O uso da alta utilização da CPU do Catalyst 3850 Series Switch pesquisa defeitos

Índice

Introdução

Este original descreve como pesquisar defeitos os interesses do USO de CPU, primeiramente devido às interrupções, no ® do Novo Cisco IOS - plataforma XE. Adicionalmente, o original introduz diversos comandos new nesta plataforma que são integrais a fim pesquisar defeitos tais problemas.

Informações de Apoio

Éimportante compreender como o Cisco IOS XE é construído. Com Cisco IOS XE, Cisco moveu-se para um kernel (centro) de Linux e todos os subsistemas foram divididos em processos. Todos os subsistemas que eram Cisco IOS interno antes - como os direcionadores dos módulos, a Alta disponibilidade (HA), e assim por diante - de agora são executado como processos de software dentro do operating system (OS) de Linux. O Cisco IOS próprio é executado como um demônio dentro do Linux OS (IOSd). O Cisco IOS XE retém não somente o mesma olhar e sensação do IOS Cisco clássico, mas igualmente seus operação, apoio, e Gerenciamento.

Estão aqui algumas definições úteis:

- **Direcionador do Forwarding Engine (ALIMENTADO)**: Este é o coração do Cisco Catalyst 3850 Series Switch e é responsável para toda a programação do hardware/transmissão.
- **IOSd**: Este é o demônio do Cisco IOS que é executado no kernel (centro) de Linux. É executado como um processo de software dentro do núcleo.
- **Sistema da entrega de pacotes (PDS)**: Este é a arquitetura e o processo de como os pacotes são entregados a e do vário subsistema. Como um exemplo, controla como os pacotes são entregados do ALIMENTADO ao IOSd e vice-versa.

- **Punho**: Um punho pode ser pensado como de um ponteiro. É meios descobrir mais informação detalhada sobre as variáveis específicas que são usadas nas saídas que a caixa produz. Isto é similar ao conceito de deslocamentos predeterminados da lógica de alvo local (LTL) no Cisco Catalyst 6500 Series Switch.

Casos Práticos: Interrupções do Address Resolution Protocol (ARP)

A pesquisa de defeitos e o processo de verificação nesta seção podem amplamente ser usados para o uso da alta utilização da CPU devido às interrupções.

Passo 1: Identifique o processo que consome ciclos de CPU

Do comando show process cpu os indicadores naturalmente como o CPU olha atualmente. Observe que o Cisco Catalyst 3850 Series Swtich usa quatro núcleos, e você veem o USO de CPU alistado 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
```

Da saída, é claro que o demônio do Cisco IOS consome uma parcela principal do CPU junto com ALIMENTADA, que é o coração desta caixa. Quando o USO de CPU é alta devido às interrupções, você vê que IOSd e o uso FED um a parcela principal do CPU, e estes subprocesses (ou um subconjunto destes) usam o CPU:

- Punject ALIMENTADO TX
- Punject ALIMENTADO RX
- Punject ALIMENTADO reabastece
- Punject ALIMENTADO TX termina

Você pode zumbir em qualquens um processos com o comando **detalhado processador central do *<process> do* processo da mostra**. Desde que IOSd é responsável para a maioria do USO de CPU, está aqui um olhar mais atento naquele.

```
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 (IOSd CPU output) mostra que a espião ARP, o processo da trilha do Host IP, e a entrada ARP são altos. Isto é geralmente - visto quando o CPU é interrompido devido aos pacotes ARP.

Passo 2: Determine a fila CPU que causa a condição do uso da alta utilização da CPU

O Cisco Catalyst 3850 Series Switch tem um número de filas que abastecem aos tipos diferentes de pacotes (ALIMENTADO mantém 32 filas RX CPU, que são as filas que vão diretamente ao CPU). É importante monitorar estas filas a fim descobrir que pacotes punted ao CPU e quais são processados pelo IOSd. Estas filas são pelo ASIC.

> **Note**: Há dois ASIC: 0 e 1. as portas 1 a 24 pertencem a ASIC 0.

A fim olhar as filas, incorpore o *<rx do sentido do <queue> do cpuq do <port-asic> das estatísticas porta-ASIC do pontapé da plataforma da mostra|*comando do *tx>*.

No comando do **sentido RX do cpuq -1 das estatísticas porta-ASIC 0 do pontapé da plataforma da mostra**, as **-1** lista de argumento todas as filas. Consequentemente, lista deste comando todas as filas de recepção para Porta-ASIC 0.

Agora, você deve identificar que enfileiram impulsos um grande número pacotes em 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**.

Uma outra maneira de descobrir a fila do culpado é inscrever o **comando client do pontapé da plataforma da mostra**.

```
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 etiqueta para que a maioria de pacotes foram atribuídos. Neste exemplo, é **65561**.

Então, incorpore 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
```

Estas saídas mostram que a fila é **espião Proto RX**.

O s antes que os **65561 na** saída do **comando client do pontapé da plataforma da mostra** significar que o punho ALIMENTADO está suspendido e oprimido pelo número de pacotes recebidos. Se o **s** não desaparece, significa que a fila está colada permanentemente.

Passo 3: Despeje o pacote enviado ao CPU

Nos resultados do **comando all da etiqueta pds da mostra**, observe um punho, **7296672**, é relatado ao lado da **espião Proto do pontapé RX**.

Use este punho no comando do **dissipador do último do pacote do *<handle> do* cliente pds da**

mostra. Observe que você deve permitir **debuga o pds pktbuf-último** antes que você use o comando. Se não você encontra 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 debugar permitido, você vê 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 dissipador, que é IOSd neste exemplo. Isto mostra que despeja o encabeçamento e pode ser descodificado com Wireshark Terminal-baseado (TShark). **O Meta-DATA** é para o uso interno pelo sistema, mas as **saídas de dados** fornecem a informação do pacote real. **O Meta-DATA**, contudo, permanece extremamente útil.

Observe a linha que começa com **0070**. Use os primeiros 16 bit em seguida que como mostrado aqui:

```
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 relação do culpado é identificada aqui. **Gig2/0/20** é onde há um gerador de tráfego esse tráfego ARP das bombas. Se você fecha este trague, a seguir resolveria o problema e minimizaria o USO de CPU.

Passo 4: Use o traçado ALIMENTADO

O único inconveniente com o método discutido na última seção é que despeja somente o último pacote que entra no dissipador, e não pôde ser o culpado.

Uma maneira melhor de pesquisar defeitos isto seria usar uma característica chamada **traçado FED**. O traçado é um método da captura de pacote de informação (que usa vários filtros) que é empurrado pelo ALIMENTADO ao CPU. O traçado FED não é tão simples quanto a característica de Netdr no Cisco Catalyst 6500 Series Switch, contudo. Aqui o processo quebra-se em etapas:

1. Permita o seguimento do detalhe. À revelia, o rastreamento de evento está **ligada**. Você deve permitir o detalhe que segue a fim capturar os pacotes reais:

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

2. Ajustar o buffer da captação. Determine como profundamente seus bufferes são para o traçado do detalhe e aumente-o como 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 mudar o tamanho de buffer com este comando:

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

Os valores disponíveis a 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. Adicionar filtros da captação. Você precisa agora de adicionar vários filtros para a captação. Você pode adicionar filtros diferentes e para escolher **combinar tudo** ou **combinar alguma d**aqueles para sua captação.

   Os filtros são adicionados com este comando:

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

   Estas opções estão atualmente disponíveis:

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

   Agora você deve ligar coisas junto. Recorde que a fila do culpado que foi identificada em etapa 2 desta pesquise defeitos o processo? Desde que a fila 16 é a fila que empurra um grande número pacotes para o CPU, faz o sentido seguir esta fila e ver que pacotes punted ao CPU por ele.

   Você pode escolher seguir toda a fila com este comando:

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

   Está aqui o comando para este exemplo:

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

   Você deve escolher **todos um fósforo** ou um **fósforo** para seus filtros e então permitir o traço:

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

4. Pacotes filtrados indicador. Você pode indicar os pacotes capturados com o comando do alimentar-punject-**detalhe das mensagens do traço de Mgmt-infra da mostra**.

```
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 saída fornece a abundância da informação e deve tipicamente ser bastante para descobrir de aonde os pacotes vêm e o que é contido nelas.

O primeiro parte da descarga do encabeçamento é outra vez o Meta-DATA 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 escolher seguir este endereço MAC de origem a fim descobrir a porta do culpado (uma vez que você identificou que esta é a maioria dos pacotes que punted da fila 16; esta saída mostra somente que um exemplo do pacote e a outra saída/pacotes estão grampeados).

Contudo, há uma maneira melhor. Observe que logs que estam presente após a informação 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 log di-lo claramente que de qual a fila e etiqueta este pacote vem. Se você não estava ciente do eariler da fila, esta é uma maneira fácil identificar que a enfileiram eram.

O segundo log é ainda mais útil, porque fornece a fábrica da interface física ID (IIF) - ID para

a interface de origem. O valor de HEX é um punho que possa ser usado a fim despejar a informação 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
       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
```

Você tem identificado mais uma vez a relação e o culpado do souce.

O traçado é uma ferramenta poderosa que seja crítica a fim pesquisar defeitos problemas do uso da alta utilização da CPU e forneça a abundância da informação a fim resolver com sucesso tal situação.

Script encaixado amostra do gerente do evento (EEM) para o Cisco Catalyst 3850 Series Switch

Use este comando a fim provocar um log a ser gerado em um ponto inicial específico:

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

O log gerado com o comando olha como 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 esta informação:

- A utilização total de CPU na altura do disparador. Isto é identificado por **CPU total Utilzation (total/Intr): 50/0 n**este exemplo.
- Processos superiores - estes são alistados no formato de **PID/CPU%**. Assim neste exemplo, estes são:

```
*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 script EEM é mostrado aqui:

```
*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**: **O comando threshold processador central do processo** não trabalha atualmente no trem 3.2.X. Um outro ponto a recordar é que este comando olha o utilzation médio CPU entre os quatro núcleos e gerencie um log quando esta média alcança a porcentagem que esteve definida no comando.

Cisco IOS XE 16.x ou liberações mais atrasadas

Se você tem os Catalyst 3850 Switch que executam a liberação 16.x do Software Cisco IOS XE ou mais tarde, veja para pesquisar defeitos o uso da alta utilização da CPU nas plataformas de Catalyst switch que executam IOS-XE 16.x.

Informações Relacionadas

- Que é Cisco IOS XE?
- Cisco Catalyst 3850 Switch - Folhas de dados e literatura
- Suporte Técnico e Documentação - Cisco Systems