Troubleshooting del uso del Catalyst 3850 Series Switch CPU elevada

Contenido

Introducción

Este documento describe cómo resolver problemas las preocupaciones del USO de la CPU, sobre todo debido a las interrupciones, en la plataforma del nuevo Cisco IOS ®-XE. Además, el documento presenta a varios comandos new en esta plataforma que sean integrales para resolver problemas tales problemas.

Antecedentes

Es importante entender cómo se construye el Cisco IOS XE. Con el Cisco IOS XE, Cisco se ha movido a un núcleo de Linux y todos los subsistemas se han analizado en los procesos. Todos los subsistemas que eran Cisco IOS interior antes - por ejemplo los drivers de los módulos, la Alta disponibilidad (HA), y así sucesivamente - de ahora se ejecutan como procesos del software dentro del operating system (OS) de Linux. El Cisco IOS sí mismo se ejecuta como daemon dentro del Linux OS (IOSd). El Cisco IOS XE conserva no sólo la misma mirada y sensación del Cisco IOS clásico, pero también su operación, soporte, y Administración.

Aquí están algunas definiciones útiles:

- **Driver de motor de reenvío (FED)**: Éste es el corazón del Cisco Catalyst 3850 Series Switch y es responsable de toda la programación/expedición del hardware.
- **IOSd**: Ésta es la daemon del Cisco IOS que se ejecuta en el núcleo de Linux. Se ejecuta como proceso del software dentro del corazón.

- **Sistema de la entrega del paquete (PDS)**: Éste es la arquitectura y el proceso de cómo los paquetes se entregan a y desde el diverso subsistema. Como un ejemplo, controla cómo los paquetes se entregan del FED al IOSd y vice versa.

- **Manija**: Una manija se puede pensar en como puntero. Es los medios de descubrir más información detallada sobre las variables específicas que se utilizan en las salidas que el cuadro produce. Esto es similar al concepto de índices de la lógica de destino local (LTL) en el Cisco Catalyst 6500 Series Switch.

Caso práctico: Interrupciones del Address Resolution Protocol (ARP)

El Troubleshooting y el proceso de verificación en esta sección se pueden utilizar ampliamente para CPU elevada el uso debido a las interrupciones.

Paso 1: Identifique el proceso que consume los ciclos de la CPU

Del comando show process cpu las visualizaciones naturalmente cómo el CPU mira actualmente. Note que las Cisco Catalyst 3850 Series conmutan las aplicaciones cuatro memorias, y usted ven el USO de la CPU enumerado para las cuatro memorias:

```
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
```

De la salida, está claro que la daemon del Cisco IOS consume una porción importante del CPU junto con el FED, que es el corazón de este cuadro. Cuando el USO de la CPU es elevado debido a las interrupciones, usted ve que IOSd y FED utilizan una porción importante del CPU, y estos subprocesos (o un subconjunto de éstos) utilizan el CPU:

- FED Punject TX
- FED Punject RX
- FED Punject llena
- FED Punject TX completo

Usted puede enfocar en ninguno de estos procesos con el comando **detallado CPU del *<process>* del proceso de la demostración**. Puesto que IOSd es responsable de la mayoría del USO de la CPU, aquí está una mirada más atenta en ésa.

```
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
```

La salida (IOSd CPU hecho salir) muestra que el fisgón ARP, el proceso de la pista del host IP, y la entrada de información ARP son altos. Esto está comúnmente - visto cuando el CPU es interrumpido debido a los paquetes ARP.

Paso 2: Determine la cola CPU que causa CPU elevada la condición del uso

El Cisco Catalyst 3850 Series Switch tiene varias colas de administración del tráfico que abastezcan a diversos tipos de paquetes (el FED mantiene 32 colas de administración del tráfico RX CPU, que son las colas de administración del tráfico que van directamente al CPU). Es importante monitorear estas colas de administración del tráfico para descubrir qué paquetes se llevan en batea al CPU y cuáles son procesados por el IOSd. Estas colas de administración del tráfico están por ASIC.

> **Note**: Hay dos Asics: 0 y 1. los puertos 1 a 24 pertenecen a ASIC 0.

Para mirar las colas, ingrese el *<rx de la dirección del <queue> del cpuq del <port-asic> de puerto-ASIC de las estadísticas de la batea de la plataforma de la demostración|*comando del *tx>*.

En las **estadísticas puerto-ASIC de la batea de la plataforma de la demostración 0** comandos del **rx de la dirección del cpuq -1**, las **-1** listas de argumento todas las colas de administración del tráfico. Por lo tanto, este listas de comandos todos los recibires cola para Puerto-ASIC 0.

Ahora, usted debe identificar qué cola avanza un gran número de paquetes a una alta velocidad. En este ejemplo, un examen de las colas de administración del tráfico reveló a este culpable:

```
<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>
```

El número de la cola es 16 y el nombre de la cola es **CPU_Q_PROTO_SNOOPING**.

Otra manera de descubrir la cola del culpable es ingresar el **comando client de la batea de la plataforma de la demostración**.

```
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 la etiqueta para la cual se han afectado un aparato la mayoría de los paquetes. En este ejemplo, es **65561**.

Entonces, ingrese 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
```

Este ouput muestra que la cola es **fisgón Proto del rx**.

El s antes de que los **65561** en la salida del **comando client de la batea de la plataforma de la demostración** signifique que la manija de FED es suspendida y abrumada por el número de

paquetes entrantes. Si el **s** no desaparece, significa que la cola está pegada permanentemente.

Paso 3: Vacie el paquete enviado al CPU

En los resultados del **comando all de la etiqueta pds de la demostración**, note una manija, **7296672**, está señalado al lado del **fisgón Proto del rx de la batea**.

Utilice esta manija en el comando del **fregadero del último del paquete del *<handle> del* cliente pds de la demostración**. Note que usted debe habilitar el pktbuf-**último pds del debug** antes de que usted utilice el comando. Si no usted encuentra este error:

```
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
```

Con el debug habilitado, usted ve esta salida:

```
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 vacia el paquete más reciente recibido por el fregadero, que es IOSd en este ejemplo. Esto muestra que vacia la encabezado y puede ser decodificada con Wireshark Terminal-basado (TShark). **Los meta datos** están para el uso interno al lado del sistema, pero la **salida de datos** proporciona la información del paquete real. **Los meta datos**, sin embargo, siguen siendo extremadamente útiles.

Línea del aviso que comienza con **0070**. Utilice los primeros 16 bits después que como se muestra aquí:

```
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
```

La interfaz del culpable se identifica aquí. Gig2/0/20 es donde hay un generador de tráfico ese tráfico ARP de las bombas. Si usted cierra esto trague, después resolvería el problema y minimizaría el USO de la CPU.

Paso 4: Utilice el seguimiento de FED

La única desventaja con el método discutido en la sección más reciente es que vacia solamente el paquete más reciente que entra el fregadero, y puede ser que no sea el culpable.

Una mejor manera de resolver problemas esto sería utilizar una característica llamada **seguimiento de FED**. El seguimiento es un método de la captura de paquetes (usando los diversos filtros) que es avanzado por el FED al CPU. El seguimiento de FED no es tan simple como la característica de Netdr en el Cisco Catalyst 6500 Series Switch, sin embargo. Aquí el proceso está roto en los pasos:

1. Seguimiento del detalle del permiso. Por abandono, el seguimiento de evento está **prendido**. Usted debe permitir al seguimiento del detalle para capturar los paquetes reales:

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

2. Ajuste el buffer de la captura. Determine cómo profundamente sus buffers están para el seguimiento del detalle y aumente según lo necesitado.

   ```
   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
```

Usted puede cambiar el tamaño de almacén intermedio con este comando:

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

Los valores disponibles para usted son:

```
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. Agregue los filtros de la captura. Usted ahora necesita agregar los diversos filtros para la captura. Usted puede agregar diversos filtros y para elegir **hacer juego todos** o **hacer juego ningunos de** ésos para su captura.

   Los filtros se agregan con este comando:

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

   Estas opciones están actualmente disponibles:

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

   Ahora usted debe conectar las cosas juntas. ¿Recuerde la cola del culpable que fue identificada en el paso 2 de este proceso del Troubleshooting? Puesto que la cola 16 es la cola que avanza un gran número de paquetes hacia el CPU, tiene sentido de localizar esta cola y de ver qué paquetes son llevados en batea al CPU por él.

   Usted puede elegir localizar cualquier cola con este comando:

```
 3850-2#set trace fed-punject-detail direction rx filter_add cpu-queue <start queue>
<end queue>
```

Aquí está el comando por este ejemplo:

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

Usted debe elegir una **coincidencia toda** o una **coincidencia** para sus filtros y después habilitar la traza:

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

4. Paquetes filtrados visualización. Usted puede visualizar los paquetes capturados con el comando del alimentar-punject-**detalle de los mensajes de seguimiento del mgmt-infra de la demostración**.

```
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>
```

Esta salida proporciona el un montón de información y debe típicamente ser bastante para descubrir de adónde los paquetes vienen y qué se contiene en ellos.

La primera parte del volcado de la encabezado es otra vez los **meta datos** que es utilizado por el sistema. La segunda parte es el paquete real.

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

Usted puede elegir localizar este MAC Address de origen para descubrir el puerto del culpable (una vez que usted ha identificado que ésta es la mayoría de los paquetes que se están llevando en batea de la cola 16; esta salida muestra solamente que el un caso del paquete y la otra salida/paquetes están acortados).

Sin embargo, hay una mejor manera. Note que registros que están presentes después de la información de encabezado:

```
[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
```

El primer registro le dice claramente que de qué cola y marque este paquete con etiqueta viene. Si usted no era consciente del eariler de la cola, esto es una forma sencilla de identificar que la hacen cola eran.

El segundo registro es aún más útil, porque proporciona la fábrica de la interfaz física ID (IIF) - ID para la interfaz de origen. El valor hex es una manija que se puede utilizar para vaciar la información sobre ese puerto:

```
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
```

Usted ha identificado de nuevo la interfaz y al culpable del souce.

El seguimiento es una herramienta potente que es crítica para resolver problemas CPU elevada los problemas del uso y proporciona el un montón de información para resolver con

éxito tal situación.

Script integrado muestra del administrador del evento (EEM) para el Cisco Catalyst 3850 Series Switch

Utilice este comando para accionar un registro que se generará en un umbral específico:

```
process cpu threshold type total rising <CPU %> interval <interval in seconds>
switch <switch number>
```

El registro generado con el comando parece esto:

```
*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
```

El registro generado proporciona esta información:

- El uso total de la CPU a la hora del activador. Esto es identificada por **CPU total Utilzation (total/Intr): 50/0** en este ejemplo.
- Procesos superiores - éstos se enumeran en el formato del **PID/CPU%**. Tan en este ejemplo, éstos son:

```
*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
```

El script EEM se muestra aquí:

```
*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
```

> **Note**: **El comando threshold CPU del proceso** no trabaja actualmente en el tren 3.2.X. Otra punta a recordar es que este comando mira el utilzation medio CPU entre las cuatro memorias y genera un registro cuando esta media alcanza el porcentaje que se ha definido en el comando.

Información Relacionada

- ¿Cuál es Cisco IOS XE?
- Cisco Catalyst 3850 Switch - Hojas de datos y literatura
- Soporte Técnico y Documentación - Cisco Systems