排除CPS中的高负载警报和建议的解决方法故障

目录

[简介](#)
[先决条件](#)
[要求](#)
[使用的组件](#)
[背景信息](#)
[问题](#)
[排除高负载故障](#)
[解决方法](#)

简介

本文档介绍高负载警报调查和思科策略套件(CPS)中推荐的解决方法。

先决条件

要求

Cisco 建议您了解以下主题：

- Linux
- CPS

思科还建议您拥有对CPS CLI的权限根访问权限。

使用的组件

本文档中的信息基于CPS 19.4

本文档中的信息都是基于特定实验室环境中的设备编写的。本文档中使用的所有设备最初均采用原始（默认）配置。如果您的网络处于活动状态，请确保您了解所有命令的潜在影响。

背景信息

负载平均值是Linux服务器上在定义时间段内的平均系统负载。换句话说，包括活动线程和空闲线程之和的服务器的CPU需求。

测量负载平均值对于了解服务器的工作方式至关重要；如果过载，您必须终止或优化消耗大量资源的流程，或提供更多资源来平衡工作负载。

通常，top或uptime命令提供服务器的平均负载，其输出如下：

```
[root@cps-194-aio-mob ~]# uptime
11:41:08 up 6 days, 5:20, 2 users, load average: 0.71, 0.35, 0.24
[root@cps-194-aio-mob ~]#

[root@cps-194-aio-mob ~]# top
top - 12:17:26 up 6 days, 5:56, 2 users, load average: 0.09, 0.12, 0.13
Tasks: 185 total, 1 running, 183 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.8 us, 0.8 sy, 0.0 ni, 98.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 12137348 total, 4128956 free, 5219860 used, 2788532 buff/cache
KiB Swap: 4194300 total, 4194300 free, 0 used. 6586848 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
7070 root 5 -15 8263680 1.3g 21728 S 12.5 11.6 561:38.74 java
1 root 20 0 191384 4320 2620 S 0.0 0.0 3:11.17 systemd
```

这些数字是系统负载在1、5和15分钟内的平均值。

在您继续前进之前，我们先了解一下所有类似Unix的系统中的以下两个重要短语：

系统负载/CPU负载 — 是Linux系统中CPU过量或利用率不足的度量；由CPU执行或处于空闲状态的进程数。
负载平均值 — 是在1、5和15分钟的给定时间段内计算的平均系统负载。

问题

每当CPS VM的负载平均值超过定义的阈值时，就会生成HighLoadAlert。高负载警报的阈值定义为VM中的1.5*NO CPU。此配置在/etc/snmp/snmpd.conf中提供：

```
load 12 12 12

# 1, 5 and 15 Minute Load Averages (UCD-SNMP-MIB la)
proxy -v 2c -c broadhop localhost .1.3.6.1.4.1.26878.200.3.2.70.1.4 .1.3.6.1.4.1.2021.10.1.5.1
proxy -v 2c -c broadhop localhost .1.3.6.1.4.1.26878.200.3.2.70.1.5 .1.3.6.1.4.1.2021.10.1.5.2
proxy -v 2c -c broadhop localhost .1.3.6.1.4.1.26878.200.3.2.70.1.6 .1.3.6.1.4.1.2021.10.1.5.3
proxy -v 2c -c broadhop localhost .1.3.6.1.4.1.26878.200.3.2.70.1.4.0 .1.3.6.1.4.1.2021.10.1.5.1
proxy -v 2c -c broadhop localhost .1.3.6.1.4.1.26878.200.3.2.70.1.5.0 .1.3.6.1.4.1.2021.10.1.5.2
proxy -v 2c -c broadhop localhost .1.3.6.1.4.1.26878.200.3.2.70.1.6.0 .1.3.6.1.4.1.2021.10.1.5.3
```

高负载警报示例：

```
2021-10-31T14:25:36.572711+05:30 XXXXX-lb01 snmptrapd[5717]: 2021-10-31 14:25:36 pcrfclient01
[UDP: [XX.XX.XX.XX]:46046->[XX.XX.XX.XX]:162]:#012DISMAN-EVENT-MIB::sysUpTimeInstance =
99307800#011SNMPv2-MIB::snmpTrapOID.0 = OID: DISMAN-EVENT-MIB::mteTriggerFired#011DISMAN-EVENT-
MIB::mteHotTrigger.0 = STRING: HighLoadAlert#011DISMAN-EVENT-MIB::mteHotTargetName.0 = STRING:
#011DISMAN-EVENT-MIB::mteHotContextName.0 = STRING: #011DISMAN-EVENT-MIB::mteHotOID.0 = OID:
UCD-SNMP-MIB::laErrorFlag.1#011DISMAN-EVENT-MIB::mteHotValue.0 = INTEGER: 1#011UCD-SNMP-
MIB::laNames.1 = STRING: Load-1#011UCD-SNMP-MIB::laErrMessage.1 = STRING: 1 min Load Average too
high (= 64.84)
```

排除高负载故障

在进一步调查之前，请确保受影响的VM具有符合标准的CPU计数。这可以通过相应的CPS安装指南来完成，其中提到每个VM所需的CPU计数。

唯一一个Linux命令是top命令，它可按进程提供平均负载和CPU利用率。为了确定导致高负载的进程，必须在覆盖高负载实例的特定持续时间内定期在受影响的VM中执行top命令。此命令每3秒提供

15000次的顶部输出（您可以根据您的场景更改该数字）：

```
#top -b -n15000 >> top.txt &

[root@cps-194-aio-mob ~]# top
top - 09:32:11 up 7 days, 3:11, 3 users, load average: 0.13, 0.16, 0.15
Tasks: 184 total, 1 running, 182 sleeping, 0 stopped, 1 zombie
%Cpu(s): 0.8 us, 0.8 sy, 0.0 ni, 98.4 id, 0.0 wa, 0.0 hi, 0.0 si, 0.0 st
KiB Mem : 12137348 total, 3911352 free, 5262096 used, 2963900 buff/cache
KiB Swap: 4194300 total, 4194300 free, 0 used. 6520076 avail Mem

PID USER PR NI VIRT RES SHR S %CPU %MEM TIME+ COMMAND
7014 redis 20 0 147356 2372 1184 S 6.7 0.0 48:15.15 redis-server
7070 root 5 -15 8263688 1.4g 21744 S 6.7 11.8 645:12.88 java
1 root 20 0 191384 4320 2620 S 0.0 0.0 3:38.65 systemd
2 root 20 0 0 0 0 S 0.0 0.0 0:00.12 kthreadd
3 root 20 0 0 0 0 S 0.0 0.0 0:04.51 ksoftirqd/0
5 root 0 -20 0 0 0 S 0.0 0.0 0:00.00 kworker/0:0H
7 root rt 0 0 0 0 S 0.0 0.0 0:01.76 migration/0
8 root 20 0 0 0 0 S 0.0 0.0 0:00.00 rcu_bh
9 root 20 0 0 0 0 S 0.0 0.0 11:53.47 rcu_sched
```

将HighLoadAlert实例与顶部命令输出紧密关联并进行比较，确定在发出警报时高度利用CPU的进程。

然后，要收集有关该进程的详细信息，请运行以下命令：

```
Command Template:
#ps -ef | grep {PID}

Sample command:
[root@cps-194-aio-mob ~]# ps -ef | grep 7070
root 7070 1 6 Dec02 ? 12:17:06 /usr/bin/java -server -XX:+UnlockDiagnosticVMOptions -
XX:+UnsyncloadClass -Xms2048m -Xmx2048m -javaagent:/opt/broadhop/qns-1/bin/jmxagent.jar -
Dqns.config.dir=/etc/broadhop/pcrf -Dqns.instancenum=1 -
Dlogback.configurationFile=/etc/broadhop/logback.xml -Djmx.port=9045 -
Dorg.osgi.service.http.port=8080 -Dsnmp.port=1161 -Dcom.broadhop.run.systemId=lab -
Dcom.broadhop.run.clusterId=cluster-1 -Dcom.broadhop.run.instanceId=cps-194-aio-mob-1 -
Dcom.broadhop.config.url=http://pcrfclient01/repos/run/ -
Dcom.broadhop.repository.credentials.isEncrypted=true -Dcom.broadhop.repository.credentials=qns-
svn/3300901EA069E81CE29D4F77DE3C85FA@pcrfclient01 -
Dcom.broadhop.referencedata.local.location=/var/broadhop/checkout -DdisableJms -
DrefreshOnChange=true -DenableRuntimePolling=true -DdefaultNasIp=127.0.0.1 -Xdebug -
Xrunjdwp:transport=dt_socket,server=y,suspend=n,address=1044 -Dua.version.2.0.compatible=true -
Denable.compression=true -Denable.dictionary.compression=true -DuseZlibCompression=true -
DenableBestCompression=true -DenableQueueSystem=false -
Dredis.keystore.connection.string=lb01:lb01:6379:6379 -
DbrokerUrl=failover:(tcp://lb01:61616,tcp://lb02:61616)?randomize=false -
DjmsFlowControlHost=lb02 -DjmsFlowControlPort=9045 -Dosgi.framework.activeThreadType=normal -jar
/opt/broadhop/qns-1/plugins/org.eclipse.equinox.launcher_1.1.0.v20100507.jar -console cps-194-
aio-mob:9091 -clean -os linux -ws gtk -arch x86_64
root 7846 7587 0 11:00 pts/0 00:00:00 grep --color=auto 7070
[root@cps-194-aio-mob ~]#
```

解决方法

一旦确定了导致HighLoadAlert的进程，则可以考虑以下解决方法：

步骤1.重新启动该过程。

```
#monit stop {Process Name}
Wait for 10 secs
#monit start {Process Name}
```

步骤2.如果流程包括日志记录，则检验具有调试日志级别的任何记录器，并将日志记录器的日志级别从调试更改为警告/错误。
步骤3.如果步骤1.和步骤2.不工作，则根据需要在开发团队的帮助下调整相应的配置文件。