Catalyst 3850シリーズスイッチでのCPU高使用率のトラブルシューティング

内容

はじめに

このドキュメントでは、新しいCisco IOS® XEプラットフォームで、主に割り込みによって発生するCPU使用率の問題をトラブルシューティングする方法について説明します。

背景説明

Cisco IOS® XEの構築方法を理解することが重要です。Cisco IOS® XEを使用して、シスコはLinuxカーネルに移行し、すべてのサブシステムがプロセスに分割されました。モジュールドライバやハイアベイラビリティ(HA)など、以前はCisco IOS内に存在していたすべてのサブシステムは、現在はLinuxオペレーティングシステム(OS)内のソフトウェアプロセスとして動作します。Cisco IOS自体は、Linux OS(IOSd)内のデーモンとして動作します。Cisco IOS® XEは、従来のCisco IOS®と同じルックアンドフィールだけでなく、その動作、サポート、および管理も保持しています。

さらに、このドキュメントでは、CPU使用率の問題のトラブルシューティングに不可欠な、このプラットフォーム上の新しいコマンドをいくつか紹介します。

いくつかの有用な定義を次に示します。

- Forwarding Engine Driver(FED)：これはCisco Catalyst 3850シリーズスイッチの中核であり、すべてのハードウェアプログラミング/転送を担当します。

- Cisco IOSd:Linuxカーネルで動作するCisco IOS®デーモンです。カーネル内のソフトウェアプロセスとして実行されます。

- Packet Delivery System（PDS；パケット配信システム）：さまざまなサブシステムとの間でパケットを配信する方法のアーキテクチャとプロセスです。たとえば、FED と IOSd の間でのパケットの送受信を制御します。

- ハンドル：ハンドルはポインタと考えることができます。これは、ボックスが生成する出力で使用される特定の変数に関する詳細情報を見つけるために使用されます。Cisco Catalyst 6500 シリーズ スイッチのローカル ターゲット ロジック（LTL）インデックスの概念に似ています。

ケーススタディ：アドレス解決プロトコル(ARP)割り込み

このセクションで説明するトラブルシューティングおよび検証プロセスは、割り込みによって高い CPU 使用率が生じている状況で幅広く使用できます。

ステップ1:CPUサイクルを消費しているプロセスを特定する

show process cpu コマンドは、CPU の現在の状態をそのまま表示します。Cisco Catalyst 3850シリーズスイッチは4つのコアを使用し、4つのコアすべてのCPU使用率が表示されます。

<#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
```

出力から、Cisco IOS®デーモンが、このボックスの中心であるFEDとともにCPUの大部分を消費していることがわかります。割り込みによってCPU使用率が高くなっている場合は、Cisco IOSdとFEDがCPUの大部分を使用しており、次のサブプロセス（またはこれらのサブセット）がCPUを使用していることがわかります。

- FED Punject TX
- FED Punject RX
- FED Punject replenish
- FED Punject TX complete

show process cpu detailed <process> コマンドを使用すると、これらのプロセスを詳しく調べることができます。CPU使用率の大部分はCisco IOSdが担当しているため、次に詳細を示します。

<#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
```

出力（Cisco IOSd CPU出力）は、ARPスヌープ、IPホストトラックプロセス、およびARP入力が高いことを示しています。これは、ARP パケットによる CPU 割り込みが発生しているときによく見られる現象です。

手順2：高CPU使用率の原因となっているCPUキューを特定する

Cisco Catalyst 3850 シリーズ スイッチには、さまざまなタイプのパケットに対応するための多数のキューがあります（FED は 32 個の RX CPU キューを保持しています。これらは CPU に直接送信されるキューです）。どのパケットがCPUにパントされ、どのパケットがCisco IOSdで処理されるかを検出するには、これらのキューを監視することが重要です。これらのキューは ASIC ごとにあります。

注:ASICには0と1の2つがあります。ポート1 ~ 24はASIC 0に属しています。

キューを表示するには、 show platform punt statistics port-asic <port-asic>cpuq <queue> direction
 コマンドを使用して、アップグレードを実行します。

show platform punt statistics port-asic 0 cpuq -1 direction rx コマンドで -1 引数を指定すると、すべてのキューが表示されます。したがって、このコマンドを実行すると、Port-ASIC 0 のすべての受信キューが表示されます。

次に、多数のパケットを高速でプッシュしているキューを特定する必要があります。この例では、キューを詳しく調べることで次のキューが原因になっていることがわかります。

```
<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>
```

キュー番号は 16 で、キュー名は CPU_Q_PROTO_SNOOPING です。

問題を引き起こしているキューを見つけるもう 1 つの方法は、show platform punt client コマンドを入力することです。

<#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
```

ほとんどのパケットが割り当てられているタグを特定します。この例では、65561 です。

続いて、次のコマンドを入力します。

<#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
```

この出力は、キューがRx Proto Snoopであることを示しています。

show platform punt client コマンドの出力の 65561 の前の s は、FED ハンドルが多数の着信パケットを処理しきれず、中断状態になっていることを示しています。s が消えない場合は、キューが完全にスタックしていることを意味します。

ステップ3:CPUに送信されるパケットのダンプ

show pds tag all コマンドの結果では、ハンドル 7296672 が Punt Rx Proto Snoop の横に表示されています。

このハンドルを show pds client <handle> packet last sink コマンドで使用します。このコマンドを使用する前に、debug pds pktbuf-last を有効にする必要があります。これを行わないと、次のエラーが発生します。

<#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
```

デバッグを有効にすると、次の出力が表示されます。

<#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 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              ............
```

このコマンドは、シンク（この例ではCisco IOSd）が最後に受信したパケットをダンプします。これはヘッダーがダンプされ、端末ベースの Wireshark（TShark）でデコードできることを示しています。Meta-data はシステムが内部で使用するものですが、Data 出力は実際のパケット情報を提供します。ただし、Meta-data は非常に役立ちます。

0070で始まる行に注目してください。その後の最初の 16 ビットを次のように使用します。

<#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
```

ここで問題を引き起こしているインターフェイスが判明します。ARP トラフィックを送り出しているトラフィック ジェネレータがあるのは Gig2/0/20 です。このインターフェイスをシャットダウンすると、問題が解決され、CPU 使用率が最小限に抑えられます。

ステップ4:FEDトレースの使用

前のセクションで説明した方法の唯一の欠点は、シンクに入る最後のパケットをダンプするだけであり、問題の原因になれないことです。

より適切なトラブルシューティング方法は、FED トレースと呼ばれる機能を使用することです。トレースとは、FED から CPU にプッシュされるパケットを (各種フィルタを使用して) キャプチャする方法です。ただし、FED トレースは Cisco Catalyst 6500 シリーズ スイッチの Netdr 機能ほどシンプルではありません。

ここでは、このプロセスをいくつかのステップに分割して説明します。

1. 詳細トラッキングを有効にする。デフォルトでは、イベント トレースはオンになっています。実際のパケットをキャプチャするには、詳細トレースを有効にする必要があります。

```
<#root>

3850-2#

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

2. キャプチャ バッファを微調整する。詳細トレースに使用するバッファのサイズを決定し、必要に応じてサイズを増やします。

```
<#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
```

バッファ サイズを変更するには、次のコマンドを使用します。

```
<#root>

3850-2#

set trace control fed-punject-detail buffer-size
```

使用できる値は次のとおりです。

```
<#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. キャプチャ フィルタを追加する。次に、キャプチャに使用する各種フィルタを追加する必要があります。複数の異なるフィルタを追加し、それらのフィルタのすべてに一致、またはいずれかに一致するパケットをキャプチャできます。

   フィルタを追加するには、次のコマンドを使用します。

   <#root>

   3850-2#

   set trace fed-punject-detail direction rx filter_add

   現在使用できるオプションは次のとおりです。

   <#root>

   3850-2#

   set trace fed-punject-detail direction rx filter_add ?

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

   次に、各要素を結び付ける必要があります。このトラブルシューティング プロセスのステップ 2 で明らかになった、問題の原因になっているキューを思い出してください。キュー 16 は、CPU に多数のパケットをプッシュしているキューです。そのため、このキューをトレースし、このキューから CPU に送られているパケットを調べることには意味があります。

   キューをトレースするには、次のコマンドを使用します。

   <#root>

   3850-2#

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

この例では、次のコマンドを使用します。

```
<#root>

3850-2#

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

フィルターのすべてに一致させるか、いずれかに一致させるかを選択し、トレースを有効にする必要があります。

```
<#root>

3850-2#

set trace fed-punject-detail direction rx match_all

3850-2#

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

4. フィルタしたパケットを表示する。キャプチャしたパケットは、show mgmt-infra trace messages fed-punject-detail コマンドで表示できます。

```
<#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>
```

この出力は多くの情報を提供し、通常はパケットの送信元と内容を検出するのに十分です。

ヘッダー ダンプの最初の部分は、先ほどと同様にシステムで使用される Meta-data です。2番目の部分は実際のパケットです。

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

この送信元MACアドレスをトレースして、原因となるポートを検出できます (これがキュー16からパントされるパケットの大部分であることが特定されたら、この出力にはパケットのインスタンスが1つだけが示され、他の出力やパケットはクリップされます) 。

ただし、さらに良い方法があります。ヘッダー情報の後にある口グに注意してください。

```
[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
```

最初の口グは、このパケットの送信元のキューとタグをはっきり示しています。以前にキューに気付かなかった場合は、この方法でキューを簡単に識別できます。

2番目の口グは、送信元インターフェイスの物理インターフェイスIDファクトリ(IIF)-IDを提供するため、さらに便利です。16 進数値は、そのポートに関する情報をダンプするために使用できるハンドルです。

<#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
```

送信元インターフェイスを再度特定し、問題を引き起こしています。

トレースは、高いCPU使用率の問題のトラブルシューティングに不可欠な強力なツールであり、このような状況を正常に解決するために多くの情報を提供します。

Cisco Catalyst 3850 シリーズ スイッチ用のサンプル、組み込みイベント マネージャ（EEM）スクリプト

特定のしきい値でログの生成をトリガーするには、次のコマンドを使用します。

```
process cpu threshold type total rising


       interval
```

```
switch
```

このコマンドを実行すると、次のようなログが生成されます。

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

生成されたログには次の情報が含まれています。

- トリガー時の合計 CPU 使用率。これは、この例ではTotal CPU Utilization(total/Intr) :50/0によって識別されます。
- 上位プロセス - これらは PID/CPU% 形式で表示されます。この例では、上位プロセスは次のとおりです。

```
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 スクリプトを次に示します。

```
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：現在、process cpu threshold コマンドは3.2.Xトレインでは機能しません。もう1つ覚えておくべき点は、このコマンドは4つのコアの平均CPU使用率を調べ、この平均がコマンドで定義されているパーセンテージに達するとログを生成するということです。

Cisco IOS XE 16.x以降のリリース

Cisco IOS® XEソフトウェアリリース16.x以降が稼働するCatalyst 3850スイッチを使用している場合は、「IOS-XE 16.xが稼働するCatalystスイッチプラットフォームでのCPU使用率が高い場合のトラブルシューティング」を参照してください。

関連情報

- Cisco IOS XE とは
- Cisco Catalyst 3850 スイッチ - データ シートおよび製品資料
- シスコのテクニカルサポートとダウンロード

翻訳について
シスコは世界中のユーザにそれぞれの言語でサポート コンテンツを提供するために、機械と人による翻訳を組み合わせて、本ドキュメントを翻訳しています。ただし、最高度の機械翻訳であっても、専門家による翻訳のような正確性は確保されません。シスコは、これら翻訳の正確性について法的責任を負いません。原典である英語版（リンクからアクセス可能）もあわせて参照することを推奨します。