Catalyst 3850系列交换机高CPU使用率故障排除

目录

简介

本文档介绍如何排除新的Cisco IOS® XE平台上CPU使用问题(主要由于中断)的故障。

背景信息

了解Cisco IOS® XE的构建方式很重要。借助Cisco IOS® XE，Cisco已迁移到Linux内核，并且所有子系统都已分解为多个进程。之前在Cisco IOS内部的所有子系统(例如模块驱动程序、高可用性(HA)等)现在都作为Linux操作系统(OS)内的软件进程运行。Cisco IOS本身作为Linux OS(IOSd)中的守护程序运行。Cisco IOS® XE不仅保留传统Cisco IOS®的外观和感觉，而且保留其操作、支持和管理功能。

此外，本文档还介绍了此平台上用于解决CPU使用问题的几个完整的新命令。

以下提供了一些有用的术语定义：

- 转发引擎驱动程序(FED)：这是Cisco Catalyst 3850系列交换机的核心，负责所有硬件编程/转发。
- Cisco IOSd：这是在Linux内核上运行的Cisco IOS®后台程序。作为内核中的软件进程运行。
- 数据包传输系统(PDS)：这是将数据包传输到各个子系统和从各个子系统传输数据包的体系结构和过程。例如，PDS 可控制如何在 FED 和 IOSd 之间传输数据包。
- 句柄：可将句柄视为指针。用于发现设备生成的输出中所使用的特定变量的更详细信息。句柄与 Cisco Catalyst 6500 系列交换机上本地目标逻辑 (LTL) 索引的概念类似。

案例研究：地址解析协议(ARP)中断

此部分所述的故障排除和验证流程可广泛用于因中断而导致的高 CPU 使用率情形。

第1步：确定消耗CPU周期的进程

show process cpu 命令可自发显示 CPU 的当前情况。请注意，Cisco Catalyst 3850系列交换机使用四个内核，并且您会看到所有四个内核的CPU使用率：

<#root>

3850-2#

show processes cpu sorted | exclude 0.0

```
Core 0: CPU utilization for five seconds: 53%; one minute: 39%;  five minutes: 41%
Core 1: CPU utilization for five seconds: 43%; one minute: 57%;  five minutes: 54%
Core 2: CPU utilization for five seconds: 95%; one minute: 60%;  five minutes: 58%
Core 3: CPU utilization for five seconds: 32%; one minute: 31%;  five minutes: 29%
PID    Runtime(ms) Invoked  uSecs  5Sec     1Min     5Min     TTY   Process
8525   472560      2345554  7525   31.37    30.84    30.83    0     iosd
5661   2157452     9234031  698    13.17    12.56    12.54    1088  fed
6206   19630       74895    262    1.83     0.43     0.10     0     eicored
6197   725760      11967089 60     1.41     1.38     1.47     0     pdsd
```

从输出中可以清楚地看到，Cisco IOS®后台守护程序与FED一起占用了CPU的主要部分，而FED是此设备的核心。当CPU使用率因中断而过高时，您会看到Cisco IOSd和FED占用CPU的主要部分，并且这些子进程（或这些子进程的子集）使用CPU:

- FED Punject 发送
- FED Punject 接收
- FED Punject 补给
- FED Punject 发送完成

可以使用 show process cpu detailed<process> 命令详细了解其中任何一个进程。由于Cisco IOSd负责大部分CPU使用，下面我们来详细了解一下这一点。

<#root>

3850-2#

show processes cpu detailed process iosd sorted | ex 0.0

```
Core 0: CPU utilization for five seconds: 36%; one minute: 39%; five minutes: 40%
Core 1: CPU utilization for five seconds: 73%; one minute: 52%; five minutes: 53%
Core 2: CPU utilization for five seconds: 22%; one minute: 56%; five minutes: 58%
Core 3: CPU utilization for five seconds: 46%; one minute: 40%; five minutes: 31%
PID    T C  TID   Runtime(ms)Invoked uSecs 5Sec   1Min   5Min  TTY  Process
                                          (%)    (%)    (%)
8525   L          556160     2356540 7526  30.42   30.77  30.83  0   iosd
8525   L 1  8525  712558     284117  0     23.14   23.33  23.38  0   iosd
59     I          1115452    4168181 0     42.22   39.55  39.33  0   ARP Snoop
198    I          3442960    4168186 0     25.33   24.22  24.77  0   IP Host Track Proce
```

```
30     I          3802130    4168183 0     24.66   27.88  27.66  0   ARP Input
283    I          574800     3225649 0     4.33    4.00   4.11   0   DAI Packet Process

3850-2#
```

show processes cpu detailed process fed sorted | ex 0.0

```
Core 0: CPU utilization for five seconds: 45%; one minute: 44%; five minutes: 44%
Core 1: CPU utilization for five seconds: 38%; one minute: 44%; five minutes: 45%
Core 2: CPU utilization for five seconds: 42%; one minute: 41%; five minutes: 40%
Core 3: CPU utilization for five seconds: 32%; one minute: 30%; five minutes: 31%
PID    T C  TID   Runtime(ms)Invoked uSecs 5Sec   1Min   5Min   TTY  Process
                                          (%)    (%)    (%)
5638   L          612840     1143306 536   13.22  12.90  12.93  1088  fed
5638   L 3  8998  396500     602433  0     9.87   9.63   9.61   0     PunjectTx
5638   L 3  8997  159890     66051   0     2.70   2.70   2.74   0     PunjectRx
```

输出（Cisco IOSd CPU输出）显示ARP监听、IP主机跟踪进程和ARP输入都很高。CPU 因 ARP 数据包而中断时，通常会出现此情况。

第2步：确定导致高CPU使用率情况的CPU队列

Cisco Catalyst 3850 系列交换机有许多队列，可满足不同类型数据包的需求（FED 维护 32 个直接进入 CPU 的接收队列）。监控这些队列非常重要，以便发现哪些数据包被传送到CPU以及哪些数据包由Cisco IOSd处理。这些队列基于 ASIC。

注：有两个ASIC:0和1。端口1至24属于ASIC 0。

要查看队列，请输入 show platform punt statistics port-asic <port-asic>cpuq <queue> direction 命令。

在 show platform punt statistics port-asic 0 cpuq -1 direction rx 命令中， -1 参数用于列出所有队列。因此，此命令可用于列出 Port-ASIC 0 的所有接收队列。

现在，您必须确定以高速率推送大量数据包的队列。在此例中，经检查队列之后，发现造成此问题的原因如下：

```
<snip>
RX (ASIC2CPU) Stats (asic 0 qn 16 lqn 16):
RXQ 16: CPU_Q_PROTO_SNOOPING
---------------------------------------
Packets received from ASIC     : 79099152
Send to IOSd total attempts    : 79099152
Send to IOSd failed count      : 1240331
RX suspend count               : 1240331
RX unsuspend count             : 1240330
RX unsuspend send count        : 1240330
RX unsuspend send failed count : 0
RX dropped count               : 0
RX conversion failure dropped  : 0
RX pkt_hdr allocation failure  : 0
```

```
RX INTACK count                : 0
RX packets dq'd after intack   : 0
Active RxQ event               : 9906280
RX spurious interrupt          : 0
<snip>
```

队列编号为 16，队列名称为 CPU_Q_PROTO_SNOOPING。

要发现问题队列，另一种方法是输入 show platform punt client 命令。

<#root>

3850-2#

show platform punt client

```
  tag      buffer        jumbo    fallback     packets   received   failures
                                              alloc   free  bytes    conv  buf
 27       0/1024/2048     0/5       0/5         0     0          0     0     0
 65536    0/1024/1600     0/0       0/512       0     0          0     0     0
 65537    0/ 512/1600     0/0       0/512    1530  1530     244061     0     0
 65538    0/   5/5        0/0       0/5         0     0          0     0     0
 65539    0/2048/1600     0/16      0/512       0     0          0     0     0
 65540    0/ 128/1600     0/8       0/0         0     0          0     0     0
 65541    0/ 128/1600     0/16      0/32        0     0          0     0     0
 65542    0/ 768/1600     0/4       0/0         0     0          0     0     0
 65544    0/  96/1600     0/4       0/0         0     0          0     0     0
 65545    0/  96/1600     0/8       0/32        0     0          0     0     0
 65546    0/ 512/1600     0/32      0/512       0     0          0     0     0
 65547    0/  96/1600     0/8       0/32        0     0          0     0     0
 65548    0/ 512/1600     0/32      0/256       0     0          0     0     0
 65551    0/ 512/1600     0/0       0/256       0     0          0     0     0
 65556    0/  16/1600     0/4       0/0         0     0          0     0     0
 65557    0/  16/1600     0/4       0/0         0     0          0     0     0
 65558    0/  16/1600     0/4       0/0         0     0          0     0     0
 65559    0/  16/1600     0/4       0/0         0     0          0     0     0
 65560    0/  16/1600     0/4       0/0         0     0          0     0     0
s65561  421/ 512/1600     0/0       0/128  79565859 131644697  478984244     0 37467
 65563    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65564    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65565    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65566    0/ 512/1600     0/16      0/256       0     0          0     0     0
 65581    0/   1/1        0/0       0/0         0     0          0     0     0
 131071   0/  96/1600     0/4       0/0         0     0          0     0     0
fallback pool: 98/1500/1600
jumbo pool:    0/128/9300
```

确定已为其分配最多数据包的标签。在此例中，此标签为 65561。

然后，输入以下命令：

<#root>

```
3850-2#
```

show pds tag all | in Active|Tags|65561

```
   Active   Client Client
     Tags   Handle Name                  TDA       SDA         FDA   TBufD       TBytD
    65561  7296672 Punt Rx Proto Snoop  79821397  79821397    0  79821397   494316524
```

此输出显示队列为Rx Proto Snoop。

在 show platform punt client 命令的输出中，65561 之前的 s 表示 FED 句柄被挂起并被大量传入数据包所淹没。如果 s 没有消失，则表示队列一直被卡住。

第3步：转储发送到CPU的数据包

在 show pds tag all 命令的结果中，请注意 Punt Rx Proto Snoop 旁边所报告的句柄 7296672。

在 show pds client <handle> packet last sink 命令中使用此句柄。请注意，必须先启用 debug pds pktbuf-last 命令，然后才能使用该命令；否则，您会遇到以下错误：

<#root>

```
3850-2#
```

show pds client 7296672 packet last sink

```
% switch-2:pdsd:This command works in debug mode only. Enable debug using
"debug pds pktbuf-last" command
```

启用调试后，输出结果如下：

<#root>

```
3850-2#
```

show pds client 7296672 packet last sink

```
Dumping Packet(54528) # 0 of Length 60
-----------------------------------------
Meta-data
0000 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0010 00 00 16 1d 00 00 00 00 00 00 00 00 55 5a 57 f0  ............UZW.
0020 00 00 00 00 fd 01 10 df 00 5b 70 00 00 10 43 00  .........[p...C.
0030 00 10 43 00 00 41 fd 00 00 41 fd 00 00 00 00 00  ..C..A...A......
0040 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0050 00 00 00 3c 00 00 00 00 00 01 00 19 00 00 00 00  ...<............
0060 01 01 b6 80 00 00 00 4f 00 00 00 00 00 00 00 00  .......O........
0070 01 04 d8 80 00 00 00 33 00 00 00 00 00 00 00 00  .......3........
0080 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
0090 00 01 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
00a0 00 00 00 00 00 00 00 00 02 00 00 00 00 00 00 00  ................
```

```
Data
0000 ff ff ff ff ff ff aa bb cc dd 00 00 08 06 00 01  ................
0010 08 00 06 04 00 01 aa bb cc dd 00 00 c0 a8 01 0a  ................
0020 ff ff ff ff ff ff c0 a8 01 14 00 01 02 03 04 05  ................
0030 06 07 08 09 0a 0b 0c 0d 0e 0f 10 11              ............
```

此命令会转储接收方收到的最后一个数据包，在本例中为Cisco IOSd。这表明此命令转储了报头，并且可以使用基于终端的 Wireshark (TShark) 进行解码。Meta-data 供系统内部使用，Data 输出则提供实际的数据包信息。但是，Meta-data 仍然非常有用。

请注意以 0070 开头的行。使用其后面的前 16 位，如下所示：

<#root>

3850-2#

show platform port-asic ifm iif-id 0x0104d88000000033

```
Interface Table
Interface IIF-ID        : 0x0104d88000000033
Interface Name          : Gi2/0/20
Interface Block Pointer : 0x514d2f70
Interface State         : READY
Interface Stauts        : IFM-ADD-RCVD, FFM-ADD-RCVD
Interface Ref-Cnt       : 6
Interface Epoch         : 0
Interface Type          : ETHER
        Port Type         : SWITCH PORT
        Port Location     : LOCAL
Slot              : 2
        Unit              : 20
        Slot Unit         : 20
        Acitve            : Y
        SNMP IF Index     : 22
        GPN               : 84
        EC Channel        : 0
        EC Index          : 0
        ASIC

              :

0
        ASIC Port         : 14
        Port LE Handle    : 0x514cd990
Non Zero Feature Ref Counts
        FID : 48(AL_FID_L2_PM), Ref Count : 1
        FID : 77(AL_FID_STATS), Ref Count : 1
        FID : 51(AL_FID_L2_MATM), Ref Count : 1
        FID : 13(AL_FID_SC), Ref Count : 1
        FID : 26(AL_FID_QOS), Ref Count : 1
Sub block information
        FID : 48(AL_FID_L2_PM), Private Data &colon; 0x54072618
        FID : 26(AL_FID_QOS), Private Data &colon; 0x514d31b8
```

其中已标识问题接口。Gig2/0/20 中存在一个流量生成器，用于泵送 ARP 流量。如果将其关闭，则

可以解决问题并最大限度地降低 CPU 使用率。

第4步：使用FED跟踪

在上一节中讨论的方法的唯一缺点是它只转储进入接收器的最后一个数据包，而不可能是罪魁祸首。

可以采用更好的方法排查此问题，即使用名为 FED 跟踪的功能。跟踪是一种由 FED 推送到 CPU 的数据包捕获方法（使用各种过滤器）。FED 跟踪不如 Cisco Catalyst 6500 系列交换机上的 Netdr 功能简单，

其中包含以下步骤：

1. 启用详情跟踪。默认情况下，事件跟踪处于开启状态。必须启用详情跟踪才能捕获实际数据包：

```
<#root>

3850-2#

set trace control fed-punject-detail enable
```

2. 微调捕获缓冲区。确定用于详情跟踪的缓冲区深度，并根据需要增加此深度。

```
<#root>

3850-2#

show mgmt-infra trace settings fed-punject-detail


One shot Trace Settings:

Buffer Name: fed-punject-detail
Default Size: 32768
Current Size: 32768
Traces Dropped due to internal error: No
Total Entries Written: 0
One shot mode: No
One shot and full: No
Disabled: False
```

可以使用以下命令更改缓冲区大小：

```
<#root>
```

```
3850-2#
set trace control fed-punject-detail buffer-size
```

可用的值如下：

```
<#root>
3850-2#
set trace control fed-punject-detail buffer-size ?

  <8192-67108864>  The new desired buffer size, in bytes
  default          Reset trace buffer size to default
```

3. 添加捕获过滤器。您现在需要添加各种捕获过滤器。添加不同的过滤器之后，可以选择匹配所有或匹配任意要捕获的那些数据包。

使用以下命令添加过滤器：

```
<#root>
3850-2#
set trace fed-punject-detail direction rx filter_add
```

当前可用选项如下所示：

```
<#root>
3850-2#
set trace fed-punject-detail direction rx filter_add ?

  cpu-queue  rxq 0..31
  field      field
  offset     offset
```

您必须把各种因素关联起来，还记得在此故障排除流程第 2 步中确定的问题队列吗？由于队列 16 将大量数据包推送到 CPU，因此适宜跟踪此队列并查看其向 CPU 推送的数据包。

您可以选择使用以下命令跟踪任何队列：

```
<#root>

 3850-2#

set trace fed-punject-detail direction rx filter_add cpu-queue
```

此例中的命令如下所示：

```
<#root>

3850-2#

set trace fed-punject-detail direction rx filter_add cpu-queue 16 16
```

您必须为过滤器选择匹配所有或匹配任意，然后启用跟踪：

```
<#root>

3850-2#

set trace fed-punject-detail direction rx match_all

3850-2#

set trace fed-punject-detail direction rx filter_enable
```

4. 显示已过滤的数据包。您可以使用 show mgmt-infra trace messages fed-punject-detail 命令显示捕获的数据包。

<#root>

3850-2#

show mgmt-infra trace messages fed-punject-detail

```
[11/25/13 07:05:53.814 UTC 2eb0c9 5661]
00 00 00 00 00 4e 00 40 07 00 02 08 00 00 51 3b
00 00 00 00 00 01 00 00 03 00 00 00 00 00 00 01
00 00 00 00 20 00 00 0e 00 00 00 00 00 01 00 74
00 00 00 04 00 54 41 02 00 00 00 00 00 00 00 00

[11/25/13 07:05:53.814 UTC 2eb0ca 5661]
ff ff ff ff ff ff aa bb cc dd 00 00 08 06 00 01
08 00 06 04 00 01 aa bb cc dd 00 00 c0 a8 01 0a
ff ff ff ff ff ff c0 a8 01 14 00 01 02 03 04 05
06 07 08 09 0a 0b 0c 0d 0e 0f 10 11 f6 b9 10 32
[11/25/13 07:05:53.814 UTC 2eb0cb 5661] Frame descriptors:
[11/25/13 07:05:53.814 UTC 2eb0cc 5661]
=========
fdFormat=0x4    systemTtl=0xe
loadBalHash1=0x8        loadBalHash2=0x8
spanSessionMap=0x0      forwardingMode=0x0
destModIndex=0x0        skipIdIndex=0x4
srcGpn=0x54     qosLabel=0x41
srcCos=0x0      ingressTranslatedVlan=0x3
bpdu=0x0        spanHistory=0x0
sgt=0x0 fpeFirstHeaderType=0x0
srcVlan=0x1     rcpServiceId=0x2
wccpSkip=0x0    srcPortLeIndex=0xe
cryptoProtocol=0x0      debugTagId=0x0
vrfId=0x0       saIndex=0x0
pendingAfdLabel=0x0     destClient=0x1
appId=0x0       finalStationIndex=0x74
decryptSuccess=0x0      encryptSuccess=0x0
rcpMiscResults=0x0      stackedFdPresent=0x0
spanDirection=0x0       egressRedirect=0x0
redirectIndex=0x0       exceptionLabel=0x0
destGpn=0x0     inlineFd=0x0
suppressRefPtrUpdate=0x0        suppressRewriteSideEfects=0x0
cmi2=0x0        currentRi=0x1
currentDi=0x513b        dropIpUnreachable=0x0
srcZoneId=0x0   srcAsicId=0x0
originalDi=0x0  originalRi=0x0
srcL3IfIndex=0x2        dstL3IfIndex=0x0
dstVlan=0x0     frameLength=0x40
fdCrc=0x7       tunnelSpokeId=0x0

=========
[11/25/13 07:05:53.814 UTC 2eb0cd 5661]
[11/25/13 07:05:53.814 UTC 2eb0ce 5661] PUNT PATH (fed_punject_rx_process_packet:
830):RX: Q: 16, Tag: 65561
[11/25/13 07:05:53.814 UTC 2eb0cf 5661] PUNT PATH (fed_punject_get_physical_iif:
```

```
579):RX: Physical IIF-id 0x104d88000000033
[11/25/13 07:05:53.814 UTC 2eb0d0 5661] PUNT PATH (fed_punject_get_src_l3if_index:
434):RX: L3 IIF-id 0x101b6800000004f
[11/25/13 07:05:53.814 UTC 2eb0d1 5661] PUNT PATH (fed_punject_fd_2_pds_md:478):
RX: l2_logical_if = 0x0
[11/25/13 07:05:53.814 UTC 2eb0d2 5661] PUNT PATH (fed_punject_get_source_cos:638):
RX: Source Cos 0
[11/25/13 07:05:53.814 UTC 2eb0d3 5661] PUNT PATH (fed_punject_get_vrf_id:653):
RX: VRF-id 0
[11/25/13 07:05:53.814 UTC 2eb0d4 5661] PUNT PATH (fed_punject_get_src_zoneid:667):
RX: Zone-id 0
[11/25/13 07:05:53.814 UTC 2eb0d5 5661] PUNT PATH (fed_punject_fd_2_pds_md:518):
RX: get_src_zoneid failed
[11/25/13 07:05:53.814 UTC 2eb0d6 5661] PUNT PATH (fed_punject_get_acl_log_direction:
695): RX: : Invalid CMI2
[11/25/13 07:05:53.814 UTC 2eb0d7 5661] PUNT PATH (fed_punject_fd_2_pds_md:541):RX:
 get_acl_log_direction failed
[11/25/13 07:05:53.814 UTC 2eb0d8 5661] PUNT PATH (fed_punject_get_acl_full_direction:
724):RX: DI 0x513b ACL Full Direction 1
[11/25/13 07:05:53.814 UTC 2eb0d9 5661] PUNT PATH (fed_punject_get_source_sgt:446):
RX: Source SGT 0
[11/25/13 07:05:53.814 UTC 2eb0da 5661] PUNT PATH (fed_punject_get_first_header_type:680):
RX: FirstHeaderType 0
[11/25/13 07:05:53.814 UTC 2eb0db 5661] PUNT PATH (fed_punject_rx_process_packet:916):
RX: fed_punject_pds_send packet 0x1f00 to IOSd with tag 65561
[11/25/13 07:05:53.814 UTC 2eb0dc 5661] PUNT PATH (fed_punject_rx_process_packet:744):
RX: **** RX packet 0x2360 on qn 16, len 128 ****
[11/25/13 07:05:53.814 UTC 2eb0dd 5661]
buf_no 0 buf_len 128

<snip>
```

此输出提供大量信息，通常足以发现数据包的来源和包含的内容。

报头转储的第一部分是前文提过的系统所使用的 Meta-data，第二部分是实际的数据包。

```
ff ff ff ff ff ff - destination MAC address
aa bb cc dd 00 00 - source MAC address
```

您可以选择跟踪此源MAC地址以发现导致错误的端口（一旦确定这是从队列16传送的大部分数据包；此输出仅显示数据包的一个实例，其他输出/数据包被剪切）。

但是，还有更好的方法。请注意报头信息后面显示的日志：

```
[11/25/13 07:05:53.814 UTC 2eb0ce 5661] PUNT PATH (fed_punject_rx_process_packet:
830):RX: Q: 16, Tag: 65561
[11/25/13 07:05:53.814 UTC 2eb0cf 5661] PUNT PATH (fed_punject_get_physical_iif:
```

```
579):RX: Physical IIF-id 0x104d88000000033
```

第一个日志清楚地显示此数据包来自哪个队列和标签。如果您之前不知道该队列，这是确定该队列是哪个队列的简单方法。

第二个日志甚至更加有用，因为它为源接口提供物理接口ID工厂(IIF)-ID。可用于转储该端口相关信息的句柄是一个十六进制值：

```
<#root>

3850-2#

show platform port-asic ifm iif-id 0x0104d88000000033


Interface Table
Interface IIF-ID        : 0x0104d88000000033
Interface Name          : Gi2/0/20
Interface Block Pointer : 0x514d2f70
Interface State         : READY
Interface Stauts        : IFM-ADD-RCVD, FFM-ADD-RCVD
Interface Ref-Cnt       : 6
Interface Epoch         : 0
Interface Type          : ETHER
        Port Type         : SWITCH PORT
        Port Location     : LOCAL
        Slot              : 2
        Unit              : 20
        Slot Unit         : 20
        Active            : Y
        SNMP IF Index     : 22
        GPN               : 84
        EC Channel        : 0
        EC Index          : 0
        ASIC              : 0
        ASIC Port         : 14
        Port LE Handle    : 0x514cd990
Non Zero Feature Ref Counts
        FID : 48(AL_FID_L2_PM), Ref Count : 1
        FID : 77(AL_FID_STATS), Ref Count : 1
        FID : 51(AL_FID_L2_MATM), Ref Count : 1
        FID : 13(AL_FID_SC), Ref Count : 1
        FID : 26(AL_FID_QOS), Ref Count : 1
Sub block information
        FID : 48(AL_FID_L2_PM), Private Data &colon; 0x54072618
        FID : 26(AL_FID_QOS), Private Data &colon; 0x514d31b8
```

您再次确定了源接口和罪魁祸首。

跟踪是一个强大的工具，对于解决高CPU使用率问题至关重要，它提供了大量信息以便成功解决此类问题。

适用于 Cisco Catalyst 3850 系列交换机的嵌入式事件管理器 (EEM) 脚本示例

使用下列命令，在达到特定阈值时触发生成日志：

```
process cpu threshold type total rising

       interval

switch
```

使用此命令生成的日志如下所示：

```
*Jan 13 00:03:00.271: %CPUMEM-5-RISING_THRESHOLD: 1 CPUMEMd[6300]:  Threshold: : 50, Total CPU Utilzati
```

生成的日志提供以下信息：

- 触发生成日志时的总 CPU 利用率。在本例中，这是通过Total CPU Utilization(total/Intr):50/0确定的。
- 排名靠前的进程以 PID/CPU% 格式列出。在此例中，如下所示：

```
8622/25 - 8622 is PID for IOSd and 25 implies that this process is using 25% CPU.
5753/12 - 5733 is PID for FED and 12 implies that this process is using 12% CPU.
```

EEM 脚本如下所示：

```
event manager applet highcpu
event syslog pattern "%CPUMEM-5-RISING_THRESHOLD"
 action 0.1 syslog msg "high CPU detected"
 action 0.2 cli command "enable"
 action 0.3 cli command "show process cpu sorted | append nvram:<filename>.txt"
 action 0.4 cli command "show process cpu detailed process <process name|process ID>
 sorted | nvram:<filename>.txt"
 action 0.5 cli command "show platform punt statistics port-asic 0 cpuq -1
 direction rx | append nvram:<filename>.txt"
 action 0.6 cli command "show platform punt statistics port-asic 1 cpuq -1
 direction rx | append nvram:<filename>.txt"
```

```
action 0.7 cli command "conf t"
action 0.8 cli command "no event manager applet highcpu"
```

注:process cpu threshold命令当前在3.2.X系列中不起作用。另外需要记住的一点是，此命令查看四个核心之间的平均CPU利用率，并且在此平均值达到命令中定义的百分比时生成日志。

Cisco IOS XE 16.x或更高版本

如果您的Catalyst 3850交换机运行Cisco IOS® XE软件版本16.x或更高版本，请参阅对运行IOS-XE 16.x的Catalyst交换机平台的CPU使用率过高进行故障排除。

相关信息

- Cisco IOS XE 是什么？
- Cisco Catalyst 3850 交换机产品手册和资料
- 思科技术支持和下载

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。