Identificar e Solucionar Problemas de Alto Uso da CPU do Switch Catalyst 3850 Series

Contents

Introdução

Este documento descreve como solucionar problemas de utilização da CPU, principalmente devido a interrupções, na nova plataforma Cisco IOS®-XE.

Informações de Apoio

É importante entender como o Cisco IOS®-XE é construído. Com o Cisco IOS®-XE, a Cisco migrou para um kernel Linux e todos os subsistemas foram divididos em processos. Todos os subsistemas que estavam dentro do Cisco IOS® antes, como os drivers de módulos, High Availability (HA) e assim por diante, agora são executados como processos de software dentro do sistema operacional Linux. O próprio Cisco IOS® é executado como um daemon no sistema operacional Linux (IOSd). O Cisco IOS®-XE mantém não apenas a mesma aparência do Cisco IOS® clássico, mas também sua operação, suporte e gerenciamento.

Além disso, o documento introduz vários comandos novos nesta plataforma que são essenciais para solucionar problemas de uso da CPU.

Aqui estão algumas definições úteis:

- Driver do mecanismo de encaminhamento (FED - Forwarding Engine Driver): este é o coração do switch Cisco Catalyst 3850 Series e é responsável por toda a programação/encaminhamento de hardware.

- Cisco IOSd®: Este é o daemon do Cisco IOS® que é executado no kernel do Linux. Ele é executado como um processo de software dentro do kernel.

- Packet Delivery System (PDS): essa é a arquitetura e o processo de como os pacotes são entregues para e de vários subsistemas. Como exemplo, ele controla como os pacotes são entregues do FED ao IOSd e vice-versa.

- Identificador: Um identificador pode ser considerado como um ponteiro. É um meio de descobrir informações mais detalhadas sobre variáveis específicas que são usadas nas saídas que a caixa produz. Isso é semelhante ao conceito de índices LTL (Local Target Logic, lógica de destino local) no switch Cisco Catalyst 6500 Series.

Estudo de caso: Interrupções do Address Resolution Protocol (ARP)

O processo de solução de problemas e verificação nesta seção pode ser amplamente utilizado para alto uso da CPU devido a interrupções.

Etapa 1: Identificar o processo que consome ciclos da CPU

O comando **show process cpu** exibe naturalmente a aparência atual da CPU. Observe que o Cisco Catalyst 3850 Series Switch usa quatro núcleos e você vê o uso da CPU listado para todos os quatro núcleos:

```
3850-2#show processes cpu sort | 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
```

Na saída, fica claro que o daemon do Cisco IOS® consome uma parte importante da CPU junto com o FED, que é o coração dessa caixa. Quando o uso da CPU é alto devido a interrupções, você vê que o Cisco IOSd® e o FED usam uma parte maior da CPU, e esses subprocessos (ou um subconjunto deles) usam a CPU:

- FED Punject TX
- FED Punject RX
- Reabastecimento de Punject FED
- FED Punject TX completo

Você pode ampliar qualquer um desses processos com o comando **show process cpu detailed <*process*>**. Como o Cisco IOSd® é responsável pela maior parte do uso da CPU, aqui está uma análise mais detalhada.

```
3850-2#show processes cpu detailed process iosd sort | 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
```

A saída (saída de CPU do Cisco IOSd®) mostra que o Snoop ARP, o Processo de rastreamento de host IP e a Entrada ARP estão altos. Isso é comumente visto quando a CPU é interrompida devido a pacotes ARP.

Etapa 2: Determine a fila da CPU que causa a condição de alto uso da CPU

O switch Cisco Catalyst 3850 Series tem várias filas que atendem a diferentes tipos de pacotes (o FED mantém 32 filas de CPU RX, que são filas que vão diretamente para a CPU). É importante monitorar essas filas para descobrir quais pacotes são lançados para a CPU e quais são processados pelo Cisco IOSd®. Essas filas são por ASIC.

Observação: há dois ASICs: 0 e 1. As portas 1 a 24 pertencem ao ASIC 0.

Para examinar as filas, insira o comando **show platform punt statistics port-asic <*port-asic*> cpuq** <queue> **direction** <rx|tx>**.**

No comando **show platform punt statistics port-asic 0 cpuq -1 direction rx**, o argumento -1 lista todas as filas. Portanto, esse comando lista todas as filas de recebimento para Port-ASIC 0.

Agora, você deve identificar qual fila envia um grande número de pacotes a uma taxa alta. Neste exemplo, um exame das filas revelou este culpado:

```
<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>
```

O número da fila é 16 e o nome da fila é CPU_Q_PROTO_SNOOPING.

Outra maneira de descobrir a fila culpada é inserir o comando **show platform punt client**.

```
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
```

Determine a tag para a qual a maioria dos pacotes foi alocada. Neste exemplo, é 65561.

Em seguida, insira este comando:

```
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
```

Esta saída mostra que a fila é Rx Proto Snoop.

O s antes do 65561 na saída do comando **show platform punt client** significa que o identificador de FED está suspenso e sobrecarregado pelo número de pacotes de entrada. Se o s não desaparecer, significa que a fila está presa permanentemente.

Etapa 3: Descartar o pacote enviado à CPU

Nos resultados do comando **show pds tag all**, observe que um identificador, 7296672, é relatado ao lado do Punt Rx Proto Snoop.

Use este identificador no comando **show pds client** <handle> **packet last sink**. Observe que você deve habilitar debug pds pktbuf-last antes de usar o comando. Caso contrário, você encontrará este erro:

```
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
```

Com a depuração ativada, você verá esta saída:

```
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 02 00 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              ............
```

Este comando despeja o último pacote recebido pelo coletor, que é IOSd neste exemplo. Isso mostra que ele despeja o cabeçalho e pode ser decodificado com o Wireshark baseado em Terminal (TShark). Os metadados são para uso interno do sistema, mas a saída de dados fornece informações reais do pacote. Os metadados, no entanto, continuam sendo extremamente úteis.

Observe a linha que começa com 0070. Use os primeiros 16 bits depois disso, como mostrado aqui:

<#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
```

A interface culpada é identificada aqui. Gig2/0/20 é onde há um gerador de tráfego que bombeia o tráfego ARP. Se você desligar, o problema seria resolvido e o uso da CPU seria minimizado.

Etapa 4: Use o rastreamento de FED

A única desvantagem do método discutido na última seção é que ele apenas despeja o último pacote que vai para o coletor e não pode ser o culpado.

Uma maneira melhor de solucionar esse problema seria usar um recurso chamado rastreamento de FED. O rastreamento é um método de captura de pacotes (usando vários filtros) que são enviados pelo FED à CPU. No entanto, o rastreamento de FED não é tão simples quanto o recurso Netdr no Cisco Catalyst 6500 Series Switch.

Aqui, o processo é dividido em etapas:

1. Habilitar rastreamento de detalhes. Por padrão, o rastreamento de eventos está ativado. Você deve habilitar o rastreamento detalhado para capturar os pacotes reais:

```
3850-2#set trace control fed-punject-detail enable
```

2. Ajuste o buffer de captura. Determine a profundidade dos buffers para rastreamento detalhado e aumente conforme necessário.

```
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
```

Você pode alterar o tamanho do buffer com este comando:

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

Os valores disponíveis para você são:

```
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. Adicione filtros de captura. Agora você precisa adicionar vários filtros para a captura. Você pode adicionar diferentes filtros e escolher corresponder todos ou qualquer um deles para sua captura.

   Os filtros são adicionados com este comando:

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

   Estas opções estão disponíveis no momento:

```
3850-2#set trace fed-punject-detail direction rx filter_add ?
  cpu-queue  rxq 0..31
  field      field
  offset     offset
```

   Agora, você deve unir as coisas. Você se lembra da fila de culpados identificada na Etapa 2 deste processo de solução de problemas? Como a fila 16 é a fila que envia um grande número de pacotes para a CPU, faz sentido rastrear essa fila e ver quais pacotes são lançados para a CPU por ela.

Você pode optar por rastrear qualquer fila com este comando:

```
3850-2#set trace fed-punject-detail direction rx filter_add cpu-queue
```

Aqui está o comando para este exemplo:

```
3850-2#set trace fed-punject-detail direction rx filter_add cpu-queue 16 16
```

Você deve escolher uma correspondência total ou qualquer correspondência para os filtros e habilitar o rastreamento:

```
3850-2#set trace fed-punject-detail direction rx match_all
3850-2#set trace fed-punject-detail direction rx filter_enable
```

4. Exibir pacotes filtrados. Você pode exibir os pacotes capturados com o comando **show mgmt-infra trace messages fed-punject-detail**.

```
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>
```

Essa saída fornece muitas informações e pode ser suficiente para descobrir de onde os pacotes vêm e o que está contido neles.

A primeira parte do dump do cabeçalho é novamente o Meta-dados que é usado pelo sistema. A segunda parte é o pacote real.

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

Você pode optar por rastrear esse endereço MAC de origem para descobrir a porta culpada (depois de identificar que essa é a maioria dos pacotes que são lançados da fila 16; essa saída mostra apenas uma instância do pacote e a outra saída/pacotes são cortados).

No entanto, há uma maneira melhor. Observe que os registros que estão presentes após as informações de cabeçalho:

```
[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
```

O primeiro registro informa claramente de qual fila e marca esse pacote vem. Se você não estava ciente da fila anteriormente, essa é uma maneira fácil de identificar qual fila ela estava.

O segundo registro é ainda mais útil porque fornece o ID de Fábrica de ID de interface (IIF) físico para a interface de origem. O valor hexadecimal é um identificador que pode ser usado para despejar informações sobre essa porta:

```
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
```

Você identificou novamente a interface de origem e o responsável.

O rastreamento é uma ferramenta poderosa que é crítica para solucionar problemas de uso elevado da CPU e fornece muitas informações para resolver com sucesso tal situação.

Exemplo de script do Embedded Event Manager (EEM) para o switch Cisco Catalyst 3850 Series

Use este comando para disparar um log a ser gerado em um limite específico:

```
process cpu threshold type total rising



        interval



switch
```

O log gerado com o comando é semelhante a este:

```
*Jan 13 00:03:00.271: %CPUMEM-5-RISING_THRESHOLD: 1 CPUMEMd[6300]:  Threshold: :
50, Total CPU Utilzation(total/Intr) :50/0, Top 3 processes(Pid/Util) : 8622/25,
```

```
5753/12, 9663/0
```

O log gerado fornece estas informações:

- A utilização total da CPU no momento do gatilho. Isso é identificado por Total CPU Utilization(total/Intr) :50/0 neste exemplo.
- Principais processos - listados no formato PID/CPU%. Neste exemplo, eles são:

```
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.
```

O script EEM é mostrado aqui:

```
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"
```

Observação: o comando **process cpu threshold** não funciona atualmente na trilha 3.2.X. Outro ponto a ser lembrado é que esse comando examina a utilização média da CPU entre os quatro núcleos e gera um log quando essa média atinge o percentual definido no comando.

Cisco IOS-XE 16.x ou versões posteriores

Se você tiver switches Catalyst 3850 que executam o Cisco® IOS-XE Software Release 16.x ou posterior, consulte Troubleshooting de Alto Uso da CPU em Catalyst Switch Platforms Executando o IOS-XE 16.x.

Informações Relacionadas

- **O que é o Cisco IOS XE?**
- **Switches Cisco Catalyst 3850 - Folha de dados técnicos e documentação**
- **Suporte Técnico e Documentação - Cisco Systems**

Sobre esta tradução
A Cisco traduziu este documento com a ajuda de tecnologias de tradução automática e humana para oferecer conteúdo de suporte aos seus usuários no seu próprio idioma, independentemente da localização.

Observe que mesmo a melhor tradução automática não será tão precisa quanto as realizadas por um tradutor profissional.

A Cisco Systems, Inc. não se responsabiliza pela precisão destas traduções e recomenda que o documento original em inglês (link fornecido) seja sempre consultado.