Содержание

Введение

Этот документ описывает, как устранить неполадки проблем использования ЦПУ, прежде всего из-за прерываний, на новой Cisco платформа IOS®-XE. Кроме того, документ представляет несколько новых команд на этой платформе, которые являются неотъемлемой частью для устренения таких проблем.

Общие сведения

Важно понять, как создан Cisco IOS XE. С Cisco IOS XE Cisco переместилась в Ядро Linux, и все подсистемы были разломаны на процессы. Все подсистемы, которые были внутренней Cisco IOS прежде - такой как драйверы модулей, Высокая доступность (HA), и так далее - теперь выполненный как программные процессы в операционной системе Linux (ОС). Сама Cisco IOS выполняется как демон в Linux OS (IOSd). Cisco IOS XE сохраняет не только тот же стиль обычного ПО Cisco IOS, но также и его операцию, поддержку и управление.

Вот некоторые полезные определения:

- **Драйвер механизма пересылки (FED)**: Это - основа Cisco Catalyst Коммутатор серии 3850 и ответственно за все аппаратное программирование/передачу.
- **IOSd**: Это - демон Cisco IOS, который работает на Ядре Linux. Это выполнено как программный процесс в ядре.
- **Система доставки пакетов (PDS)**: Это - архитектура и процесс того, как пакетам отправляют и от различной подсистемы. Как пример, это управляет, как пакеты освобождены от FED до IOSd и наоборот.
- **Маркер**: маркер может считаться указателем. Это - средство обнаружить более подробную информацию об определенных переменных, которые используются в

выходных данных, которые производит коробка. Это подобно понятию индексов Логики локальной цели (LTL) на коммутаторе Cisco Catalyst серии 6500.

Примеры практического применения: Прерывания протокола ARP

Устранение неполадок и процесс проверки в этом разделе могут широко использоваться для высокой загрузки ЦП из-за прерываний.

Шаг 1: Определите Процесс, который Использует Циклы ЦПУ

Команда show process cpu естественно отображается, как в настоящее время смотрит ЦП. Заметьте, что Cisco Catalyst, Swtich серии 3850 использует четыре ядра, и вы видите использование ЦПУ, перечисленное для всех четырех ядер:

```
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
```

От выходных данных ясно, что демон Cisco IOS использует главную часть ЦП наряду с FED, который является основой этой коробки. Когда использование ЦПУ является причиной высокой загрузки к прерываниям, вы видите, что IOSd и FED используют главную часть ЦП, и эти подпроцессы (или подмножество их) используют ЦП:

- FED TX Punject
- FED RX Punject
- FED Punject пополняет
- FED завершенный TX Punject

Можно масштабировать в любой из этих процессов с **подробным *<process* ЦПУ покажите процесс>** команда. Так как IOSd ответственен за большинство использования ЦПУ, вот более внимательное рассмотрение в это.

```
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
```

Выходные данные (выходные данные IOSd CPU) показывают, что Ищейка ARP, Процесс Дорожки IP-узла и Ввод ARP высоки. Когда ЦП прерван из-за пакетов ARP, это обычно замечается.

Шаг 2: Определите очередь ЦП который Причины Условие Высокой загрузки ЦП

Коммутатор Cisco Catalyst серии 3850 имеет много очередей, которые угождают различным типам пакетов (FED поддерживает 32 очереди ЦП RX, которые являются очередями, которые идут непосредственно в ЦП). Важно контролировать эти очереди для обнаружения, какие пакеты плывутся на плоскодонке к ЦП и которые обработаны IOSd. Эти очереди на ASIC.

> **Примечание:** Существует две ASIC-схемы: _____ 0 и 1. Порты 1 - 24 принадлежат ASIC 0.

Для рассмотрения очередей введите **asic порта статистики избыточного направления show platform** ***<asic порта>*** **cpuq** ***<очередь>*** команда ***<rx|tx>*** **направления**.

В **asic порта статистики избыточного направления show platform 0 cpuq-1** команда **rx направления**, этот**-1** список аргументов все очереди. Поэтому это списки команд все очереди приема для ASIC порта 0.

Теперь, необходимо определить, какая очередь выдвигает большое число пакетов в высокой скорости. В данном примере исследование очередей показало этого преступника:

```
<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>
```

Номер очереди равняется 16, и название очереди является **CPU_Q_PROTO_SNOOPING**.

Другой способ обнаружить очередь преступника состоит в том, чтобы ввести **избыточное направление show platform клиентская** команда.

```
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
```

Определите метку, для которой было выделено большинство пакетов. В данном примере это **65561**.

Затем введите эту команду:

```
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
```

Этот выход показывает, что очередь является **Ищейкой Proto Rx**.

S перед **65561** в выходных данных **избыточного направления show platform, клиентская** команда означает, что маркер FED приостановлен и разбит количеством входящих пакетов. Если **s** не исчезает, это означает, что очередь застревает постоянно.

Шаг 3: Формируйте дамп пакета, переданного к ЦП

В результатах **показа фунты помечают всю** команду, замечают, что о маркере, **7296672**, сообщают рядом с **Ищейкой Proto Rx Избыточного направления**.

Используйте этот маркер в **клиенте фунтов показа *<маркер>* пакет последняя** команда **приемника**. Заметьте, что необходимо включить **фунты отладки pktbuf-в-последний-раз** перед использованием команды. В противном случае вы встречаетесь с этой ошибкой:

```
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
```

С включенной отладкой вы видите эти выходные данные:

```
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             ............
```

Эта команда формирует дамп последнего пакета, полученного приемником, который является IOSd в данном примере. Это показывает, что формирует дамп заголовка, и это может декодироваться с Основанным на терминале Wireshark (TShark). Метаданные для внутреннего пользования системой, но Исходящие данные предоставляют информацию о действительном пакете. Метаданные, однако, остаются чрезвычайно полезными.

Заметьте линию, которая запускается с 0070. Используйте первые 16 битов после этого как показано здесь:

```
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
```

Интерфейс преступника определен здесь. **Gig2/0/20** - то, где существует генератор трафика, который качает трафик ARP. Если бы вы завершаете работу этого, то это решило бы проблему и минимизировало бы использование ЦПУ.

Шаг 4: Используйте отслеживание FED

Единственный недостаток с методом, обсужденным в последнем разделе, - то, что это только формирует дамп последнего пакета, который входит в приемник, и это не мог бы быть преступник.

Лучший способ устранить неполадки этого состоял бы в том, чтобы использовать функцию, названную **отслеживанием FED**. Отслеживание является методом захвата пакета (использующий различные фильтры), которые выдвинуты FED к ЦП. Отслеживание FED не так просто как функция Netdr на коммутаторе Cisco Catalyst серии 6500, как бы то ни было. Здесь процесс разделен на шаги:

1. Включите подробное отслеживание. По умолчанию трассировка события **идет**. Необходимо включить подробное отслеживание для получения действительных пакетов:

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

2. Подстройте накопительный буфер. Определите, как глубоко ваши буферы для подробного отслеживания и увеличения по мере необходимости.

   ```
   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
   ```

   Можно изменить размер буфера с этой командой:

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

   Значения, доступные вам:

   ```
   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. Добавьте фильтры перехвата. Теперь необходимо добавить различные фильтры для

перехвата. Можно добавить другие фильтры и или принять решение **совпасть со всеми** или **совпасть с любым** из тех для перехвата.

Фильтры добавлены с этой командой:

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

Эти опции в настоящее время доступны:

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

Теперь необходимо соединить вещи. Помните очередь преступника, которая была определена в Шаге 2 этого процесса устранения неполадок? Так как очередь 16 является очередью, которая выдвигает большое число пакетов к ЦП, это целесообразно отслеживать эту очередь и видеть, какие пакеты плывутся на плоскодонке к ЦП им.

Можно принять решение отследить любую очередь с этой командой:

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

Вот команда для данного примера:

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

Необходимо выбрать или **соответствие все** или **соответствие любой** для фильтров и затем разрешить трассировку:

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

4. Отобразите фильтруемые пакеты. Можно отобразить пакеты, перехваченные с **показом, инфра mgmt трассировка передает fed-punject-detail** команду.

```
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>
```

Эти выходные данные предоставляют много информации и должны, как правило, быть достаточно для обнаружения, куда пакеты прибывают из и что содержится в них.

Первой частью дампа заголовка являются снова Метаданные, которые используются системой. Вторая часть является действительным пакетом.

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

Можно принять решение отследить этот источник с MAC-адресом для обнаружения порта преступника (как только вы определили это, это - большинство пакетов, которые плывутся на плоскодонке от очереди 16; эти выходные данные только показывают, что один экземпляр пакета и других выходных данных/пакетов отсечен).

Однако существует лучший путь. Заметьте, что регистрирует, которые присутствуют после информации заголовка:

```
[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
```

Первый журнал ясно говорит вам, от которой очереди и помечают этот пакет, прибывает. Если вы не знали об очереди ранее, это - простой способ для определения, какой очередью это было.

Второй журнал еще более полезен, потому что он предоставляет Фабрику ID физического интерфейса (IIF) - ID для исходного интерфейса. Шестнадцатеричное значение является маркером, который может использоваться для формирования дампа информации о том порте:

```
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
```

Вы еще раз определили исходный интерфейс и преступника.

Отслеживание является мощным программным средством, которое важно, чтобы устранять проблемы высокой загрузки ЦП и предоставляет много информации для успешного решения такой ситуации.

Типовой сценарий встроенного диспетчера событий (EEM) для Cisco Catalyst коммутатор серии 3850

Используйте эту команду для инициирования журнала, который будет генерироваться в определенном пороге:

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

Журнал, генерируемый с командой, похож на это:

```
*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
```

Генерируемый журнал предоставляет эту информацию:

- Общее использование CPU во время триггера. Это определено **Общим ЦП Utilzation(total/Intr):50/0** в данном примере.
- Главные процессы - они перечислены в формате **% PID/ЦП**. Таким образом в данном примере, это:
  ```
  *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
  ```

Сценарий EEM показывают здесь:

```
*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
```

> **Примечание: Команда threshold ЦПУ процесса** в настоящее время не работает в 3.2. X серий. Другая точка для запоминания - то, что эта команда посмотрела на средний ЦП utilzation среди этих четырех ядер и генерирует журнал, когда это среднее число достигает процента, который был определен в команде.

Дополнительные сведения

- Что такое Cisco IOS XE?

- Cisco Catalyst 3850 коммутаторов - таблицы данных и литература
- Cisco Systems – техническая поддержка и документация