Comprenez l'utilisation du CPU élevé signalée par vManage pour le vEdge 5000/1000/100B et les Plateformes de nuage de vEdge

Contenu

Introduction

Ce document décrit pourquoi vous pourriez voir l'utilisation du CPU élevée signalée dans le vManage pour le vEdge 5000/1000/100B et des Plateformes de nuage de vEdge en dépit de la représentation des Plateformes étant normales sans la CPU de haute signalée comme visualisée dans le **dessus**.

Comprenez l'utilisation du CPU élevé qui est signalée sur le vEdge 5000/1000/100B et les Plateformes de nuage de vEdge

Avec le 17.2.x et les versions ultérieures, on peut observer une consommation plus élevée de CPU et mémoire pour des Plateformes de vEdge et de nuage de vEdge. Ceci est noté sur le tableau de bord de vManage pour un périphérique donné. Dans certains cas, ceci mène également à un nombre accru d'alertes et d'avertissements dans le vManage.

Explication

La raison pour l'utilisation du CPU élevée signalée quand le périphérique exécute normalement avec normal, les basses, ou aucun chargement est due à un changement de la formule utilisée afin de calculer l'utilisation. Avec les 17.2 releases, l'utilisation du processeur est calculée a basé sur la **moyenne de charge de l'état de show system** sur le vEdge.

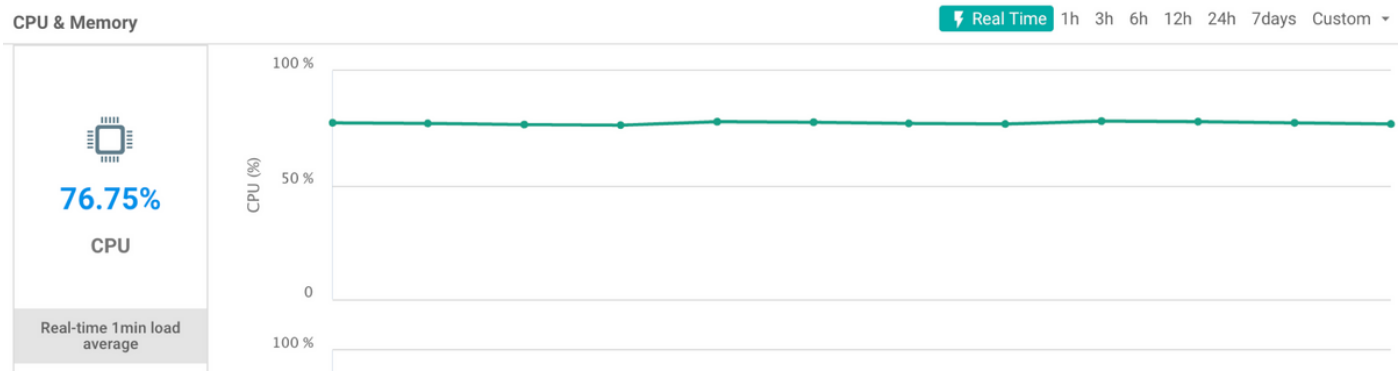
le vManage affiche l'utilisation du processeur en temps réel pour un périphérique. Il tire la **1 moyenne minute [min1_avg]** et **5 minutes [min5_avg] moyen** basées sur des données historiques. **La moyenne de charge**, par définition, inclut les divers choses et pas simplement cycles CPU qui contribuent au calcul d'utilisation. Par exemple, temps d'attente E/S, de processus en attendant le temps, et d'autres valeurs sont considérées quand vous présentez cette valeur pour la plate-forme. Dans ce cas, vous ignorez les valeurs affichées pour les états CPU et des valeurs CPU dans la commande **supérieure du** vShell.

Voici un exemple sur la façon dont l'utilisation du processeur, qui est réellement la **1 moyenne de charge minute**, obtient calculé et affiché dans le tableau de bord de vManage :

Quand vous vérifiez le chargement d'un vEdge CLI, 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 processeur est calculée a basé en moyenne**/# de noyaux (vCPUs)**. Pour cet exemple, le noeud a 4 noyaux. La moyenne de charge est alors convertie par un facteur de 100 avant que vous vous divisiez par le nombre de noyaux. Quand vous faites la moyenne de la moyenne de charge de tous les noyaux et vous multipliez par 100, vous arrivez sur une valeur de ~310. Prenez cette valeur et divisez par 4 rendements, une lecture CPU de CPU 77.5%, qui aligne avec la valeur vue dans le graphique en temps réel dans le vManage capturé autour du temps où la sortie CLI a été collectée et suivant les indications de l'image.



Afin de voir les moyennes de charge et le nombre de cores du CPU dans le système, la sortie du **dessus** peut être consultée du vShell sur le périphérique.

Dans l'exemple ci-dessous, le vEdge contient 4 vCPUs. Le premier noyau (**Cpu0**) est utilisé pour le **contrôle** (vu l'utilisation inférieure d'utilisateur) tandis que les 3 noyaux demeurants sont utilisés pour des **données** :

```
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 du vEdge CLI, cette commande peut être utilisée :

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

Un autre exemple du calcul pour la valeur affichée dans le vManage sur le vEdge 1000 est fourni ici. Après que vous émettiez le **dessus du** vShell, **I** est intéressé afin d'afficher le chargement pour

tous les noyaux :

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

Puisqu'un vEdge 1000 a seulement un core du CPU disponible, le chargement signalé ici est 55% (0.55*100).

Utilisation du CPU élevée avec point de gel-um le processus

Vous pouvez également parfois noter à partir du **dessus** que point de gel-**um le** processus exécute élevé et affiche la CPU jusqu'à de 100%. Ceci est prévu sur les cores du CPU qui sont utilisés pour le traitement de plan de données.

De la commande de **dessus** référencée plus tôt, 3 noyaux fonctionne à la CPU de 100% et 1 noyau affiche l'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 noyau (Cpu0) est utilisé pour le **contrôle** et les trois noyaux restants utilisés pour des **données**. Comme vous pouvez voir dans la liste de processus, point de gel-**um le** processus utilise ces ressources.

point de gel-um est un processus qui utilise un gestionnaire de balayage-mode, ainsi il signifie qu'il repose et vote le port sous-jacent pour des paquets constamment de sorte qu'il puisse traiter n'importe quelle trame dès qu'il sera reçu. Ce processus manipule l'expédition et est équivalent à l'expédition de rapide-chemin dans le vEdge 1000, le vEdge 2000, et le vEdge 100.
Cette architecture de balayage-mode est utilisée par Intel pour le traitement de paquets efficace basé sur le cadre DPDK. Puisque le transfert de paquet est mis en application dans une boucle serrée, la CPU reste ou de près de 100% à tout moment. Bien que ceci soit fait, aucune latence n'est introduite par ces CPU car c'est comportement prévu.

L'information générale sur l'interrogation DPDK peut il fondent ici.

Le nuage de vEdge et les Plateformes du vEdge 5000 utilisent la même architecture d'expédition et montrent le même comportement à cet égard. Voici un exemple d'un vEdge 5000 tiré de la sortie **supérieure**. Il a 28 noyaux, dont 2 (Cpu0 et Cpu1) sont utilisés pour le **contrôle** (comme le vEdge 2000) et 26 sont utilisés pour des **données**.

```
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 sur les 28 processeurs fonctionnent à 100% dû point de gel-**um au** processus.

Conclusion

L'utilisation du CPU signalée dans le vManage pour les releases 17.2.x avant 17.2.7 n'est pas l'utilisation du CPU réelle mais est à la place calculée basé sur la moyenne de charge. Ceci peut mener à la confusion pour comprendre la valeur signalée et mener aux fausses alertes liées à la CPU de haute tandis que la plate-forme fonctionne normalement avec normal, bas, ou à l'aucun trafic/charge du réseau réels.

Ce comportement est changé/modifié avec les 17.2.7 et les 18.2 libère tels que la lecture CPU peut maintenant être précise basée sur la lecture de cpu_user à partir du **dessus**.

La question est également mentionnée dans les 17.2 notes de mise à jour.