Verzamel en grafiek CPU-statistieken met behulp van " PERF" Tool in NSO

Inhoud

Inleiding

In dit document wordt beschreven hoe u de perf-tool op NSO-hosts kunt gebruiken om prestatieproblemen te onderzoeken.

Voorwaarden

Vereisten

Cisco raadt kennis van de volgende onderwerpen aan:

- Basisgebruik van Linux/Unix-opdrachtregel
- NSO (Network Services Orchestrator) systeemarchitectuur en -werking
- CPU profilering en analyse concepten
- Vertrouwdheid met workflows voor het oplossen van prestatieproblemen

Gebruikte componenten

De informatie in dit document is gebaseerd op de volgende software- en hardware-versies:

- NSO-systeem of lokale installatie op een ondersteunde Unix/Linux-host
- Linux-distributies zoals Ubuntu, Debian, Fedora of RedHat-derivaten
- perf tool (Linux performance analysis tool)

De informatie in dit document is gebaseerd op apparaten in een specifieke laboratoriumomgeving. Alle apparaten die in dit document worden beschreven, hadden een opgeschoonde

(standaard)configuratie. Als uw netwerk live is, moet u zorgen dat u de potentiële impact van elke opdracht begrijpt.

Achtergrondinformatie

Perf is een krachtige prestatie-analyse tool in Linux, voornamelijk gebruikt voor CPU profilering. Het biedt inzicht in waar de CPU momenteel aan werkt door de belasting van functies op een lager niveau vast te leggen en te analyseren. Dit helpt te identificeren welke functies of processen de CPU bezet houden en is essentieel voor het opsporen van prestatieknelpunten.

Perf kan ook vlamgrafieken genereren, dit zijn speciale grafieken die visueel weergeven welke delen van een programma de meeste CPU-tijd gebruiken. Vlamgrafieken maken het gemakkelijker om gebieden in code te herkennen die moeten worden geoptimaliseerd.

Belangrijk is dat perf ook is opgenomen in de belangrijkste checklist voor gegevensverzameling voor OOM-gevallen (Out of Memory), zoals aanbevolen door de Business Unit (BU) van de NSO. Voor meer gedetailleerde richtlijnen voor het oplossen van OOM-problemen neemt u contact op met Cisco TAC.

Problemen met Perf-gebruik oplossen voor NSO-prestatieproblemen

Deze sectie biedt een uitgebreide workflow voor het installeren, gebruiken en analyseren van gegevens van de perf-tool op NSO-hosts om prestatieproblemen op te lossen.

PERF installeren

Stap 1: Installeer perf op uw Linux-distributie. Gebruik de juiste opdracht voor uw besturingssysteem:

Voor Ubuntu:

```
apt-get update && apt-get -y install linux-tools-generic
```

Voor Debian:

```
apt-get update && apt-get -y install linux-perf
```

Voor Fedora/RedHat-derivaten:

```
dnf install -y perf
```

Neem voor meer informatie over bekende waarschuwingen tijdens het installeren van perf contact op met het Cisco TAC-team.

Bemonstering van de gegevens

Stap 1: Identificeer het belangrijkste NSO-proces.

Gebruik de onderstaande opdracht om het NSO-proces (ncs.smp) te lokaliseren:

```
ps -ef | grep ncs\.smp
```

Voorbeeld van uitvoer:

```
root    120829       1  16 13:23 ? 00:11:08 /opt/ncs/current/lib/ncs/erts/bin/ncs.smp -K true -P 277140
root    121424   120604  0 14:30 pts/0 00:00:00 grep --color=auto ncs.smp
```

Stap 2: Als alternatief moet u de PID van het belangrijkste Java-proces gebruiken dat is gekoppeld aan NSO, vooral als u zich richt op Java-bewerkingen. Voer uit:

```
ps -ef | grep NcsJVMLauncher
```

Voorbeeld van uitvoer:

```
root    120903   120833  6 13:32 ? 00:03:40 java -classpath :/opt/ncs/current/java/jar/* -Dhost=127.0.0
root    121435   120604  0 14:33 pts/0 00:00:00 grep --color=auto NcsJVMLauncher
```

Stap 3: Voer de problematische testcase of use-case uit om het prestatiescenario te valideren.

Stap 4: Voer op een ander terminalvenster perf uit tegen de relevante proces-ID's (PID's). Gebruik het onderstaande opdrachtformaat en vervang XX, YY, ZZ door de hierboven verkregen PID's:

```
perf record -F 100 -g -p XX,YY,ZZ
```

Bijvoorbeeld om het hele systeem te profileren en oproepgrafieken op 99Hz te verzamelen voor specifieke PID's:

```
perf record -a -g -F 99 -p 120829,120903
```

Voorbeeld van uitvoer:

```
Warning:
PID/TID switch overriding SYSTEM
```

Optiebeschrijvingen:

- -a: Alle CPU's; systeembrede verzameling van alle CPU's (standaard als geen doel is opgegeven).
- -g: Oproepgrafieken vastleggen (stapelsporen). Identificeert waar functies worden aangeroepen.
- -F: bemonsteringsfrequentie in Hz. Hogere frequenties verhogen de precisie, maar voegen overhead toe.
- -p: Hiermee geeft u de proces-ID('s) op.

Stap 5: Wanneer u klaar bent met het verzamelen van monsters, stopt u perf met Ctrl + C:

```
^C
[ perf record: Woken up 1 times to write data ]
[ perf record: Captured and wrote 0.646 MB perf.data (4365 samples) ]
```

U ziet nu een perf.data bestand in de huidige directory.

Stap 6: Genereer een overzichtsrapport met deze opdracht:

```
perf report -n --stdio > perf_report.txt
```

Optiebeschrijvingen:

- -n: Symbolen weergeven zonder groepering (vlakke weergave).
- --studio: Uitgang naar standaarduitvoer (de terminal).

Op dit punt moet u beide bestanden opslaan (perf.data en perf_report.txt) en deze delen met uw ondersteuningscontactpersoon voordat u verder gaat met verdere analyse.

Als de opname succesvol was, toont perf_report.txt een boomachtige structuur die een hiërarchische aanroepgrafiek weergeeft. Percentages helpen u hotspots te identificeren waar de meeste CPU-tijd wordt besteed.

Voorbeeldfragment:

```
# Children      Self       Samples  Command          Shared Object         Symbol
# ........  ........  ............  ...............  ....................  ............................
#    30.61%     0.00%             0  C2 CompilerThre  libc.so.6             [.] start_thread
#            ---start_thread
#               thread_native_entry(Thread*)
#               Thread::call_run()
#               JavaThread::thread_main_inner()
#               CompileBroker::compiler_thread_loop()
#                           --30.58%--CompileBroker::invoke_compiler_on_method(CompileTask*)
#                                     --30.47%--C2Compiler::compile_method(ciEnv*, ciMethod*, int, bool
#                                     Compile::Compile(ciEnv*, ciMethod*, int, bool, bool, bool, bool, 
#                                               |--17.57%--Compile::Code_Gen()
#                                               |          |--12.46%--PhaseChaitin::Register_Allocate()
#                                               |          |          |--2.79%--PhaseChaitin::build_ifg_
#                                               |          |          |          --1.05%--PhaseChaitin
#                                               |          |--1.49%--PhaseChaitin::Split(unsigned int, 
#                                               |          |--1.26%--PhaseChaitin::post_allocate_copy_r
```

Interpretatie:

- Proces/Thread: De C2 CompilerThread wordt geanalyseerd.
- Totaal CPU-gebruik: deze thread is verantwoordelijk voor 30,61% van de CPU-tijd.
- Functiestroom: De thread begint met `start_thread` en delegeert werk over verschillende lagen. Het grootste deel van de CPU-tijd (30,47%) wordt besteed in `C2Compiler::compile_method`, wat een potentiële hotspot aangeeft.

Een vlamgrafiek genereren

Stap 1: Genereer een prestatiemonster van alle CPU's en processen over een bepaald interval (bijv. 60 seconden):

```
perf record -a -g -F 99 sleep 60
```

Voorbeeld van uitvoer:

```
[ perf record: Woken up 32 times to write data ]
[ perf record: Captured and wrote 10.417 MB perf.data (67204 samples) ]
```

Stap 2: Kopieer of verplaats dit perf.data bestand naar een host van waaruit u de flamegraph template repository kunt downloaden.

Stap 3: Converteer het bestand perf.data naar een tekstformaat:

```
perf script > data.perf
```

Stap 4: Kloon de FlameGraph GitHub repository en plaats data.perf in deze directory:

```
cp data.perf $PWD/FlameGraph/.
```

Stap 5: Vouw de stapelsporen in voor de verwerking van de flamegraaf:

<#root>

```
cat data.perf | ./stackcollapse-perf.pl > data.perf-folded
```

Stap 6: Genereer de vlam grafiek SVG-bestand:

<#root>

```
./flamegraph.pl data.perf-folded > data.svg
```

Opmerking: Als u de fout "can not locate open.pm in @INC" op CentOS of RHEL tegenkomt, installeert u de vereiste Perl-module:

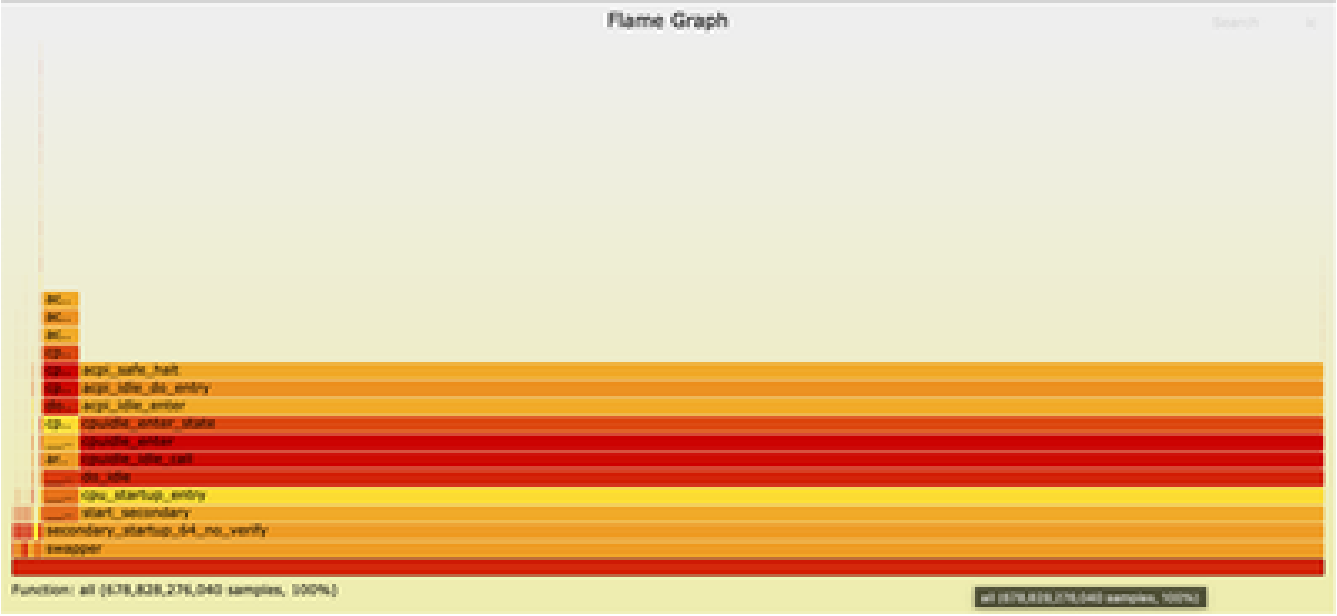
```
yum install perl-open.noarch
```

Stap 7: Open het bestand data.svg in uw favoriete webbrowser om de vlamgrafiek te visualiseren.

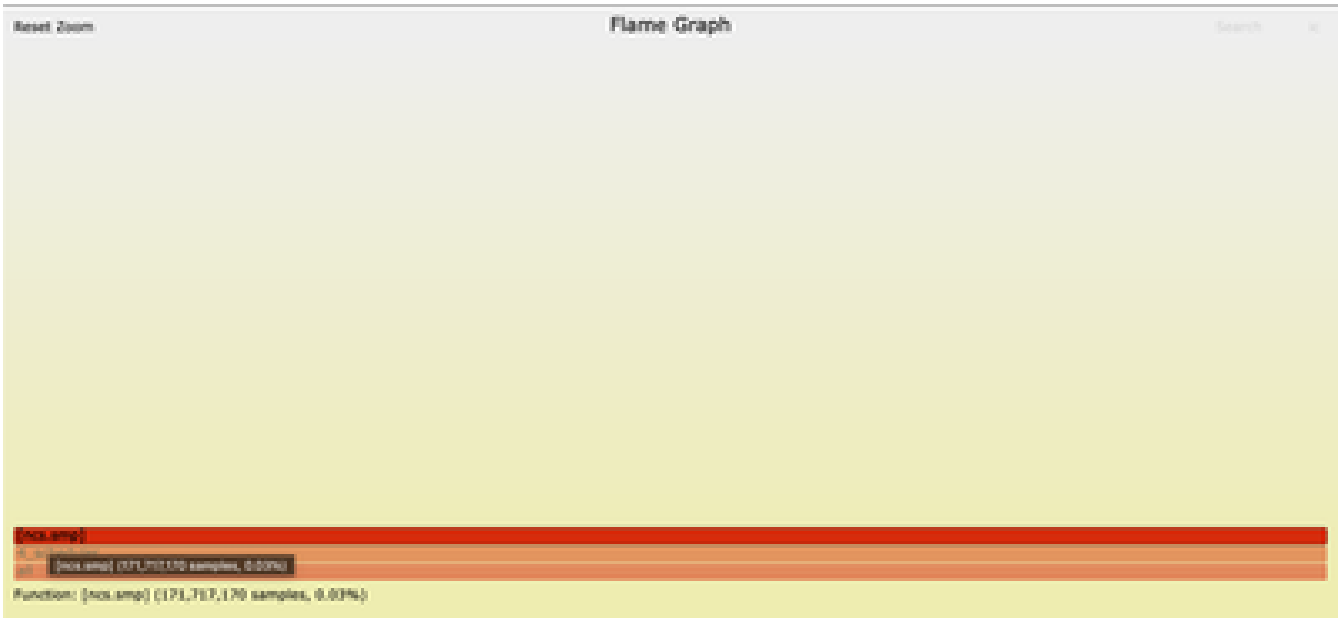
Blader door de Flame Graph

Zodra het flame graph-bestand in uw browser is geopend, kunt u ermee communiceren door op een willekeurig vak te klikken om in te zoomen op die functie en de aanroepstapel. De lengte van elke box geeft de hoeveelheid CPU-tijd weer die in die functie wordt doorgebracht en de aanroepstapel. Deze visualisatie maakt het gemakkelijk om hotspots en gebieden voor

optimalisatie te identificeren.



Zoomed op ncs.smp:



Gerelateerde informatie

- Linux Perf Bekende voorbehouden
- Cisco Technical Support en downloads

Over deze vertaling

Cisco heeft dit document vertaald via een combinatie van machine- en menselijke technologie om onze gebruikers wereldwijd ondersteuningscontent te bieden in hun eigen taal. Houd er rekening mee dat zelfs de beste machinevertaling niet net zo nauwkeurig is als die van een professionele vertaler. Cisco Systems, Inc. is niet aansprakelijk voor de nauwkeurigheid van deze vertalingen en raadt aan altijd het oorspronkelijke Engelstalige document (link) te raadplegen.