## 案例五：监控磁盘使用率

在我的运维职业生涯中，由于磁盘写满导致业务不正常的事故至少有5次，虽然对于磁盘的监控属于最基础的监控，但是很多时候往往因为运维的疏忽而忽略监控磁盘，最终导致事故发生。希望读到这篇文章的朋友，一定要把监控磁盘这件事重视起来。本案例的要求如下：

1）每分钟检测一次磁盘状况

2）当磁盘空间使用率或者inode使用率高于90%，需要发邮件告警，假设收件邮箱为admin@admin.com

3）统计使用率超过90%的分区所有子目录的大小，并把排名前3的子目录写到邮件内容中发给上面的邮箱

4）第一次告警后，如果我们没有及时处理，则需要每隔30分钟告警一次

5）每分钟脚本执行时，需检测该脚本是否已执行完，如果没有完成则本次不执行

知识点一：查看磁盘使用

命令 : df

df查看已挂载磁盘的总容量、使用容量、剩余容量等，可以不加任何参数，默认是按k为单位显示的，如下：

# df

文件系统 1K-块 已用 可用 已用% 挂载点

/dev/sda3 14347632 1490876 12127924 11% /

tmpfs 163308 0 163308 0% /dev/shm

/dev/sda1 99150 26808 67222 29% /boot

df命令常用选项有：-i, -h, -k, -m。-i选项查看inodes使用状况，如下所示：

# df -i

文件系统 Inode 已用(I) 可用(I) 已用(I)%% 挂载点

/dev/sda3 912128 66195 845933 8% /

tmpfs 40827 1 40826 1% /dev/shm

/dev/sda1 25688 38 25650 1% /boot

-h选项会使用合适的单位显示，例如G或者M：

# df -h

文件系统 容量 已用 可用 已用%% 挂载点

/dev/sda3 14G 1.5G 12G 11% /

tmpfs 160M 0 160M 0% /dev/shm

/dev/sda1 97M 27M 66M 29% /boot

-k、-m选项分别以K、M 为单位显示：</p>

# df -k

文件系统 1K-块 已用 可用 已用% 挂载点

/dev/sda3 14347632 1490880 12127920 11% /

tmpfs 163308 0 163308 0% /dev/shm

/dev/sda1 99150 26808 67222 29% /boot

# df -m

文件系统 M-块 已用 可用 已用% 挂载点

/dev/sda3 14012 1456 11844 11% /

tmpfs 160 0 160 0% /dev/shm

/dev/sda1 97 27 66 29% /boot

简单介绍一下各列所表示的含义，第一列是分区的名字，第二列为该分区总共的容量，第三列为已经使用了多少，第四列为还剩下多少，第五列为已经使用百分比，最后一列为挂载点。

知识点二：查看目录或文件大小

命令 : du<br/>du命令用来查看某个目录或文件所占空间大小。

语法 : du [-abckmsh] [文件或者目录名] 常用的参数有：

-a：全部文件与目录大小都列出来。如果不加任何选项和参数只列出目录（包含子目录）大小，例如：

# du dirb

4 dirb/dirc

12 dirb

# du -a dirb

4 dirb/filee

4 dirb/dirc

12 dirb

如果du不指定单位的话，默认显示单位为K。

-b：列出的值以bytes为单位输出。

-k：以KB为单位输出，和默认不加任何选项的输出值是一样的。

-m：以MB为单位输出

-h：系统自动调节单位，例如文件太小可能就几K，那么就以K为单位显示，如果大到几G，则就以G为单位显示。

# du -b /etc/passwd

1181 /etc/passwd

# du -k /etc/passwd

4 /etc/passwd

# du -m /etc/passwd

1 /etc/passwd

# du -h /etc/passwd

4.0K /etc/passwd

-c： 最后加总

# du -c dirb

4 dirb/dirc

12 dirb

12 总用量

# du dirb

4 dirb/dirc

12 dirb

-s：只列出总和

# du -s dirb

12 dirb

最常用的方式是du -sh filename。

结合find查找某个目录下所有子目录，并统计大小，命令为：

# find /dir/ -type d |sed '1d'|xargs du -sm

说明：用sed '1d'删除第一行，原因是第一行是/dir/目录本身，而我们要统计的是子目录。

知识点三：查看进程

在windows下我们查看进程在任务栏点右键，选择任务管理器就可以了非常方便。而在Linux下也有一些命令可以查看系统里的进程。

1）top

# top

top - 17:44:29 up 1 day, 17:17, 2 users, load average: 0.00, 0.01, 0.05

Tasks: 134 total, 1 running, 133 sleeping, 0 stopped, 0 zombie

%Cpu(s): 2.7 us, 16.2 sy, 0.0 ni, 81.1 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 2907940 total, 1103312 free, 266492 used, 1538136 buff/cache

KiB Swap: 2097148 total, 2096848 free, 300 used. 2337468 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

1 root 20 0 193604 6576 4008 S 5.6 0.2 0:08.58 systemd

43211 root 20 0 161984 2180 1528 R 5.6 0.1 0:00.04 top

2 root 20 0 0 0 0 S 0.0 0.0 0:00.35 kthreadd

3 root 20 0 0 0 0 S 0.0 0.0 0:12.25 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 0:01.23 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 0:19.25 rcu\_sched

10 root rt 0 0 0 0 S 0.0 0.0 0:02.11 watchdog/0

11 root rt 0 0 0 0 S 0.0 0.0 0:02.40 watchdog/1

12 root rt 0 0 0 0 S 0.0 0.0 0:01.83 migration/1

13 root 20 0 0 0 0 S 0.0 0.0 0:03.84 ksoftirqd/1

17 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 khelper

18 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kdevtmpfs

19 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 netns

20 root 20 0 0 0 0 S 0.0 0.0 0:00.11 khungtaskd

21 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 writeback

22 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kintegrityd

23 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 bioset

24 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kblockd

25 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 md

31 root 20 0 0 0 0 S 0.0 0.0 0:00.94 kswapd0

32 root 25 5 0 0 0 S 0.0 0.0 0:00.00 ksmd

33 root 39 19 0 0 0 S 0.0 0.0 0:01.43 khugepaged

34 root 20 0 0 0 0 S 0.0 0.0 0:00.00 fsnotify\_mark

35 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 crypto

43 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kthrotld

45 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kmpath\_rdacd

47 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kpsmoused

49 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 ipv6\_addrconf

68 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 deferwq

102 root 20 0 0 0 0 S 0.0 0.0 0:00.05 kauditd

282 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 ata\_sff

283 root 20 0 0 0 0 S 0.0 0.0 0:00.01 scsi\_eh\_0

这个命令用于动态监控进程所占系统资源，每隔3秒变一次。这个命令的特点是把占用系统资源（CPU，内存，磁盘IO等）最高的进程放到最前面。top命令打印出了很多信息，包括系统负载（loadaverage）、进程数（Tasks）、cpu使用情况、内存使用情况以及交换分区使用情况。其实上面这些内容可以通过其他命令来查看，所以用top重点查看的还是下面的进程使用系统资源详细状况。这部分东西反映的东西还是比较多的，不过需要你关注的也就是几项：RES, %CPU, %MEM, COMMAND。RES 这一项为进程所占内存大小，%CPU为进程使用CPU百分比，%MEM 为进程使用内存百分比，COMMAND为具体进程名字。

在 top 状态下，按"shift m"组合键, 可以按照内存使用大小排序。按数字'1'可以列出各颗CPU的使用状态。我经常用的一个命令为"top -bn1"，

它表示非动态打印系统资源使用情况，可以用在shell脚本中:

# top -bn1

top - 17:47:56 up 1 day, 17:21, 2 users, load average: 0.00, 0.01, 0.05

Tasks: 134 total, 1 running, 133 sleeping, 0 stopped, 0 zombie

%Cpu(s): 2.9 us, 5.7 sy, 0.0 ni, 91.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st

KiB Mem : 2907940 total, 1103712 free, 266052 used, 1538176 buff/cache

KiB Swap: 2097148 total, 2096848 free, 300 used. 2337956 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND

43254 root 20 0 161848 2080 1516 R 6.2 0.1 0:00.03 top

1 root 20 0 193604 6576 4008 S 0.0 0.2 0:08.59 systemd

2 root 20 0 0 0 0 S 0.0 0.0 0:00.35 kthreadd

3 root 20 0 0 0 0 S 0.0 0.0 0:12.25 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 0:01.23 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 0:19.39 rcu\_sched

10 root rt 0 0 0 0 S 0.0 0.0 0:02.11 watchdog/0

11 root rt 0 0 0 0 S 0.0 0.0 0:02.41 watchdog/1

12 root rt 0 0 0 0 S 0.0 0.0 0:01.83 migration/1

13 root 20 0 0 0 0 S 0.0 0.0 0:03.84 ksoftirqd/1

17 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 khelper

18 root 20 0 0 0 0 S 0.0 0.0 0:00.01 kdevtmpfs

19 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 netns

top还有一个-c选项，它可以显示具体的命令，也就是说在COMMAND这一列显示的内容会更加详细。

2）ps

ps命令用来汇报当前系统进程的状况，常见的用法有ps aux和ps -elf

# ps aux

USER PID %CPU %MEM VSZ RSS TTY STAT START TIME COMMAND

root 1 0.0 0.2 193604 6576 ? Ss 9月20 0:08 /usr/lib/systemd/systemd --system --deserialize 24

root 2 0.0 0.0 0 0 ? S 9月20 0:00 [kthreadd]

root 3 0.0 0.0 0 0 ? S 9月20 0:12 [ksoftirqd/0]

root 5 0.0 0.0 0 0 ? S< 9月20 0:00 [kworker/0:0H]

root 7 0.0 0.0 0 0 ? S 9月20 0:01 [migration/0]

root 8 0.0 0.0 0 0 ? S 9月20 0:00 [rcu\_bh]

root 9 0.0 0.0 0 0 ? S 9月20 0:19 [rcu\_sched]

root 10 0.0 0.0 0 0 ? S 9月20 0:02 [watchdog/0]

root 11 0.0 0.0 0 0 ? S 9月20 0:02 [watchdog/1]

root 12 0.0 0.0 0 0 ? S 9月20 0:01 [migration/1]

# ps -elf

F S UID PID PPID C PRI NI ADDR SZ WCHAN STIME TTY TIME CMD

4 S root 1 0 0 80 0 - 48401 ep\_pol 9月20 ? 00:00:08 /usr/lib/systemd/systemd --system --deserialize 24

1 S root 2 0 0 80 0 - 0 kthrea 9月20 ? 00:00:00 [kthreadd]

1 S root 3 2 0 80 0 - 0 smpboo 9月20 ? 00:00:12 [ksoftirqd/0]

1 S root 5 2 0 60 -20 - 0 worker 9月20 ? 00:00:00 [kworker/0:0H]

1 S root 7 2 0 -40 - - 0 smpboo 9月20 ? 00:00:01 [migration/0]

1 S root 8 2 0 80 0 - 0 rcu\_gp 9月20 ? 00:00:00 [rcu\_bh]

1 S root 9 2 0 80 0 - 0 rcu\_gp 9月20 ? 00:00:19 [rcu\_sched]

5 S root 10 2 0 -40 - - 0 smpboo 9月20 ? 00:00:02 [watchdog/0]

5 S root 11 2 0 -40 - - 0 smpboo 9月20 ? 00:00:02 [watchdog/1]

1 S root 12 2 0 -40 - - 0 smpboo 9月20 ? 00:00:01 [migration/1]

1 S root 13 2 0 80 0 - 0 smpboo 9月20 ? 00:00:03 [ksoftirqd/1]

1 S root 17 2 0 60 -20 - 0 rescue 9月20 ? 00:00:00 [khelper]

5 S root 18 2 0 80 0 - 0 devtmp 9月20 ? 00:00:00 [kdevtmpfs]

1 S root 19 2 0 60 -20 - 0 rescue 9月20 ? 00:00:00 [netns]

第一个用法中的STAT列或第二个用法中的S列表示进程的状态，进程状态分为以下几种：

D 不能中断的进程（通常为IO）

R 正在运行中的进程<br/>S 已经中断的进程，通常情况下，系统中大部分进程都是这个状态

T 已经停止或者暂停的进程，如果我们正在运行一个命令，按一下ctrl z组合键让他暂停，用ps查看就会显示T这个状态

W 表示为没有足够的内存页分配

X 已经死掉的进程（几乎从不出现）

Z 僵尸进程，杀不掉，打不死的垃圾进程，占系统一小点资源，不过没有关系。如果太多，就有问题了。一般不会出现。

< 高优先级进程

N 低优先级进程

L 在内存中被锁了内存分页

s 主进程

l 多线程进程

+代表在前台运行的进程

在本案例中，我们检查某个进程是否存在，可以这样做：

# ps aux |grep '进程名'

假如，本脚本的名字为mon\_disk.sh，则需要这样统计：

# ps aux |grep 'mon\_disk.sh' |grep -vE "$$|grep"

说明：这里的$$为本进程PID，之所以要排除掉它，是因为我们需要检查之前的旧进程而不是本次的进程，并且需要把grep这个进程也排除掉

知识点四：告警收敛思路分析

本案例中有要求，如果发生告警后，下次再告警应该是30分钟后。脚本本应该是一分钟执行一次，告警邮件也会一分钟发一次，如果我们不能在短时间内修复完问题，则会造成邮件骚扰。

这里我的思路是引入一个计数器，而且需要考虑以下几个场景：

1）脚本从来没有告警过，第一次告警

这种情况，不用考虑太多东西，直接发邮件，但需要做两件事情，第一需要记录此时的时间戳到一个临时文件中，第二需要建立一个临时文件记录告警次数，告警发生但不一定发邮件，要区分差异。

2）脚本之前告警过，距离上一次告警超过30分钟

判断距离上一次告警多久需要借助记录时间戳的临时文件，求本次告警时间戳和上一次告警时间戳的差值，是否大于1800。根据需求，只要大于30分钟就可以立即发邮件，同时记录时间戳和告警次数到两个不同的临时文件中。

3）脚本之前告警过，距离上一次告警不超过30分钟

不超过30分钟则需要查看记录告警次数的临时文件，只有告警次数大于等于30才会再一次发邮件。

本案例参考脚本

结合以上相关的知识点，最终的参考脚本如下：

#!/bin/bash

##监控磁盘使用情况，并做告警收敛（30分钟发一次邮件）

##作者：

##日期：2018-09-25

##版本：v0.1

#把脚本名字存入变量s\_name

s\_name=`echo $0|awk -F '/' '{print $NF}'`

#定义收件人邮箱

mail\_user=admin@admin.com

#定义检查磁盘空间使用率函数

chk\_sp()

{

df -m|sed '1d' |awk -F '%| +' '$5>90 {print $7,$5}'>/tmp/chk\_sp.log

n=`wc -l /tmp/chk\_sp.log|awk '{print $1}'`

if [ $n -gt 0 ]

then

tag=1

for d in `awk '{print $1}' /tmp/chk\_sp.log`

do

find $d -type d |sed '1d'|xargs du -sm |sort -nr |head -3

done > /tmp/most\_sp.txt

fi

}

#定义检查inode使用率函数

chk\_in()

{

df -i|sed '1d'|awk -F '%| +' '$5>90 {print $7,$5}'>/tmp/chk\_in.log

n=`wc -l /tmp/chk\_in.log|awk '{print $1}'`

if [ $n -gt 0 ]

then

tag=2

fi

}

#定义告警函数（这里的mail.py是案例二中那个脚本）

m\_mail() {

log=$1

t\_s=`date +%s`

t\_s2=`date -d "1 hours ago" +%s`

if [ ! -f /tmp/$log ]

then

#创建$log文件

touch /tmp/$log

#增加a权限，只允许追加内容，不允许更改或删除

chattr +a /tmp/$log

#第一次告警，可以直接写入1小时以前的时间戳

echo $t\_s2 >> /tmp/$log

fi

#无论$log文件是否是刚刚创建，都需要查看最后一行的时间戳

t\_s2=`tail -1 /tmp/$log|awk '{print $1}'`

#取出最后一行即上次告警的时间戳后，立即写入当前的时间戳

echo $t\_s>>/tmp/$log

#取两次时间戳差值

v=$[$t\_s-$t\_s2]

#如果差值超过1800，立即发邮件

if [ $v -gt 1800 ]

then

#发邮件，其中$2为mail函数的第二个参数，这里为一个文件

python mail.py $mail\_user "磁盘使用率超过90%" "`cat $2`" 2>/dev/null

#定义计数器临时文件，并写入0

echo "0" > /tmp/$log.count

else

#如果计数器临时文件不存在，需要创建并写入0

if [ ! -f /tmp/$log.count ]

then

echo "0" > /tmp/$log.count

fi

nu=`cat /tmp/$log.count`

#30分钟内每发生1次告警，计数器加1

nu2=$[$nu+1]

echo $nu2>/tmp/$log.count

#当告警次数超过30次，需要再次发邮件

if [ $nu2 -gt 30 ]

then

python mail.py $mail\_user "磁盘使用率超过90%持续30分钟了" "`cat $2`" 2>/dev/null

#第二次告警后，将计数器再次从0开始

echo "0" > /tmp/$log.count

fi

fi

}

#把进程情况存入临时文件，如果加管道求行数会有问题

ps aux |grep "$s\_name" |grep -vE "$$|grep">/tmp/ps.tmp

p\_n=`wc -l /tmp/ps.tmp|awk '{print $1}'`

#当进程数大于0，则说明上次的脚本还未执行完

if [ $p\_n -gt 0 ]

then

exit

fi

chk\_sp

chk\_in

if [ $tag == 1 ]

then

m\_mail chk\_sp /tmp/most\_sp.txt

elif [ $tag == 2 ]

then

m\_mail chk\_in /tmp/chk\_in.log

fi