

vManage에서 보고한 vEdge 5000/2000/1000/100B 및 vEdge 클라우드 플랫폼의 높은 CPU 사용률 이해

목차

[소개](#)

[vEdge 5000/2000/1000/100B 및 vEdge 클라우드 플랫폼에서 보고된 높은 CPU 사용률 이해](#)

[설명](#)

[fp-um 프로세스를 통한 높은 CPU 사용량](#)

[결론](#)

소개

이 문서에서는 플랫폼의 성능이 정상이고 높은 CPU가 없는 vEdge 5000/2000/1000/100B 및 vEdge Cloud 플랫폼의 성능이 정상임에도 불구하고 vManage for vEdge 5000/2000/1000/100B 및 vEdge Cloud 플랫폼에 높은 CPU 사용량이 보고되는 이유를 설명합니다.

vEdge 5000/2000/1000/100B 및 vEdge 클라우드 플랫폼에서 보고된 높은 CPU 사용률 이해

17.2.x 이상 릴리스에서는 vEdge 및 vEdge 클라우드 플랫폼의 CPU 및 메모리 소비량이 더 높습니다. 이는 해당 디바이스의 vManage 대시보드에서 확인할 수 있습니다. 경우에 따라 vManage의 경고 및 경고 수도 증가할 수 있습니다.

설명

디바이스가 정상, 낮음 또는 부하가 없는 상태에서 정상적으로 수행되는 경우 CPU 사용량이 높게 보고된 이유는 사용량 계산에 사용된 공식이 변경되었기 때문입니다. 17.2 릴리스의 경우 CPU 사용률은 vEdge의 시스템 상태 표시에서 로드 평균에 따라 계산됩니다.

vManage는 디바이스에 대한 실시간 CPU 사용률을 표시합니다. 기록 데이터를 기반으로 1분 평균 [min1_avg] 및 5분 평균 [min5_avg]을 가져옵니다. 로드 평균은 정의에 따라 사용률 계산에 기여하는 CPU 주기뿐만 아니라 다양한 항목을 포함합니다. 예를 들어, 플랫폼에 대해 이 값을 제공할 때 IO 대기 시간, 프로세스 보류 시간 및 기타 값이 고려됩니다. 이 경우 vShell의 top 명령에서 CPU 상태 및 CPU 값에 대해 표시된 값을 무시합니다.

실제로 1분 로드 평균인 CPU 사용률이 계산되어 vManage 대시보드에 표시되는 방법에 대한 예는 다음과 같습니다.

vEdge CLI에서 로드를 확인하면 다음과 같이 표시됩니다.

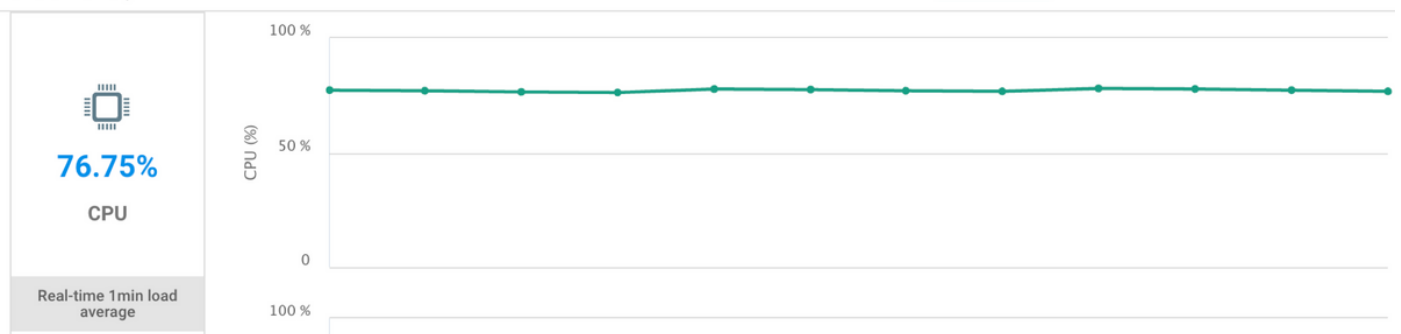
```
vEdge# show system status | include Load
```

```
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
```

이 경우 CPU 사용률은 로드 평균/코어 수(vCPU)를 기반으로 계산됩니다. 이 예에서 노드에는 4개의 코어가 있습니다. 로드 평균은 100의 인수로 변환된 후 코어 수로 나눕니다. 모든 코어에서 로드 평균을 평균하고 100을 곱하면 ~310의 값이 됩니다. 이 값을 사용하여 4로 나누면 CPU 읽기 77.5% CPU가 됩니다. 이는 CLI 출력이 수집될 때 및 이미지에 표시된 대로 수집된 vManage의 실시간 그래프에 표시된 값과 일치합니다.

CPU & Memory

Real Time 1h 3h 6h 12h 24h 7days Custom



시스템의 로드 평균 및 CPU 코어 수를 확인하기 위해 디바이스의 vShell에서 top의 출력을 참조할 수 있습니다.

이 예에서 vEdge는 4개의 vCPU를 포함합니다. 첫 번째 코어(Cpu0)는 컨트롤(사용자 활용률 감소)에 사용되고 나머지 3개 코어는 데이터에 사용됩니다.

```
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

vEdge CLI에서 CPU 수를 가져오려면 다음 명령을 사용할 수 있습니다.

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

vEdge 1000의 vManage에 표시된 값에 대한 계산의 다른 예가 여기에 나와 있습니다. vShell에서 top을 실행한 후 모든 코어에 대한 로드를 표시하기 위해 관심을 갖습니다.

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

vEdge 1000에는 CPU 코어가 하나만 있으므로 여기에 보고된 로드는 55%(0.55*100)입니다.

fp-um 프로세스를 통한 높은 CPU 사용량

fp-um 프로세스가 높은 수준으로 실행되어 최대 100% CPU를 표시한다는 사실도 간혹 확인할 수 있습니다. 이는 데이터 플레인 처리에 사용되는 CPU 코어에서 발생합니다.

앞에서 언급한 top 명령으로 3개의 코어가 100% CPU에서 작동하며 1개의 코어가 정상적인 사용량을 나타냅니다.

```
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
...											

이 첫 번째 코어(Cpu0)는 Control에 사용되고 나머지 3개의 코어는 Data에 사용됩니다. 프로세스 목록에서 볼 수 있듯이 fp-um 프로세스는 이러한 리소스를 사용합니다.

fp-um은 폴링 모드 드라이버를 사용하는 프로세스로, 패킷이 수신되는 즉시 임의의 프레임을 처리할 수 있도록 지속적으로 패킷의 기본 포트를 폴링합니다. 이 프로세스는 포워딩을 처리하며 vEdge 1000, vEdge 2000 및 vEdge 100에서의 빠른 경로 포워딩과 같습니다. 이 폴링 모드 아키텍처는 DPDK(Data Plane Development Kit) 프레임워크를 기반으로 하는 효율적인 패킷 처리에 사용됩니다. 패킷 전달이 빠듯한 루프로 구현되므로 CPU는 항상 100% 또는 거의 100%를 유지합니다. 이 작업은 완료되었지만, 이 동작은 예상된 동작이므로 이러한 CPU를 통해 레이턴시가 유입되지 않습니다.

DPDK 폴링에 대한 배경 정보는 [여기서](#) 확인할 수 있습니다.

vEdge Cloud 및 vEdge 5000 플랫폼은 동일한 포워딩 아키텍처를 사용하며 이와 관련하여 동일한 동작을 나타냅니다. 다음은 상위 출력에서 가져온 vEdge 5000의 예입니다. 28개의 코어가 있으며, 이중 2개(Cpu0 및 Cpu1)는 Control에 사용되고(vEdge 2000과 유사), 26개는 데이터에 사용됩니다.

```
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

여기서 Load Average는 fp-um 프로세스로 인해 28개 프로세서 중 26개가 100%로 실행되기 때문에 항상 높은 편이다.

결론

17.2.7 이전의 17.2.x 릴리스에 대해 vManage에서 보고된 CPU 사용량은 실제 CPU 사용량이 아니라 로드 평균을 기반으로 계산됩니다. 이로 인해 보고된 값을 이해하는 데 혼란이 발생할 수 있으며, 플랫폼이 정상, 낮음 또는 실제 트래픽/네트워크 로드가 없는 상태에서 정상적으로 작동하는 동안 높은 CPU와 관련된 잘못된 경보가 발생할 수 있습니다.

이 동작은 17.2.7 및 18.2 릴리스로 변경/수정되어 이제 CPU 읽기가 위에서 읽은 cpu_user를 기준으로 정확해질 수 있습니다.

이 문제는 [17.2 릴리스 노트에서도 언급되어 있습니다.](#)

이 번역에 관하여

Cisco는 전 세계 사용자에게 다양한 언어로 지원 콘텐츠를 제공하기 위해 기계 번역 기술과 수작업 번역을 병행하여 이 문서를 번역했습니다. 아무리 품질이 높은 기계 번역이라도 전문 번역가의 번역 결과물만큼 정확하지는 않습니다. Cisco Systems, Inc.는 이 같은 번역에 대해 어떠한 책임도 지지 않으며 항상 원본 영문 문서(링크 제공됨)를 참조할 것을 권장합니다.