了解vEdge 5000/2000/1000/100B和vEdge云平台的vManage报告的高CPU利用率

目录

简介

了解vEdge 5000/2000/1000/100B和vEdge云平台上报告的高CPU利用率

- 说明

使用fp-um进程时的CPU使用率较高

结论

简介

本文档说明为什么您可能会看到针对vEdge 5000/2000/1000/100B和vEdge云平台的vManage中报告的高CPU使用率，尽管这些平台的性能正常，并且没有报告高CPU使用率，如顶端所见。

了解vEdge 5000/2000/1000/100B和vEdge云平台上报告的高CPU利用率

使用17.2.x及更高版本，可以发现vEdge和vEdge云平台的CPU和内存消耗较高。这会在给定设备的vManage控制面板上看到。在某些情况下，这也会导致vManage中的警报和警告数量增加。

说明

当设备在正常、低负载或无负载下正常运行时，之所以报告高CPU使用率，是因为用于计算使用率的公式发生更改。对于17.2版本，根据show system status中的Load Average计算CPU使用率。

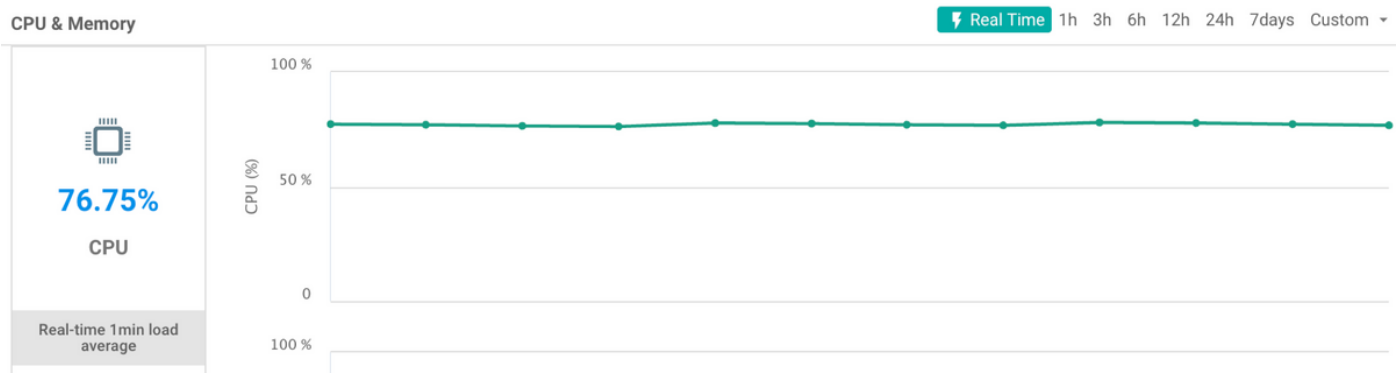
vManage显示设备的实时CPU利用率。它根据历史数据提取1分钟平均值[min1_avg]和5分钟平均值[min5_avg]。根据定义，“平均负载”包括各种因素，而不只是影响利用率计算的CPU周期。 例如，当您为平台显示此值时，会考虑IO等待时间、进程挂起时间和其他值。在这种情况下，将忽略vShell中top命令中显示的CPU状态和CPU值。

下面是一个示例，介绍如何在vManage控制面板中计算并显示CPU利用率（实际上是1分钟负载平均值）：

当您从vEdge CLI检查负载时，可以看到以下内容：

```
vEdge# show system status | include Load
Load average:            1 minute: 3.10, 5 minutes: 3.06, 15 minutes: 3.05
Load average:            1 minute: 3.12, 5 minutes: 3.07, 15 minutes: 3.06
Load average:            1 minute: 3.13, 5 minutes: 3.08, 15 minutes: 3.07
Load average:            1 minute: 3.10, 5 minutes: 3.07, 15 minutes: 3.05
```

在这种情况下，CPU使用率是根据平均负载/核心数(vCPU)计算的。 在本示例中，节点有4个核心。然后将负载平均值转换为100倍，再除以内核数。当计算所有内核的负载平均值并乘以100时，得到的值为~310。将此值除以4得出，CPU读数为77.5%，与在收集CLI输出时捕获的vManage实时图形中的值一致，如图所示。



要查看系统中的负载平均值和CPU核心数，可以从设备上的vShell查看top的输出。

在此处的示例中，vEdge包含4个vCPU。第一个内核(Cpu0)用于控制（通过较低的用户利用率看出来），而其余3个内核用于Data:

```
top - 01:14:57 up 1 day,  3:15,  1 user,  load average: 3.06, 3.06, 3.08
Tasks: 219 total,   5 running, 214 sleeping,   0 stopped,   0 zombie
Cpu0  :  1.7%us,  4.0%sy,  0.0%ni, 94.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 56.0%us, 44.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 54.2%us, 45.8%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 59.3%us, 40.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   7382664k total,  2835232k used,  4547432k free,   130520k buffers
Swap:        0k total,        0k used,        0k free,   587880k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
  978 root      20   0 3392m 664m 127m R  100  9.2   1635:21 fp-um-2
  692 root      20   0 3392m 664m 127m R  100  9.2   1635:18 fp-um-1
  979 root      20   0 3392m 664m 127m R  100  9.2   1634:51 fp-um-3
  694 root      20   0 1908m 204m 131m S    1  2.8  15:29.95 ftmd
  496 root      20   0  759m  72m 3764 S    0  1.0   1:31.50 confd
```

要从vEdge CLI获取CPU数量，可以使用以下命令：

```
vEdge# show system status | display xml | include total_cpu
    <total_cpu_count>4</total_cpu_count>
```

下面提供了计算vEdge 1000上的vManage中所示值的另一个示例。从vShell发出top后，I对于显示所有内核的负载感兴趣：

```
top - 18:19:49 up 19 days,  1:37,  1 user,  load average: 0.55, 0.71, 0.73
```

由于vEdge 1000只有一个CPU核心可用，因此此处报告的负载为55%(0.55*100)。

使用fp-um进程时的CPU使用率较高

您有时也会从顶部注意到fp-um进程运行高，并且显示多达100%的CPU。在用于数据平面处理的CPU核心上预期会出现这种情况。

从前面引用的top命令中，3个内核以100%的CPU运行，1个内核显示正常利用率：

```
top - 01:14:57 up 1 day,  3:15,  1 user,  load average: 3.06, 3.06, 3.08
Tasks: 219 total,   5 running, 214 sleeping,   0 stopped,   0 zombie
Cpu0  :  1.7%us,  4.0%sy,  0.0%ni, 94.3%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  : 56.0%us, 44.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 54.2%us, 45.8%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 59.3%us, 40.7%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:   7382664k total,  2835232k used,  4547432k free,   130520k buffers
Swap:        0k total,        0k used,        0k free,   587880k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
  978 root      20   0 3392m 664m 127m R  100  9.2   1635:21 fp-um-2
  692 root      20   0 3392m 664m 127m R  100  9.2   1635:18 fp-um-1
  979 root      20   0 3392m 664m 127m R  100  9.2   1634:51 fp-um-3
...
```

第一个内核(Cpu0)用于Control，其余三个内核用于Data。正如您在进程列表中看到的，fp-um进程使用这些资源。

fp-um是一个使用轮询模式驱动程序的进程，这意味着它持续地驻留在基础端口上并轮询数据包，以便在收到任何帧后立即对其进行处理。此进程处理转发，相当于vEdge 1000、vEdge 2000和vEdge 100中的快速路径转发。此轮询模式体系结构由英特尔用于基于数据平面开发套件(DPDK)框架的高效数据包处理。由于数据包转发是在紧密环路中实施的，因此CPU始终保持或接近100%。虽然这样做了，但不会通过这些CPU引入延迟，因为这是预期行为。

有关DPDK轮询的背景信息可在此处找到。

vEdge Cloud和vEdge 5000平台使用相同的转发架构，并且在这方面表现出相同的行为。下面是从顶部输出中抽取的vEdge 5000的示例。它有28个内核，其中2个（Cpu0和Cpu1）用于Control（与vEdge 2000类似），26个用于Data。

```
top - 02:18:30 up 1 day,  7:33,  1 user,  load average: 26.24, 26.28, 26.31
Tasks: 382 total,  27 running, 355 sleeping,   0 stopped,   0 zombie
Cpu0  :  0.7%us,  1.3%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu1  :  0.7%us,  1.3%sy,  0.0%ni, 98.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu2  : 79.4%us, 20.6%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu3  : 73.4%us, 26.6%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu4  : 73.4%us, 26.6%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu5  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu6  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
```

```
Cpu7  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu8  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu9  :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu10 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu11 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu12 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu13 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu14 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu15 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu16 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu17 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu18 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu19 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu20 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu21 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu22 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu23 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu24 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu25 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu26 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Cpu27 :100.0%us,  0.0%sy,  0.0%ni,  0.0%id,  0.0%wa,  0.0%hi,  0.0%si,  0.0%st
Mem:  32659508k total, 10877980k used, 21781528k free,   214788k buffers
Swap:        0k total,        0k used,        0k free,  1039104k cached

  PID USER      PR  NI  VIRT  RES  SHR S %CPU %MEM    TIME+  COMMAND
 2028 root      20   0 12.1g 668m 124m R  100  2.1   1897:21 fp-um-3
 2029 root      20   0 12.1g 668m 124m R  100  2.1   1897:22 fp-um-4
 2030 root      20   0 12.1g 668m 124m R  100  2.1   1897:12 fp-um-5
 2031 root      20   0 12.1g 668m 124m R  100  2.1   1897:22 fp-um-6
 2032 root      20   0 12.1g 668m 124m R  100  2.1   1897:22 fp-um-7
 2034 root      20   0 12.1g 668m 124m R  100  2.1   1897:22 fp-um-9
 2035 root      20   0 12.1g 668m 124m R  100  2.1   1897:21 fp-um-10
 2038 root      20   0 12.1g 668m 124m R  100  2.1   1897:21 fp-um-13
 2040 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-15
 2041 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-16
 2043 root      20   0 12.1g 668m 124m R  100  2.1   1897:22 fp-um-18
 2045 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-20
 2052 root      20   0 12.1g 668m 124m R  100  2.1   1897:18 fp-um-27
 2033 root      20   0 12.1g 668m 124m R  100  2.1   1897:21 fp-um-8
 2036 root      20   0 12.1g 668m 124m R  100  2.1   1897:21 fp-um-11
 2037 root      20   0 12.1g 668m 124m R  100  2.1   1897:21 fp-um-12
 2039 root      20   0 12.1g 668m 124m R  100  2.1   1897:09 fp-um-14
 2042 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-17
 2044 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-19
 2046 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-21
 2047 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-22
 2048 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-23
 2049 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-24
 2050 root      20   0 12.1g 668m 124m R  100  2.1   1897:22 fp-um-25
 2051 root      20   0 12.1g 668m 124m R  100  2.1   1897:23 fp-um-26
 1419 root      20   0  116m 5732 2280 S    0  0.0   0:02.00 chmgrd
 1323 root      20   0  753m  70m 3764 S    0  0.2   1:51.20 confd
 1432 root      20   0 1683m 172m 134m S    0  0.5   0:58.91 fpmd
```

这里，负载平均值始终较高，因为28个处理器中的26个由于fp-um进程以100%的速率运行。

结论

17.2.7之前的17.2.x版本的vManage中报告的CPU使用率不是实际CPU使用率，而是根据负载平均值计算。当平台在正常、低或无实际流量/网络负载下正常运行时，这可能导致对报告值的理解混乱，并导致与高CPU相关的错误警报。

使用17.2.7和18.2版本更改/修改此行为，这样，CPU读取现在可以根据cpu_user从顶部读取的准确度进行。

17.2版本说明中也提到了此问题。

关于此翻译
思科采用人工翻译与机器翻译相结合的方式将此文档翻译成不同语言，希望全球的用户都能通过各自的语言得到支持性的内容。

请注意：即使是最好的机器翻译，其准确度也不及专业翻译人员的水平。

Cisco Systems, Inc. 对于翻译的准确性不承担任何责任，并建议您总是参考英文原始文档（已提供链接）。