Comprendre l'utilisation élevée du CPU signalée par vManage pour les plates-formes cloud vEdge 5000/2000/100/100B et vEdge

Contenu

Introduction

Ce document décrit pourquoi vous pourriez voir une utilisation élevée du CPU signalée dans vManage pour les plates-formes vEdge 5000/2000/100/100B et vEdge Cloud malgré les performances normales des plates-formes sans CPU élevé signalé comme affiché en **haut**.

Comprendre l'utilisation élevée du CPU signalée sur les plates-formes cloud vEdge 5000/2000/1000/100B et vEdge

Avec les versions 17.2.x et ultérieures, la consommation de CPU et de mémoire pour les plates-formes cloud vEdge et vEdge est plus élevée. Ceci est remarqué sur le tableau de bord vManage d'un périphérique donné. Dans certains cas, cela entraîne également un nombre accru d'alertes et d'avertissements dans vManage.

Explication

La raison de l'utilisation élevée du CPU signalée lorsque le périphérique fonctionne normalement avec une charge normale, faible ou nulle est due à une modification de la formule utilisée pour calculer l'utilisation. Avec les versions 17.2, l'utilisation du CPU est calculée en fonction de la **moyenne de charge** à partir de **show system status** sur vEdge.

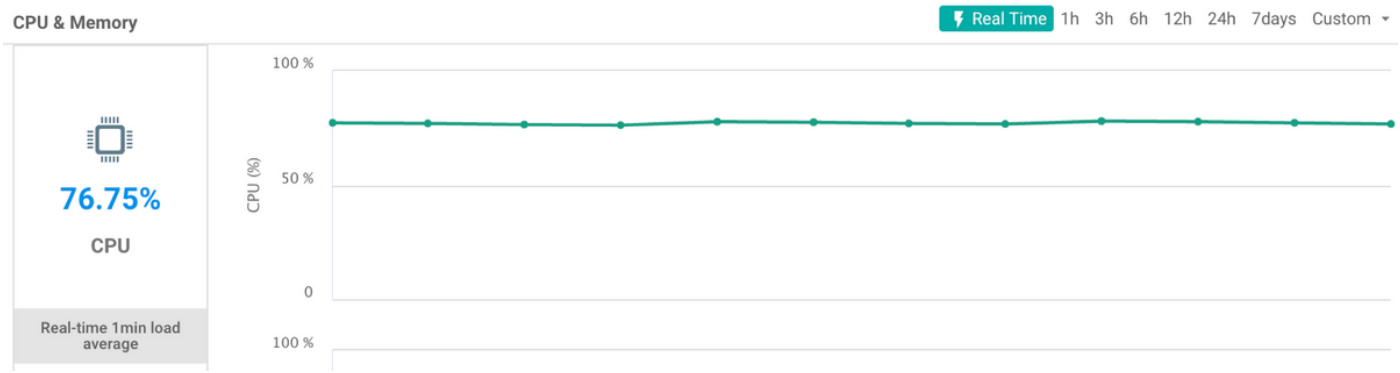
vManage affiche l'utilisation du processeur en temps réel pour un périphérique. Il extrait la **moyenne de 1 minute [min1_avg]** et la moyenne de **5 minutes [min5_avg]** en fonction des données historiques. **Load Average**, par définition, inclut différentes choses et pas seulement les cycles CPU qui contribuent au calcul de l'utilisation. Par exemple, le temps d'attente des E/S, le temps d'attente du traitement et d'autres valeurs sont pris en compte lorsque vous présentez cette valeur pour la plate-forme. Dans ce cas, vous ignorez les valeurs affichées pour les états du processeur et les valeurs du processeur dans la commande **top** de vShell.

Voici un exemple de la façon dont l'utilisation du CPU, qui est en fait la **moyenne de charge d'1 minute**, est calculée et affichée dans le tableau de bord vManage :

Lorsque vous vérifiez la charge à partir d'une CLI vEdge, ceci peut être vu :

```
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
```

Dans ce cas, l'utilisation du CPU est calculée en fonction de la **moyenne de charge / nombre de coeurs (vCPU)**. Pour cet exemple, le noeud a 4 coeurs. La moyenne de charge est ensuite convertie par un facteur de 100 avant de diviser par le nombre de coeurs. Lorsque vous faites la moyenne de charge de tous les coeurs et que vous multipliez par 100, vous obtenez une valeur de ~310. Prenez cette valeur et divisez par 4 rendements, une lecture CPU de 77,5% CPU, qui correspond à la valeur vue dans le graphique en temps réel dans vManage capturé au moment où la sortie CLI a été collectée et comme indiqué dans l'image.



Afin de voir les moyennes de charge et le nombre de coeurs de CPU dans le système, la sortie de **top** peut être consultée à partir de vShell sur le périphérique.

Dans l'exemple ci-dessous, le vEdge contient 4 vCPU. Le premier coeur (**Cpu0**) est utilisé pour **Control** (vu par une utilisation utilisateur plus faible) tandis que les 3 coeurs restants sont utilisés pour **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
```

Afin d'obtenir le nombre de CPU de l'interface de ligne de commande vEdge, cette commande peut être utilisée :

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

Un autre exemple de calcul de la valeur affichée dans vManage sur vEdge 1000 est fourni ici.

Après avoir émis **top** de vShell, **I** est intéressé afin d'afficher la charge pour tous les coeurs :

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

Comme un vEdge 1000 ne dispose que d'un seul coeur de processeur, la charge signalée ici est de 55 % (0,55*100).

Utilisation élevée du CPU avec le processus fp-um

Vous pouvez également parfois remarquer **en haut** que le processus **fp-um** fonctionne à grande vitesse et affiche jusqu'à 100 % de CPU. Ceci est attendu sur les coeurs de CPU utilisés pour le traitement du plan de données.

Àpartir de la commande **top** référencée précédemment, 3 coeurs fonctionnent à 100 % du CPU et 1 coeur affiche une utilisation 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
...
```

Ce premier coeur (Cpu0) est utilisé pour **Control** et les trois coeurs restants pour **Data**. Comme vous pouvez le voir dans la liste des processus, le processus **fp-um** utilise ces ressources.

fp-um est un processus qui utilise un pilote en mode interrogation, ce qui signifie qu'il se trouve et interroge constamment le port sous-jacent pour les paquets afin qu'il puisse traiter n'importe quelle trame dès qu'elle est reçue. Ce processus gère le transfert et est équivalent au transfert par chemin rapide dans vEdge 1000, vEdge 2000 et vEdge 100. Cette architecture en mode interrogation est utilisée par Intel pour un traitement efficace des paquets basé sur le cadre Data Plane Development Kit (DPDK). Étant donné que le transfert de paquets est mis en oeuvre en boucle étroite, le processeur reste à 100 % ou proche de ce chiffre en tout temps. Bien que cela soit fait, aucune latence n'est introduite par ces processeurs, car ce comportement est attendu.

Des informations générales sur le sondage DPDK sont disponibles ici.

Les plates-formes vEdge Cloud et vEdge 5000 utilisent la même architecture de transfert et présentent le même comportement à cet égard. Voici un exemple tiré d'un vEdge 5000 de la sortie **supérieure**. Il comporte 28 coeurs, dont 2 (Cpu0 et Cpu1) sont utilisés pour **Control** (comme le vEdge 2000) et 26 pour **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
```

Ici, la moyenne de charge est toujours élevée parce que 26 des 28 processeurs fonctionnent à 100 % en raison du processus **fp-um**.

Conclusion

L'utilisation de CPU signalée dans vManage pour les versions 17.2.x antérieures à 17.2.7 n'est pas l'utilisation de CPU réelle, mais est calculée en fonction de la moyenne de charge. Cela peut entraîner une confusion dans la compréhension de la valeur signalée et conduire à de fausses alarmes liées à un CPU élevé alors que la plate-forme fonctionne normalement avec une charge normale, faible ou nulle réelle du trafic/réseau.

Ce comportement est modifié/modifié avec les versions 17.2.7 et 18.2 de sorte que la lecture du CPU puisse maintenant être précise en fonction de la lecture cpu_user en **haut**.

Le problème est également mentionné dans les notes de version 17.2.