量。

有时我们还需要去查看当前所有的进程已打开和允许打开 fd 数量，这个时候我们可以通过 cat 命令查看另外一个文件，叫作 file-nr，它会展示出操作系统当前的进程打开及允许打开的文件句柄。

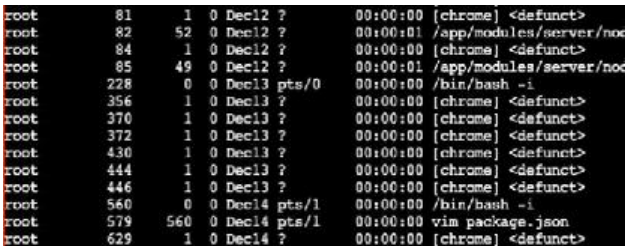
接下来，如果我想把范围精确到某一个进程允许打开的 fd 数量，因为操作系统除了对所有进程打开的d 文件有限制以外，对单个进程也会有限制。所以如果我们想了解单个进程打开文件句柄的限制，可通过插入 ulimit 命令，然后加 –n 参数，来展示单个进程允许打开的文件句柄数量。

刚讲到的是单个进程允许打开的文件句柄数量，同样我们想了解某一个进程当前打开了哪些 fd，我们以使用 ls -l，然后在 proc 目录下对应的进程 pid 目录，再到进程 ID（pid ）目录的 fd 目录下（/pro /{pid}/{fd}/），存放着当前打开的文件句柄，我们可以通过 ls -l 把所有的内容全部罗列出来，然后用 wc 命令进行统计，就能知道当前进程(PID)打开的 fd 数量。

如果进程它所打开的文件句柄过多，超过了操作系统的限制，就可能导致进程或服务出现影响和问题这个时候我们就需要去进行分析和调整了。关于调整操作系统的文件句柄限制，我们在课时 8 中学习何进行操作系统初始化的时候，给你讲到了如何去调整操作系统对文件句柄打开的设置，这里就不多了。如果要用第 2 种方式，也就是架构、程序优化的角度去解决的话，就要从程序自身排查，比如思它为什么会打开这么多文件句柄，以及打开的文件句柄是否合理。

通常来说，一些程序可能因为设计不合理，在本地创建了很多的临时碎片文件，这都有可能导致自己文件句柄过多，从而影响服务。

3.5、僵尸状态进程

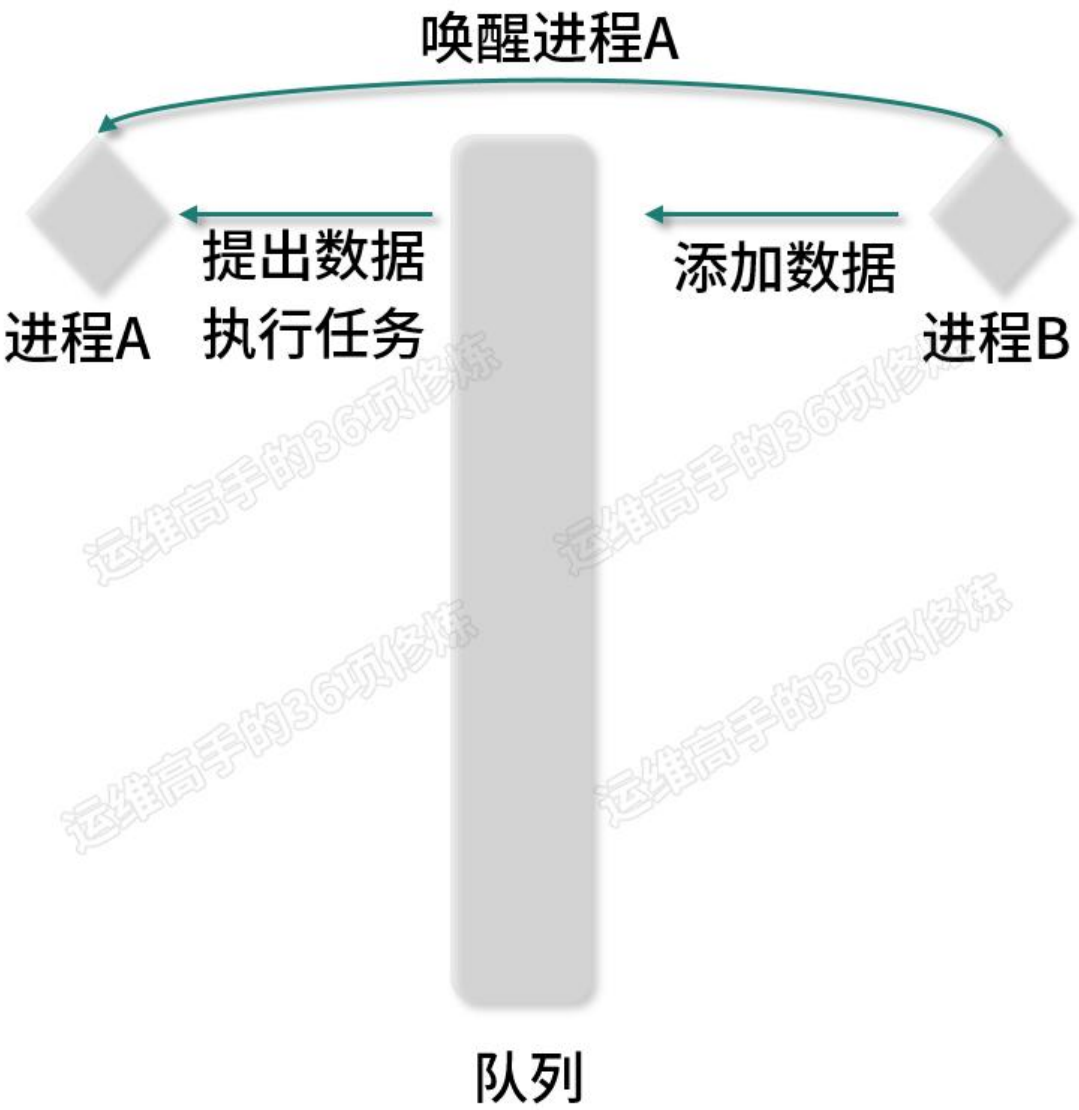


接下来我们讲下操作系统僵尸进程分析和处理，我们知道LInux的父子进程，任何一个子进程(init除外在exit()之后，并非马上就消失掉，而是留下一个称为僵尸进程(Zombie)的数据结构，等待父进程处。这是每个 子进程在结束时都要经过的阶段。如果子进程在exit()之后，父进程没有来得及处理，这用ps命令就能看到子进程的状态是“Z”

我们通过 ps -ef +管道符号（|），grep 一个“defunct”关键词，就会展示所有僵尸状态的进程。果是通过 top 命令，也可以查看当前操作系统上的异常状态进程个数。

通过"ps -e -o ppid,stat | grep Z | cut -d” ” -f2 | xargs kill -9 "这段组合代码用来进行处理僵尸程，原理是要把父进程的 pid 清理掉，才能够把僵尸进程回收。

3.6、进程不可中断睡眠状态



刚讲到的是僵尸进程状态，第 2 个异常进程状态是进程不可中断睡眠状态。在操作系统的睡眠状态分可中断睡眠状态和不可中断睡眠状态。可中断睡眠状态通常是以“S”表示，而不可中断睡眠状态通是以“D”关键字符进行表示。

我们首先来分析一下，不可中断睡眠状态产生的场景，这里我画了一张简单的图示：

我们会看到这里有两个进程，分别是进程 A 和进程 B，中间是一个队列，进程 B 负责添加数据，添完数据以后，它要唤醒进程 A 来提取数据，并且执行相应的任务。假设在 B 唤醒 A 的过程中，A 进正在处理上一次的任务，此时 A 就无法响应 B 的这次唤醒，这个时候就会导致进程 A 进入等待状态从而进入不可中断的睡眠状态。

可中断的程序设计过程是这样的，它会只等待某个条件为真，不论是产生硬件中断，或释放进程等待统资源，还是传递一个信号量，都可以作为唤醒进程的条件。而不可中断进程只能等待原有硬件终端需要的资源被唤醒，如果没有得到唤醒的话，那么它就不响应操作系统上的信号量。

所以如果我们想关闭不可中断进程状态的话，通过 kill -9 命令关闭是做不到的。这种情况下，只能通重启操作系统进行恢复或者所需资源。

再举一个真实的例子，我们常见到的客户端挂载 NFS 这种共享存储服务来给到客户端场景。假设把NS 服务端关闭之时，未先 umount 相关目录，在 NFS 客户端执行 df 命令，这个时候我们会看到在操系统的前端， df 命令会一直进入不可中断的状态，即使用 kill -9 也无法把 df 命令关闭。这时正确

处理方式是，需要先把 NFS 服务端的服务重新启用，才能够唤醒前端进程的响应，这个是我们常见的一种情况。