Troubleshooting High CPU Utilization

翻译：陈丹丹　武洋

这篇文档包含如下几个部分：

- CPU 利用率概述
- 何时高 CPU 利用率有隐患
- 判断根本原因
- 有用信息
- 文档获取和提交服务请求

CPU 利用率概述

当交换机启动后，交换机 CPU 会同时执行两项任务:

- 相应系统进程的各种中断请求
- 接收发送数据包

当系统进程的中断请求或需要转发的数据包增加时，CPU 利用率就会增加。

在正常的操作环境下，一个非堆叠交换机的 CPU 利用率在 5%一下。对于堆叠交换机，CPU 利用率最小会在 7%-8%之间，CPU 利用率只由主交换机测量，并且交换机堆叠的数量也会影响 CPU 利用率。

由于 Cisco 系统的后台计时器每秒会运行很多次，即使在最简单的部署中，CPU 利用率也不会显示 0%。

提醒	正常的包转发是由硬件完成的，并不涉及 CPU，所以包的转发不会受高 CPU 利用率的影响。

当需要 CPU 处理的数据包很多，或系统进程占用了过长的 CPU 处理时间，CPU 负载就会过高。当任何一种使用 CPU 资源的交换机功能受到攻击时，CPU 负载也会过高。例如：如果网络中存在广播风暴，CPU 就会接收到过多的数据包，并处理它们，进而使其他系统进程得不到 CPU 处理。

查看 CPU 利用率，输入 show cpu processes sorted 命令。输出会显示过去 5 秒、一分钟、5 分钟的 CPU 利用率。输出也会显示每个系统进程在各个周期下的利用率。

Switch# show processes cpu sorted

CPU utilization for five seconds: 5%/0%; one minute: 6%; five minutes: 5%

PID	Runtime(ms)	Invoked	uSecs	5Sec	1Min	5Min	TTY	Process
1	4539	89782	50	0.00%	0.00%	0.00%	0	Chunk Manager
2	1042	1533829	0	0.00%	0.00%	0.00%	0	Load Meter
3	0	1	0	0.00%	0.00%	0.00%	0	DiagCard3/-1
4	14470573	1165502	12415	0.00%	0.13%	0.16%	0	Check heaps
5	7596	212393	35	0.00%	0.00%	0.00%	0	Pool Manager
6	0	2	0	0.00%	0.00%	0.00%	0	Timers
7	0	1	0	0.00%	0.00%	0.00%	0	Image Licensing
8	0	2	0	0.00%	0.00%	0.00%	0	License Client N

9	1442263	25601	56336	0.00%	0.08%	0.02%	0 Licensing Auto U
10	0	1	0	0.00%	0.00%	0.00%	0 Crash writer
11	979720	2315501	423	0.00%	0.00%	0.00%	0 ARP Input
12	0	1	0	0.00%	0.00%	0.00%	0 CEF MIB API

<output truncated>

在输出中，过去 5 秒 CPU 利用率显示了两个数值（5%/0%）。

- 第一个数值，5%，表示 CPU 在过去 5 秒中的利用率。这个数值是所有活动的系统进程占用的整体 CPU 利用率，包括各种中断占用的时间。
- 第二个数值，0%，显示过去 5 秒中各种中断的利用率。中断利用率是 CPU 用来处理数据包的时间。

另外两个重要的数值显示在同一行上：过去一分钟的平均利用率和过去 5 分钟的平均利用率。这些数值适用于典型的小型且稳定的非堆叠交换环境。

在任何时间点，CPU 都有可能处理数以百计的活动系统进程。系统进程的数量是根据不同的交换模块、IOS 版本、特性集及交换机的堆叠数量（如果使用堆叠）而有所变化的。例如：在堆叠的 3750 交换机上运行 IP base 的 IOS，通常会有 475 个活动系统进程。运行 LAN base 的 2960 的系统进程就会少于堆叠的 3750 交换机。通常，IOS 的特性集越丰富，系统进程越多。

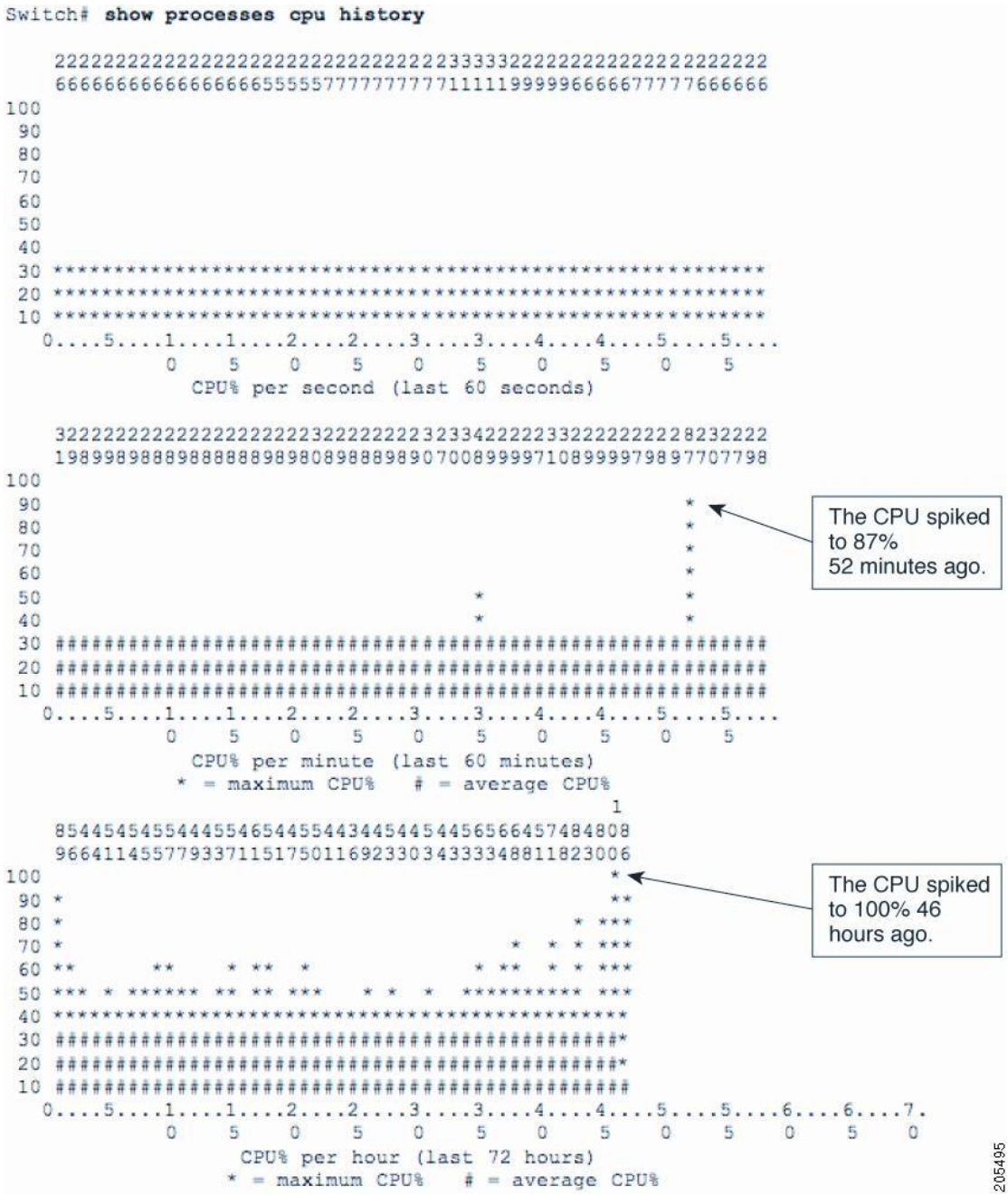
何时高 CPU 利用率有问题

这部分说明如何识别 CPU 的高利用率并判断是否有隐患:

- 高 CPU 利用率的正常情况
- 高 CPU 利用率的网络征兆
- 判断中断比率

在某些情况下，高 CPU 利用率是正常的，不会引起网络故障。当交换机失去了预期的正常运行，高 CPU 利用率才会引发问题。

通过输入 show processes cpu history 命令可以查看过去 60 秒、60 分钟和 72 小时的 CPU 利用率。这条命令的输出提供了图形化的视图显示 CPU 繁忙程度。你可以看见 CPU 是否持续繁忙，或者利用率是否达到峰值。



由已知网络时间或活动引起的 CPU 峰值是正常的。有规律的 87%的峰值是可接受的。例如：网络管理员保存配置引起的 CPU 峰值是可接受的；由于 2 层网络拓扑发生变化导致的峰值也是正常的反映。

在网络正常状态下,一段时间内持续的CPU利用率可以被认为是正常的基线。可以通过show processes cpu history 命令查看过去 72 小时内，CPU 正常利用率的基线。CPU 利用率由于未知的原因而超过此基线，被认为是存在问题。

在前例中，CPU 利用率在过去 37 小时内保持在 40%-56%之间。我们认为任何事情使 CPU 利用率在 50%以下的，都是可接受的。CPU 利用率持续在 50%以上则被认为是存在问题的。

频繁的非预知的超出正常基线的 CPU 利用率，或者突然的 CPU 利用率跳动，应该是值得关注的。

高 CPU 利用率的正常情况

在某些环境下，高 CPU 利用率是正常的。通常，2 层或 3 层的网络越大，CPU 处理相关网络流量的需求越大。以下是潜在的高 CPU 利用率操作：

- 生成树协议
- 路由表更新
- IOS 命令
- 其他导致高 CPU 利用率事件

生成树协议

一个 2 层生成树实例是 2 层交换机的 PVST 特性。生成树导致的 CPU 利用率是由生成树的实例个数以及活动接口的数量决定的。实例和活动接口越多，CPU 利用率越高。

路由表更新

启用了 3 层路由功能的交换机需要处理路由更新信息。CPU 利用率取决于以下几个因素：

- 路由更新信息的大小
- 更新的频率
- 接受路由更新的路由进程数量
- Route map 和 filter 的使用

IOS 命令

一些 IOS 命令也能导致 CPU 利用率峰值的产生，包括：

- Show tech-support
- Write memory（尤其是在堆叠交换机中）
- Show running-configuration（在堆叠交换机中）
- Debug
- 高频或大量的 IGMP requests（CPU 处理 IGMP 消息）
- 大量的 IP SLAs 监控会话（CPU 产生 ICMP 或 traceroute 包）
- SNMP，尤其是 MIB 浏览（IOS SNMP 引擎执行 SNMP 请求）
- 同时有大量的 DHCP 请求（当交换机作为 DHCP server 时）
- ARP 广播风暴
- 以太网广播风暴

高 CPU 利用率的网络征兆

高 CPU 利用率会影响系统进程的正常使用。当系统进程没有被执行，交换机（或直连网络设备）会导致网络问题。例如 2 层网络，生成树会发生重新收敛；3 层网络，路由拓扑会发生变化。

已知的高 CPU 利用率会产生的问题：

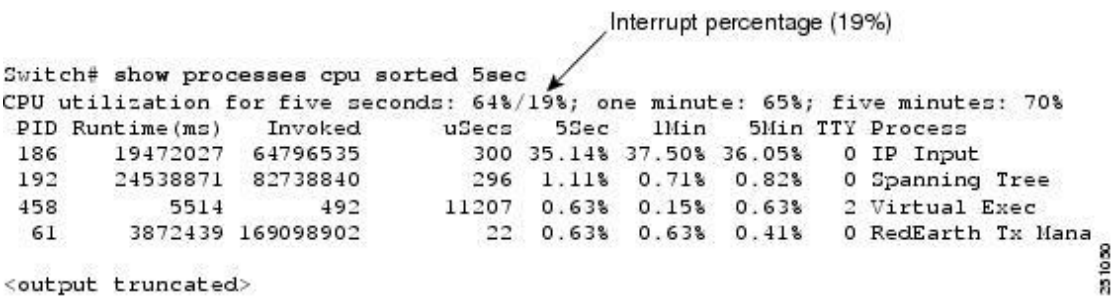
- 生成树拓扑变化（当 2 层网络设备没有在规定周期内从 root port 收到 BPDUs，它会认为根交换机 down，并尝试新的路径。生成树重新收敛产生。）
- 路由拓扑变化（例如 BGP 或 OSPF 的路由翻动）
- 以太网通道跳动(当以太网通道另一端的网络设备没有收到保持以太网通道的协议数据包时，它认为联络 down)
- 以太网停止响应正常网络管理请求：
  - ICMP ping 请求
  - SNMP 超时
  - Telnet 或 SSH 会话速度变慢或无法开始
- UDLD 翻动
- 由于 SLAs 响应超过可接受的门限导致的 IP SLAs 失效
- DHCP 或 IEEE 802.1x 失败（如果交换机不能转发或响应响应的请求）
- 需要软件进行路由的数据包将被丢弃或延时增加
- HSRP 翻动

判断中断比率

CPU 利用率的历史信息只是显示了在过去一段时间内整体的 CPU 利用率。它并不显示 CPU 所消耗的中断时间。要知道，CPU 在中断上所花费的时间对 CPU 利用率起着决定性的作用。CPU 利用率历史信息显示 CPU 何时持续收到数据包，并不显示原因。

输入 show processes cpu 命令显示当前 CPU 利用率以及每个进程所占用的 CPU 时间。在这个例子中，由于 CPU 的利用率超过了 50%的基线而变得繁忙。

箭头指向中断比率值，过去 5 秒利用率中的第二个数值。



它的正常数值在 0%-5%之间。超过 10%的中断比率就应该被审查。

判断根本原因

当你怀疑 CPU 利用率过高，先要判断是否是因为系统进程占用了太多的 CPU 时间，或者是因为接收到过多的数据包。如何判断是那种原因的方法是不一样的。这部分告诉大家如何判断并排查原因：

- 判断原因-系统进程还是网络流量
- 分析网络流量
- Debug 活动进程
- SNMP 处理进程

提醒	经常查阅相关平台和软件版本的相关文档，获取已知的 IOS bug。可以从排错的过程中排除这些问题。

当交换机 CPU 繁忙时，像 Telnet 或 SSH 这样的管理工具通常就不是很有用了。建议使用 console 解决 CPU 利用率问题。

判断原因-系统进程还是网络流量

查看 CPU 繁忙程度及哪个进程占用大量的 CPU 时间，输入 show processes cpu 命令。在输出中，过去 5 秒的 CPU 利用率中第二个数值是中断比率。使用中断比率判断问题是由系统进程产生的还是高网络流量产生的。

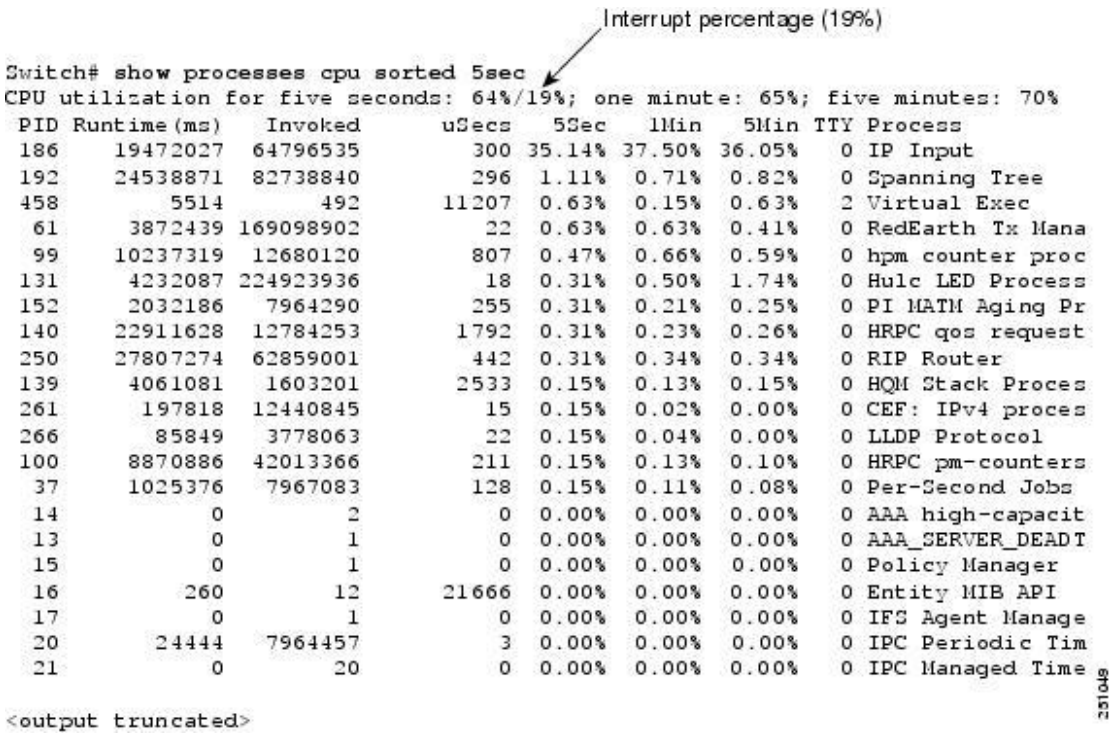
如果中断比率与整体 CPU 利用率相比，非常高，CPU 利用率问题就是由接收到过多的数据包导致。

高中断比率指示出高网络流量。这是最常见的高 CPU 利用率原因。

高 CPU 利用率和低中断比率表明是系统进程有问题。

当两项都很高时，或者不能判断是什么原因，先分析网络流量再分析系统进程。

在下面的例子中，CPU 利用率是 64%，中断比率是 19%，两者都很高。这个 CPU 利用率问题就是由收到过多的数据包导致。



在另一个例子中，中断比率相比 CPU 利用率就比较低（5%与 82%相比）。这个 CPU 利用率问题表明一个或多个系统进程占用了太多的 CPU 时间。

Switch# show processes cpu sorted 5sec

CPU utilization for five seconds: 82%/5%; one minute: 40%; five minutes: 34%

PID	Runtime(ms)	Invoked	uSecs	5Sec	1Min	5Min	TTY	Process
217	135928429	493897689	275	45.68%	18.61%	16.78%	0	SNMP ENGINE
47	61840574	480781517	128	23.80%	8.63%	7.43%	0	hrpc <-response
158	58014186	265701225	218	1.11%	1.36%	1.35%	0	Spanning Tree
46	1222030	67734870	18	0.47%	0.14%	0.08%	0	hrpc -> request
75	1034724	8421764	122	0.15%	0.06%	0.02%	0	hpm counter proc
223	125	157	796	0.15%	0.13%	0.03%	2	Virtual Exec
213	2573	263	9783	0.15%	2.43%	0.71%	1	Virtual Exec
150	578692	3251272	177	0.15%	0.02%	0.00%	0	CDP Protocol
114	8436933	3227814	2613	0.15%	0.17%	0.16%	0	HRPC qos request
105	1002819	96357752	10	0.15%	0.10%	0.06%	0	Hulc LED Process
28	701287	68160	10288	0.15%	0.01%	0.00%	0	Per-minute Jobs
215	9757808	42169987	231	0.15%	0.58%	0.56%	0	IP SNMP
12	0	1	0	0.00%	0.00%	0.00%	0	IFS Agent Manage

13	8	67388	0	0.00%	0.00%	0.00%	0 IPC

!<Output truncated>

分析网络流量

当中断比率很高，根本原因是 CPU 处理了大量的数据包。为了解决这个问题，应该先查找数据包的源，并终止该流量，同时修改交换机配置。包括一下内容:

- 系统进程和网络流量
- 系统进程和调度流量
- 判断 CPU 处理的网络流量
- 限制 CPU 处理的网络流量
- 判断交换机硬件中的调度流量
- 判断 TCAM 利用率
- 解决 TCAM 利用率

系统进程和网络流量

你可以识别那些交换机用来管理数据包的系统进程所发送给 CPU 的数据包类型。当 CPU 中断比率相比 CPU 利用率很高时，输入 show processes cpu 查看最活跃的系统进程。CPU 会通过多个系统进程接收多种数据包类型。命令会把最活跃的系统进程先输出出来。最活跃的系统进程可能正在响应接收到的数据包。

Switch# show processes cpu sorted 5sec

CPU utilization for five seconds: 64%/19%; one minute: 65%; five minutes: 70%

PID	Runtime(ms)	Invoked	uSecs	5Sec	1Min	5Min	TTY	Process
186	19472027	64796535	300	35.14%	37.50%	36.05%	0	IP Input
192	24538871	82738840	296	1.11%	0.71%	0.82%	0	Spanning Tree
458	5514	492	11207	0.63%	0.15%	0.63%	2	Virtual Exec
61	3872439	169098902	22	0.63%	0.63%	0.41%	0	RedEarth Tx Mana
99	10237319	12680120	807	0.47%	0.66%	0.59%	0	hpm counter proc
131	4232087	224923936	18	0.31%	0.50%	1.74%	0	Hulc LED Process
152	2032186	7964290	255	0.31%	0.21%	0.25%	0	PI MATM Aging Pr
140	22911628	12784253	1792	0.31%	0.23%	0.26%	0	HRPC qos request
250	27807274	62859001	442	0.31%	0.34%	0.34%	0	RIP Router

!<Output truncated>

下表列出了一些常见系统进程及其数据包类型。如果其中一种系统进程是最活跃进程，则相关的数据包很有可能就被发送到 CPU 处理。

Table 1 Processes Associated with Network Packet Processing	
System Process Name	Packet Types
IP Input	IP packets (includes ICMP)
IGMPSN	IGMP snooping packets
ARP Input	IP ARP packets
SNMP Engine	SNMP packets

系统进程和调度流量

在 3 层交换中，当 IP 路由未知时，交换机硬件将数据包调度给 CPU 做处理。被调度的数据包作中断处理，占有 CPU 资源。如果中断比率很高，但是最活跃的系统进程不在上述列表中，或者没有进程看起来足够活跃用以证明 CPU 利用率，那么，高 CPU 利用率很有可能是被调度的数据包所引起，就像如下出入：

Switch# show processes cpu sorted 5sec

CPU utilization for five seconds: 53%/28%; one minute: 48%; five minutes: 45%

PID	Runtime(ms)	Invoked	uSecs	5Sec	1Min	5Min	TTY	Process
78	461805	220334990	2	11.82%	11.53%	10.37%	0	HLFM address lea
309	99769798	1821129	54784	5.27%	1.53%	1.39%	0	RIP Timers
192	19448090	72206697	269	1.11%	0.87%	0.81%	0	Spanning Tree
250	25992246	58973371	440	0.63%	0.27%	0.29%	0	RIP Router
99	6853074	11856895	577	0.31%	0.46%	0.44%	0	hpm counter proc
131	3184794	210112491	15	0.31%	0.13%	0.12%	0	Hulc LED Process
140	20821662	11950171	1742	0.31%	0.28%	0.26%	0	HRPC qos request
139	3166446	1498429	2113	0.15%	0.11%	0.11%	0	HQM Stack Proces
67	2809714	11642483	241	0.15%	0.03%	0.00%	0	hrpc <- response

223	449344	16515401	27	0.15%	0.03%	0.00%	0 Marvell wk-a Pow
10	0	1	0	0.00%	0.00%	0.00%	0 Crash writer
11	227226	666257	341	0.00%	0.00%	0.00%	0 ARP Input

下表列出了当 CPU 忙于处理被调度的数据包时最活跃的系统进程。

Table 2 Processes Indicating Punted Packet Handling	
System Process Name	Packet Types
HLFM address lea	IP forwarding manager process
Check heaps	Memory collection process
Virtual Exec	Cisco IOS CLI process
RedEarth Tx Mana	Microprocessor communication process
hpm counter proc	Statistics collection

判断 CPU 处理的网络流量

一下用来判断发送给 CPU 的数据包类型的方法，可以互为补充，也可以单独使用。

- 交换机硬件会记录每个 CPU 接收队列的包的个数。可以通过这个数值判断何种数据包正在被接收。
- 使用 debug 命令输出所有 CPU 接收到的数据包。Debug 可以针对每个队列收集信息。
- 交换机统计所有发送和接收到的数据包。这个信息对于判断高数值、增长快的数据包类型非常有帮助。

➢ 监控 CPU 接收队列的数据包统计

如果交换机将指点的数据包调度给 CPU，交换机硬件会把数据包放入适当的 CPU 队列中并做统计。用 show controllers cpu-interface 命令查看。

Switch # show controllers cpu-interface

ASIC Rxbiterr Rxunder Fwdctfix Txbuflos Rxbufloc Rxbufdrain

ASIC0 0 0 0 0 0 0

ASIC1 0 0 0 0 0 0

cpu-queue-frames retrieved dropped invalid hol-block stray

---------- ---------- ---------- ---------- ---------- ----------

rpc 726325 0 0 0 0

stp 16108 0 0 0 0

ipc 56771 0 0 0 0

routing protocol 3949 0 0 0 0

L2 protocol 827 0 0 0 0

remote console 58 0 0 0 0

sw forwarding 0 0 0 0 0

host 0 0 0 0 0

broadcast 382 0 0 0 0

cbt-to-spt 0 0 0 0 0

igmp snooping 3567 0 0 0 0

icmp 11256 0 0 0 0

logging 0 0 0 0 0

rpf-fail 0 0 0 0 0

dstats 0 0 0 0 0

cpu heartbeat 322409 0 0 0 0

<output truncated>

交换机也会统计那些由于冲突而导致被 CPU 丢弃的数据包。每个 CPU 队列都有最大接收上限。当接收队列已满，交换机硬件会该队列丢弃额外数据包。交换机为每个队列均做统计。丢弃数量的增加意味着该 CPU 队列过载使用。

输入 show platform port-asic stat drop 命令查看 CPU 接收队列的丢弃统计情况，并标示那些正在丢弃数据包的队列。这个命令不像 show controllers cpu-interface 命令那样有用，因为它输出的队列而不是名字，并且只能查看丢弃统计。由于交换机硬件认为被 CPU 队列丢弃的数据包时发送给引擎的，所以这些包在命令输出中被叫做 Supervisor TxQueue Drop Statistics。

Switch #show platform port-asic stats drop

Port-asic Port Drop Statistics - Summary

==========

RxQueue Drop Statistics Slice0

RxQueue 0 Drop Stats Slice0: 0

RxQueue 1 Drop Stats Slice0: 0

RxQueue 2 Drop Stats Slice0: 0

RxQueue 3 Drop Stats Slice0: 0

RxQueue Drop Statistics Slice1

RxQueue 0 Drop Stats Slice1: 0

RxQueue 1 Drop Stats Slice1: 0

RxQueue 2 Drop Stats Slice1: 0

RxQueue 3 Drop Stats Slice1: 0

!<Output truncated>

Port 27 TxQueue Drop Stats: 0

Supervisor TxQueue Drop Statistics

Queue 0: 0

Queue 1: 0

Queue 2: 0

Queue 3: 0

Queue 4: 0

Queue 5: 0

Queue 6: 0

Queue 7: 0

Queue 8: 0

Queue 9: 0

Queue 10: 0

Queue 11: 0

Queue 12: 0

Queue 13: 0

Queue 14: 0

Queue 15: 0

! <Output truncated>

在输出中显示的队列号与 show controllers cpu-interface 命令输出中的队列名字是相同的顺序。例如：队列 0 对应 rpc，队列 15 对应 cpu heartbeat。

这些状态不会被重置，只能通过多次输入命令查看数值变化。这条命令还显示了其他的丢弃状态，后面会有介绍。

- **通过 CPU 接收队列排查数据包**

硬件会把待调度的数据包放到 16 个不同的 CPU 队列中。可以通过 debug 命令排查某个指定的队列。先用 show controllers cpu-interface 命令确定要排查的队列。

如果不能通过 show controllers cpu-interfac 命令确定需要 debug 的位置，建议每个队列用一次 debug 命令，直到出现 debug 信息。用不同的命令为每个队列打开和关闭 debug。

在开始排查队列之前，使用如下的配置防止其他应用占用 console 和增加系统日志缓存并打上时间戳。当 debug 结束，debug 信息都保存在系统日志中。

	Command	Purpose
Step 1	configure terminal	Enter global configuration mode.
Step 2	no logging console	Disable logging to the console terminal.
Step 3	logging buffered 128000	Enable system message logging to a local buffer, and set the buffer size to 12800 bytes.
Step 4	service timestamps debug datetime msecs localtime	Configure the system to apply a timestamp to debugging messages or system logging messages.
Step 5	exit	Return to privileged EXEC mode.

确保使用 undebug all 命令关闭所有 debug。尽管可能看不见命令提示符，交换机在任何时候都会执行 undebug all 命令。在输入此命令后，等一会，让 debug 信息停止并清空消息缓存。

Debug 信息可以帮助判断数据包的源地址，这对于具有类型或相同源的数据包非常有用。

下面的例子是 debug 信息的部分输出:

```
Switch# debug platform cpu-queues ?
  broadcast-q          Debug packets received by Broadcast Q
  cbt-to-spt-q         Debug packets received by cbt-to-spt Q
  cpuhub-q             Debug packets received by CPU heartbeat Q
  host-q               Debug packets received by host Q
  icmp-q               Debug packets received by ICMP Q
  igmp-snooping-q      Debug packets received by IGMP snooping Q
  layer2-protocol-q    Debug packets received by layer2 protocol Q
  logging-q            Debug packets received by logging Q
  remote-console-q     Debug packets received by remote console Q
  routing-protocol-q   Debug packets received by routing protocol Q
  rpffail-q            Debug packets received by RPF fail Q
  software-fwd-q       Debug packets received by software forward Q
  stp-q                Debug packets received by STP Q

Switch# debug platform cpu-queues broadcast-q
debug platform cpu-queue broadcast-q debugging is on
Switch#
Switch# debug platform cpu-queues cbt-to-spt-q
debug platform cpu-queue cbt-sbt-q debugging is on
Switch#
Switch# debug platform cpu-queues cpuhub-q
debug platform cpu-queue cpuhb debugging is on
Switch#
Switch# debug platform cpu-queues host-q
debug platform cpu-queue host-q debugging is on
Switch#
*Mar  2 22:48:06.227: L2B-Q:Dropped Null L3Hwidb: Local Port Fwding L3If:
L2If:GigabitEthernet4/0/23 DI:0x6F5, LT:1, Vlan:139   SrcGPN:185, SrcGID:185,
ACLLogIdx:0x4, MacDA:ffff.ffff.ffff, MacSA: 0009.9b00.edfe   ARP:
00010800_06040001_00099B00_EDFEAC14_8B850000_00000000_AC148B81
   TPFFD:E10000B9_008B008B_00800044-000406F5_1A1A0000_00000000

Switch# debug platform cpu-queues icmp-q
debug platform cpu-queue icmp-q debugging is on
Switch#
*Mar  2 22:48:16.947: ICMP-Q:Dropped Throttle timer not awake: Remote Port Blocked
L3If:Vlan200 L2If:GigabitEthernet1/0/3 DI:0xB4, LT:7, Vlan:200   SrcGPN:3, SrcGID:3,
ACLLogIdx:0x0, MacDA:001d.46be.7541, MacSA: 0000.0300.0101   IP_SA:10.10.200.1
IP_DA:10.10.200.5 IP_Proto:1
   TPFFD:ED000003_008B00C8_00B00222-000000B4_00060000_03090000

*Mar  2 22:48:16.947: ICMP-Q:Dropped Throttle timer not awake: Remote Port Blocked
L3If:Vlan200 L2If:GigabitEthernet1/0/3 DI:0xB4, LT:7, Vlan:200   SrcGPN:3, SrcGID:3,
ACLLogIdx:0x0, MacDA:001d.46be.7541, MacSA: 0000.0300.0101   IP_SA:10.10.200.1
IP_DA:10.10.200.5 IP_Proto:1
   TPFFD:ED000003_008B00C8_00B00222-000000B4_00050000_03090000

*Mar  2 22:48:16.947: ICMP-Q:Dropped Throttle timer not awake: Remote Port Blocked
L3If:Vlan200 L2If:GigabitEthernet1/0/3 DI:0xB4, LT:7, Vlan:200   SrcGPN:3, SrcGID:3,
ACLLogIdx:0x0, MacDA:001d.46be.7541, MacSA: 0000.0300.0101   IP_SA:10.10.200.1
IP_DA:10.10.200.5 IP_Proto:1
   TPFFD:ED000003_008B00C8_00B00222-000000B4_00040000_03090000
```

A single packet was received on the host queue. This is normal

205496

上例中的动作序列如下:

- 在执行 debug platform cpu-queue host-q 命令后,只接收到一个数据包。这是正常的。
- 当执行 debug platform cpu-queue icmp-q 命令后，debug 信息开始弹出。在 icmp-q 队列上接收到的包都是相同的。这里只显示了三个包的信息。因此，CPU 正在接收 ICMP 包的泛洪。
- 检查输出中关于数据源的信息，包含了 VLAN（200）和源 MAC 地址（0000.0300.0101）。
- 输入 show mac address-table 命令查看此 vlan 下的 mac 地址表，并找出该地址是从何处学来的。下面的输出显示了该地址是从 G1/0/3 学来的。

Switch# show mac address-table dynamic vlan 200

Mac Address Table

Vlan Mac Address Type Ports

---- ----------- -------- -----

200 0000.0300.0101 DYNAMIC Gi1/0/3

!<Output truncated>

对不同类型的数据包，当 CPU 被泛洪时，如下步骤可以让你查看单独的信息处理过程。持续对不同的 CPU 队列使用 debug 命令，直到有输出为止。

	Command	Purpose
Step 1	terminal length 0	Set the number of lines on the terminal screen for the current session to 0.
Step 2	show logging	Display the contents of the standard system logging buffer.
Step 3	terminal length 30	Set the terminal length to 30, or reset back to the original value.
Step 4	exit	Exit the CLI.

提醒 如果在开启 debug 之前修改了系统日志的缓存大小或添加时间戳，确保在 debug 结束后将配置改回默认值。

- **监控 IP 流量计数器**

交换机会统计所有 CPU 接收到的数据包。这些计数器不记录硬件转发的数据包。输入 show ip traffic 命令查看 IP 数据包类型计数器。输出结果将 IP 包分为 4 种类型（ICMP、multicast、IGMP、ARP）。当某类计数器数值迅速增加时，该类型数据包可能被泛洪到 CPU。

Switch# show ip traffic

IP statistics:

Rcvd: 12420483 total, 840467 local destination

0 format errors, 0 checksum errors, 0 bad hop count

0 unknown protocol, 222764 not a gateway

0 security failures, 0 bad options, 0 with options

Opts: 0 end, 0 nop, 0 basic security, 0 loose source route

0 timestamp, 0 extended security, 0 record route

0 stream ID, 0 strict source route, 0 alert, 0 cipso, 0 ump

0 other

Frags: 0 reassembled, 0 timeouts, 0 couldn't reassemble

0 fragmented, 0 couldn't fragment

Bcast: 0 received, 0 sent

Mcast: 0 received, 0 sent

Sent: 834640 generated, 928020828 forwarded

Drop: 189206 encapsulation failed, 0 unresolved, 0 no adjacency

0 no route, 0 unicast RPF, 0 forced drop

0 options denied, 0 source IP address zero

ICMP statistics:

Rcvd: 0 format errors, 0 checksum errors, 834640 redirects, 0 unreachable

0 echo, 0 echo reply, 0 mask requests, 0 mask replies, 0 quench

0 parameter, 0 timestamp, 0 info request, 0 other

0 irdp solicitations, 0 irdp advertisements

Sent: 834640 redirects, 0 unreachable, 0 echo, 0 echo reply

0 mask requests, 0 mask replies, 0 quench, 0 timestamp

0 info reply, 0 time exceeded, 0 parameter problem

0 irdp solicitations, 0 irdp advertisements

TCP statistics:

Rcvd: 5830 total, 0 checksum errors, 0 no port

Sent: 0 total

UDP statistics:

Rcvd: 0 total, 0 checksum errors, 0 no port

Sent: 0 total, 0 forwarded broadcasts

PIMv2 statistics: Sent/Received

Total: 0/0, 0 checksum errors, 0 format errors

Registers: 0/0 (0 non-rp, 0 non-sm-group), Register Stops: 0/0, Hellos: 0/0

Join/Prunes: 0/0, Asserts: 0/0, grafts: 0/0

Bootstraps: 0/0, Candidate_RP_Advertisements: 0/0

State-Refresh: 0/0

IGMP statistics: Sent/Received

Total: 0/0, Format errors: 0/0, Checksum errors: 0/0

Host Queries: 0/0, Host Reports: 0/0, Host Leaves: 0/0

DVMRP: 0/0, PIM: 0/0

EIGRP-IPv4 statistics:

Rcvd: 0 total

Sent: 0 total

ARP statistics:

Rcvd: 0 requests, 0 replies, 0 reverse, 0 other

Sent: 92 requests, 87 replies (0 proxy), 0 reverse

Drop due to input queue full: 444087

限制 CPU 处理的网络流量

我们可以通过在进栈接口处阻止此类数据包来防止 CPU 利用率被影响:

- 使用 storm-control {broadcast | multicast | unicast} level {level [level-low] | bps bps [bps-low] | pps pps [pps-low]}接口命令限制以太网广播泛洪或多播泛洪
- 如果导致高 CPU 利用率的根本原因是 2 层环路，可能需要重新配置生成树。

- 策略可以限制进入交换机的数据包数量。策略可以阻止进栈流量，以 bps 的级别限制速率，或者允许某些流量同时限制某些流量。策略可以基于 MAC 地址、IPv4 头、IPv6 头，或者 4 层端口号。
- 在 3 层交换机上为了防止 ARP 对 CPU 利用率的影响，可以配置 Dynamic ARP Inspection，即 DAI，通过 ip arp inspection limit {rate pps [burst interval seconds] | none} 接口命令启用该特性。

判断交换机硬件中的调度流量

当 3 层交换机正常工作是，没有进入硬件处理的路由信息，硬件将相应数据包调度给 CPU 做路由处理。被调度的数据包是正常的并且是可预测的，但是过多的数据包被调度，CPU 就会很繁忙。

交换机统计每个被硬件调度给 CPU 做路由处理的数据包。可以通过 show controllers cpu 命令查看进入每个 CPU 队列中的数据包数量。当交换机硬件调度数据包时，名字为“sw forwarding”的数值就会增加。通过重复输入此命令可以查看该数值是否快速的增长。

Switch# show controllers cpu-interface

ASIC	Rxbiterr	Rxunder	Fwdctfix	Txbuflos	Rxbufloc	Rxbufdrain
ASIC0	0	0	0	0	0	0
ASIC1	0	0	0	0	0	0

cpu-queue-frames	retrieved	dropped	invalid	hol-block	stray
rpc	2811788	0	0	0	0
stp	944641	0	0	0	0
ipc	280645	0	0	0	0
routing protocol	813536	0	0	0	0
L2 protocol	8787	0	0	0	0
remote console	2808	0	0	0	0
sw forwarding	65614320	0	0	0	0
host	25	0	0	0	0
broadcast	794570	0	0	0	0
cbt-to-spt	0	0	0	0	0
igmp snooping	18941	0	0	0	0

icmp	0	0	0	0	0
logging	0	0	0	0	0
rpf-fail	0	0	0	0	0
dstats	0	0	0	0	0
cpu heartbeat	1717274	0	0	0	0

我们可以使用 show platform ip unicast statistics 命令查看到关于调度数据包的相同信息。被调度的数据包被称为 CPUAdj 的计数器记录。

Switch# show platform ip unicast statistics

Global Stats:

HWFwdLoc:0 HWFwdSec:0 UnRes:0 UnSup:0 NoAdj:0

EncapFail:0 CPUAdj:1344291253 Null:0 Drop:0

Prev Global Stats:

HWFwdLoc:0 HWFwdSec:0 UnRes:0 UnSup:0 NoAdj:0

EncapFail:0 CPUAdj:1344291253 Null:0 Drop:0

这些状态信息没 2、3 秒更新一次。通过重复输入此命令可以查看 CPUAdj 是否有变化。如果该数值迅速增加，说明有很多数据包正在被调度给 CPU 做路由处理。

判断 TCAM 利用率

在 3 层交换机上，硬件使用 TCAM 保存路由数据库。TCAM 存储 3 层路由信息的空间是有限的。当空间已满时，新学习到的路由信息就不能被放到 TCAM 内。如果交换机硬件接收到的数据包目标地址不再 TCAM 中，硬件就会将该数据包调度给 CPU。

可以用 show platform tcam utilization 命令查看 TCAM 是否已满。

Switch# show platform tcam utilization

CAM Utilization for ASIC# 0	Max	Used
	Masks/Values	Masks/values
Unicast mac addresses:	6364/6364	31/31
IPv4 IGMP groups + multicast routes:	1120/1120	1/1
IPv4 unicast directly-connected routes:	6144/6144	4/4
IPv4 unicast indirectly-connected routes:	2048/2048	2047/2047

IPv4 policy based routing aces:	452/452	12/12
IPv4 qos aces:	512/512	21/21
IPv4 security aces:	964/964	30/30

提醒	此处每个 TCAM 所列出来的条目都是经过复杂计算的。上面的信息仅提供了当前 TCAM 的利用率。

在上例中，尽管输出中显示了在 2048 中只使用了 2047 的空间，但 IP indirectly-connected routes 资源已满，如果交换机的 TCAM 已满，硬件只能转发在 TCAM 中存在目标地址的数据包。所有其他没有与 TCAM 匹配的数据包都被调度到 CPU。在 show controllers cpu-interface 命令的输出中，“sw forwarding”数值的增加，与一个满的 TCAM，意味着被调度的数据包引起了高 CPU 利用率。

Switch# show platform ip unicast counts

# of HL3U fibs 2426

# of HL3U adjs 4

# of HL3U mpaths 0

# of HL3U covering-fibs 0

# of HL3U fibs with adj failures 0

Fibs of Prefix length 0, with TCAM fails: 0

Fibs of Prefix length 1, with TCAM fails: 0

Fibs of Prefix length 2, with TCAM fails: 0

Fibs of Prefix length 3, with TCAM fails: 0

Fibs of Prefix length 4, with TCAM fails: 0

Fibs of Prefix length 5, with TCAM fails: 0

Fibs of Prefix length 6, with TCAM fails: 0

Fibs of Prefix length 7, with TCAM fails: 0

Fibs of Prefix length 8, with TCAM fails: 0

Fibs of Prefix length 9, with TCAM fails: 0

Fibs of Prefix length 10, with TCAM fails: 0

Fibs of Prefix length 11, with TCAM fails: 0

Fibs of Prefix length 12, with TCAM fails: 0

Fibs of Prefix length 13, with TCAM fails: 0

Fibs of Prefix length 14, with TCAM fails: 0

Fibs of Prefix length 15, with TCAM fails: 0

Fibs of Prefix length 16, with TCAM fails: 0

Fibs of Prefix length 17, with TCAM fails: 0

Fibs of Prefix length 18, with TCAM fails: 0

Fibs of Prefix length 19, with TCAM fails: 0

Fibs of Prefix length 20, with TCAM fails: 0

Fibs of Prefix length 21, with TCAM fails: 0

Fibs of Prefix length 22, with TCAM fails: 0

Fibs of Prefix length 23, with TCAM fails: 0

Fibs of Prefix length 24, with TCAM fails: 0

Fibs of Prefix length 25, with TCAM fails: 0

Fibs of Prefix length 26, with TCAM fails: 0

Fibs of Prefix length 27, with TCAM fails: 0

Fibs of Prefix length 28, with TCAM fails: 0

Fibs of Prefix length 29, with TCAM fails: 0

Fibs of Prefix length 30, with TCAM fails: 0

Fibs of Prefix length 31, with TCAM fails: 0

Fibs of Prefix length 32, with TCAM fails: 693

Fibs of Prefix length 33, with TCAM fails: 0

输出中显示了 693 个匹配失败。我们可以通过这个状态判断，此时 TCAM 需要多少额外的资源来保存路由信息。使用 show ip route summary 命令查看每种路由协议有多少路由条目。

Switch# show ip route summary

IP routing table name is Default-IP-Routing-Table(0)

IP routing table maximum-paths is 32

Route Source	Networks	Subnets	Overhead	Memory (bytes)
connected	5	0	320	760
static	0	0	0	0
rip	0	2390	152960	363280
internal	1			1172

Total	6	2390	153280	365212

解决 TCAM 利用率

我满推荐用两种方法解决:

- 修改 SDM 模板
- 优化 IP 路由

修改 SDM 模板

Switch database management（SDM）模板为不同的转发类型分配了有限的 TCAM 资源。要想解决 TCAM 利用率的问题，应该适当的调整 SDM 模板。

输入 show sdm prefer 命令查看活动的 SDM 模板:

```
Switch# show sdm prefer
 The current template is "desktop default" template.
 The selected template optimizes the resources in
 the switch to support this level of features for
 8 routed interfaces and 1024 VLANs.
  number of unicast mac addresses:                  6K
  number of IPv4 IGMP groups + multicast routes:    1K
  number of IPv4 unicast routes:                    8K
    number of directly-connected IPv4 hosts:        6K
    number of indirect IPv4 routes:                 2K
  number of IPv4 policy based routing aces:         0
  number of IPv4/MAC qos aces:                      0.5K
  number of IPv4/MAC security aces:                 1K
```

交换机上默认的 TCAM 模板只允许保存 2048 条 3 层路由。为了能给 3 层路由分配更多的资源，必须减少 TCAM 其他的资源。用 sdm prefer template-name 全局命令修改资源分配。

输入 show sdm templates all 命令查看交换机上有哪些 SDM 模板:

```
Switch# show sdm templates all
 Id   Type          Name
  0   desktop       desktop default
  1   desktop       desktop vlan
  2   desktop       desktop routing
```

3	aggregator	aggregate default
4	aggregator	aggregate vlan
5	aggregator	aggregate routing
6	desktop	desktop routing pbr
8	desktop	desktop IPv4 and IPv6 default
9	desktop	desktop IPv4 and IPv6 vlan
10	aggregator	aggregate IPv4 and IPv6 default
11	aggregator	aggregate IPv4 and IPv6 vlan
12	desktop	desktop access IPv4
13	aggregator	aggregator access IPv4
14	desktop	desktop IPv4 and IPv6 routing
15	aggregator	aggregator IPv4 and IPv6 routing
16	desktop	desktop IPe
17	aggregator	aggregator IPe

- **优化 IP 路由**

当不能恰当的为 3 层交换机调整 SDM 模板时，我们应该通过路由汇总或路由过滤减少 TCAM 中的路由条目。

在相邻路由器上,用路由汇总减小路由表大小。RIP 和 EIGRP 是自动做路由汇总的,而 OSPF 则不是。

我们也可以使用路由过滤阻止那些不需要的路由进入 TCAM。

Debug 活动进程

如果 CPU 利用率很高而中断比率很低,那么高 CPU 利用率就是由一个或多个系统进程占用系统资源导致的。这并没有由于接收到数据包导致的高 CPU 利用率常见。当一个系统进程占用大量 CPU 资源时，肯定是有某个事件触发了该进程。

下表列出了一些进程占用 CPU 资源的正常情况、高百分比、可能的根本原因和建议操作:

Table 3 System Processes that Consume CPU Resources

Process Name	Percentage of Active Resources		Possible Root Cause	Suggested Action
	Normal	Considered		

		High		
Hulc LED Process	0 to 2 %	24 ports or less: More than 5 % 48 ports: More than 8 %	Physical link flapping.	Review the syslog for a physical link losing and gaining connectivity.
Inline Power Twt	0	More than 5 %	Bad power supply.	Review the syslog for reports of a failed power controller.
HACL	0	More than 50 %	Too many ACLs configured on the switch in a short time period. This can occur when the ACLs are being applied in an automated fashion (from a script).	Consider changing the SDM template.
SNMP Engine	0	More than 40%	See the "SNMP Engine Process" section.	

SNMP 处理进程

SNMP 进程只在交换机接收到 SNMP 请求时运行。请求所占用 CPU 时间与 SNMP 请求数据包的数量成正比。在请求数据包转发到 SNMP 进程前，CPU 都将做中断操作。当 SNMP 进程很忙时，在 show processes cpu 的输出中显示的中断比率通常都为非零。SNMP 进程很少将中断比率与 CPU 利用率进行比较。

当评估 SNMP 占用 CPU 利用率百分比时，交换机 CPU 利用率的基线就显得很重要。交换机通常是周期性的接收 SNMP 的请求，而且 SNMP 进程会占用大量的 CPU 资源。请求间隔可以通过 show processes cpu history 命令查看。

以下 SNMP 进程行为都会增加 CPU 的使用：

- 多个服务器同时做 SNMP 请求
- 请求交换机文件系统信息（SNMP Gets 或 SNMP GetNext 操作对于文件系统的访问是 CPU 敏感的）

- SNMP 做 MIB 的扫描

有用信息

CPU 接收队列

根据数据包的类型，硬件将数据包放入 16 个 CPU 队列中的一个。每个队列都优先级，只有当高优先级队列执行完毕才能执行低优先级。每个队列在内存中都有预留的空间用以存放数据包，这样避免一个队列或一种类型的包占用全部的内存。

下面是 CPU 对了和它们的用途：

- rpc—Remote procedure call. Used by Cisco system processes to communicate across the stack.
- stp—Spanning Tree Protocol. A Layer 2 protocol with its own queue.
- ipc—Interprocess communication. Used by Cisco system processes to communicate across the stack.
- routing protocol—Used for routing protocol packets received by other network devices.
- L2 protocol—Used for protocol packets such as LACP, UDLD, and so on.
- remote console—Used for packets when you enter the session switch-number privileged EXEC command on a stack master switch to open the console on another switch member.
- sw forwarding—Used for packets punted by the hardware for the CPU to route.
- host—Used for packets with a destination IP address matching any switch IP address. Also IP broadcast packets.
- broadcast—Receives Layer2 broadcast packets.
- cbt-to-spt—Receives multicast packets for PIM_SM.
- igmp snooping—A queue for IGMP packets.
- icmp—A queue for ICMP redirect packets.
- logging—Used for receive packets generated by hardware for ACL logging.
- rpf fail—A queue for reverse path forwarding failures.
- dstats—drop stats. Not used during normal operation.
- cpu heartbeat—Used by the CPU receiving the packets it sends to itself.

Table 4 show and debug Commands for CPU Troubleshooting

Command	Purpose	Usage
show controllers cpu-interface	Shows packet counts for all CPU receive queues.	Identify the type of packets that are flooding the CPU.
show ip route summary	Shows the number of route entries used by each protocol.	Use to help determine if additional TCAM resources are needed for IP routing.
show ip traffic	Shows a count of IP packet types received by the switch.	A rapidly incrementing packet type indicates that packet type is flooding the IP stack.
show platform ip unicast counts	Shows routes that are not programmed into the TCAM.	Use to determine how many additional TCAM resources are needed to hold the current network routes.
show platform ip unicast statistics	CPUAdj output identifies punted packets.	Enter the command several times. If the CPUAdj value increments rapidly, packets are being punted from switch hardware.
show platform port-asic stats drop	Shows CPU packets discarded due to congestion.	Identify CPU receive queues that are dropping packets due to flooding.
show platform tcam utilization	Shows TCAM maximum capacity and usage.	Determine if the TCAM is full. If the IPv4 unicast indirectly-connected routes output is at maximum, the IP routing database is full, and packets to other IP addresses are punted.
show processes cpu history	Shows a history of CPU utilization for 60 seconds, 60 minutes, and 72 hours.	Determine baseline CPU usage, and identify when spikes occur.

show processes cpu sorted [5sec]	Shows percentages for CPU utilization and CPU time spent on interrupts and lists the most active system processes in order of CPU utilization.	Determine if a CPU utilization problem is the result of too many network packets or of an active system process running on the switch.
show sdm templates all	Lists the SDM templates available on the switch.	Enter to determine if you can reallocate TCAM resources as needed.
debug platform cpu-queue queue	Debugs CPU queues.	Enter the command for each suspect CPU receive queue. The console floods at the problem queue.

当问题没有解决

如果以上排查步骤没能查找到导致高 CPU 利用率的根本问题，可以联系思科的 Technical Assistance Center （TAC）。思科的 TAC 工程师也会查看你所获得的相关信息。在联系 TAC 之前，保留这些信息，会缩短问题的排除时间。

文档获取和提交服务请求

想要获得文档、提交服务请求、收集更多的信息，访问 http://www.cisco.com/en/US/docs/switches/lan/catalyst3750/software/troubleshooting/cpu_util.html#wp1026206