Comprendere l'elevato utilizzo della CPU segnalato da vManage per le piattaforme vEdge 5000/2000/1000/100B e vEdge Cloud

Sommario

Introduzione

In questo documento viene descritto il motivo per cui l'utilizzo elevato della CPU potrebbe essere segnalato in vManage per le piattaforme vEdge 5000/2000/1000/100B e vEdge Cloud, nonostante le prestazioni delle piattaforme siano normali e non sia segnalato alcun utilizzo elevato della CPU come visualizzato nella **parte superiore**.

Comprendere l'elevato utilizzo della CPU riportato sulle piattaforme vEdge 5000/2000/1000/100B e vEdge Cloud

Con la versione 17.2.x e successive è possibile osservare un maggiore consumo di CPU e memoria per le piattaforme vEdge e vEdge Cloud. Questo viene rilevato sul dashboard vManage per un determinato dispositivo. In alcuni casi, ciò comporta anche un aumento del numero di avvisi e avvertenze in vManage.

Spiegazione

Il motivo dell'elevato utilizzo della CPU segnalato quando le prestazioni del dispositivo sono normali, basse o senza carico è dovuto a una modifica della formula utilizzata per calcolare l'utilizzo. Nelle release 17.2, l'utilizzo della CPU viene calcolato in base al **carico medio** da **show system status** sul vEdge.

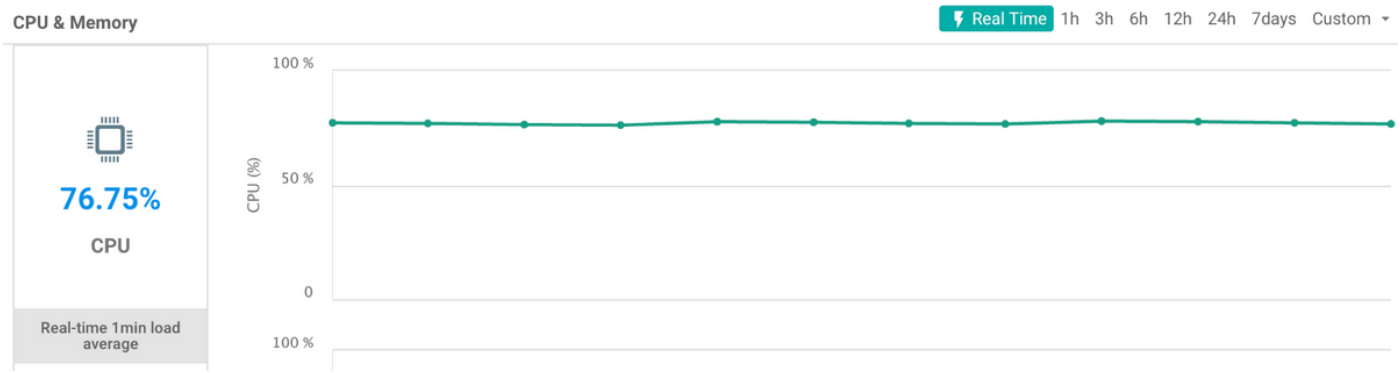
vManage mostra l'utilizzo in tempo reale della CPU per un dispositivo. Esegue il pull della **media di 1 minuto [min1_avg]** e della **media di 5 minuti [min5_avg]** in base ai dati cronologici. **Load Average**, per definizione, include vari elementi e non solo cicli della CPU che contribuiscono al calcolo dell'utilizzo. Ad esempio, il tempo di attesa I/O, il tempo di attesa processo e altri valori vengono considerati quando si presenta questo valore per la piattaforma. In questo caso, ignorare i valori mostrati per gli stati della CPU e i valori della CPU nel comando **top** di vShell.

Di seguito è riportato un esempio di come l'utilizzo della CPU, che corrisponde in realtà alla **media di carico di 1 minuto**, viene calcolato e mostrato nel dashboard vManage:

Quando si controlla il caricamento da una CLI vEdge, è possibile verificare quanto segue:

```
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
```

In questo caso, l'utilizzo della CPU viene calcolato in base al **carico medio / numero di core (vCPU)**. Per questo esempio, il nodo ha 4 core. La media del carico viene quindi convertita di un fattore 100 prima di dividerla per il numero di core. Quando si calcola la media del carico di tutti i core e la si moltiplica per 100, si ottiene un valore di ~310. Prendere questo valore e dividerla per 4 rendimenti, una lettura della CPU del 77,5%, che corrisponde al valore visualizzato nel grafico in tempo reale in vManage acquisito nel periodo di tempo in cui l'output CLI è stato raccolto e come mostrato nell'immagine.



Per visualizzare le medie del carico e il numero di core CPU nel sistema, l'output di **top** può essere consultato da vShell sul dispositivo.

Nell'esempio riportato di seguito, vEdge contiene 4 vCPU. Il primo core (**Cpu0**) viene utilizzato per il **controllo** (visto attraverso il minore utilizzo da parte dell'utente), mentre gli altri 3 core vengono utilizzati per i **dati**:

```
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
```

Per ottenere il numero di CPU dalla CLI di vEdge, è possibile utilizzare questo comando:

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

Di seguito è riportato un altro esempio di calcolo per il valore visualizzato in vManage su vEdge 1000. Dopo aver eseguito il comando **top** da vShell, I desidera visualizzare il carico per tutti i core:

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

Poiché un vEdge 1000 dispone di un solo core CPU, il carico riportato in questo caso è del 55% (0,55*100).

Utilizzo CPU elevato con processo fp-um

Dall'**alto** è inoltre possibile notare che il processo **fp-um** è molto veloce e mostra fino al 100% della CPU. Ciò è previsto nei core CPU utilizzati per l'elaborazione del piano dati.

Dal comando **top** a cui si è fatto riferimento in precedenza, 3 core funzionano al 100% della CPU e 1 core mostra un utilizzo normale:

```
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
...
```

Questo primo core (Cpu0) viene utilizzato per il **controllo** e i tre core rimanenti per i **dati**. Come si può vedere nell'elenco dei processi, il processo **fp-um** utilizza queste risorse.

fp-um è un processo che utilizza un driver in modalità poll, ovvero siede ed esegue il polling della porta sottostante per i pacchetti in modo costante in modo da poter elaborare qualsiasi frame non appena ricevuto. Questo processo gestisce l'inoltro ed è equivalente all'inoltro rapido dei percorsi in vEdge 1000, vEdge 2000 e vEdge 100. Questa architettura in modalità poll viene utilizzata da Intel per un'elaborazione efficiente dei pacchetti basata sul framework Data Plane Development Kit (DPDK). Poiché l'inoltro dei pacchetti viene implementato in un ciclo ristretto, la CPU rimane sempre al 100% o quasi. Anche se questa operazione viene eseguita, non viene introdotta alcuna latenza attraverso queste CPU, in quanto si tratta di un comportamento previsto.

Qui sono disponibili informazioni generali sul sondaggio DPDK.

Le piattaforme vEdge Cloud e vEdge 5000 utilizzano la stessa architettura di inoltro e presentano lo stesso comportamento a questo proposito. Di seguito viene riportato un esempio di un vEdge 5000 estratto dall'output **superiore**. Dispone di 28 core, di cui 2 (Cpu0 e Cpu1) vengono utilizzati per il **controllo** (come vEdge 2000) e 26 per i **dati**.

```
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
```

In questo caso, la media del carico è sempre elevata perché 26 dei 28 processori sono in esecuzione al 100% a causa del processo **fp-um**.

Conclusioni

L'utilizzo della CPU segnalato in vManage per le versioni 17.2.x precedenti alla 17.2.7 non è l'utilizzo effettivo della CPU, ma viene invece calcolato in base alla media del carico. Ciò può

generare confusione nella comprensione del valore riportato e causare falsi allarmi relativi all'elevata CPU mentre la piattaforma funziona normalmente con traffico/carico di rete normale, basso o nessun carico effettivo.

Questo comportamento viene modificato con le versioni 17.2.7 e 18.2 in modo che la lettura della CPU possa essere accurata in base alla lettura di cpu_user dall'**alto**.

Il problema è menzionato anche nelle note di rilascio 17.2.